Correlated Data Notes

Brianna Heggeseth

2025-07-30

Table of contents

Preface

Greetings!

This course is about three types of **correlated data**. In most of the statistical methods and models you've learned in past classes, you assumed that observations are independently drawn from a population through random sampling or independently generated from a random process. For this to be true, the observed value of a randomly chosen unit or subject cannot influence or be systematically related to the observed value of another unit or subject.

There are many circumstances in which the **independence assumption** is not valid or realistic to assume.

- If you collect data on biological siblings, the children with similar genetics and home environment will be more similar to each other than randomly selected children.
- Educational data collected in schools is not independent. Students in the same classroom will be more similar in their learning than students from different classrooms because they have a common teacher and curriculum.
- Data collected on the same individuals over time is correlated; the repeated measurements on an individual will be more similar than measurements across individuals.

One of the learning goals of this course is to understand the consequences of incorrectly making the independence assumption in a statistical model and the potential impact it has on our conclusions.

We'll also learn about appropriate statistical models and methods for analyzing data generated with natural dependencies and correlation.

1 Introduction

1.1 Data Types

In this class, we focus on **temporal data**, in which we have repeated measurements over time on the same units, and **spatial data**, in which the measurement location plays a meaningful role in the analysis.

There are two types of temporal data we discuss in this class.

1. We call temporal data a **time series** if we have measurements on a smaller number of units or subjects taken at many (typically > 20) regular and equally-spaced times.
2. We call temporal data **longitudinal data** if we have measurements on many units or subjects taken at approximately 2 to 20 observation times (potentially irregular, unequally-spaced times) that may differ between subjects. If we have repeated measurements on each subject in different conditions, rather than necessarily over some time, we call this data **repeated measures data**, but the methods will be the same for both longitudinal and repeated measures data.

Spatial data can be measured as

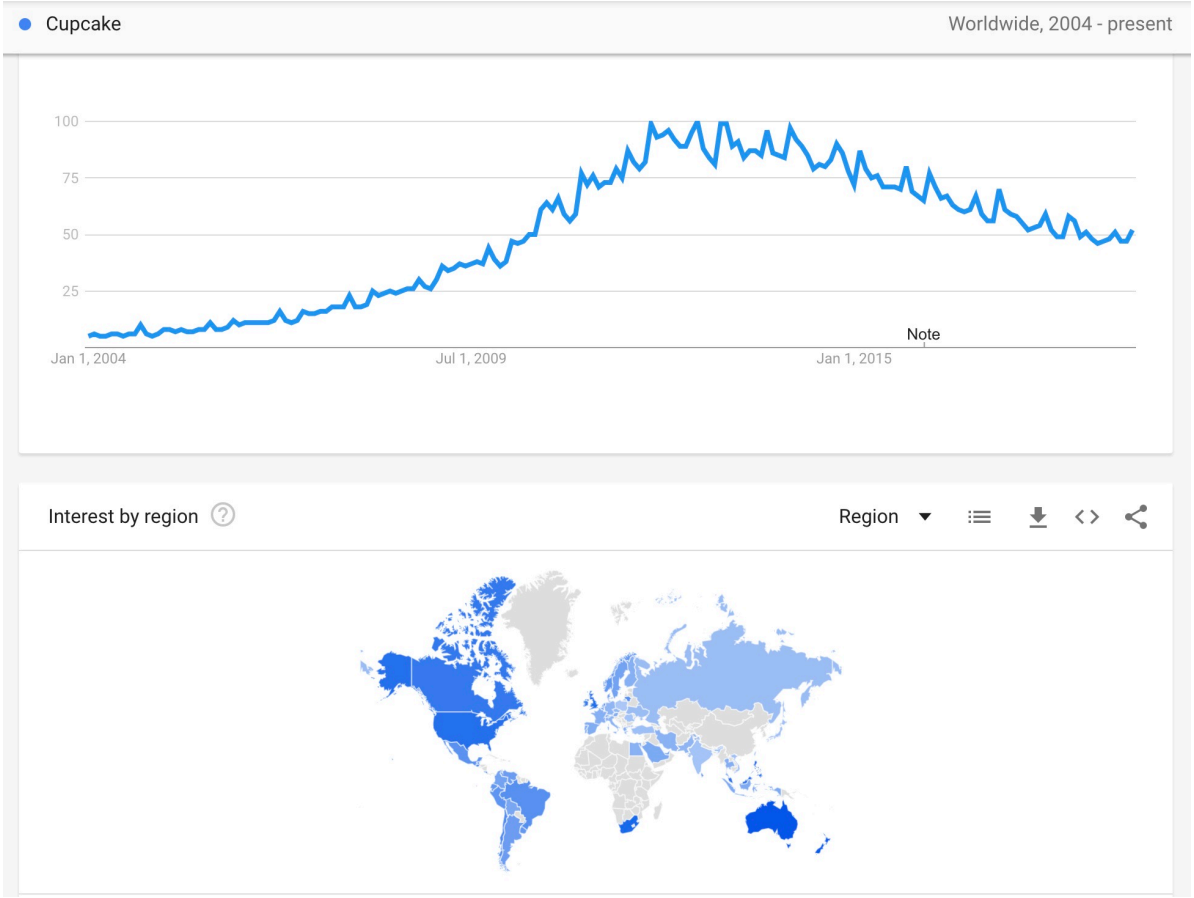
1. observations at a **point in space**, typically measured using a longitude and latitude coordinate system, or
2. **areal units**, which are aggregated summaries based on natural or societal boundaries such as county districts, census tracts, postal code areas, or any other arbitrary spatial partition.

The common thread between these types of data is that observations measured closer in time or space tend to be more similar (more positively correlated) than observations measured further away in time or space.

1.1.1 Data Type Examples

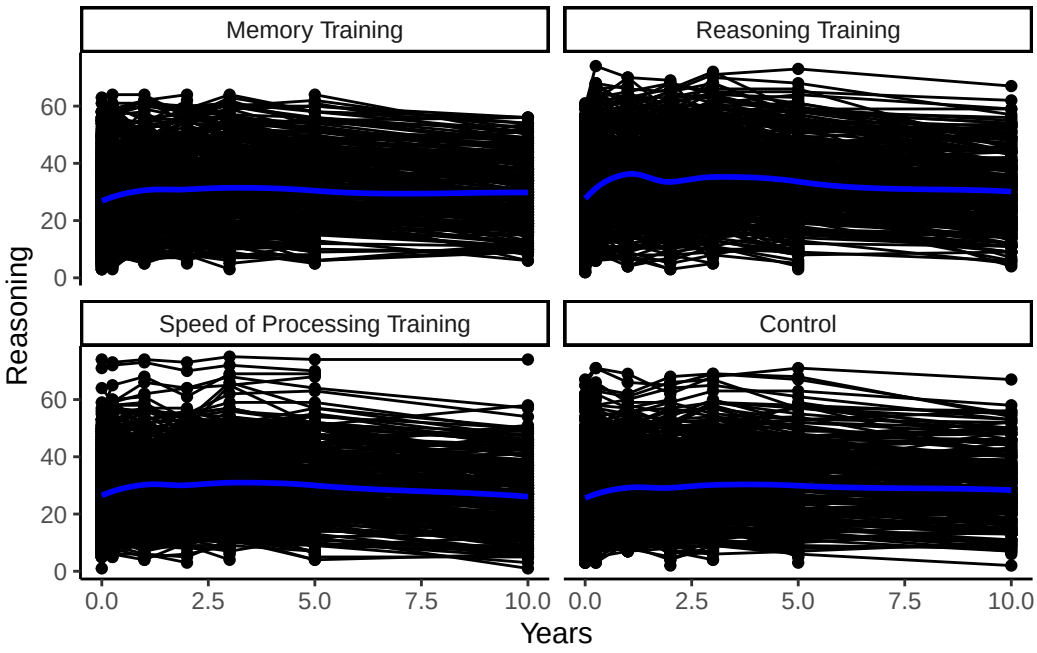
Here are some examples of these types of data.

- **Time series**: Below are the daily use frequencies for the search term "cupcake" from Google Trends (data source). While there is a spatial component (areal units are countries), we could focus solely on the time series and ignore the country. We notice a larger overall trend of increase and then a slight decrease in the search frequency. We also note there may be a cyclic pattern that may indicate that there are predictable times of the year in which searches for "cupcake" might be more or less popular.

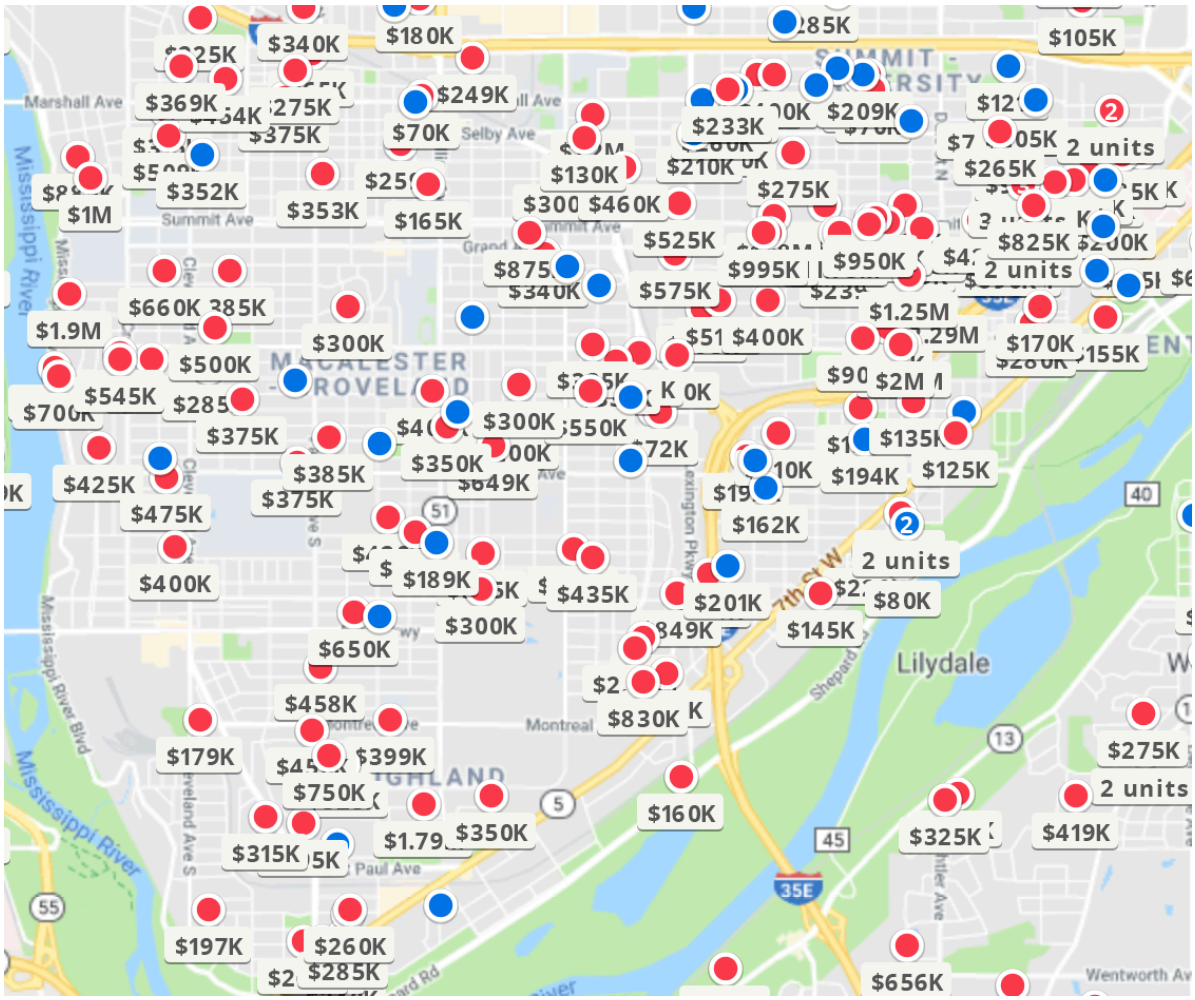


- **Longitudinal Data**: Below are measurements of reasoning ability over time on a group of subjects from the ACTIVE clinical trial. In the plot, each individual is represented by one connected line segment or trajectory (data source). We see that among the subjects that were randomized to the Reasoning Training group, on average, their reasoning measure increased after training and then leveled off, similar to other groups. We also see quite a bit of variation in reasoning abilities between subjects and across time within a subject.

```r
source('Active_Cleaning.R')

activeLong %>%
  ggplot(aes(x = Years, y = Reasoning)) +
  geom_point() +
  geom_line(aes(group = factor(AID))) +
  geom_smooth(method = 'loess',color = 'blue' , se = FALSE) +
  facet_wrap(~ INTGRP)
```



- **Spatial Point Referenced Data**: Below are homes for sale in St. Paul and are spread out in space (longitude, latitude) (data source). We might be able to explain why some houses cost more than others using building characteristics (number of bedrooms, bathrooms, etc.). Even after accounting for those differences, houses close together have similar values due to other intangible factors about the location. We'll need to account for this dependency in models to predict home prices.



1.2 Motivation for Methods

To motivate why this class is important, let's consider applying linear regression (least squares) models that you learned in your introductory statistics course, Stat 155. We will call the method of least squares regression you used **ordinary least squares (or OLS, for short)**.

1.2.1 Simulated Data

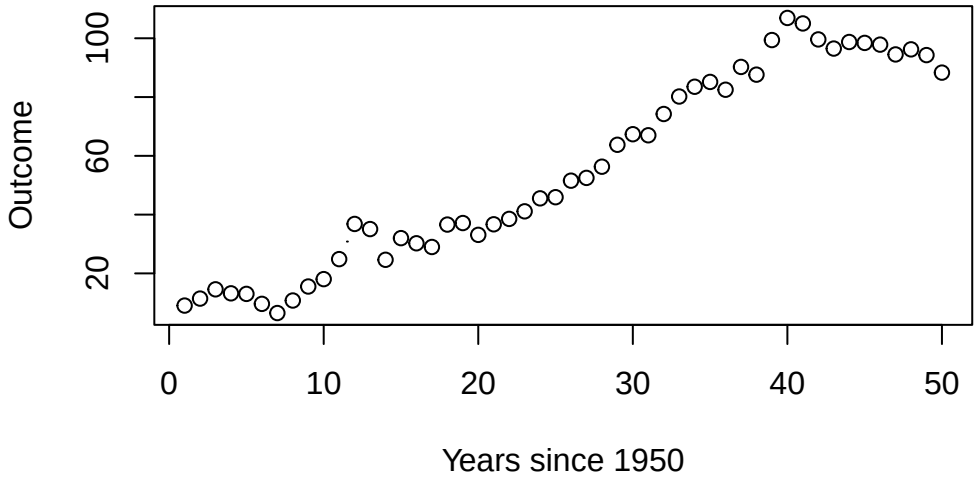
Suppose we have 50 observations over time where our outcome of interest is observed yearly, and it linearly increases over time, but there is random variation from year to year. If the random fluctuation is large and positive one year, it is more likely to be large and positive

next year. In particular, let's assume that our outcome y_t roughly doubles every year, such that

$$y_t = 2 * x_t + \epsilon_t$$

where $x_1 = 1, x_2 = 2, ..., x_{50} = 50$ represents the year since 1950 and the random fluctuation, or **the noise**, ϵ_t are positively correlated from time t to $t+1$ and generated from an autoregressive process – we'll talk about this process later.

Let's see a plot of one possible realization (for one subject or unit) generated from this random process. The overall trend is a line with an intercept at 0 and a slope of 2, but we can see that the observations don't just randomly bounce around that line but rather stick close to where the past value was.



1.2.2 OLS Estimation

If we were to ignore the correlation in the random fluctuation across time, we could fit a simple linear regression model,

$$y_t = \beta_0 + \beta_1 x_t + \epsilon_t, \quad \epsilon_t \overset{iid}{\sim} N(0, \sigma^2)$$

using OLS to estimate the general or overall relationship with year, we'll call that big picture relationship **the trend**.

```
lm(y ~ x) %>% summary()
```

```
Call:
lm(formula = y ~ x)

Residuals:
     Min       1Q   Median       3Q      Max
-20.8574  -6.1287  -0.3483   6.6895  19.7229

Coefficients:
            Estimate Std. Error t value Pr(>|t|)
(Intercept) -0.69654    2.36584  -0.294     0.77
x            2.19734    0.08074  27.213   <2e-16 ***
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

Residual standard error: 8.239 on 48 degrees of freedom
Multiple R-squared:  0.9391,    Adjusted R-squared:  0.9379
F-statistic: 740.6 on 1 and 48 DF,  p-value: < 2.2e-16
```

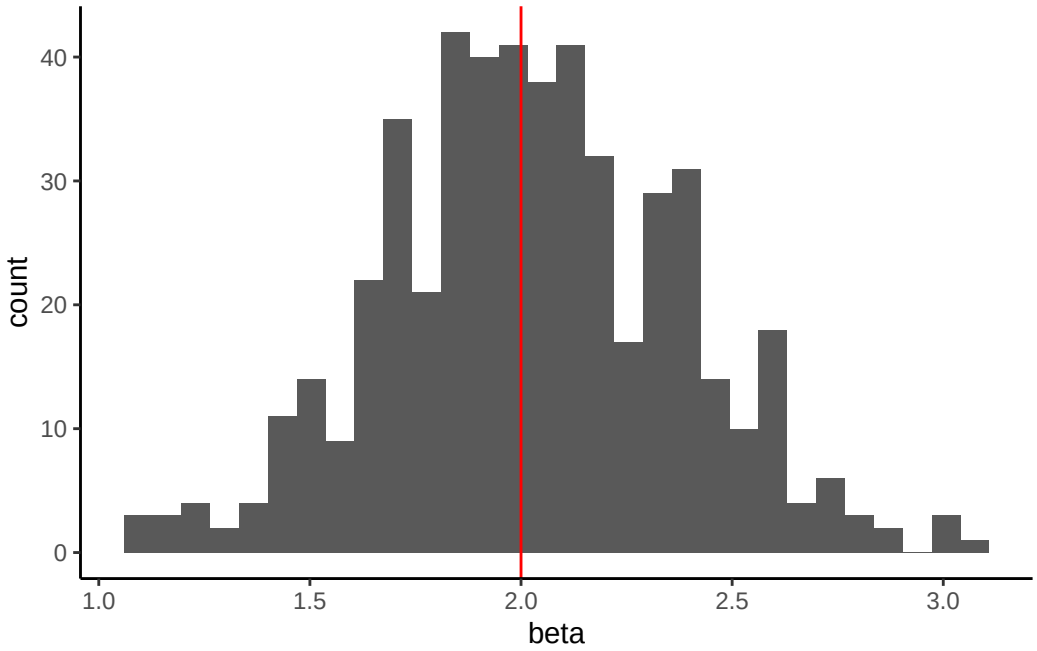
Even with linear regression, we do well estimating the slope ($\hat{\beta}_1 = 2.19$ when the true slope value in how we generated the data is $\beta_1 = 2$). But let's consider the **standard error of that slope**, the estimated variability of that slope estimate. The `lm()` function gives a standard error of 0.08074.

WARNING: This is not a valid estimate of the variability in the slope with correlated data. Let's generate more data to see why!

Let's simulate this same process 500 times (get different random fluctuations) so we can get a sense of how much the estimated slope (which was a "good" estimate) might change.

We can look at all of the estimated slopes by looking at a histogram of the values in `beta`.

```
sim_cor_data_results %>%
  ggplot(aes(x = beta)) +
  geom_histogram() +
  geom_vline(xintercept = 2, color = 'red') #true value used to generate the data
```

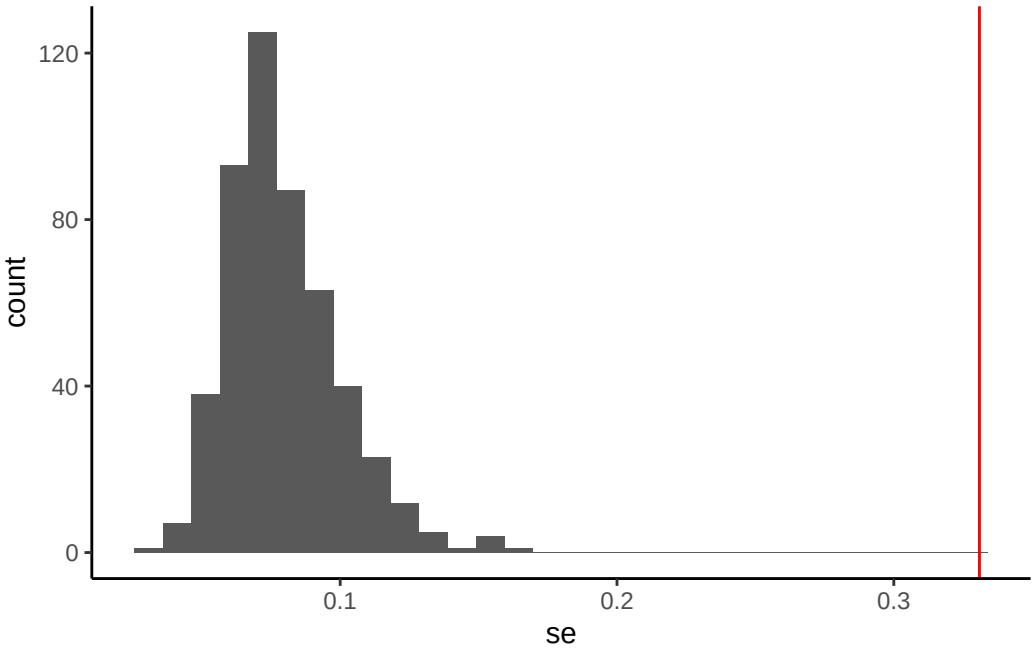


It is centered at the true slope of 2 (great!). This indicates that our estimate is **unbiased** (on average, OLS using the `lm()` function gets the true value of 2). The estimate is unbiased if the expected value of our estimated slope equals the true slope we used to generate the data,

$$E(\hat{\beta}_1) = \beta_1$$

Now, let's look at the standard errors (SE) that OLS using `lm()` gave us.

```
sim_cor_data_results %>%
  ggplot(aes(x = se)) +
  geom_histogram() +
  geom_vline(xintercept = sd(beta), color = 'red') #true variability
```



The true variability of the slopes is estimated by the standard deviation of `beta` from the simulations (around 0.33), but the values provided from `lm()` are all between 0.03 and 0.1.

- The function `lm()`, which assumes the observations are independent, *underestimates* the true variation of the slope estimate.

Why does this happen?

If the data were truly independent, then each random data point would give you unique, important information. If the data are correlated, the observations contain overlapping information (e.g., knowing today's interest in cupcakes tells you something about tomorrow's interest in cupcakes). Thus, your **effective sample size** for positively correlated data is going to be less than the sample size of independent observations. You could get the same amount of information about the phenomenon with fewer data points. You could space them apart further in time so they are almost independent.

The variability of our estimates depends on the information available. Typically, the sample size of independent observations captures the measure of information, but if we have correlated observations, the effective sample size (which is usually smaller than the actual sample size) gives a more accurate measure of the information available.

In this specific simulation of $n = 50$ from an autoregressive order 1 process, the effective sample size is calculated as $50/(1 + 2 * 0.9) = 17.9$, where 0.9 was the correlation used to generate the autoregressive noise. The 50 correlated observations contain about the same amount of information as about 18 independent observations.

If you are interested in the theory of calculating effective sample size, check out these two blog posts, Andy Jones Blog and the Stan handbook

What is the big takeaway?

When we use `lm()`, we are using the ordinary least squares (OLS) method of estimating a regression model. In this method, we assume that the observations are independent of each other.

If our data is actually correlated (not independent, the OLS slope estimates are good (unbiased), but the inference based on the standard errors (including the test statistics, the p-values, and confidence intervals) **is wrong**.

In this case, the true variability of the slope estimates is much higher than the OLS estimates given by `lm()` because the effective sample size is much smaller.

1.3 General Themes

With any of the data examples above or the examples we talk about in class, observations taken closer in time or space are typically going to be more similar than observations taken further apart in space or time.

We may be able to explain why data points closer together are more similar using predictors or explanatory variables, but there may be unmeasured characteristics or inherent dependence that we can't explain with our collected data.

To be more precise, we will assume that the observed outcome at time t (*we will generalize this notation to spatial data*) can be modeled as

$$y_t = \underbrace{f(x_t)}_{\text{trend}} + \underbrace{\epsilon_t}_{\text{noise}}$$

where the trend can be modeled as a deterministic function based on predictors, and there is leftover random noise. This noise might include both serial autocorrelation due to observations being observed close in time, plus random variability or measurement error from the data collection instrument.

Time Series Example

Consider the Google search frequency for "cupcake" data example.

The number of people who are searching for the term "cupcake" should be a function of general interest in cupcakes. This interest could change throughout the year by season, or it may be reflected in the number of cupcake shops in business or the number of mentions of cupcakes

on network television. We could use these measured predictors in modeling the overall trend in search frequency.

What else may explain differences in the interest in cupcakes over time? Even if we can collect and account for these other cultural characteristics, the number of searches for cupcakes will be similar from one day to the next because culture and general interest typically do not change overnight (unless an extreme event happens).

Spatial Example

Consider the home sale prices from Zillow. Price will be determined by a combination of the home characteristics (e.g., number of bedrooms, bathrooms, size, home quality) as well as neighborhood characteristics (e.g., walking distance to amenities, perception of school reputation). These characteristics could be used to model the general trends of sale prices. Even after controlling for these measurable qualities, homes that are next to each other or on the same block will have a similar price.

For each data type, we discuss these two components, the trend and the noise. In particular, we can't assume the noise is independent, so we need to model the **covariance** and **correlation** of the noise, treating it as a series of random variables.

1.3.1 Questions of Interest

1. Dependence of the random variables in the process: 'How do future values depend on past values? How do values depend on neighboring values?'

- Think: Covariance and correlation of random variables

2. Long-Term Averages: 'What is the average value at a particular point in time (or space)?'

- Think: Expected value of random variables (we'll call the overall long-term average the **trend**)

3. Cycles: 'Are there recurring patterns in the average values?'

- Think: Cycles in the expected value of random variables (we'll call the local cycles **seasonality**)

We'll come back to questions 2 and 3 for each sub-field. Let's spend some time thinking about the covariance and correlation in the context of a random process.

With all three of the correlated data types, we explicitly or implicitly model the covariance between observations, so we need to be quite familiar with the probability theory of covariance.

1.4 Outline

Let's go on a journey together, learning the foundational approaches to dealing with correlated (temporal or spatial) data!

We'll start with a review of important probability concepts and review/learn some basic matrix notation that simplifies our probability model notation by neatly organizing our model information. Then we'll spend some time thinking about how we define or encode dependence between observations in models.

The course will be structured so that we spend a few weeks with each type of data structure. We'll learn the characteristics and structure that define that type of correlated data and the standard models and approaches used to deal with the dependence over time or space. By the end of this course, you should have a foundational understanding of how to analyze correlated data and be able to learn more advanced methodologies within each of these data types.

2 Probability Review

With correlated data, we often have repeated measures of some characteristic of a group of subjects or, more generally, of a set of units, collected with some random mechanism.

Before we collect data, we *imagine* those observations to be random variables.

Let's make sure we are all on the same page with the probability concepts that are vital for us to understand the foundational models for correlated data.

2.1 Random Variable

A **random variable** is a variable whose value is subject to variation due to chance.

Note: Mathematically, a random variable $X : \Omega \to E$ is a measurable function from the sample space set of possible random outcomes, Ω, to some set E where usually, $E = \mathbb{R}$.

A **discrete random variable** is a random variable that can take only a countable number of distinct values (so the set E is countable).

- **Example:** A Binomial random variable is a discrete random variable X where X can take values $0, 1, 2, ..., n$ and $P(X = x) = \binom{n}{x} p^x (1-p)^{n-x}$
- **Example:** A Poisson random variable is a discrete random variable X where X can take values $0, 1, 2, ...$ and $P(X = x) = \frac{\lambda^k e^{-\lambda}}{x!}$

The **distribution** of a discrete random variable is described as a list of values and their associated probabilities, usually described with a **probability mass function**, $p(x) = P(X = x)$.

A **continuous random variable** is a random variable that can take an uncountable (think infinite) number of distinct numerical values (the set E is a subset of $\mathbb{R}$).

The **distribution** of a continuous random variable is described with probabilities of intervals of values (rather than distinct values) through the **cumulative distribution function** (cdf), $F(x) = P(X < x)$.

For most of the distributions we will talk about, the distribution can also be described with a probability density function (PDF) such that the cdf is defined as an integral over the pdf, $F(x) = \int_{-\infty}^{x} f(y)dy$ where $f(y) \geq 0$ is the pdf.

- **Example:** Uniform distribution on $[a, b]$ where $f(x) = \frac{1}{b-a}$ if $a \leq x \leq b; 0$ otherwise .
- **Example:** Gaussian distribution on the real line, $\mathbb{R}$, where $f(x) = \frac{1}{\sigma\sqrt{2\pi}} e^{-\frac{1}{2}\left(\frac{x-\mu}{\sigma}\right)^2}$.
- **Example:** Beta distribution on $[0, 1]$ where $f(x) = \frac{\Gamma(\alpha+\beta)}{\Gamma(\alpha)\Gamma(\beta)} x^{\alpha-1}(1-x)^{\beta-1}$.

2.2 Moments

2.2.1 Expectation

The **expected value** of a random variable is the long-run average and calculated as a weighted sum of possible values (weighted by the probability)

$$\mu = E(X) = \sum_{\text{all } x} x \cdot P(X = x)$$

for a discrete random variable and

$$\mu = E(X) = \int_{-\infty}^{\infty} x f(x) dx$$

for a continuous random variable where $f(x)$ is the probability density function.

Properties

For random variables X and Y and constant a, the following are true (you can show them using the definitions of expected value),

$$E(X + a) = E(X) + a$$
$$E(aX) = aE(X)$$
$$E(X + Y) = E(X) + E(Y)$$

Example Proof of $E(X + a) = E(X) + a$

Let's assume X is a discrete random variable. Then, by the definition of expected value,

$$
\begin{aligned}
E(X+a) &= \sum_{\text{all } x}(x+a)P(X=x) && (2.1)\\
&= \sum_{\text{all } x}(xP(X=x)+aP(X=x)) && (2.2)\\
&= \sum_{\text{all } x} xP(X=x) + \sum_{\text{all } x} aP(X=x) && (2.3)\\
&= E(X) + a\sum_{\text{all } x} P(X=x) && (2.4)\\
&= E(X) + a*1 && (2.5)\\
& && (2.6)
\end{aligned}
$$

Let's assume X is a continuous random variable. Then, by the definition of expected value,

$$
\begin{aligned}
E(X+a) &= \int (x+a)f(x)dx && (2.7)\\
&= \int (xf(x)+af(x))dx && (2.8)\\
&= \int xf(x)dx + \int af(x)dx && (2.9)\\
&= E(X) + a\int f(x)dx && (2.10)\\
&= E(X) + a*1 && (2.11)\\
& && (2.12)
\end{aligned}
$$

2.2.2 Covariance and Variance

The **covariance** between two random variables is a measure of linear dependence (i.e., average product of how far you are from the mean or expected value in each variable). The theoretical covariance between two random variables, X and Y, is

$$Cov(X,Y) = E((X-\mu_X)(Y-\mu_Y))$$

where the means are defined as $\mu_X = E(X)$ and $\mu_Y = E(Y)$.

The **variance** is the covariance of a random variable with itself,

$$Var(X) = Cov(X,X) = E((X-\mu_X)(X-\mu_X)) = E((X-\mu_X)^2)$$

Covariance of a Sequence or Series of Random Variables

In this class, we will often work with a sequence or series of random variables that are indexed or ordered. Imagine we have a series of indexed random variables $X_1, ..., X_n$. The subscripts indicate the order. For any two of those random variables, X_l and X_k, the covariances are defined as

$$Cov(X_l, X_k) = E((X_l - \mu_l)(X_k - \mu_k))$$

where $\mu_l = E(X_l)$ and $\mu_k = E(X_k)$.

Note: the order of the variables does not matter, $Cov(X_l, X_k) = Cov(X_k, X_l)$, due to the commutative properties of multiplication.

Notation

We'll use the Greek letter, sigma, σ, to represent covariance. With a series of indexed random variables $X_1, ..., X_n$, we use the subscripts or indices on the σ as a short-hand for the covariance between those two random variables,

$$\sigma_{lk} = Cov(X_l, X_k)$$

where $l, k \in \{1, 2, ..., n\}$.

If the index is the same, $l = k$, then the covariance of the variable with itself is the **variance**, a measure of the spread of a random variable.

Let us denote the variance as

$$\sigma_l^2 = Cov(X_l, X_l) = Var(X_l)$$

It is the average squared distance from the mean,

$$\sigma_l^2 = Var(X_l) = Cov(X_l, X_l) = E((X_l - \mu_l)(X_l - \mu_l)) = E((X_l - \mu_l)^2)$$

The standard deviation (SD) of X_l is the square root of the variance,

$$\sigma_l = SD(X_l) = \sqrt{\sigma_l^2}$$

We often interpret the standard deviation because the units of the SD are the units of the random variable and not in squared units.

Theorem: If X_l and X_k are independent, then $Cov(X_l, X_k) = 0$. (See the technical note below to help you prove this.) The converse is not true.

Properties

For random variables X_l, X_j, X_k and constants a, b, and c, the following are true (you can show them using the definitions),

$$Cov(aX_l, bX_k) = abCov(X_l, X_k)$$
$$Cov(aX_l + c, bX_k) = abCov(X_l, X_k)$$

$$Cov(aX_l + bX_j, cX_k) = acCov(X_l, X_k) + bcCov(X_j, X_k)$$

Thus, we have the following properties of variance,

$$Var(aX_l) = Cov(aX_l, aX_l) = aaCov(X_l, X_l) = a^2Var(X_l)$$

$$\begin{aligned} Var(aX_l + bX_j) &= Cov(aX_l + bX_j, aX_l + bX_j) \\ &= Cov(aX_l, aX_l + bX_j) + Cov(bX_j, aX_l + bX_j) \\ &= Cov(aX_l, aX_l) + Cov(aX_l, bX_j) + Cov(bX_j, aX_l) + Cov(bX_j, bX_j) \\ &= a^2Var(X_l) + b^2Var(X_j) + 2abCov(X_j, X_l) \end{aligned}$$

2.2.3 Correlation

The standardized version of the covariance is the **correlation**. The theoretical correlation between two random variables is calculated by dividing the covariance by the standard deviations of each variable,

$$Cor(X_l, X_k) = \rho_{lk} = \frac{\sigma_{lk}}{\sigma_l \sigma_k} = \frac{Cov(X_l, X_k)}{SD(X_l)SD(X_k)} = \frac{Cov(X_l, X_k)}{\sqrt{Var(X_l)}\sqrt{Var(X_k)}}$$

2.3 Joint Probability Distributions

For a set of discrete random variables $(X_1, ..., X_k)$, the **joint probability mass function** of $X_1, ..., X_k$ is defined as the function such that for every point $(x_1, ..., x_k)$ in the k-dimensional space,

$$p(x_1, ..., x_k) = P(X_1 = x_1, ..., X_k = x_k) \geq 0$$

If $(x_1, ..., x_k)$ is not one of the possible values for the set of random variables, then $p(x_1, ..., x_k) = 0$. Since there can be at most countably many points with $p(x_1, ..., x_k) > 0$ and since these points must account for all the probabilities, we know that

$$\sum_{\text{all } (x_1,...,x_k)} p(x_1, ..., x_k) = 1$$

and for a subset of points called A,

$$P((x_1, ..., x_k) \in A) = \sum_{(x_1,...,x_k) \in A} p(x_1, ..., x_k)$$

The **joint distribution function** of a set of continuous random variables $(X_1, ..., X_k)$ is defined in terms of the cumulative joint distribution function,

$$F(x_1, ..., x_k) = P(X_1 < x_1, ..., X_k < x_k)$$

For a set of continuous random variables $(X_1, ..., X_k)$, the **joint density function** of $X_1, ..., X_k$ is defined as the non-negative function defined for every point $(x_1, ..., x_k)$ in the k-dimensional space such that for every subset A of the space,

$$P((x_1, ..., x_k) \in A) = \int ... \int_A f(x_1, ..., x_k) dx_1 ... dx_k$$

In order to be a joint probability density function, f must be a non-negative function and

$$\int_{-\infty}^{\infty} \cdots \int_{-\infty}^{\infty} f(x_1, ..., x_k) dx_1 ... dx_k = 1$$

2.3.0.1 Technical Note

For two discrete random variables, the covariance can be written as

$$Cov(X_l, X_k) = \sum_{\text{all } x_l} \sum_{\text{all } x_k} (x_l - \mu_l)(x_k - \mu_k) p_{lk}(x_l, x_k)$$

where $p_{lk}(x_l, x_k)$ is the **joint probability distribution** such that $p_{lk}(x_l, x_k) = P(X_l = x_l \text{ and } X_k = x_k)$ and $\sum_{\text{all } x_l} \sum_{\text{all } x_k} p_{lk}(x_l, x_k) = 1$.

For two continuous random variables, the covariance can be written as

$$Cov(X_l, X_k) = \int_{-\infty}^{\infty} \int_{-\infty}^{\infty} (x_l - \mu_l)(x_k - \mu_k) f(x_l, x_k) dx_l dx_k$$

where $f(x_l, x_k)$ is the **joint density function** such that $\int_{-\infty}^{\infty} \int_{-\infty}^{\infty} (x_l, x_k) dx_l dx_k = 1$. We'll see some examples of joint densities soon.

Two random variables are said to be **statistically independent** if and only if

$$f(x_l, x_k) = f_l(x_l) f_k(x_k)$$

for all possible values of x_l and x_k for continuous random variables and

$$P(X_l = x_l, X_k = x_k) = P(X_l = x_l) P(X_k = x_k)$$

for discrete random variables.

A finite set of random variables is said to be **mutually statistically independent** if and only if

$$f(x_1, x_2, ...x_k) = f_1(x_1) \cdots f_k(x_k)$$

for all possible values for continuous random variables and

$$P(X_1 = x_1, X_2 = x_2, ..., X_k = x_k) = P(X_1 = x_1) P(X_2 = x_2) \cdots P(X_k = x_k)$$

for discrete random variables.

2.4 Random Vectors and Matrices

For a set of random variables, we can model the joint distribution of those random variables.

Matrix notation can help us organize distributional information about the set of random variables.

2.4.1 Random Vectors

A **random vector** is a vector (a collection) of m indexed random variables such that

$$\mathbf{X} = \begin{pmatrix} X_1 \\ X_2 \\ \vdots \\ X_m \end{pmatrix}$$

The expected value of a random vector $\mathbf{X}$, written as $E(\mathbf{X})$, is a vector of expected values,

$$\mu = E(\mathbf{X}) = \begin{pmatrix} E(X_1) \\ E(X_2) \\ \vdots \\ E(X_m) \end{pmatrix} = \begin{pmatrix} \mu_1 \\ \mu_2 \\ \vdots \\ \mu_m \end{pmatrix}$$

where $E(X_1) = \mu_1$,...,$E(X_m) = \mu_m$

The covariances of all pairs of values in a random vector $\mathbf{X}$ can be organized in a covariance matrix,

$$\Sigma = Cov(\mathbf{X}) = \begin{pmatrix} \sigma_1^2 & \sigma_{12} & \cdots & \sigma_{1m} \\ \sigma_{21} & \sigma_2^2 & \cdots & \sigma_{2m} \\ \vdots & \vdots & \ddots & \vdots \\ \sigma_{m1} & \sigma_{m2} & \cdots & \sigma_m^2 \end{pmatrix}$$

using the notation introduced above that $\sigma_l^2 = Var(X_l)$ and $\sigma_{lk} = Cov(X_l, X_k)$. We'll talk much more about the covariance matrix.

2.4.2 Random Matrices

Let X_{ij} be an indexed random variable where index $i = 1, ..., n$ refers to different subjects (different units) and the index $j = 1, ..., m$ refers to different observations over time or space. We can organize these variables in random vectors and then into a matrix for convenience.

The **random matrix X** is written as

$$\mathbf{X} = \begin{pmatrix} X_{11} & X_{12} & \cdots & X_{1m} \\ X_{21} & X_{22} & \cdots & X_{2m} \\ \vdots & \vdots & \ddots & \vdots \\ X_{n1} & X_{n2} & \cdots & X_{nm} \end{pmatrix}$$

in which each column represents one observation time and the rows represent the n subjects or individual units.

The expected value of a random matrix $\mathbf{X}$, written as $E(\mathbf{X})$, is a matrix of expected values,

$$E(\mathbf{X}) = \begin{pmatrix} E(X_{11}) & E(X_{12}) & \cdots & E(X_{1m}) \\ E(X_{21}) & E(X_{22}) & \cdots & E(X_{2m}) \\ \vdots & \vdots & \ddots & \vdots \\ E(X_{n1}) & E(X_{n2}) & \cdots & E(X_{nm}) \end{pmatrix}$$

2.4.2.1 Properties

For random matrices or vectors of the same size $\mathbf{X}$ and $\mathbf{Y}$ and constant (not random) compatible matrices or vectors $\mathbf{A}$ and $\mathbf{B}$, the following are true,

$$E(\mathbf{X}+\mathbf{Y}) = E(\mathbf{X}) + E(\mathbf{Y})$$
$$E(\mathbf{AXB}) = \mathbf{A}E(\mathbf{X})\mathbf{B}$$

- Note: We will use **bold** typeface for random **vectors** and **matrices** and normal typeface for random variables.

2.5 Multivariate Normal Distribution

The **normal probability density function** for random variable X with mean μ and standard deviation σ is

$$f(x) = \frac{1}{\sqrt{2\pi\sigma^2}} \exp\left(-\frac{1}{2}\left(\frac{x-\mu}{\sigma}\right)^2\right)$$

Note:

$$\left(\frac{x-\mu}{\sigma}\right)^2 = (x-\mu)(\sigma^2)^{-1}(x-\mu)$$

The **multivariate normal probability density function** for a k-dimensional random vector $\mathbf{X} = (X_1, ..., X_k)$ with mean vector μ and covariance matrix Σ is

$$f(\mathbf{x}) = \frac{1}{(2\pi)^{k/2}|\Sigma|^{1/2}} \exp\left(-\frac{1}{2}(\mathbf{x}-\mu)^T\Sigma^{-1}(\mathbf{x}-\mu)\right)$$

Example - Bivariate Normal

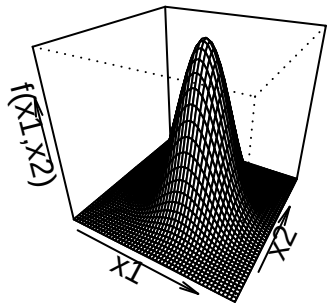
When $k = 2$, our random vector includes X_1 and X_2. Let ρ_{12} be the correlation between the two variables X_1 and X_2.

$$f(x_1, x_2) = \frac{1}{2\pi\sigma_1\sigma_2\sqrt{1-\rho_{12}^2}} e^{-\frac{1}{2(1-\rho_{12}^2)}\left[\left(\frac{x_1-\mu_1}{\sigma_1}\right)^2 + 2\rho_{12}\left(\frac{x_1-\mu_1}{\sigma_1}\right)\left(\frac{x_2-\mu_2}{\sigma_2}\right) + \left(\frac{x_2-\mu_2}{\sigma_2}\right)^2\right]}$$

See below for 3D plots of the bivariate density function under two correlation values ($\rho_{12} = 0$ indicating no correlation and $\rho_{12} = 0.8$ indicating a strong positive correlation) with standardized assumptions for the variance and mean, $\sigma_1^2 = \sigma_2^2 = 1$ and $\mu_1 = \mu_2 = 0$.

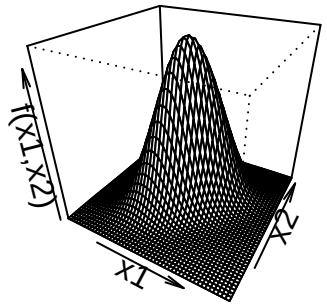
```
rho <- 0
bivn <- MASS::mvrnorm(5000, mu = c(0, 0), Sigma = matrix(c(1, rho, rho, 1), nrow = 2))
bivn.kde <- MASS::kde2d(bivn[,1], bivn[,2], h =2, n = 50) #density function values (we'll dis
persp(bivn.kde, phi = 30, theta = 30,xlab='x1',ylab='x2',zlab='f(x1,x2)',main = expression(rh
```

$\rho_{12} = 0$




```
rho <- 0.8
bivn <- MASS::mvrnorm(5000, mu = c(0, 0), Sigma = matrix(c(1, rho, rho, 1), nrow = 2))
bivn.kde <- MASS::kde2d(bivn[,1], bivn[,2], h =2, n = 50)
persp(bivn.kde, phi = 30, theta = 30,xlab='x1',ylab='x2',zlab='f(x1,x2)',main = expression(rh
```

$\rho_{12} = 0.8$



2.5.1 Contours of Density

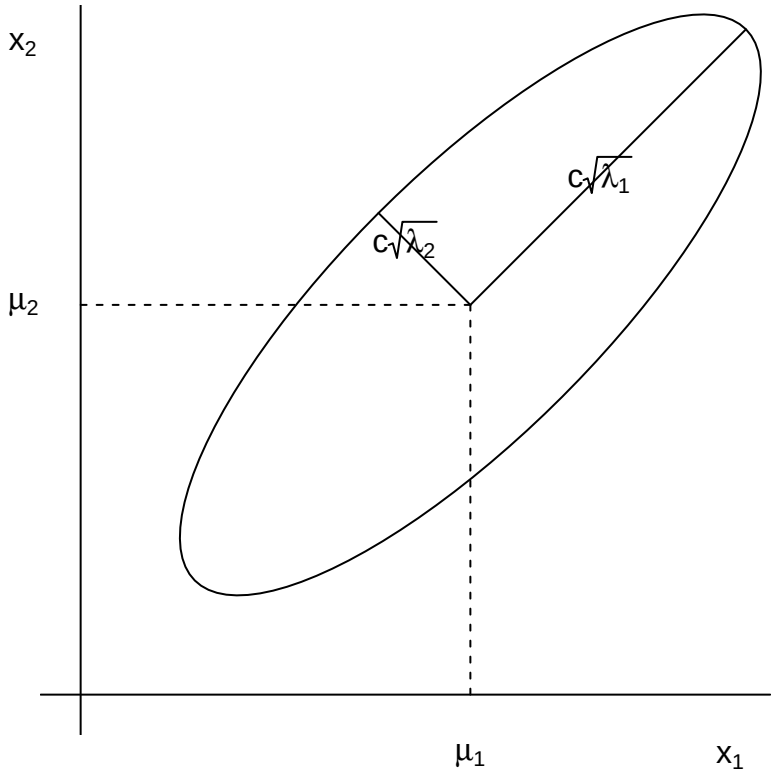
The multivariate normal density function is constant on surfaces where the square of the Mahalanobis-type distance $(\mathbf{x}-\mu)^T\Sigma^{-1}(\mathbf{x}-\mu)$ is constant and these are called **contours**.

The contours are ellipsoids defined by

$$(\mathbf{x}-\mu)^T\Sigma^{-1}(\mathbf{x}-\mu) = c^2$$

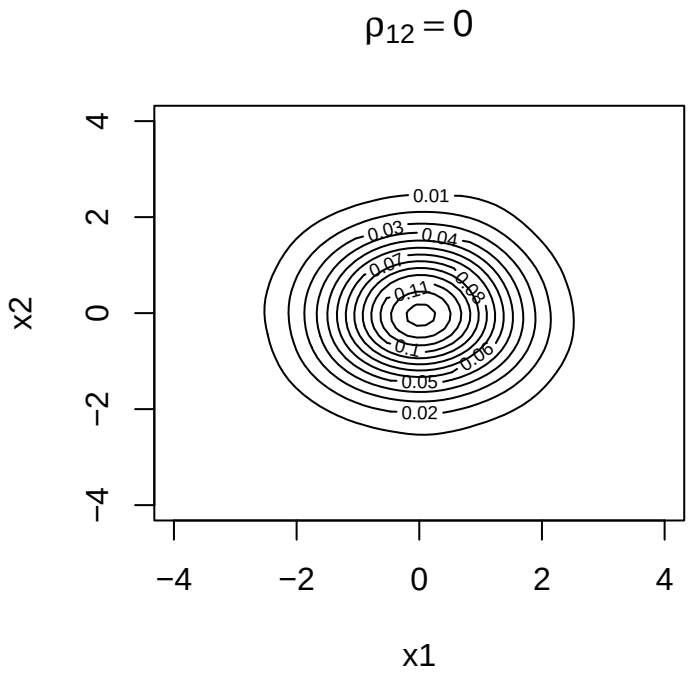
such that the ellipsoids are centered at μ and have axes $\pm c\sqrt{\lambda_i}\mathbf{e}_i$ where $\Sigma\mathbf{e}_i = \lambda_i\mathbf{e}_i$ for $i = 1, ..., k$. Thus, $c\sqrt{\lambda_i}$ are the radii of the axes (λ_i are eigenvalues of Σ) and the eigenvectors $\mathbf{e}_i$ are the directions of the axes.

Also, in this context, c is the Mahalanobis distance between points in this distribution. We may refer to c as the radius, but note that this is the radius of the ellipse only when it is a circle and $\lambda_i = 1$.

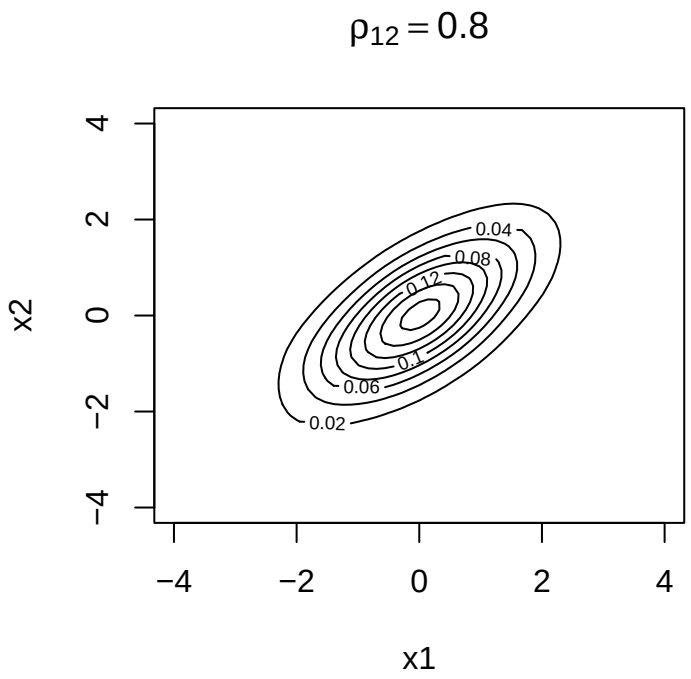


Below are contours of bivariate normal density functions with the values representing the density function (height).

```
rho = 0
bivn <- MASS::mvrnorm(5000, mu = c(0, 0), Sigma = matrix(c(1, rho, rho, 1), 2))
bivn.kde <- MASS::kde2d(bivn[,1], bivn[,2], h =2, n = 50)
contour(bivn.kde,xlab='x1',ylab='x2',main = expression(rho[12] == 0),xlim=c(-4,4),ylim=c(-4,
```




```
rho = 0.8
bivn <- MASS::mvrnorm(5000, mu = c(0, 0), Sigma = matrix(c(1, rho, rho, 1), 2))
bivn.kde <- MASS::kde2d(bivn[,1], bivn[,2], h =2, n = 50)
contour(bivn.kde,xlab='x1',ylab='x2',main = expression(rho[12] == 0.8),xlim=c(-4,4),ylim=c(-4
```



These should look familiar if you looked at a topographic map for hiking in mountainous areas.

2.5.2 Properties of Multivariate Normal

Linear Combinations

If $\mathbf{X}$ is distributed as a k-dimensional normal distribution with mean μ and covariance Σ, then any linear combination of variables (where $\mathbf{a}$ is a vector of constants), $\mathbf{a}^T\mathbf{X} = a_1X_1 + a_2X_2 + \cdots + a_kX_k$ is distributed as normal with mean $\mathbf{a}^T\mu$ and covariance $\mathbf{a}^T\Sigma\mathbf{a}$

Chi-Squared Distribution The **chi-squared distribution** with k degrees of freedom is the distribution of the sum of the squares of k independent standard normal random variables. *You typically prove this is true in Probability.*

Theorem: If a random vector $\mathbf{X} \in \mathbb{R}^k$ follows a multivariate normal distribution with mean μ and covariance Σ, then $(\mathbf{X}-\mu)^T\Sigma^{-1}(\mathbf{X}-\mu)$ follows a Chi-squared distribution with k degrees of freedom.

Sketch of proof: Using the Cholesky decomposition, we get $\Sigma = \mathbf{L}\mathbf{L}^T$. Then define a random vector $\mathbf{Y}$ such that

$$\mathbf{Y} = \mathbf{L}^{-1}(\mathbf{X} - \mu)$$

Show that $\mathbf{Y}$ is a vector of independent standard normal random variables. You can use the properties of linear combinations above.

Then show that we have a sum of independent standard normals,

$$(\mathbf{X} - \mu)^T\Sigma^{-1}(\mathbf{X} - \mu) = \mathbf{Y}^T\mathbf{Y} = \sum Y_i^2$$

Extensions of Intervals to Multiple Dimensions - Ellipse Percentiles Now that we know about the contours of a multivariate normal distribution and we know the probability distribution for the contours, we can extend the ideas of intervals (such as confidence intervals or intervals of uncertainty) to a multivariate situation.

The contours containing α of the probabilities under the multivariate normal distribution are the values of $\mathbf{x}$ that satisfy

$$(\mathbf{x} - \mu)^T\Sigma^{-1}(\mathbf{x} - \mu) = \chi^2_k(\alpha)$$

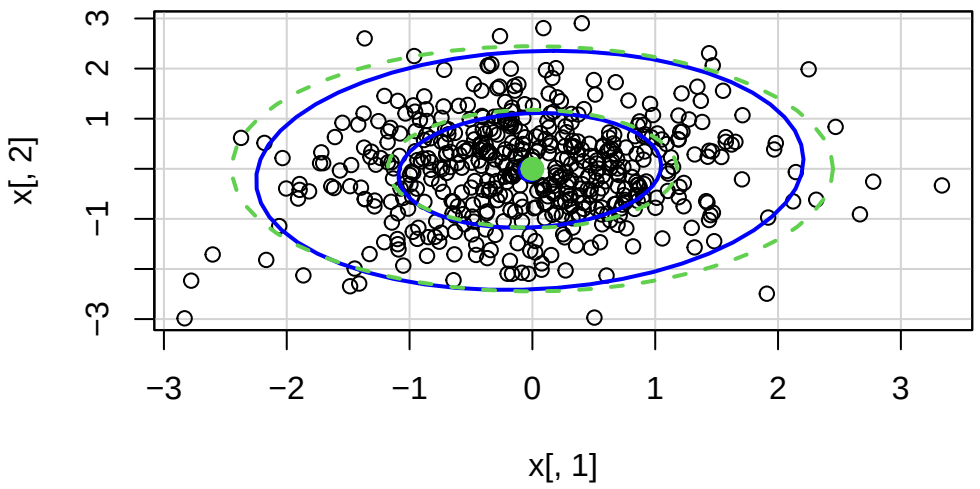
where $\chi^2_k(\alpha)$ is the $\alpha \cdot 100$ percentile of a chi-squared distribution with k degrees of freedom. Therefore,

$$P((\mathbf{x} - \mu)^T\Sigma^{-1}(\mathbf{x} - \mu) \leq \chi^2_k(\alpha)) = \alpha$$

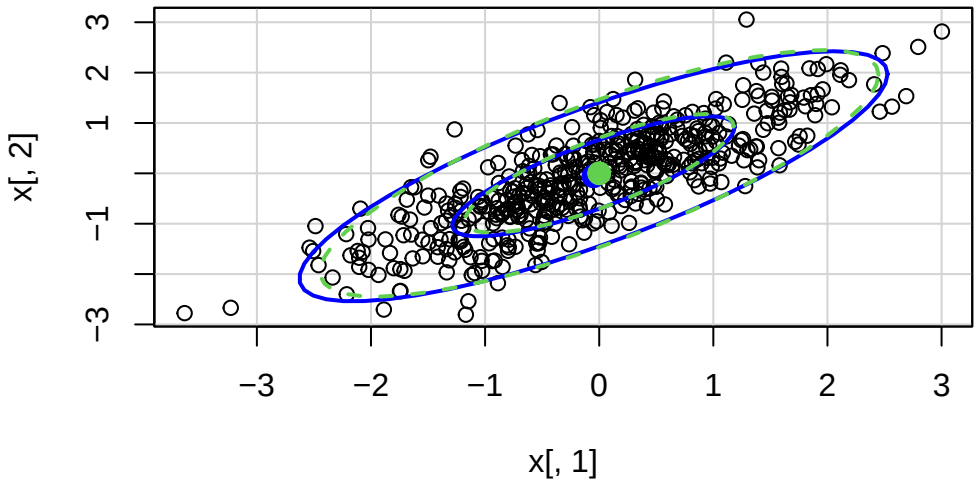
That means that we can calculate ellipses that correspond to a probability of say $\alpha = 0.95$, for example (probability = volume under the surface). In a univariate situation, these ellipses correspond to intervals such that the area under the curve equals the probability of 0.95.

For a bivariate normal with mean vector (0,0), variances equal to 1, and correlation equal to 0 (and then 0.8), there is R code below to calculate the 0.5 and 0.95 ellipses based on the theoretical mean and covariance (green lines) or estimated based on sample data (red lines).

```
x <- MASS::mvrnorm(500, mu = c(0,0), Sigma = matrix(c(1,0,0,1),2))
tmp <- car::dataEllipse(x[,1],x[,2]) #based on data
c2 <- qchisq(.5,df = 2)
car::ellipse(c(0,0), matrix(c(1,0,0,1),2),sqrt(c2), col=3, lty=2) #theoretical
c2 <- qchisq(.95,df = 2)
car::ellipse(c(0,0), matrix(c(1,0,0,1),2),sqrt(c2), col=3, lty=2)
```




```
x <- MASS::mvrnorm(500, mu = c(0,0), Sigma = matrix(c(1,.8,.8,1),2))
tmp <- car::dataEllipse(x[,1],x[,2]) #based on data
c2 <- qchisq(.5,df = 2)
car::ellipse(c(0,0), matrix(c(1,.8,.8,1),2),sqrt(c2), col=3, lty=2) #how we generated data
c2 <- qchisq(.95,df = 2)
car::ellipse(c(0,0), matrix(c(1,.8,.8,1),2),sqrt(c2), col=3, lty=2)
```


x[, 2]
3
2
1
−1
−3
−3
−2
−1
0
1
2
3
x[, 1]

3 Modeling Covariance

For this course, all of the methods we discuss have a way to model the inherent dependence in the data, directly or indirectly. Ultimately, they are modeling the covariance of pairs of random variables. Before we discuss models for a specific data type, let's think about modeling covariance more generally.

In previous sections, we discussed a random vector, a set of random variables. These random variables could represent different quantities or characteristics. We hinted that these variables could be ordered or indexed in some way.

We will formalize that here. We focus now on a random process, a series of random variables representing the same characteristic ordered or indexed by time or space.

3.1 Random Process

A **random process** is a series of random variables indexed by time or space, $\{Y_t\}_{t \in T}$, defined over a common probability space. We can organize a finite set of these variables in a random vector,

$$\mathbf{Y} = \begin{pmatrix} Y_1 \\ Y_2 \\ \vdots \\ Y_m \end{pmatrix}$$

- For time series data, the set of indices $T = \{0, 1, 2, ..., n\}$ indicates the time ordering of the random variables.
- For longitudinal data, we index our data for one unit/subject with a random process where $T = \{1, 2, ..., m\}$ indicates the ordering of the repeated random variables over time.
- For spatial data, T refers to a set of points in space measured by (longitude, latitude).

Key Point: A random process can capture how a characteristic evolves over time (or space or both), and Y represents the random value of that characteristic.

3.2 Autocovariance

Remember that the covariance between two random variables, X and Y, is defined as

$$Cov(X, Y) = E((X - E(X))(Y - E(Y))) = E(XY) - E(X)E(Y)$$

The covariance values range over the entire real line $(-\infty, \infty)$.

3.2.1 Autocovariance Function

For a random process $\{Y_t\}_{t \in T}$, we can summarize the covariance between two random variables in the process with different indices, Y_t and Y_s, as a function of the times/spaces, t and s. The **autocovariance function** of s and t is the covariance of the random variables at those times/spaces,

$$\Sigma_Y(t, s) = Cov(Y_t, Y_s) = E((Y_t - \mu_t)(Y_s - \mu_s)) = E(Y_s Y_t) - \mu_s \mu_t$$

where $\mu_s = E(Y_s)$ and $\mu_t = E(Y_t)$.

Note:

- For times series, t and s will be integer indices of time.
- For longitudinal data, t and s will refer to observations times.
- For spatial data, t and s will be points in space, (longitude, latitude).

Throughout this course, we'll use the capital Greek letter Sigma, Σ, to denote covariance. We'll use it in a few different ways, but it will always refer to covariance.

For example, we can refer to the covariance between the first and second random variable with $\Sigma_Y(1, 2)$ and the covariance between the second and fourth with $\Sigma_Y(1, 4)$. We'll come back to thinking about simplifying this autocovariance function.

3.2.2 Autocorrelation Function

In general, correlation is easier to interpret than covariance since correlation values are restricted to be between -1 and 1 (inclusive).

Remember that the correlation of two random variables X and Y is defined as

$$Cor(X, Y) = \frac{Cov(X, Y)}{SD(X)SD(Y)}$$

We can define an autocorrelation function as

$$\rho_Y(s,t) = \frac{\Sigma_Y(s,t)}{\sqrt{\Sigma_Y(s,s)\Sigma_Y(t,t)}}$$

We use ρ to notate theoretical correlation (in contrast to r, which refers to the sample correlation coefficient).

For example, we can refer to the correlation between the first and second random variable with $\rho_Y(1,2)$ and the correlation between the second and fourth with $\rho_Y(1,4)$. Again, we'll come back to thinking about simplifying this autocorrelation function.

3.2.3 Covariance Matrix

If we have m random variables $\mathbf{Y} = (Y_1, ..., Y_m)$ of a random process, we might be interested in knowing the covariance between every pair of indexed variables. We could plug the indices into the autocovariance function. The covariance between the ith observation and the jth observation in the process would be

$$\sigma_{ij} = \Sigma_Y(i,j)$$

Among the m variables, there are $m(m-1)/2$ unique pair-wise covariances and m variances and they could be organized into a $m \times m$ symmetric **covariance matrix**,

$$\Sigma_Y = Cov(\mathbf{Y}) = \begin{pmatrix} \sigma_1^2 & \sigma_{12} & \cdots & \sigma_{1m} \\ \sigma_{21} & \sigma_2^2 & \cdots & \sigma_{2m} \\ \vdots & \vdots & \ddots & \vdots \\ \sigma_{m1} & \sigma_{m2} & \cdots & \sigma_m^2 \end{pmatrix}$$

where $\sigma_i^2 = \sigma_{ii}$. Note that $\sigma_{ij} = \sigma_{ji}$.

The covariance matrix can also be written using the definitions of covariance and the properties of matrix algebra as

$$\begin{aligned} \Sigma_Y = Cov(\mathbf{Y}) &= E((\mathbf{Y} - E(\mathbf{Y}))(\mathbf{Y} - E(\mathbf{Y}))^T) \\ &= E(\mathbf{Y}\mathbf{Y}^T) - \mu\mu^T \end{aligned}$$

where $E(\mathbf{Y}) = \mu$.

3.2.3.1 Properties

For a $p \times m$ constant matrix $\mathbf{A}$ and m dimensional random vector $\mathbf{Y}$,

$$Cov(\mathbf{AY}) = \mathbf{A}Cov(\mathbf{Y})\mathbf{A}^T$$

This is an important property because it gives us the covariance of a linear combination of our random process values: $\mathbf{AY}$. This is also very useful when working with regression estimates (we'll come back to this).

3.2.3.2 Positive Semidefiniteness

For the autocovariance function to be valid, it must be **positive semidefinite**. We can check this by evaluating it at any set of m indices and checking to see if the resulting covariance matrix is positive semidefinite. A symmetric matrix, Σ, is positive semidefinite if and only if

$$\mathbf{x}^T\Sigma\mathbf{x} \geq 0 \text{ for all } \mathbf{x} \in \mathbb{R}^m$$

or equivalently,

$$\text{The eigenvalues of } \Sigma \text{ are all } \geq 0$$

In practice, this is important to be aware of when modeling covariance, as the covariance matrix must be positive semidefinite.

To ensure this matrix is positive semidefinite, we could use the **Cholesky Decomposition** in the modeling process because the decomposition is only valid for positive definite ($>$ instead of $\geq$ in the definitions above) matrices,

$$\Sigma = \mathbf{L}\mathbf{L}^T$$

where $\mathbf{L}$ is a lower triangular matrix (upper diagonal is all 0), we'll return to this.

3.2.4 Correlation Matrix

If we have m random variables $Y = (Y_1, ..., Y_m)$, we might be interested in knowing the autocorrelation between every pair of observations. The correlation between the ith observation and the jth observation would be

$$\rho_{ij} = \rho_Y(i, j)$$

These $m(m-1)/2$ correlations could be organized into a $m \times m$ symmetric **correlation matrix**,

$$\mathbf{R}_Y = Cor(\mathbf{Y}) = \begin{pmatrix} 1 & \rho_{12} & \cdots & \rho_{1m} \\ \rho_{21} & 1 & \cdots & \rho_{2m} \\ \vdots & \vdots & \ddots & \vdots \\ \rho_{m1} & \rho_{m2} & \cdots & 1 \end{pmatrix}$$

Note that $\rho_{ij} = \rho_{ji}$.

If you have the covariance matrix Σ, you can extract the correlation matrix,

$$\mathbf{R}_Y = \mathbf{D}^{-1}\Sigma_Y\mathbf{D}^{-1}$$

where $\mathbf{D} = \sqrt{diag(\Sigma_Y)}$, a diagonal matrix with standard deviations along the diagonal because

$$\rho_{ij} = \frac{\sigma_{ij}}{\sigma_i\sigma_j}$$

3.3 Models, Simplifications, & Constraints

We've been discussing covariance and correlation in terms of probability theory so far. What about if we have data?!

We need replicates to estimate any parameter (mean, covariance, correlation, etc.). For the covariance of pairs of variables, that means we need either:

- multiple realizations of a random process (which could be multiple subjects), OR
- simplifying assumptions about the structure of the autocovariance function.

3.3.1 Common Constraints

When we model covariance, we often make simplifying assumptions and use model constraints to estimate the covariance from the data. Below are some of the common constraints or assumptions we use. Like any assumption, we should check that it is a valid assumption to make with a dataset before reporting on the conclusions of the model.

3.3.1.1 Weak Stationarity

If the autocovariance function of a random process only depends on the difference in time/space, $s - t$ (e.g., covariance depends only on the difference in observation times and not time itself), then we say that the autocovariance is **weakly stationary**, such that

$$\Sigma_Y(t, s) = \Sigma_Y(s - t)$$

For example, this means that the covariance between the 1st and 4th observation is the same as the 2nd and 5th, 3rd and 6th, etc. This simplifies our function as we only need to know the difference in time/space rather than the exact time/space.

Additionally, a random process that is weakly stationary has a constant mean, $\mu = E(Y_t) = E(Y_s)$, and constant variance, $\sigma^2 = Var(Y_t) = Var(Y_s)$. When we use the term "stationary" from now on, we refer to a "weak stationary" process.

A **random process is weakly stationary** if it has a

1. Constant Mean $\mu = E(Y_t) = E(Y_s)$
2. Constant Variance $\sigma^2 = Var(Y_t) = Var(Y_s)$
3. Autocovariance (and autocorrelation) Function is *only* a function of the vector difference in time/space

In a covariance matrix, the constant variance of stationarity, $\sigma_i^2 = \sigma^2$, means that we can write the covariance matrix as the variance times correlation matrix,

$$\Sigma_Y = \sigma^2 \mathbf{R}_Y$$

where $\mathbf{R}_Y$ is the correlation matrix.

Additionally, with the stationary assumption, the correlation matrix can be written in terms of the vector difference in time/space, similar to the covariance matrix.

If we are considering spatial data where space is defined by longitude and latitude, a weakly stationary process means that the covariance depends on the difference between two points in space, which is defined by both the distance (as the crow flies Euclidean distance) and the direction (e.g., a point on a compass such as NE or West or defined as the angle degrees from North). In linear algebra terminology, the difference is the **vector difference**.

3.3.1.2 Isotropy

A stronger assumption than weak stationarity is that the autocovariance function *depends only on the distance in time/space,* $||s-t||$.

If this is true, then we typically say that the autocovariance function and random process are **isotropic**, such that the dependence doesn't change as a function of the angle of the difference in space (NE v. West) but just the **vector distance** or **vector norm**. That is,

$$\Sigma_Y(t,s) = \Sigma_Y(||s-t||)$$

Notes:

- This distinction only applies to spatial data in which our index is a vector of length two or more.
- Many spatial models we will use assume an isotropic covariance function.

3.3.1.3 Intrinsic Stationarity

If assuming a process is weak stationary is too strong a constraint, a slightly weaker constraint is that a random process is **intrinsic stationary** if

$$Var(Y_{s+h} - Y_s) \text{ depends only on h}$$

meaning that the variance of the difference of random variables only depends on the vector difference, h, in time or space.

This is similar to the concept of weak stationarity, but weak stationarity implies intrinsic stationarity, but not vice versa.

3.3.2 Common Model Structures

Here are some common correlation functions that are all weakly stationary and isotropic:

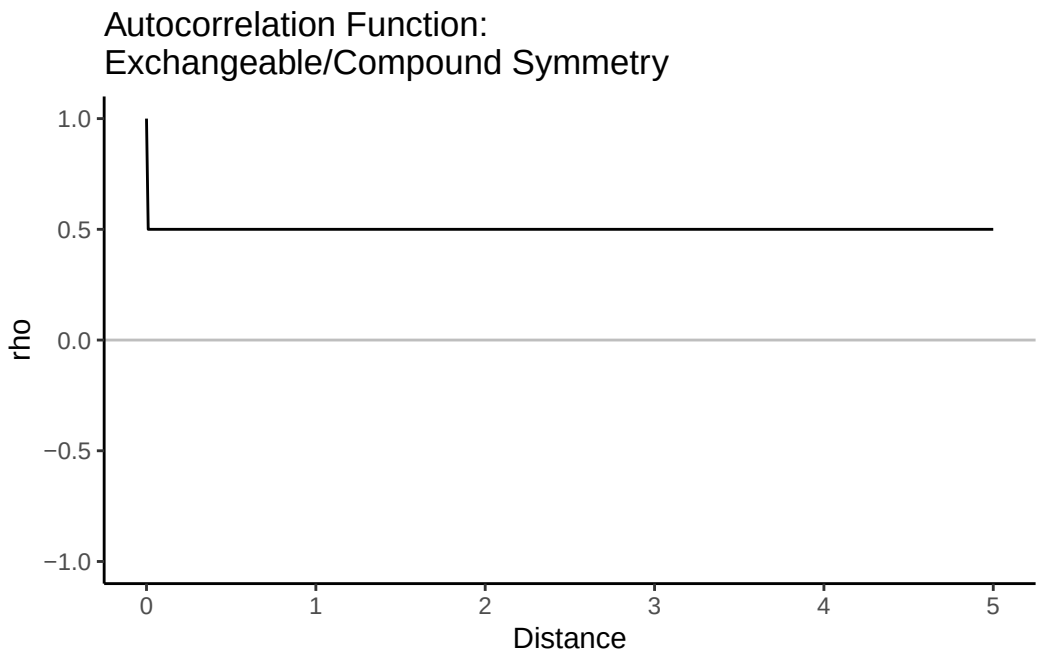
3.3.2.1 Compound Symmetry / Exchangeable correlation

The correlation between two random variables is constant, no matter the difference in time/space.

The autocorrelation function is defined as

$$\rho_Y(s-t) = \begin{cases} \rho \text{ if } s-t \neq 0 \\ 1 \text{ if } s-t = 0 \end{cases}$$

```
tibble(
  h = seq(0,5,length = 500), #distance
  rho = c(1,rep(0.5, 499))
) %>%
  ggplot(aes(x = h, y = rho)) +
  geom_line() +
  geom_hline(yintercept = 0,color='grey') +
  ylim(-1,1) +
  labs(title = 'Autocorrelation Function:\nExchangeable/Compound Symmetry', x = 'Distance') +
  theme_classic()
```



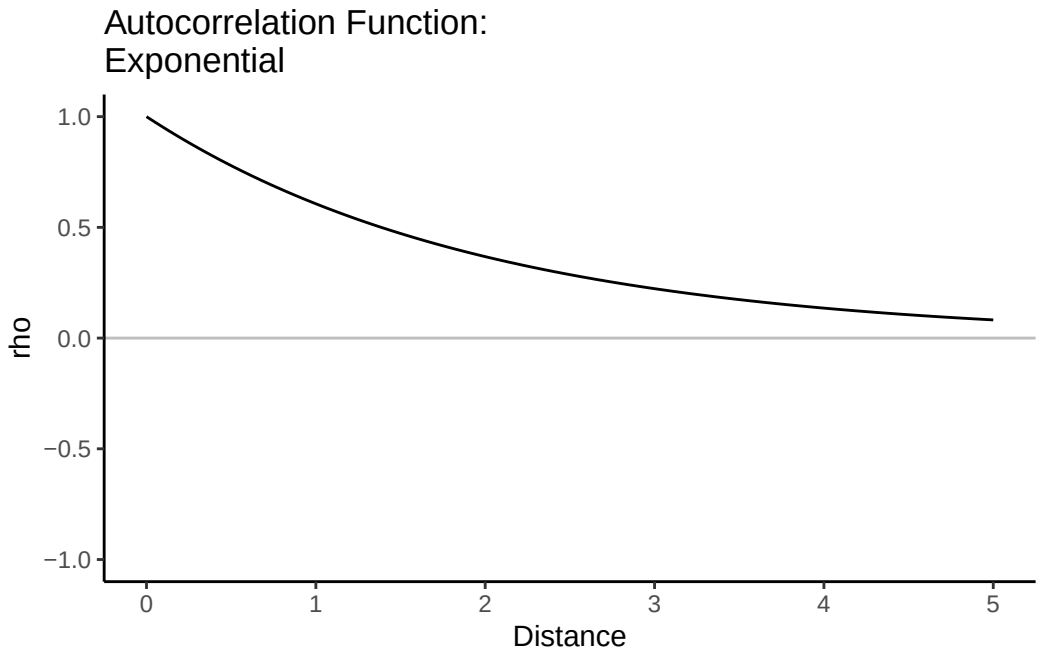
3.3.2.2 Exponential Correlation

The correlation between two random variables decays to 0 exponentially as the difference in time/space increases.

$$\rho_Y(s-t) = e^{-||s-t||/\phi}$$

where $\phi > 0$

```
phi = 2
tibble(
  h = seq(0,5,length = 500) #distance
) %>%
  mutate(rho = exp(-h/phi)) %>%
  ggplot(aes(x = h, y = rho)) +
  geom_line() +
  geom_hline(yintercept = 0,color='grey') +
  ylim(-1,1) +
  labs(title = 'Autocorrelation Function:\nExponential', x = 'Distance') +
  theme_classic()
```



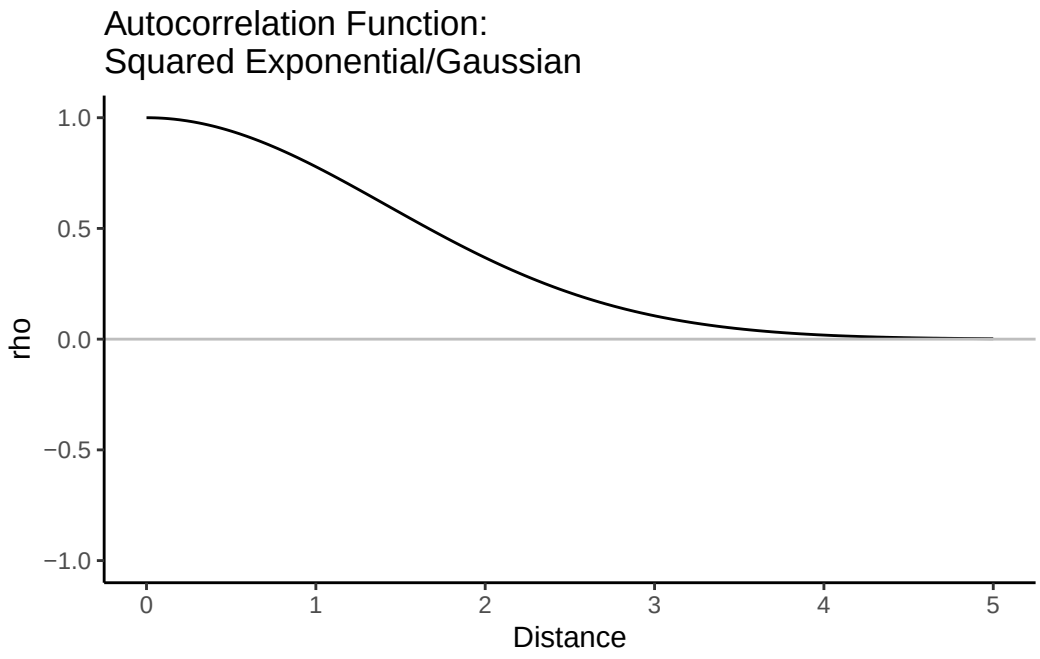
3.3.2.3 Squared Exponential / Gaussian Correlation

The correlation between two random variables decays (slightly different decay exponentially) to 0 as the difference in time/space increases. It decays more slowly than exponentially for differences $< \phi$, but for differences $> \phi$, it decays faster.

$$\rho_Y(s-t) = e^{-(||s-t||/\phi)^2}$$

where $\phi > 0$

```
phi = 2
tibble(
  h = seq(0,5,length = 500) #distance
) %>%
  mutate(rho = exp(-(h/phi)^2)) %>%
  ggplot(aes(x = h, y = rho)) +
  geom_line() +
  geom_hline(yintercept = 0,color='grey') +
  ylim(-1,1) +
  labs(title = 'Autocorrelation Function:\nSquared Exponential/Gaussian', x = 'Distance') +
  theme_classic()
```



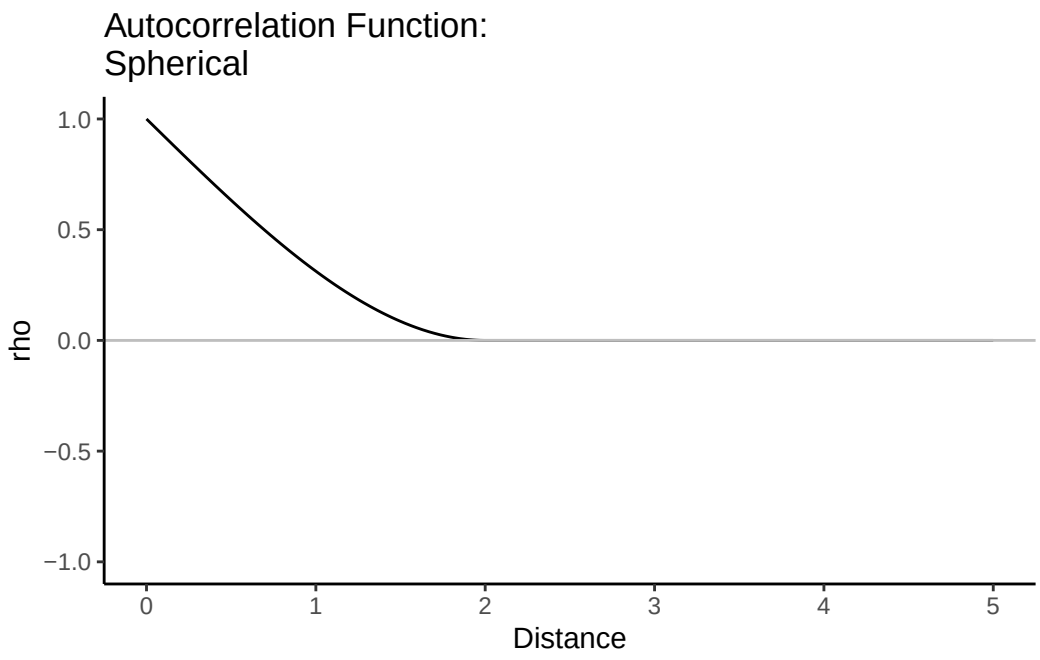
3.3.2.4 Spherical Correlation

The correlation between two random variables decays (different decay than exponentially) to 0 as the difference in time/space increases.

$$\rho_Y(s-t) = \begin{cases} 1 - 1.5(||s-t||/\phi) + 0.5(||s-t||/\phi)^3 \text{ if } ||s-t|| < \phi \\ 0 \text{ otherwise} \end{cases}$$

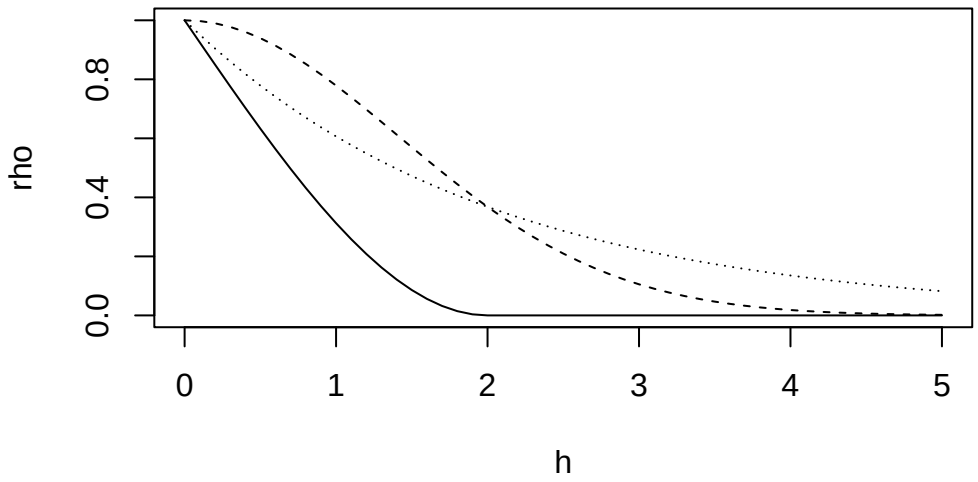
where $\phi > 0$

```
phi = 2
tibble(
  h = seq(0,5,length = 500) #distance
) %>%
  mutate(rho = if_else(h<phi,1 - 1.5*(h/phi) + 0.5*(h/phi)^3,0)) %>%
  ggplot(aes(x = h, y = rho)) +
  geom_line() +
  geom_hline(yintercept = 0,color='grey') +
  ylim(-1,1) +
  labs(title = 'Autocorrelation Function:\nSpherical', x = 'Distance') +
  theme_classic()
```



Let's look at the last three on the same graph. With the same value of ϕ, the correlation function can look quite different.

```r
h = seq(0,5,by=.1) #distance
phi = 2
rho = rep(0, length(h))
rho[h<phi] = 1 - 1.5*(h[h<phi]/phi) + 0.5*(h[h<phi]/phi)^3
rho2 = exp(-(h/phi)^2)
rho3 = exp(-h/phi)
plot(h, rho, type ='l', ylim=c(0,1))
lines(h, rho2, lty=2)
lines(h, rho3, lty=3)
```



3.4 Estimating with Data

In this chapter, we have discussed the theory around autocovariance in function and matrix format and models, simplifications, and constraints that can be imposed and assumed.

In this last section, we'll introduce the idea of estimating the numerical values of covariance and correlation based on data after assuming models, simplifications, and constraints.

The three estimators mentioned below can be used outside a larger statistical model, so we'll start here.

3.4.1 Sample Covariance Matrix

To estimate the covariance matrix, Σ, with observed data sampled from the larger population as n realizations of these m random variables (imagine n individual people with m observations over time), we can calculate the **sample covariance matrix**,

$$\mathbf{S}_Y = \frac{1}{n-1}\sum_{i=1}^{n}(\mathbf{y}_i - \bar{\mathbf{y}})(\mathbf{y}_i - \bar{\mathbf{y}})^T$$

where $\mathbf{y}_i = (y_{i1}, y_{i2}, ..., y_{im})$ and $\bar{\mathbf{y}} = n^{-1}\sum_{i=1}^{n}\mathbf{y}_i$.

Therefore, the sample variances (on the diagonal of $\mathbf{S}_Y$) are the familiar estimates we saw in our introductory course,

$$s_j^2 = \frac{1}{n-1}\sum_{i=1}^{n}(y_{ij} - \bar{y}_j)^2$$

Note: This estimation is only possible for longitudinal data if we observed m repeated measurements for each case at the same time. We typically don't have more than one realization of the process for time series and spatial data. For longitudinal data, if we have unbalanced data collected at irregular times, we'll need to use one of the common model structures to estimate the covariance. In the Longitudinal Data chapter, we'll return to this.

3.4.2 Sample Autocovariance Function

If we assume that observed data come from a stationary process, we can estimate the autocovariance function with the **sample autocovariance function** (ACVF),

$$c_k = \frac{1}{n}\sum_{t=1}^{n-|k|}(y_t - \bar{y})(y_{t+|k|} - \bar{y})$$

where $\bar{y}$ is the sample mean, averaged across time/space, because we are assuming it is constant.

For any sequence of observations of the random process, $y_1, ..., y_n$, the estimated ACVF has the following properties:

1. $c_k = c_{-k}$
2. $c_0 \geq 0$ and $|c_k| \leq c_0$

In the Time Series Data chapter, we'll return to this.

3.4.3 Sample Semivariogram

If we assume that observed data come from a stationary process (or at least an intrinsic stationary process), we can estimate the **semivariogram**,

$$\gamma(h) = \frac{1}{2} Var(Y_{s+h} - Y_s)$$

To estimate the semivariogram, let $H_1, ..., H_k$ be a partition of the space of possible distances or lags h, with h_u being a representative spatial lag/distance in H_u. Then use your stationary process y_t to estimate the **sample/empirical semivariogram**.

$$\hat{\gamma}(h_u) = \frac{1}{2 \cdot |\{i - j \in H_u\}|} \sum_{\{i-j \in H_u\}} (y_i - y_j)^2$$

4 Model Components

As we introduced at the beginning of the course, most models we use for data are written in terms of model components of a deterministic or systematic component plus noise or error,

$$Y_t = \underbrace{f(x_t)}_{\text{deterministic}} + \underbrace{\epsilon_t}_{\text{random noise}}$$

The **deterministic component** would generally give the average outcome as a function of x_t, which could represent time/space and/or explanatory variables measured across time/space.

- In Stat 155 (Introduction to Statistical Modeling), we model the deterministic component with linear combinations of explanatory variables, $f(\mathbf{x}) = \beta_0 + \beta_1 x_1 + \beta_2 x_2 + \cdots + \beta_p x_p$, and we assume the noise was independent.
- In Stat 253 (Statistical Machine Learning), we learn parametric and nonparametric tools to model the deterministic component (polynomials, splines, local regression, KNN, trees/forests, etc.) for prediction.

We'll build on your existing knowledge and learn a few more tools for modeling the deterministic components that arise with correlated data. We'll use the notation of time series to discuss model components, but these ideas apply to longitudinal and spatial data.

Typically, we write a time series model with three components (not just two) and model them separately:

$$Y_t = \underbrace{f(x_t)}_{\text{deterministic trend}} + \underbrace{s(x_t)}_{\text{deterministic seasonality}} + \underbrace{\epsilon_t}_{\text{random noise}}$$

Let's define these model components and the types of questions that are driving the modeling.

1. **Trend:** This is the long-range average outcome pattern over time. We often ask ourselves, "Is the data generally increasing or decreasing over time? How is it changing? Linearly? Exponentially?"

2. **Seasonality:** This refers to cyclical patterns related to calendar time, such as seasons, quarters, months, days of the week, or time of day, etc. We might wonder, "Are there daily cycles, weekly cycles, monthly cycles, annual cycles, and/or multi-year cycles?" (e.g., amount of sunshine has a daily and annual cycle, the climate has multi-year El Nino and La Nina climate cycles on top of annual seasonality of temperature)
3. **Noise:** There is high-frequency variability in the outcome not attributed to trend or seasonality. In other words, noise is what is left over. We might break up this noise into two noise components:

- serial correlation: the size and direction of the noise today is likely to be similar tomorrow
- independent measurement error: due to natural variation in the measurement device

We often ask, "Are there structures/patterns in the noise so that we could use probability to model the serial correlation? Is there a certain range of time or space in which we have dependence? Is the magnitude of the variability constant across time?"

4.1 Trend

Let's start by focusing on the trend component. Our goal here is to estimate the overall outcome pattern over time.

We'll plot the residuals (what is left over) over time to see if we've successfully estimated and removed the trend. We'll see if the average residual is fairly constant across time. If not, we must try a different approach to model/remove the trend.

4.1.1 Parametric Approaches

4.1.1.1 Global Transformations

If the trend, $f(x_t)$, is linearly increasing or decreasing in time (with no seasonality), then we could use a linear regression model to estimate the trend with the following model,

$$Y_t = \beta_0 + \beta_1 x_t + \epsilon_t$$

where x_t is some measure of time (e.g., age, time from baseline, etc.).

If the overall mean trend is quadratic, we could include a x_t^2 term in the regression model,

$$Y_t = \beta_0 + \beta_1 x_t + \beta_2 x_t^2 + \epsilon_t$$

Any standard regression technique for non-linear relationships (polynomials, splines, etc.) could also be used here to model the trend. As discussed in Chapter 1, we can use ordinary least squares (OLS), `lm()` in R, to get an unbiased estimate of these coefficients in linear regression. But remember, the standard errors and subsequent inference (p-values, confidence intervals) may not be valid due to correlated noise.

Let's look at some example data and see how we can model the trend.

Data Example 1

Below are the quarterly earnings per share of the company Johnson and Johnson from 1960-1980 in the `astsa` package (Stoffer 2025). The first observation is the 1st quarter of 1960 ($t = 1$) and the second is the 2nd quarter of 1960 ($t = 2$)

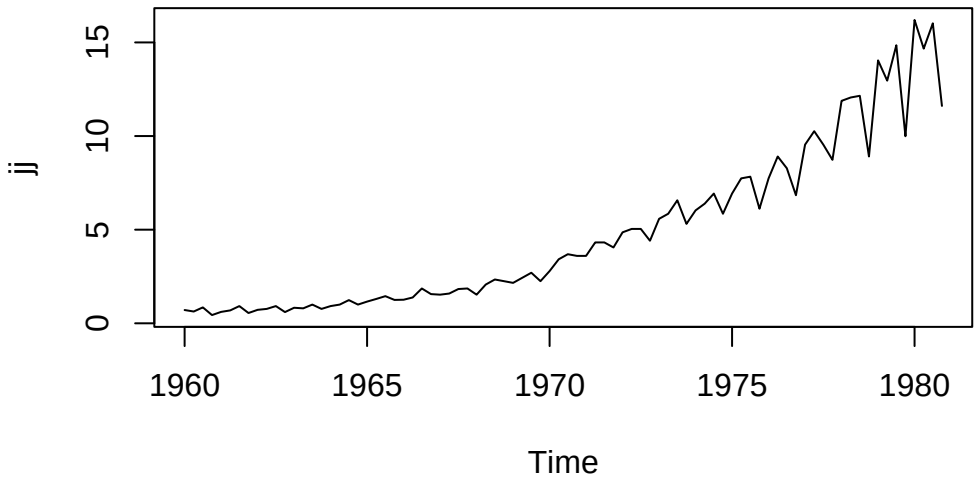
```
library(astsa) # R package for time series
jj
```

```
         Qtr1      Qtr2      Qtr3      Qtr4
1960  0.710000  0.630000  0.850000  0.440000
1961  0.610000  0.690000  0.920000  0.550000
1962  0.720000  0.770000  0.920000  0.600000
1963  0.830000  0.800000  1.000000  0.770000
1964  0.920000  1.000000  1.240000  1.000000
1965  1.160000  1.300000  1.450000  1.250000
1966  1.260000  1.380000  1.860000  1.560000
1967  1.530000  1.590000  1.830000  1.860000
1968  1.530000  2.070000  2.340000  2.250000
1969  2.160000  2.430000  2.700000  2.250000
1970  2.790000  3.420000  3.690000  3.600000
1971  3.600000  4.320000  4.320000  4.050000
1972  4.860000  5.040000  5.040000  4.410000
1973  5.580000  5.850000  6.570000  5.310000
1974  6.030000  6.390000  6.930000  5.850000
1975  6.930000  7.740000  7.830000  6.120000
1976  7.740000  8.910000  8.280000  6.840000
1977  9.540000 10.260000  9.540000  8.729999
1978 11.880000 12.060000 12.150000  8.910000
1979 14.040000 12.960000 14.850000  9.990000
1980 16.200000 14.670000 16.020000 11.610000
```

```
class(jj) # ts stands for time series R class
```

```
[1] "ts"
```

```r
plot(jj) # plot() is smart enough to make the right x-axis because jj is ts object
```

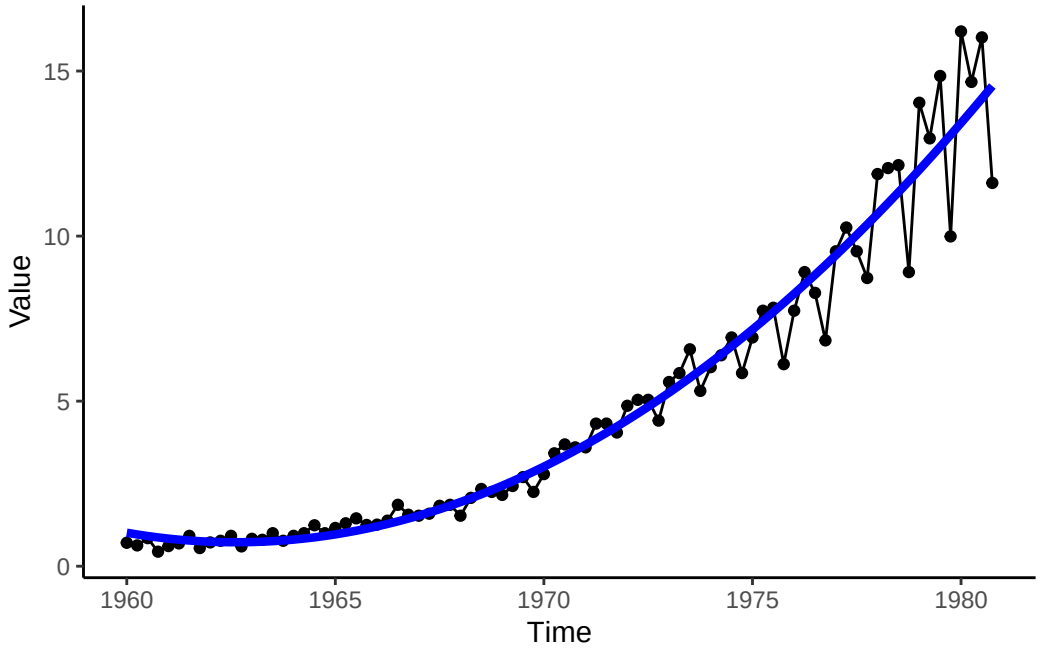


We see that the earnings are overall increasing over time, but a bit faster than a linear relationship (there is some curvature in the trend). Let's transform this dataset into a data frame to use ggplot2, an alternative graphing package in R.

```r
library(ggplot2)
library(dplyr)

jjTS <- data.frame(
  Value = as.numeric(jj),
  Time = time(jj), # time() works on ts objects
  Quarter = cycle(jj)) # cycle() works on ts objects

quadTrendMod <- lm(Value ~ poly(Time, degree = 2, raw = TRUE), data = jjTS) # poly() fits a p

jjTS <- jjTS %>%
  mutate(Trend = predict(quadTrendMod)) # Add the estimated trend to the dataset

jjTS %>%
  ggplot(aes(x = Time, y = Value)) +
  geom_point() +
  geom_line() +
  geom_line(aes(x = Time, y = Trend), color = 'blue', size = 1.5) +
  theme_classic()
```

```
Warning: Using `size` aesthetic for lines was deprecated in ggplot2 3.4.0.
i Please use `linewidth` instead.
```

```
Don't know how to automatically pick scale for object of type <ts>. Defaulting
to continuous.
```



Data Example 2

Below are the monthly counts (in thousands) of live births in the United States from 1948 to 1979. The first measurement is from January 1948 ($t = 1$), then February 1948 ($t = 2$), etc.

```
birth
```

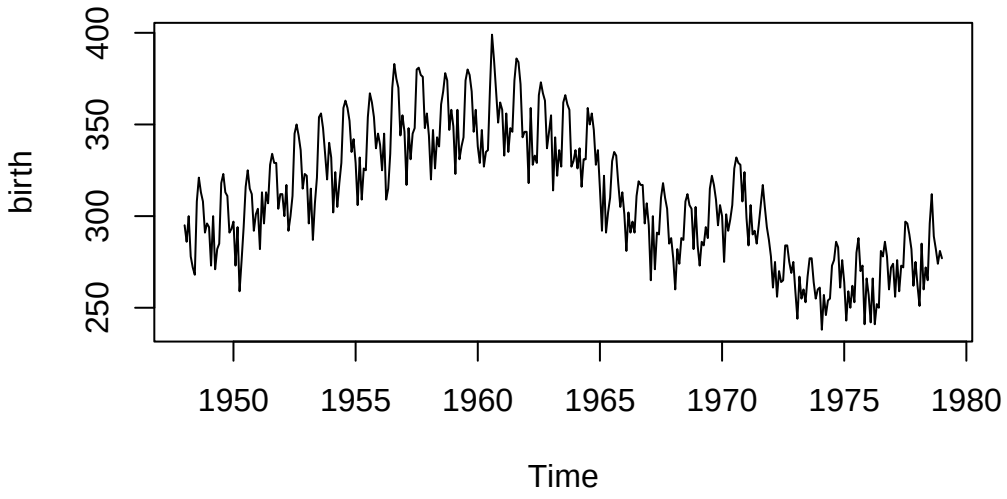
```
     Jan Feb Mar Apr May Jun Jul Aug Sep Oct Nov Dec
1948 295 286 300 278 272 268 308 321 313 308 291 296
1949 294 273 300 271 282 285 318 323 313 311 291 293
1950 297 273 294 259 276 294 316 325 315 312 292 301
1951 304 282 313 296 313 307 328 334 329 329 304 312
1952 312 300 317 292 300 311 345 350 344 336 315 323
1953 322 296 315 287 307 321 354 356 348 334 320 340
1954 332 302 324 305 318 329 359 363 359 352 335 342
1955 329 306 332 309 326 325 354 367 362 354 337 345
1956 339 325 345 309 315 334 370 383 375 370 344 355
1957 346 317 348 331 345 348 380 381 377 376 348 356
1958 344 320 347 326 343 338 361 368 378 374 347 358
1959 349 323 358 331 338 343 374 380 377 368 346 358
1960 338 329 347 327 335 336 370 399 385 368 351 362
```

```
1961 358 333 356 335 348 346 374 386 384 372 343 346
1962 346 318 359 328 333 329 366 373 367 363 337 346
1963 355 314 343 322 336 327 362 366 361 358 327 330
1964 336 326 337 316 331 331 359 350 356 347 328 336
1965 315 292 322 291 302 310 330 335 333 318 305 313
1966 301 281 302 291 297 291 311 319 317 317 296 307
1967 295 265 300 271 291 290 310 318 310 304 285 288
1968 277 260 282 274 288 287 308 312 306 304 282 305
1969 284 273 286 284 294 288 315 322 317 309 295 306
1970 300 275 301 292 298 306 326 332 329 328 308 324
1971 299 284 306 290 292 285 295 306 317 305 294 287
1972 278 261 275 256 270 264 265 284 284 275 269 275
1973 259 244 267 255 260 253 267 277 277 264 255 260
1974 261 238 257 246 254 255 273 276 286 283 261 276
1975 264 243 259 250 262 253 280 288 270 273 241 266
1976 257 242 266 241 252 250 281 278 286 278 260 272
1977 274 256 276 259 273 272 297 296 290 282 262 275
1978 262 251 285 260 272 265 296 312 289 282 274 281
1979 277
```

```
class(birth) # ts stands for time series R class
```

```
[1] "ts"
```

```
plot(birth) # plot() is smart enough to make the right x-axis because birth is ts object
```

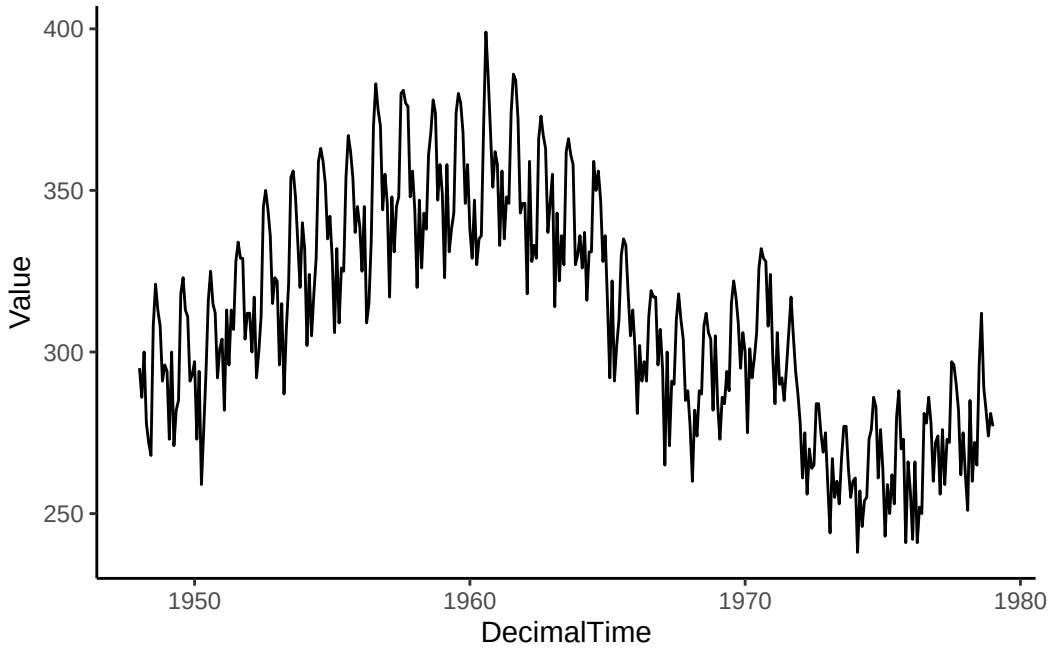


Looking at the plot (time-ordered points are connected with a line segment), we note a positive linear relationship from about 1948 to 1960 and a somewhat negative relationship between 1960

to 1968, with another slight increase and then a drop around 1972. This is not a polynomial trend over time. We'll need another approach!

Let's also transform this dataset into a data frame to use ggplot2.

```
birthTS <- data.frame(
  Year = floor(as.numeric(time(birth))), # time() works on ts objects
  Month = as.numeric(cycle(birth)), # cycle() works on ts objects
  Value = as.numeric(birth))

birthTS <- birthTS %>%
  mutate(DecimalTime = Year + (Month-1)/12)

birthTS %>%
  ggplot(aes(x = DecimalTime, y = Value)) +
  geom_line() +
  theme_classic()
```



4.1.1.2 Splines

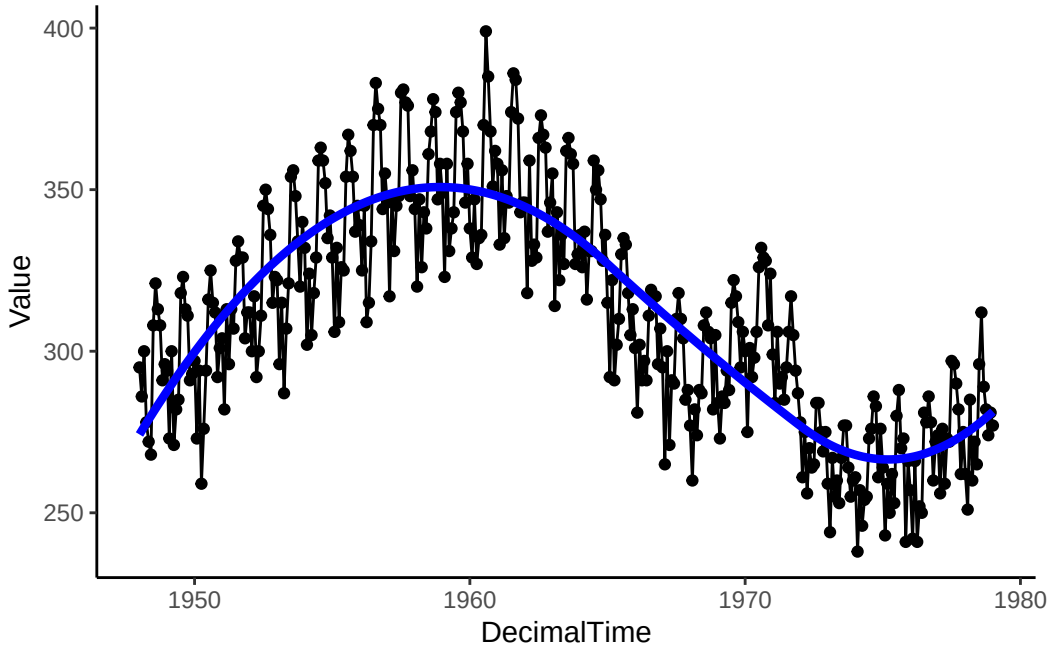
Let's try a regression basis spline, which is a smooth piecewise polynomial. This means we break up the x-axis into intervals using knots or breakpoints and model a different polynomial in each interval. In particular, let's fit a quadratic polynomial with knots/breaks at 1965 and 1972.

```r
library(splines)

TrendSpline <- lm(Value ~ bs(DecimalTime, degree = 2, knots = c(1965,1972)), data = birthTS)
summary(TrendSpline)
```

```
Call:
lm(formula = Value ~ bs(DecimalTime, degree = 2, knots = c(1965,
    1972)), data = birthTS)

Residuals:
    Min      1Q  Median      3Q     Max
-46.178 -12.789  -0.974  12.350  49.973

Coefficients:
                                                  Estimate Std. Error t value
(Intercept)                                        274.249      3.527  77.762
bs(DecimalTime, degree = 2, knots = c(1965, 1972))1  119.206      6.817  17.486
bs(DecimalTime, degree = 2, knots = c(1965, 1972))2   25.459      3.961   6.428
bs(DecimalTime, degree = 2, knots = c(1965, 1972))3  -20.140      6.115  -3.294
bs(DecimalTime, degree = 2, knots = c(1965, 1972))4    7.062      6.006   1.176
                                                  Pr(>|t|)
(Intercept)                                        < 2e-16 ***
bs(DecimalTime, degree = 2, knots = c(1965, 1972))1  < 2e-16 ***
bs(DecimalTime, degree = 2, knots = c(1965, 1972))2 4.02e-10 ***
bs(DecimalTime, degree = 2, knots = c(1965, 1972))3  0.00108 **
bs(DecimalTime, degree = 2, knots = c(1965, 1972))4  0.24039
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

Residual standard error: 18.46 on 368 degrees of freedom
Multiple R-squared:  0.7287,    Adjusted R-squared:  0.7258
F-statistic: 247.1 on 4 and 368 DF,  p-value: < 2.2e-16
```

```r
birthTS <- birthTS %>%
  mutate(SplineTrend = predict(TrendSpline))

birthTS %>%
  ggplot(aes(x = DecimalTime, y = Value)) +
  geom_point() +
  geom_line() +
```

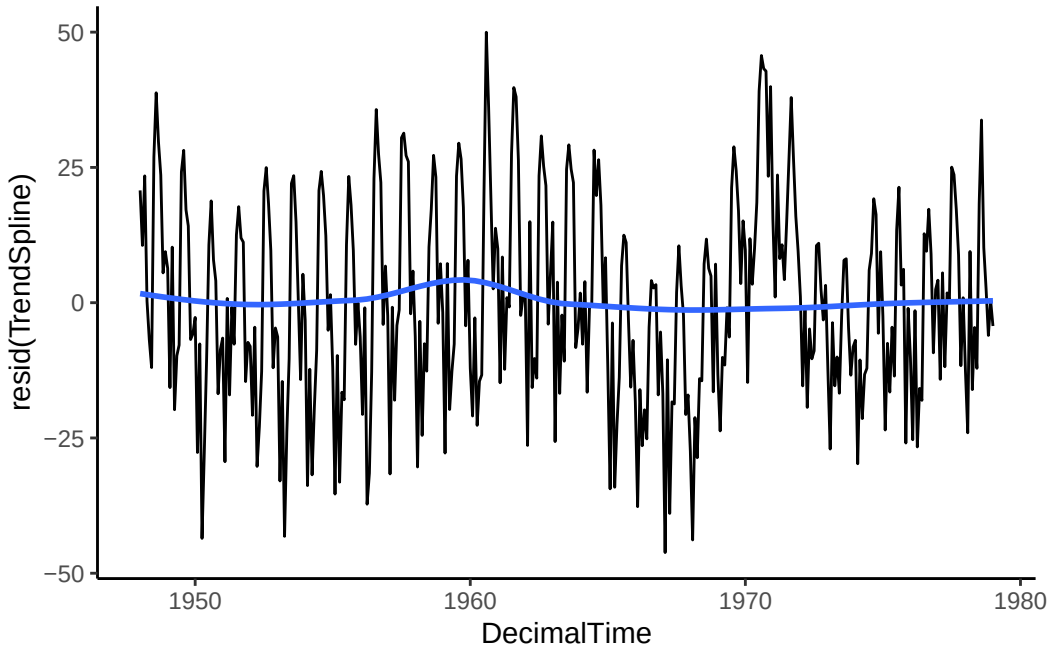
```
  geom_line(aes(x = DecimalTime, y = SplineTrend), color = 'blue', size = 1.5) +
  theme_classic()
```



Then let's see what is left over by plotting the residuals after removing the estimated trend. Is the mean constant? If so, we can ensure we've estimated and removed the long-range trend.

```
# Plotting residuals
birthTS %>%
  ggplot(aes(x = DecimalTime, y = resid(TrendSpline))) +
  geom_line() +
  geom_smooth(se = FALSE) +
  theme_classic()
```

```
`geom_smooth()` using method = 'loess' and formula = 'y ~ x'
```



These parametric approaches give us an estimated function to plug in times and get a predicted trend value, but they might not quite pick up the full trend.

What are alternative methods for estimating non-linear trends?

4.1.2 Nonparametric Approaches

Nonparametric methods for estimating a trend use algorithms (steps of a process) rather than parametric functions (linear, polynomial, splines) with a finite number of parameters to estimate.

4.1.2.1 Local Regression

Have you ever wondered how geom_smooth() estimates that blue smooth curve?

If you look at the documentation for `geom_smooth()`, you'll see that it uses the `loess()` function for smaller data sets. This trend estimation method is locally estimated scatterplot smoothing, also called local regression.

The loess algorithm is to predict the outcome (estimate the trend) at a point on the x-axis, x_0, following the steps below:

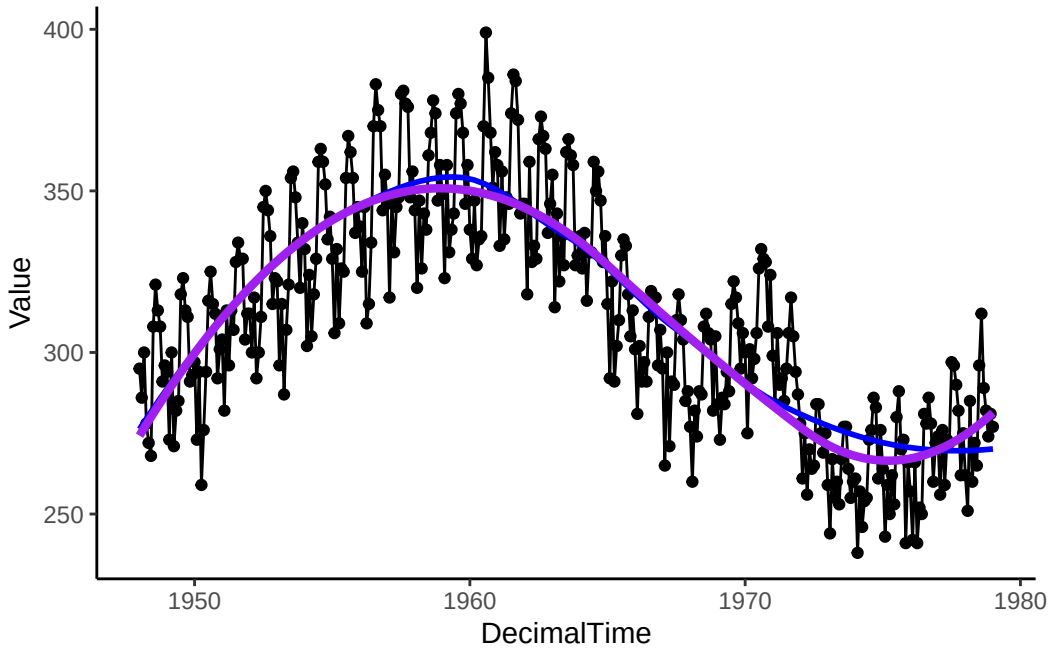
- Distance: calculate the distance between x_0 and each observed points x, $d(x, x_0) = |x - x_0|$.

- Find Local Area: keep the closest s proportion of observed data points (s determined by span parameter)
- Fit Model: fit a linear or quadratic regression model to the local area, weighting points close to x_0 higher (weighting determined by a chosen weighting kernel function)
- Predict: use that local model to predict the outcome at x_0

That process is repeated for a sequence of values along the x-axis to make a plot like the one below (blue: loess, purple: spline).

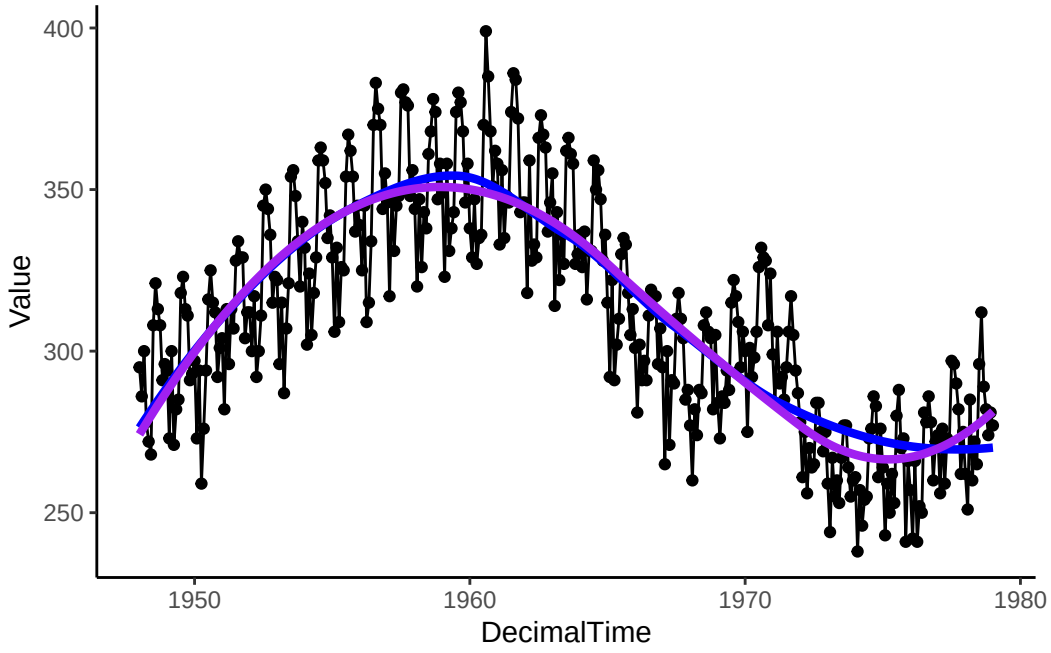
```
birthTS %>%
  ggplot(aes(x = DecimalTime, y = Value)) +
  geom_point() +
  geom_line() +
  geom_smooth(se = FALSE, color = 'blue') + # uses loess
  geom_line(aes(x = DecimalTime, y = SplineTrend), color = 'purple', size = 1.5) +
  theme_classic()
```

```
`geom_smooth()` using method = 'loess' and formula = 'y ~ x'
```



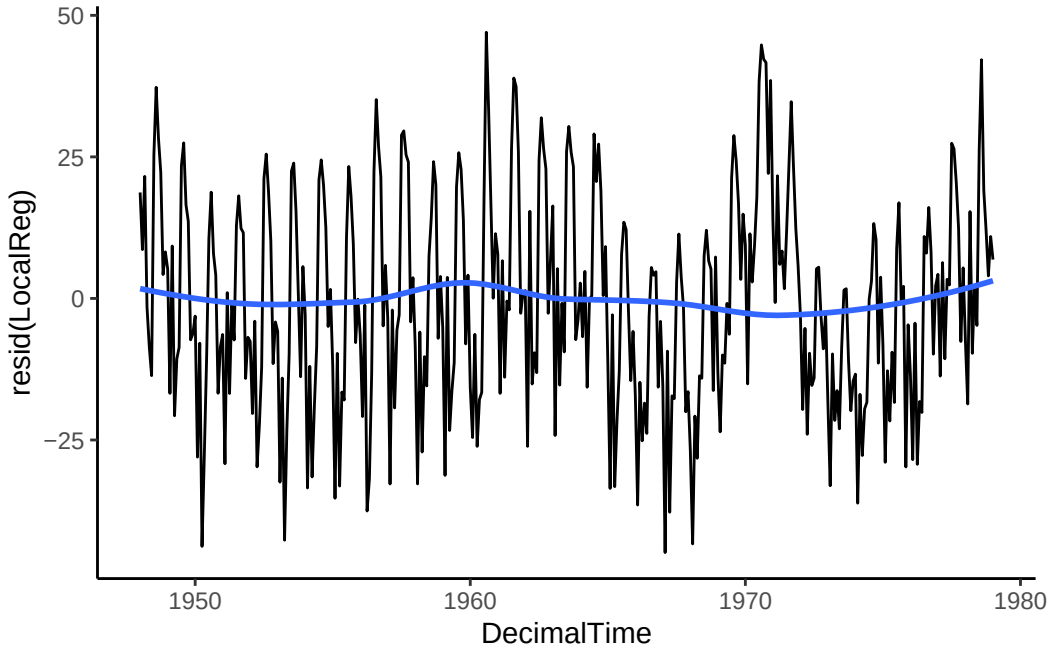
We can also get the estimated trend with the `loess()` function.

```r
# Running loess separately
LocalReg <- loess(Value ~ DecimalTime, data = birthTS, span = .75)
birthTS <- birthTS %>%
  mutate(LoessTrend = predict(LocalReg))

birthTS %>%
  ggplot(aes(x = DecimalTime, y = Value)) +
  geom_point() +
  geom_line() +
  geom_line(aes(x = DecimalTime, y = LoessTrend), color = 'blue', size = 1.5) +
  geom_line(aes(x = DecimalTime, y = SplineTrend), color = 'purple', size = 1.5) +
  theme_classic()
```




```r
# Plotting residuals
birthTS %>%
  ggplot(aes(x = DecimalTime, y = resid(LocalReg))) +
  geom_line() +
  geom_smooth(se = FALSE) +
  theme_classic()
```

```
`geom_smooth()` using method = 'loess' and formula = 'y ~ x'
```

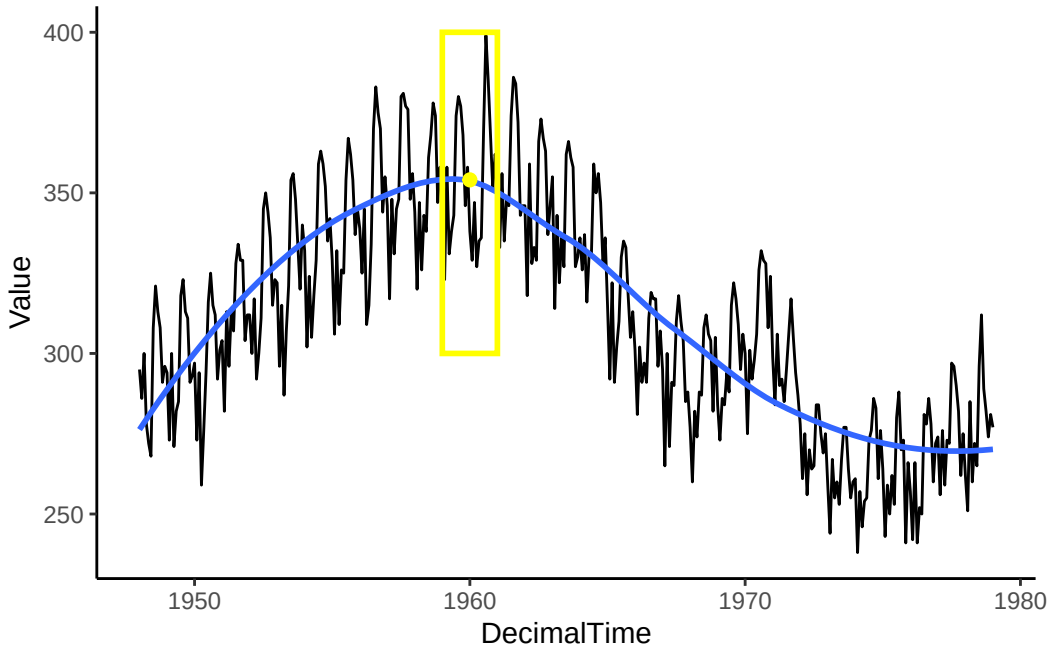


4.1.2.2 Moving Average Filter

Local regression is an extension of a simple idea called a **moving average filter** (See Chatfield and Xing 2019 for a brief introduction to moving averages and linear filtering in general).

Imagine a window that only shows you what is "local" or right in front of you, and we can adjust the window width to include more or less data. In the illustration below, we have drawn a 2-year window around January 1960. When considering information around Jan 1960, the window limits our view to only those one year before and one year after (from Jan 1959-Jan 1961).

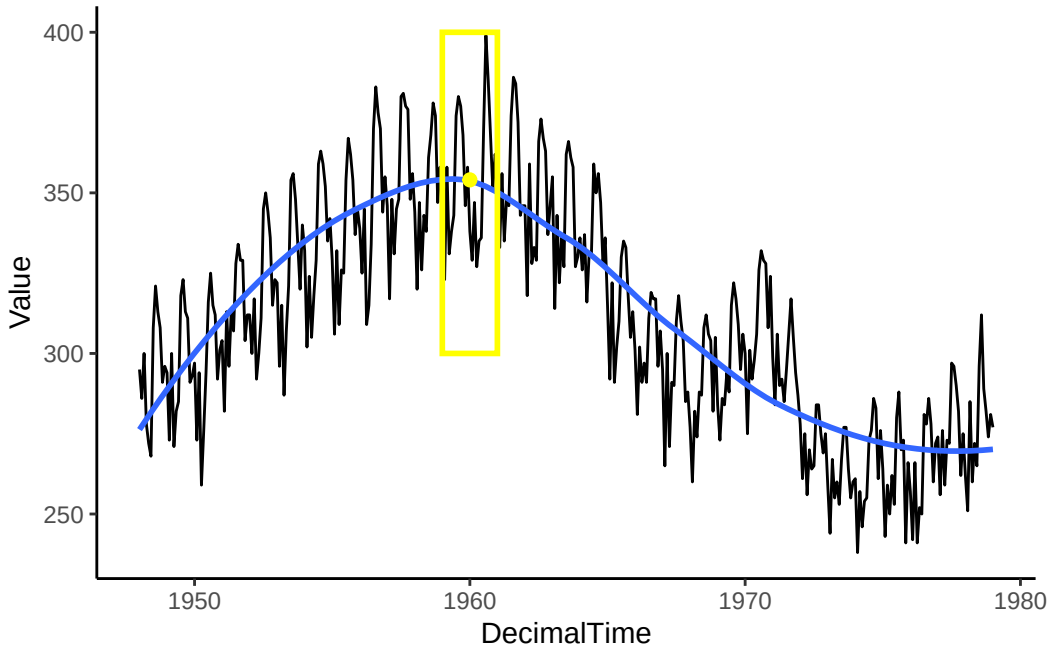
```
`geom_smooth()` using method = 'loess' and formula = 'y ~ x'
```



For our circumstances, the "local" window will include an odd number of observations (number of points $= 1 + 2a$ for a given a value). Among that small local group of observations, we take the average to estimate the mean at the center of those points (rather than fit a linear regression model),

$$\hat{f}(x_t) = (1 + 2a)^{-1} \sum_{k=-a}^{a} y_{t+k}$$

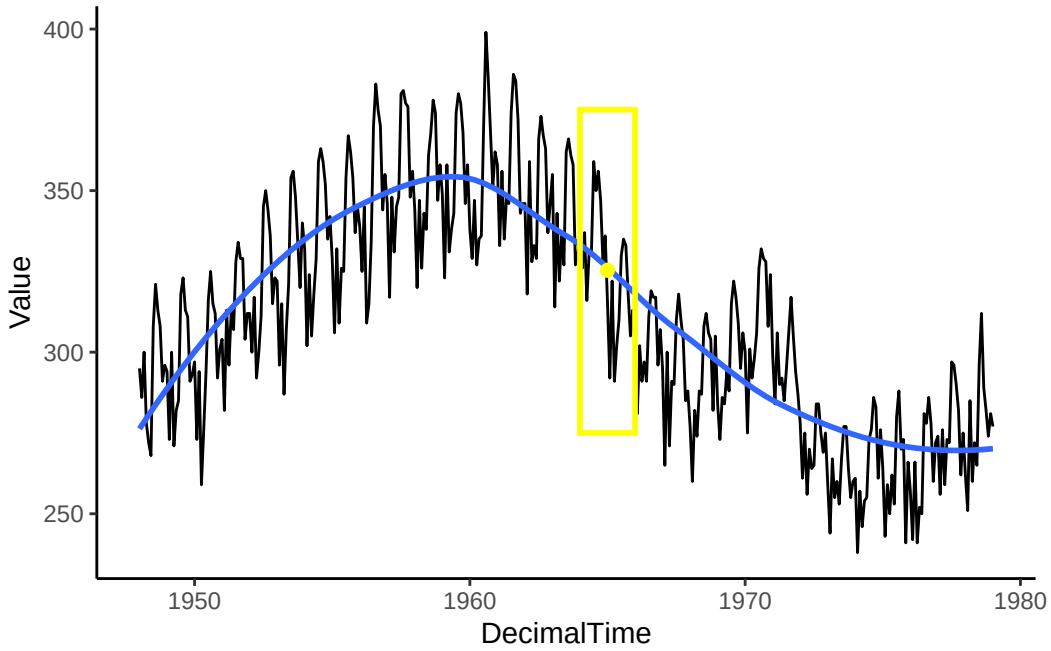
```
`geom_smooth()` using method = 'loess' and formula = 'y ~ x'
```



Then, we move the window and calculate the mean within that window. Continue until you have a smooth curve estimate for each point in the time range of interest.

```
Box <- data.frame(x = 1965, y = 325, w = 2) #2-year window around 1980
Mean <- birthTS %>%
  filter(DecimalTime < 1965 + 1 & DecimalTime > 1965 - 1) %>%
  summarize(M = mean(Value)) %>% #Mean value of points in the window
  mutate(x = 1965)

birthTS %>%
  ggplot(aes(x = DecimalTime, y = Value)) +
  geom_line() +
  geom_smooth(se = FALSE) +
  geom_tile(data = Box, aes(x = x,y=y,width=w),height = 100, fill=alpha("grey",0), colour =
  geom_point(data = Mean, aes(x = x, y = M), colour = 'yellow', size = 2)+
  theme_classic()
```

```
`geom_smooth()` using method = 'loess' and formula = 'y ~ x'
```

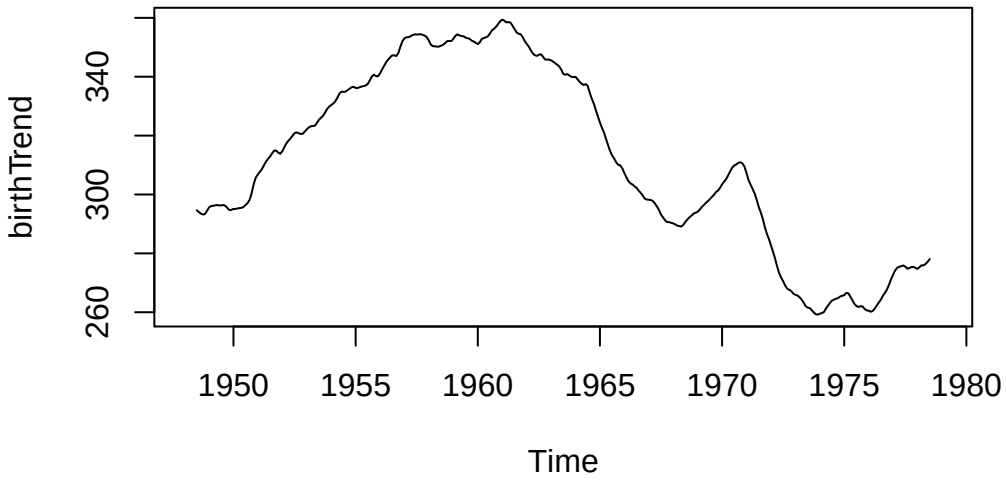


This method works well for windows of odd-numbered length (equal number of points around the center), but it can be adjusted if you desire an even-numbered length window by adding 1/2 of the two extreme lags so that the middle value lines up with time t.

For the above monthly data example, we might want a 12-point moving average, so we would have

$$\hat{f}(x_t) = \frac{0.5y_{t-6} + y_{t-5} + y_{t-4} + y_{t-3} + y_{t-2} + y_{t-1} + y_t + y_{t+1} + y_{t+2} + y_{t+3} + y_{t+4} + y_{t+5} + 0.5y_{t+6}}{12}$$

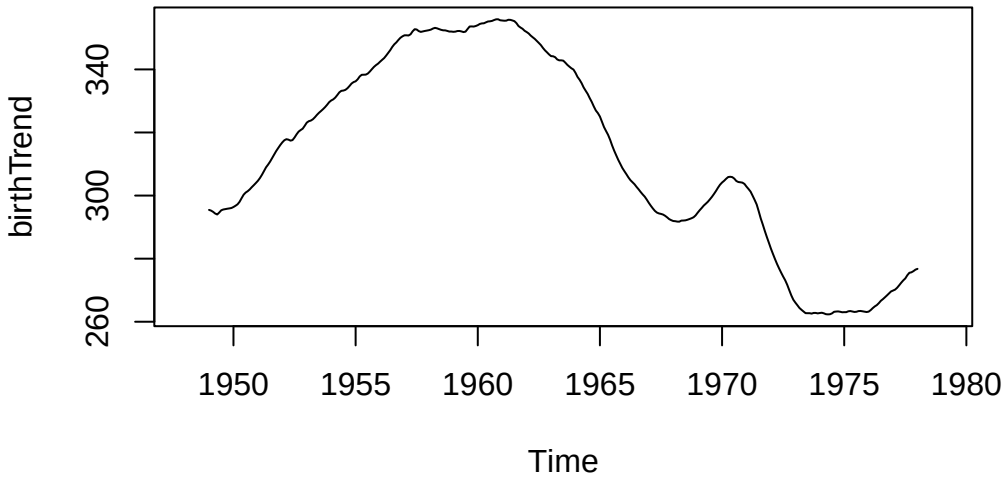
```
fltr <- c(0.5, rep(1, 11), 0.5)/12 # weights for the weighted average above
birthTrend <- stats::filter(birth, filter = fltr, method = 'convo', sides = 2) #use the ts ol
plot(birthTrend)
```



Now, if we increase the window's width (24-point moving window), you should see a smoother curve.

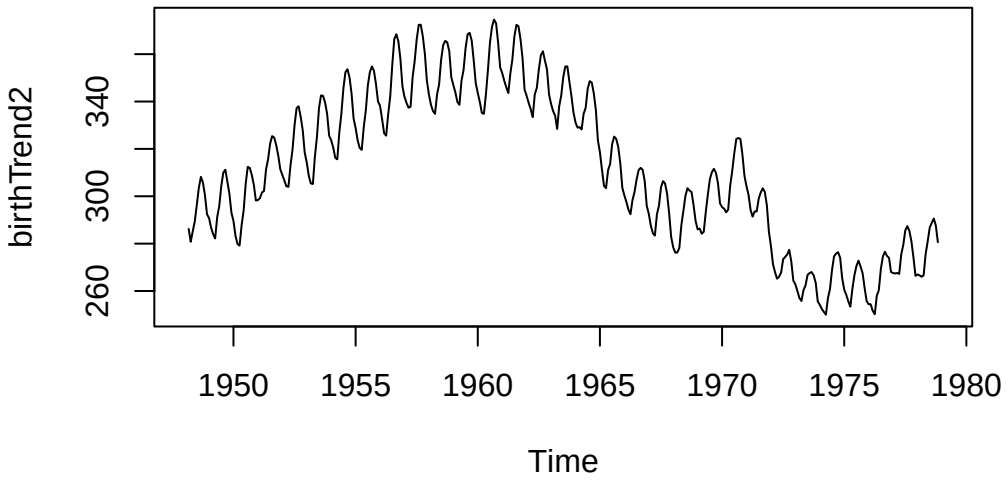
$$\hat{f}(x_t) = \frac{0.5y_{t-12} + y_{t-11} + ... + y_{t-1} + y_t + y_{t+1} + ... + y_{t+11} + 0.5y_{t+12}}{24}$$

```
fltr <- c(0.5, rep(1, 23), 0.5)/24 # weights for moving average
birthTrend <- stats::filter(birth, filter = fltr, method = 'convo', sides = 2) # use the ts c
plot(birthTrend)
```



What if we did a 5-point moving average instead of a 12 or 24-point moving average?

```
fltr <- rep(1, 5)/5
birthTrend2 <- stats::filter(birth, filter = fltr, method = 'convo', sides = 2)
plot(birthTrend2)
```



We see the seasonality (repeating cycles) because the 5-point window averages about half of the year's data for each estimate. Thus, the highs and lows don't balance each other out.

You must be mindful of the window width for the moving average, depending on the context of the data and the existing cycles in the data.

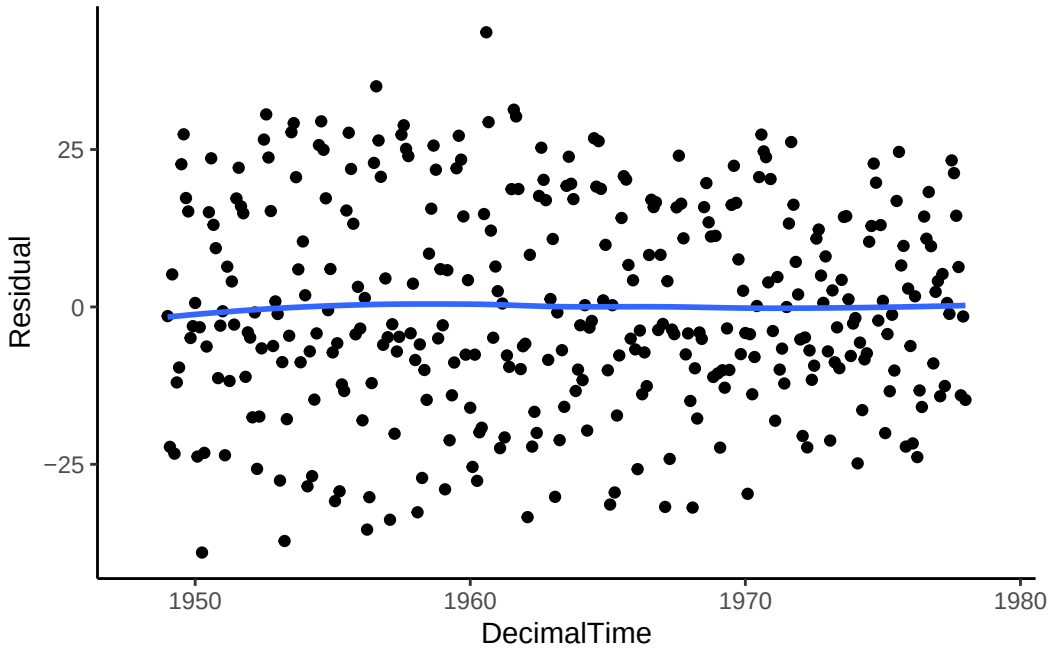
Let's see what is left over. Is the mean of the residuals constant across time (indicating that we have fully removed the trend)?

```
birthTS <- birthTS %>%
  mutate(Trend = birthTrend) %>%
  mutate(Residual = Value - Trend)

birthTS %>%
  ggplot(aes(x = DecimalTime, y = Residual)) +
  geom_point() +
  geom_smooth(se = FALSE) +
  theme_classic()
```

```
Don't know how to automatically pick scale for object of type <ts>. Defaulting
to continuous.
`geom_smooth()` using method = 'loess' and formula = 'y ~ x'
```

```
Warning: Removed 24 rows containing non-finite outside the scale range
(`stat_smooth()`).
```

```
Warning: Removed 24 rows containing missing values or values outside the scale range
(`geom_point()`).
```



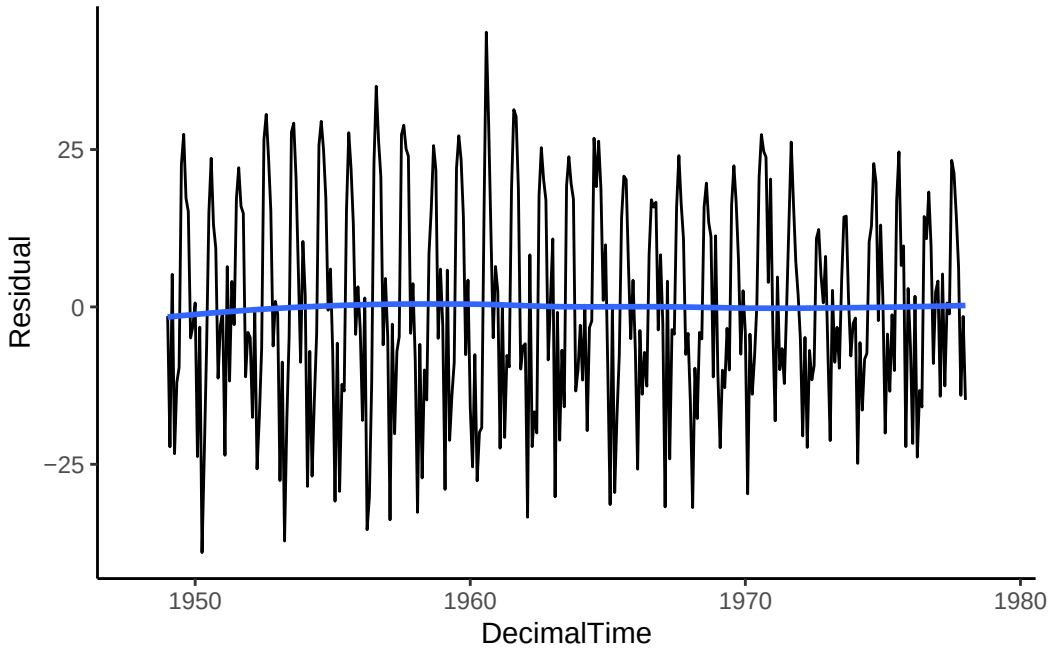
If we connect the points, we might see a cyclic pattern...

```
birthTS %>%
  ggplot(aes(x = DecimalTime, y = Residual)) +
  geom_line() +
  geom_smooth(se = FALSE) +
  theme_classic()
```

```
Don't know how to automatically pick scale for object of type <ts>. Defaulting
to continuous.
`geom_smooth()` using method = 'loess' and formula = 'y ~ x'
```

```
Warning: Removed 24 rows containing non-finite outside the scale range
(`stat_smooth()`).
```

```
Warning: Removed 24 rows containing missing values or values outside the scale range
(`geom_line()`).
```



We still have to deal with seasonality (we'll get there), but the moving average does a fairly good job at removing the trend because we see that the trend of the residuals is 0 on average!

The moving average filter did a decent job of estimating the trend because of its flexibility, but we can't write down a function statement for that trend. This is the downside of nonparametric methods.

4.1.2.3 Differencing to Remove Trend

If we don't necessarily care about estimating the trend for prediction, we could use **differencing** to explicitly remove a trend.

We'll talk about first and second-order differences, which are applicable if we have a linear or quadratic trend.

Linear Trend - First Difference

If we have a linear trend such that $Y_t = b_0 + b_1 t + \epsilon_t$, the difference between neighboring outcomes (1 time unit apart) will essentially remove that trend.

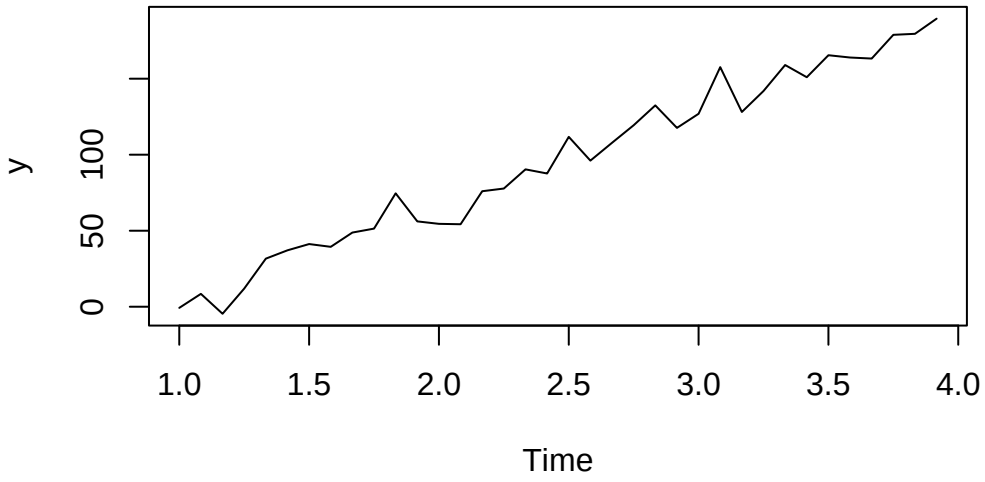
Take the outcome at time t and an outcome that lags behind one time unit:

$$\begin{aligned} Y_t - Y_{t-1} &= (b_0 + b_1 t + \epsilon_t) - (b_0 + b_1(t-1) + \epsilon_{t-1}) \\ &= b_0 + b_1 t + \epsilon_t - b_0 - b_1 t + b_1 - \epsilon_{t-1} \\ &= b_1 + \epsilon_t - \epsilon_{t-1} \end{aligned}$$

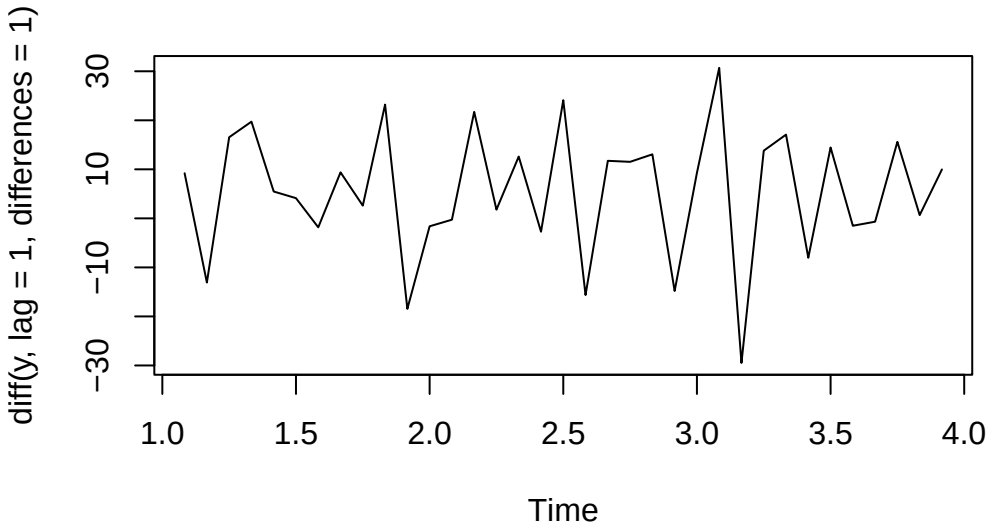
which is constant with respect to time, t.

See below for a simulated example.

```
t <- 1:36
y <- ts(2 + 5*t + rnorm(36,sd = 10), frequency=12)
plot(y)
```




```
plot(diff(y, lag = 1, differences = 1))
```



Quadratic Trend - Second Order Differences

If we have a quadratic relationship such that $Y_t = b_0 + b_1 t + b_2 t^2 + \epsilon_t$, then taking the **second order difference** between neighboring outcomes will remove that trend.

Take the outcome at time t and an outcome that lags behind one time unit (first difference):

$$(Y_t - Y_{t-1}) = (b_0 + b_1 t + b_2 t^2 + \epsilon_t) - (b_0 + b_1(t-1) + b_2(t-1)^2 + \epsilon_{t-1})$$

$$= b_0 + b_1 t + b_2 t^2 + \epsilon_t - b_0 - b_1 x_t + b_1 - b_2 t^2 + 2b_2 t - b_2^2 - \epsilon_{t-1}$$

$$= b_1 - b_2^2 + 2b_2 t + \epsilon_t - \epsilon_{t-1}$$

and thus, the second-order difference is the difference in the neighboring first differences:

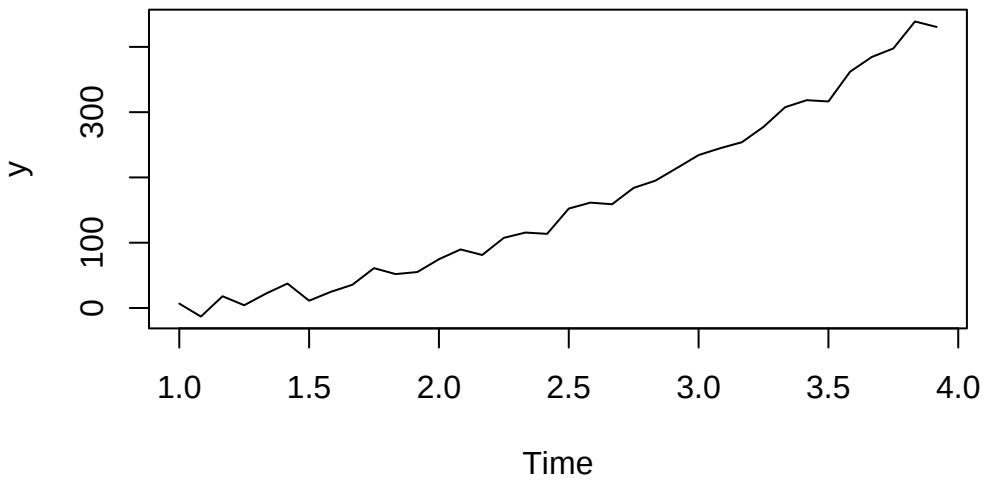
$$(Y_t - Y_{t-1}) - (Y_{t-1} - Y_{t-2}) = (b_1 - b_2^2 + 2b_2 t + \epsilon_t - \epsilon_{t-1}) - (b_1 - b_2^2 + 2b_2(t-1) + \epsilon_{t-1} - \epsilon_{t-2})$$

$$= 2b_2 + \epsilon_t - 2\epsilon_{t-1} + \epsilon_{t-2}$$

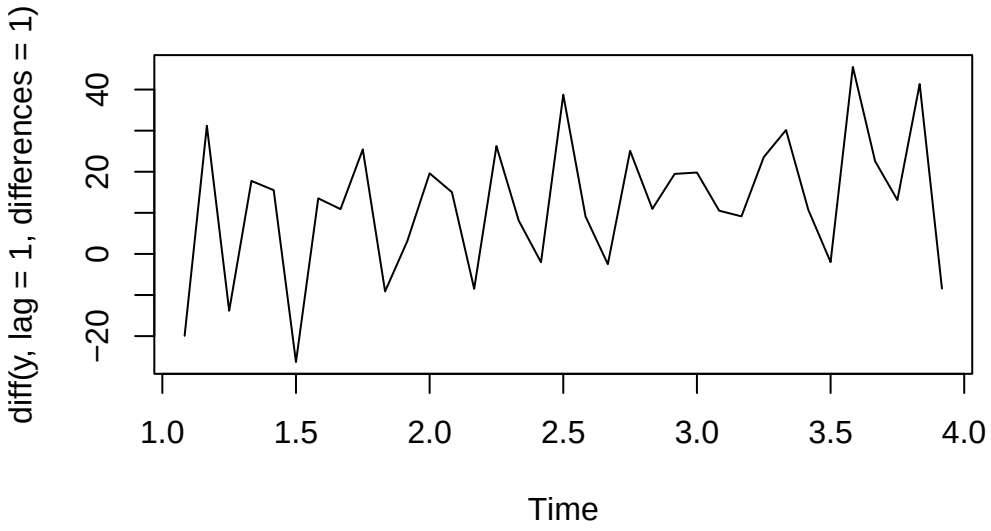
which is constant with respect to time, t.

See below for an example of simulated data.

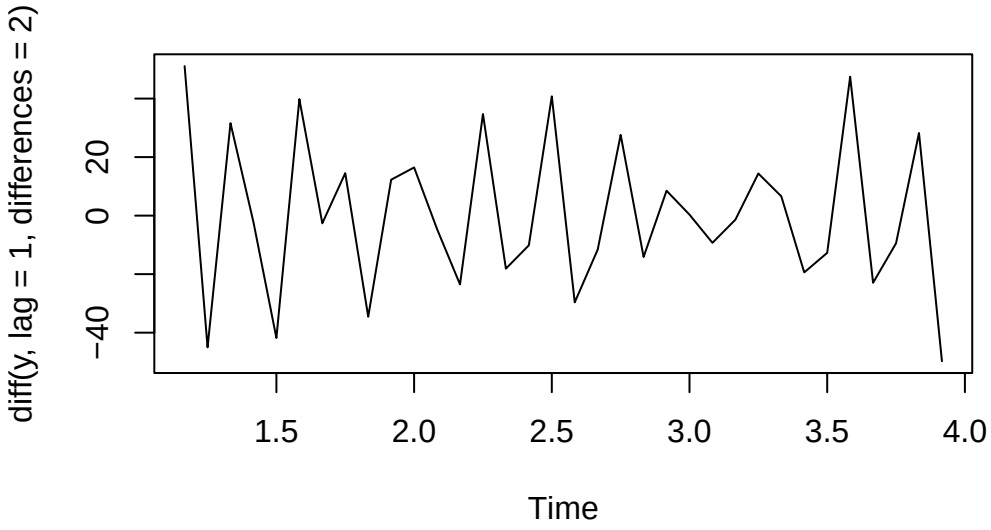
```
t <- 1:36
y <- ts(2 + 1.5*t + .3*t^2 + rnorm(36,sd = 10), frequency=12)
plot(y)
```




```
plot(diff(y, lag = 1, differences = 1)) #first difference
```




```
plot(diff(y, lag = 1, differences = 2)) #second order difference
```



Let's see what we get when we use differencing on the real birth count data. It looks like the first differences resulted in data with no overall trend despite the complex trend over time!

```
birthTS %>%
  mutate(Diff = c(NA, diff(Value, lag = 1, differences = 1))) %>%
  ggplot(aes(x = DecimalTime, y = Diff)) +
  geom_point() +
  geom_line() +
  geom_hline(yintercept = 0) +
```

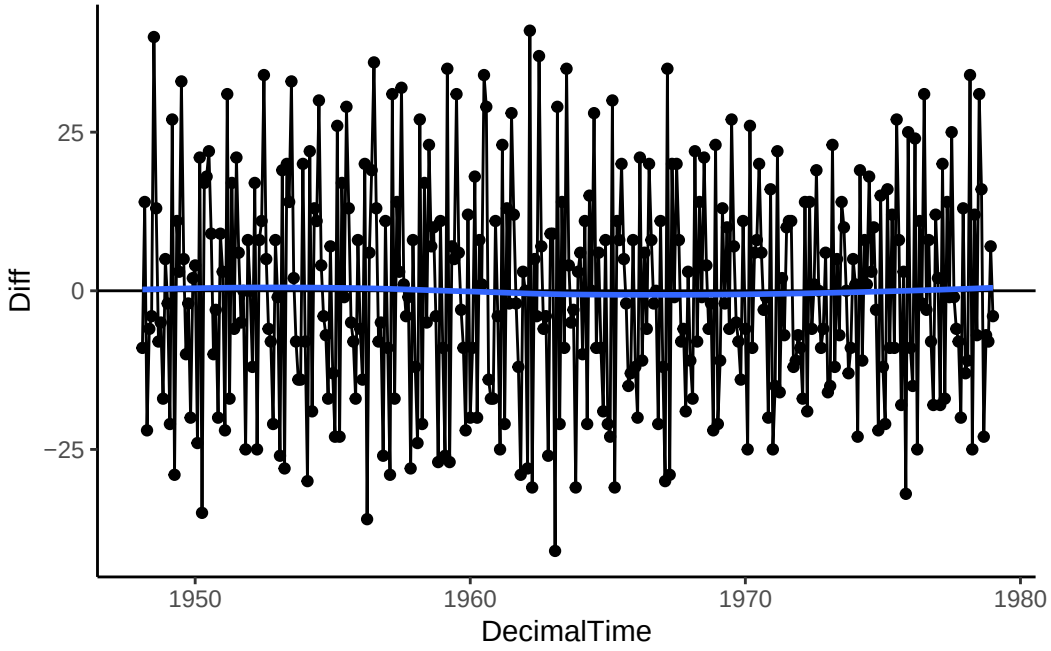
```
  geom_smooth(se = FALSE) +
  theme_classic()
```

```
`geom_smooth()` using method = 'loess' and formula = 'y ~ x'
```

```
Warning: Removed 1 row containing non-finite outside the scale range
(`stat_smooth()`).
```

```
Warning: Removed 1 row containing missing values or values outside the scale range
(`geom_point()`).
```

```
Warning: Removed 1 row containing missing values or values outside the scale range
(`geom_line()`).
```



4.1.3 In Practice: Estimate vs. Remove

We have a few methods to estimate the trend,

1. Parametric Regression: Global Transformations (polynomials)
2. Parametric Regression: Spline basis
3. Nonparametric: Local Regression

4. Nonparametric: Moving Average Filter

Among the estimation methods, two are **parametric** in that we estimate slope parameters or coefficients (regression techniques), and two are **nonparametric** in that we do not have any parameters to estimate but rather focus on the whole function.

Things to consider when choosing an estimation method:

- How well does it approximate the true underlying trend? Does it systematically under- or overestimate the trend for certain periods, or are the residuals on average zero across time?
- Do you want to use past trends to predict the future? If so, you may want to consider a parametric approach.

We have two approaches to removing the trend,

1. **Estimate, then Residuals.** Estimate the trend first, then subtract the trend from observations to get residuals, $y_t - \hat{f}(x_t)$.
2. **Difference.** Skip the estimating and try first or second-order differencing of neighboring observations.

Things to consider when choosing an approach for removing the trend:

- How well does it remove the underlying trend? Are the residuals (what is left over) close to zero on average, or are there still long-range patterns in the data?
- Do you want to report on the estimated trend? Then estimate first.
- If your main goal is prediction, try differencing.

4.2 Seasonality

Once we've estimated and removed the trend, we may notice cyclic patterns left in the data. Depending on the cyclic pattern's form, we could use a few different techniques to model the seasonal pattern.

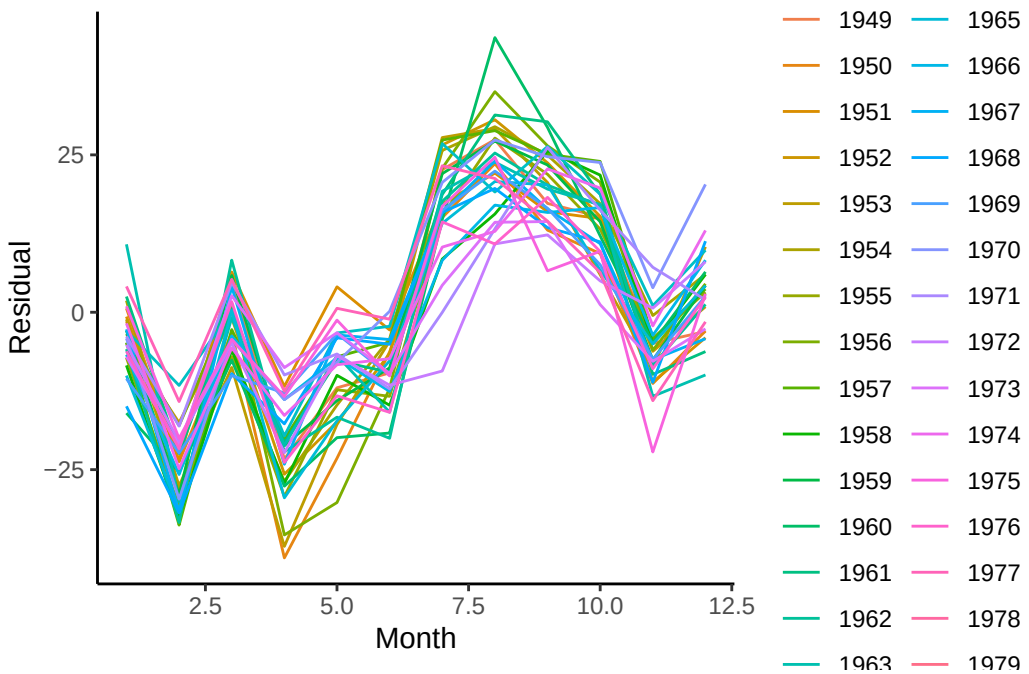
First, let's visualize the variability in the seasonality between different years for the birth data set to see if there is a consistent cycle. To do so, we'll work with the residuals after removing the trend (for now, let's use the moving average filter). Since we are interested in the cycles, we want to know how the number of births varies across the months within a year.

Using the ggplot2 package, we can visualize this with `geom_line()` by plotting the residuals by the month and having it color each line according to the year. To do this, it creates one line per year. Interestingly, August and September have the highest residuals (approximately nine months after winter break). In this case, we see that every year follows the same pattern.

```
birthTS %>%
  ggplot(aes(x = Month, y = Residual, color = factor(Year))) +
  geom_line() +
  theme_classic()
```

```
Don't know how to automatically pick scale for object of type <ts>. Defaulting
to continuous.
```

```
Warning: Removed 24 rows containing missing values or values outside the scale range
(`geom_line()`).
```



4.2.1 Parametric Approaches

Similar to the trend, we can use some of our parametric tools to model the cyclic patterns in the data using our linear regression structures.

4.2.1.1 Indicator Variables

To estimate the seasonality that does not have a recognizable cyclic functional pattern (sine or cosine curve), we can model each month (or appropriate time unit within a cycle) to have

its own average by including indicator variables for each month (or appropriate unit within a cycle) in a linear regression model.

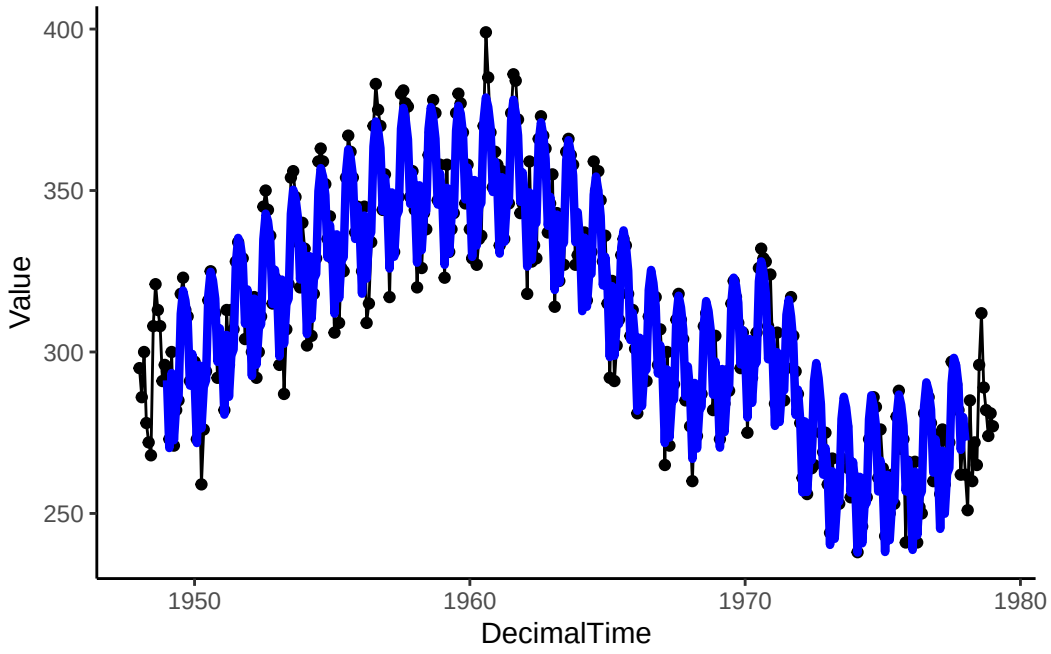
Like when we estimate the trend, we want to ensure that we have fully estimated the seasonality, but check the residuals to ensure that there is no seasonality in what is left over.

```
birthTS <- birthTS %>%
  mutate(Month = factor(Month))

SeasonModel <- lm(Residual ~  Month, data = birthTS)
SeasonModel
```

```
Call:
lm(formula = Residual ~ Month, data = birthTS)

Coefficients:
(Intercept)       Month2       Month3       Month4       Month5       Month6
     -4.231      -20.734        2.884      -17.412       -5.832       -4.814
     Month7       Month8       Month9      Month10      Month11      Month12
     20.738       27.650       24.434       18.024       -2.110        7.567
```

```
birthTS <- birthTS %>%
  mutate(Season = predict(SeasonModel, newdata = data.frame(Month = birthTS$Month))) %>%
  mutate(ResidualTS = Residual - Season)

# Estimated Trend + Seasonality
birthTS %>%
  ggplot(aes(x = DecimalTime, y = Value)) +
  geom_point() +
  geom_line() +
  geom_line(aes(x = DecimalTime, y = Trend + Season), color = 'blue', size = 1.5) +
  theme_classic()
```

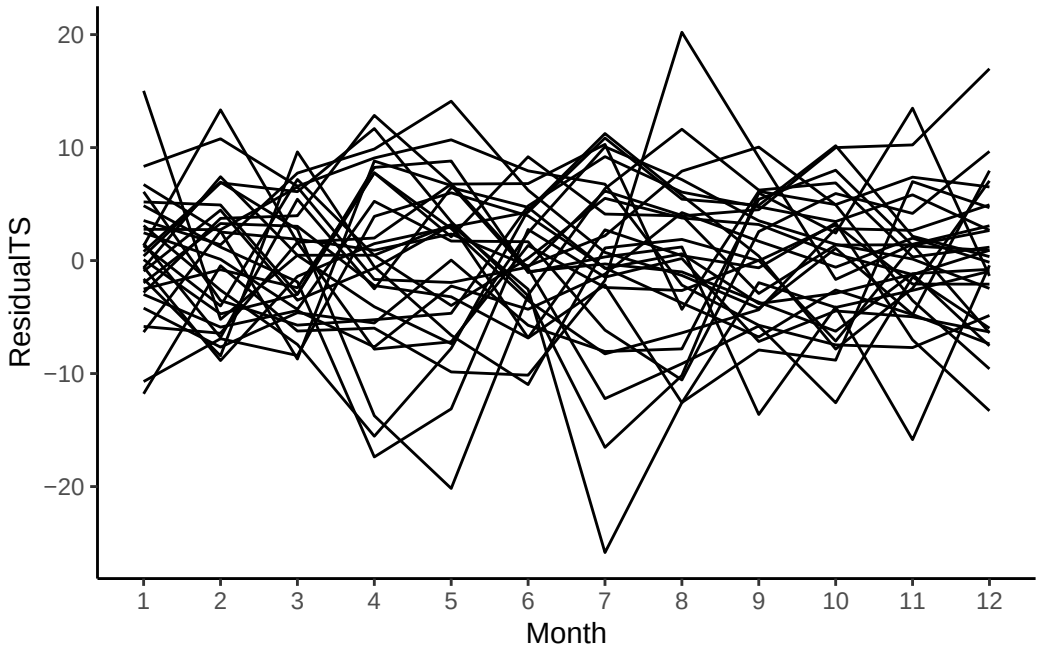
```
Warning: Removed 24 rows containing missing values or values outside the scale range
(`geom_line()`).
```




```
# Residuals
birthTS %>%
  ggplot(aes(x = Month, y = ResidualTS)) +
  geom_line(aes(group = factor(Year))) +
  geom_smooth(se = FALSE) +
  theme_classic()
```

```
Don't know how to automatically pick scale for object of type <ts>. Defaulting
to continuous.
`geom_smooth()` using method = 'loess' and formula = 'y ~ x'
```

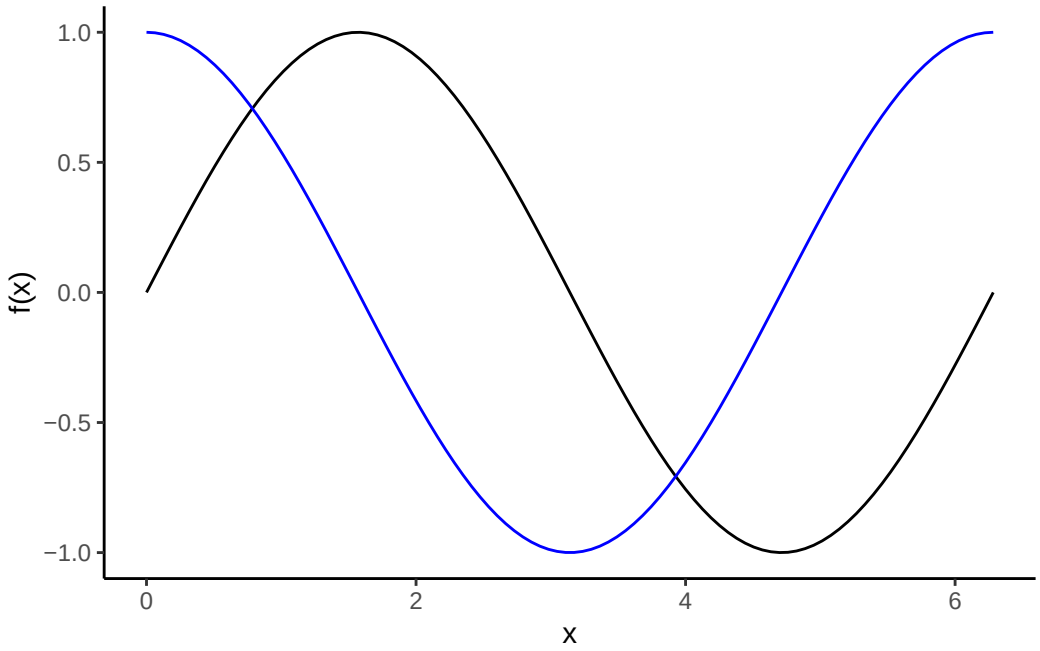
```
Warning: Removed 24 rows containing non-finite outside the scale range
(`stat_smooth()`).
Removed 24 rows containing missing values or values outside the scale range
(`geom_line()`).
```



4.2.1.2 Sine and Cosine Curves

While it isn't the case for this birth data set, if the cyclic pattern is like a sine curve, we could use a sine or cosine curve to model the relationship.

```
time <- seq(0,2*pi,length = 100)
tibble(
  t = time,
  sin = sin(time),
  cos = cos(time)
) %>%
  ggplot(aes(x = t, y = sin)) +
  geom_line() +
  geom_line(aes(x = t, y = cos),color = 'blue') +
  labs(x = 'x',y = 'f(x)') +
  theme_classic()
```



Some properties of sine and cosine curves:

- $\cos(0) = 1$, $\cos(\pi/2) = 0$, $\cos(\pi) = -1$, $\cos(3\pi/2) = 0$ $\cos(2\pi) = 1$
- $\sin(0) = 0$, $\sin(\pi/2) = 1$, $\sin(\pi) = 0$, $\sin(3\pi/2) = -1$ $\sin(2\pi) = 0$
- The period (length of 1 full cycle) of a sine or cosine function is 2π
- We can write a sine curve model as $A\sin(2\pi ft + s)$ where A is the amplitude (peak deviation from 0 instead of 1), f is the frequency (number of cycles that occur each unit of time), t is the time, and s is the shift in the x-axis.

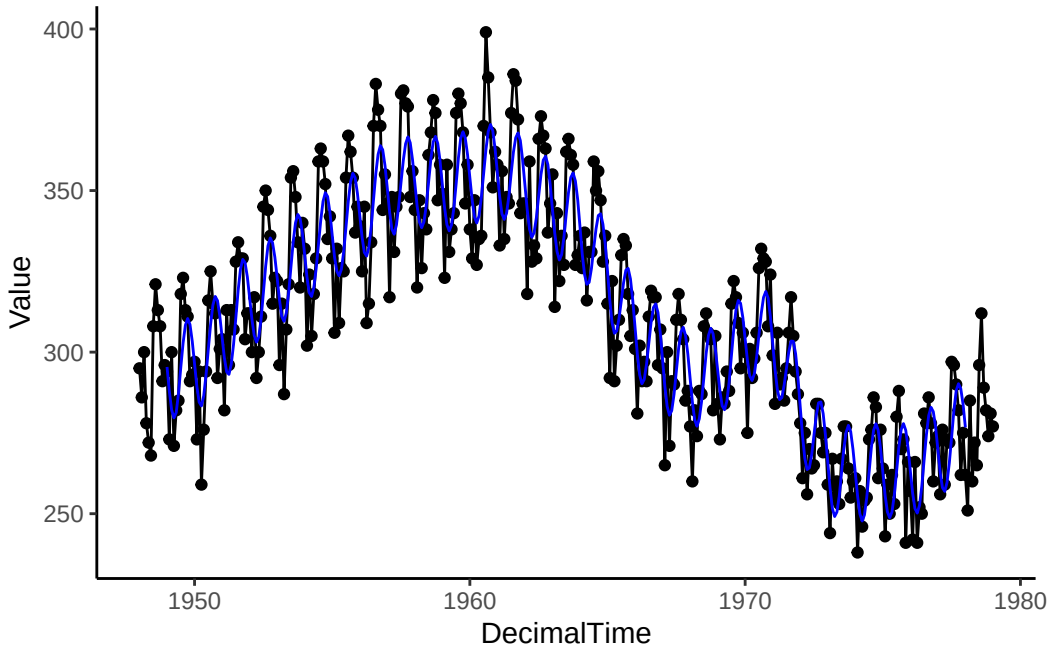
Now, let's try to apply it to the birth data.

```
SineSeason <- lm(Residual ~ sin(2*pi*(DecimalTime) ), data = birthTS) # frequency is one beca
SineSeason
```

```

Call:
lm(formula = Residual ~ sin(2 * pi * (DecimalTime)), data = birthTS)

Coefficients:
                (Intercept)  sin(2 * pi * (DecimalTime))
                   -0.04286                     -14.61751
```

```
# Estimate Trend + Seasonality
birthTS %>%
  mutate(SeasonSine  = predict(SineSeason, newdata = data.frame(DecimalTime = birthTS$Decimal
  ggplot(aes(x = DecimalTime, y = Value)) +
  geom_point() +
  geom_line() +
  geom_line(aes(x = DecimalTime, y = Trend + SeasonSine), color = 'blue') +
  theme_classic()
```

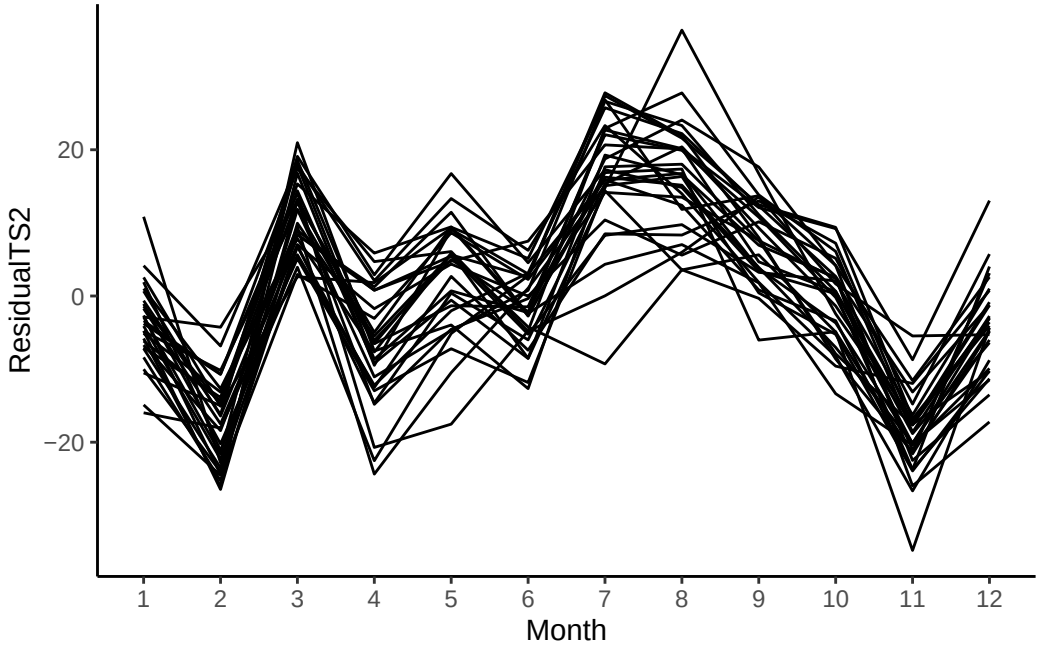
```
Warning: Removed 24 rows containing missing values or values outside the scale range
(`geom_line()`).
```




```
# Residuals
 birthTS %>%
   mutate(ResidualTS2  = Residual - predict(SineSeason, newdata = data.frame(DecimalTime = bi
   ggplot(aes(x = Month, y = ResidualTS2)) +
   geom_line(aes(group = factor(Year))) +
   geom_smooth(se = FALSE) +
   theme_classic()
```

```
Don't know how to automatically pick scale for object of type <ts>. Defaulting
to continuous.
`geom_smooth()` using method = 'loess' and formula = 'y ~ x'
```

```
Warning: Removed 24 rows containing non-finite outside the scale range
(`stat_smooth()`).
Removed 24 rows containing missing values or values outside the scale range
(`geom_line()`).
```

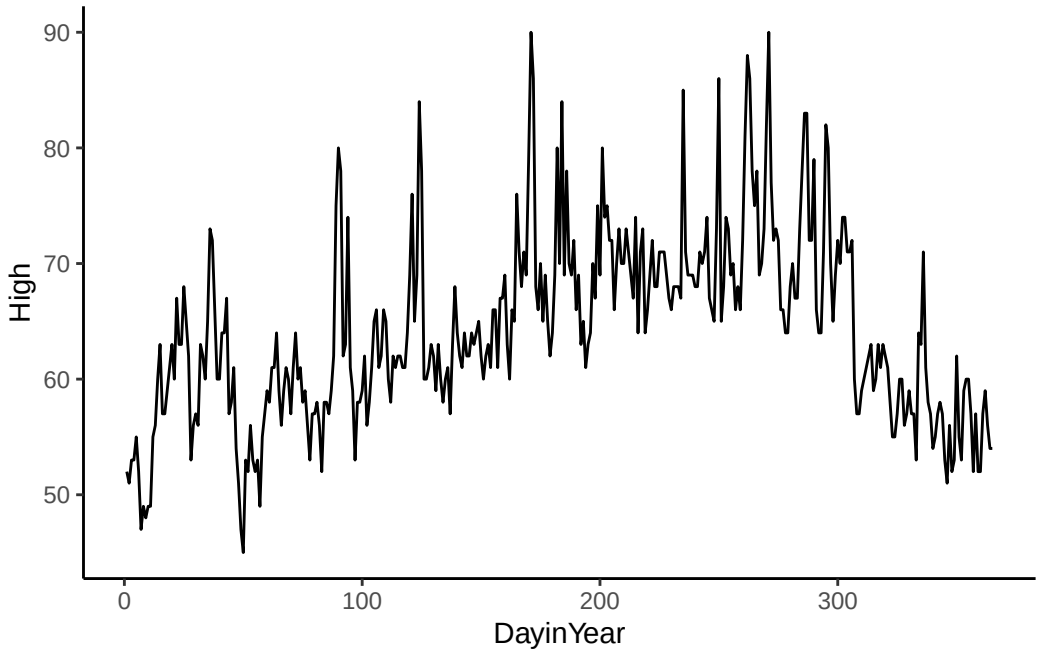


Since we still see a pattern in these residuals, using a sine/cosine curve is not an accurate way to estimate the cyclic pattern in this particular data set.

Data Example 3

Here is the daily temperature data in San Francisco in 2011. We only have one year, but we can see that the cycle shape is closer to a cosine curve. The **period**, the length of one cycle, of a cosine or sine curve is $2 * \pi$, so we need to adjust the period when we fit a cosine or sine curve to our data. In the case below, the period is 365 since our time variable is `DayinYear`, where 1 unit is one day in the year (Jan 1 is one and Dec 31 is 365).

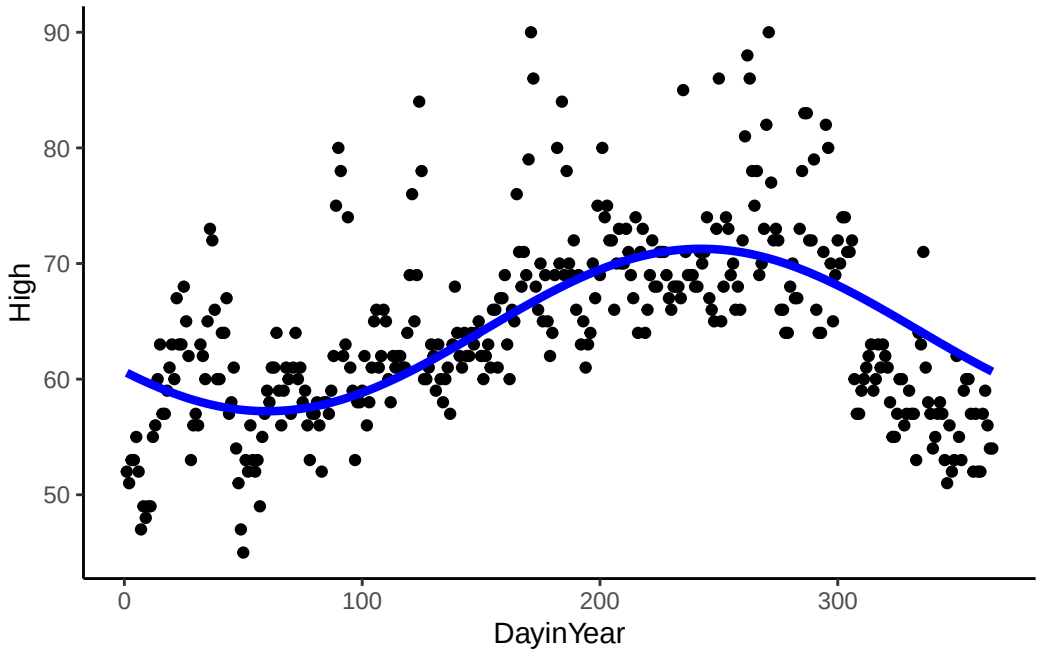
```
load('./data/sfoWeather2011.RData')
sfoWeather <- sfoWeather %>%
  mutate(DayinYear = 1:365)
sfoWeather %>%
  ggplot(aes(x = DayinYear, y = High)) +
  geom_line() +
  theme_classic()
```




```
CosSeason <- lm(High ~ cos(2*pi*(DayinYear-60)/365), data = sfoWeather) # frequency is 1/365
CosSeason
```

```
Call:
lm(formula = High ~ cos(2 * pi * (DayinYear - 60)/365), data = sfoWeather)

Coefficients:
                      (Intercept)  cos(2 * pi * (DayinYear - 60)/365)
                           64.249                              -7.022
```

```
sfoWeather %>%
  mutate(Season = predict(CosSeason)) %>%
  ggplot(aes(x = DayinYear, y = High)) +
  geom_point() +
  geom_line(aes(x = DayinYear, y = Season), color = 'blue', size = 1.5) +
  theme_classic()
```



To recap:

For a cosine function,

$$A\cos(Bx + C) + D$$

the period is $\frac{2*\pi}{|B|}$, the **amplitude** is A (the coefficient in our linear model), the **phase shift** is C (included in our model above) and the **vertical shift** is D (intercept in our linear model).

4.2.2 Nonparametric Approaches

Similar to the trend, we could use local regression or moving averages to estimate the trend and seasonality as long as we define the local area small enough to pick up the cycles.

The disadvantage of this approach is that we will not be able to use the estimate to make predictions in the future because we do not learn general patterns of trend and seasonality.

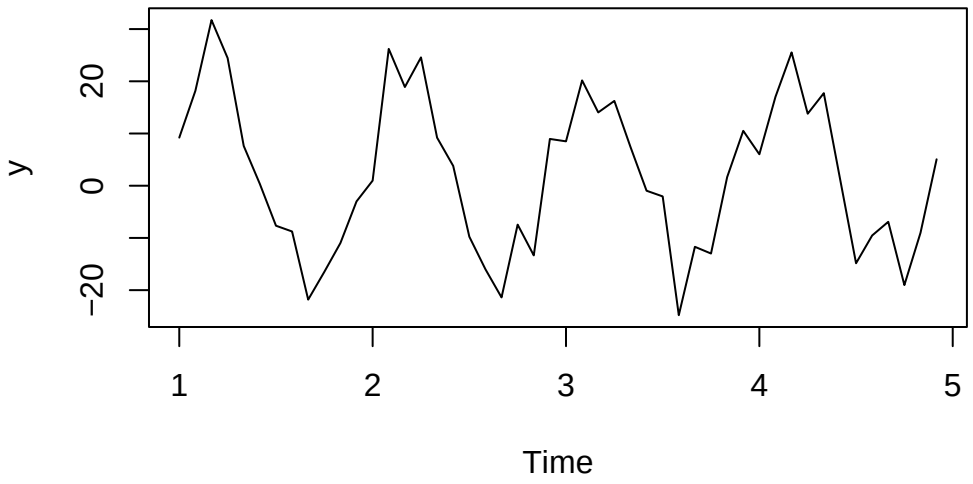
4.2.2.1 Differencing to Remove Seasonality

Like removing a trend, we can use differencing to remove seasonality if we don't need to estimate it. Rather than a difference between neighboring observations, we will take differences between observations that are approximately one period apart. The period, length of one cycle,

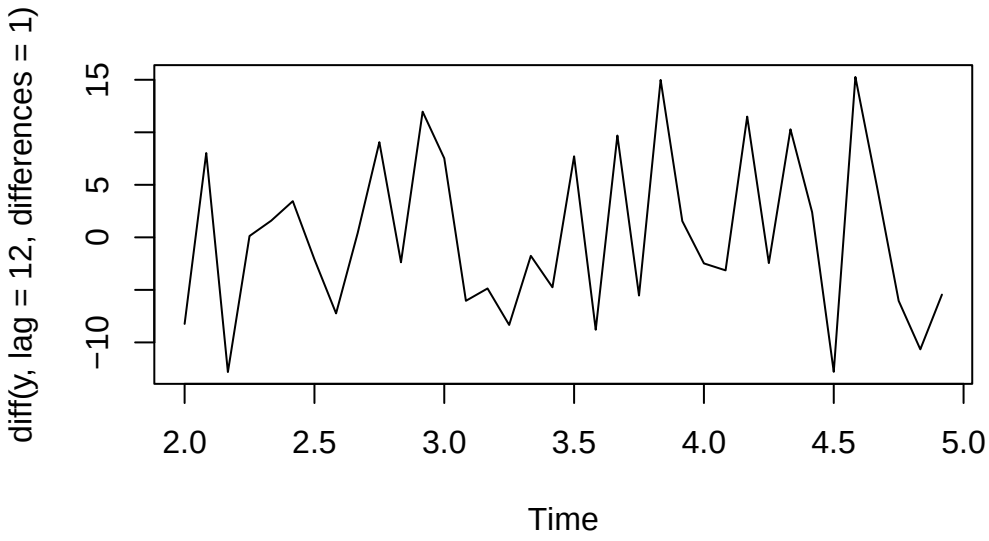
is k observations such that y_t is similar to y_{t-k} and y_{t-2k}, then we'll take differences between observations that are k observations apart.

For example, if you have monthly data and an annual cycle, you'd want to take differences between observations that are 12 lags apart (12 months apart). See the simulated monthly data below.

```
t <- 1:48
y <- ts(2 + 20*sin(2*pi*t/12) + rnorm(48,sd = 5), frequency=12)
plot(y)
```

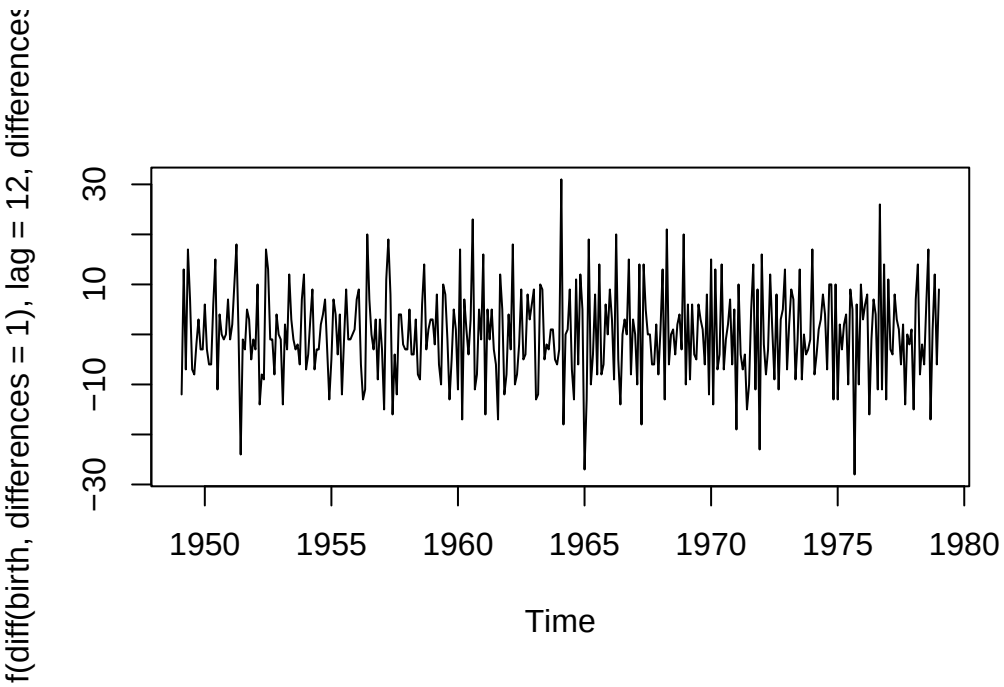



```
plot(diff(y, lag = 12, differences = 1)) #first order difference of lag 12
```



Let's try this approach with the birth data since we saw annual seasonality by month. First, we take first differences of neighboring values to remove the trend. Then we'll take the first differences of lag 12 to remove the seasonality. No obvious trends or seasonality remain in the data, and it is centered around 0.

```
plot(diff(diff(birth,differences = 1), lag = 12, differences = 1))
```



What is left over, the random errors, are what we need to worry about now.

We'll first talk about how we might model the errors from time series data and then talk about translating these ideas to other types of correlated data.

5 Time Series Data

We will start with **univariate time series,** which is defined as a sequence of measurements of the same characteristic collected over time at regular intervals. We'll notate this time series as

$$y_t, \text{ for } t = 1, 2, ..., n$$

where t indexes time in a chosen unit.

For example, if data are collected yearly, $t = 1$ indicates the first year, etc. If data are collected every day, then $t = 1$ indicates the first day of data collection. In other settings, we may have monthly or hourly measurements such that t indices months or hours from the beginning of the study period.

There are two main approaches to analyzing time series data: time-domain and frequency-domain analysis. In this class, we'll focus on the time-domain analysis. For frequency domain approaches, check out the Other Time Series References.

5.1 R: Time Series Objects

As you've seen, R has a special format (an object class) for time series data called a `ts` object. If the data are not already in that format, you can create a `ts` object with the `ts()` function. Besides the data, it requires two pieces of information.

The first is `frequency`. The name is a bit of a misnomer because it does not refer to the number of cycles per unit of time but rather the number of observations/samples per cycle.

We typically work with one day or one year as the cycle. So, if the data were collected each hour of the day, then `frequency = 24`. If the data is collected annually, `frequency = 1`; quarterly data should have `frequency = 4`; monthly data should have `frequency = 12`; weekly data should have `frequency = 52`.

The second piece of information is `start,` and it specifies the time of the first observation in terms of (cycle, frequency). In most use cases, it is (day, hour), (year, month), (year, quarter), etc. So, for example, if the data were collected monthly beginning in November of 1969, then `frequency = 12` and `start = c(1969, 11)`. If the data were collected annually, you specify

start as a scalar (e.g., `start = 1991`) and omit frequency (i.e., R will set `frequency = 1` by default).

This is a useful format for us because the `plot.ts()` or `plot(ts())` functions will automatically correctly label the x-axis according to time. Additionally, there are special functions that work on ts objects such as `decompose()` that visualizes the basic decomposition of the series into trend, season, and error.

If you have multiple characteristics or variables measured over time, we could combine them in one `ts` object by considering the intersection (overlapping periods) with `ts.intersect()` or the union (all times) of the two time series with `ts.union()`.

As you've seen above, it may also be useful to have data in a `data.frame()` format instead of a `ts` object if you want to use `ggplot()` or `lm()`. You should be familiar with both data formats to go back and forth as necessary.

5.2 ACF: Autocorrelation Function

As we discussed earlier, the key feature of correlated data is the covariance and correlation of the data. We have decomposed the data for time series into trend, seasonality, and noise or error. We will now work to model the dependence in the errors.

Remember: For a stationary random process, Y_t (constant mean, constant variance), the autocovariance function is **only** dependent on the different in time, which we will refer to as the lag h, so

$$\Sigma(h) = Cov(Y_t, Y_{t+|h|}) = E[(Y_t - \mu_t)(Y_{t+|h|} - \mu_{t+|h|})]$$

for any time t.

Most time series are not stationary because they do not have a constant mean across time. By removing the trend and seasonality, we attempt to get errors (also called residuals) with a constant mean around 0. We'll discuss the constant variance assumption later.

If we assume that the residuals, $y_1, ..., y_n$, are generated from a stationary process, we can estimate the autocovariance function with the **sample autocovariance function** (ACVF),

$$c_h = \frac{1}{n} \sum_{t=1}^{n-|h|} (y_t - \bar{y})(y_{t+|h|} - \bar{y})$$

where $\bar{y}$ is the sample mean, averaged across time, because we are assuming the mean is constant.

There are a few useful properties of the ACVF function. For any sequence of observations, $y_1, ..., y_n$,

1. $c_h = c_{-h}$
2. $c_0 \geq 0$ and $|c_h| \leq c_0$

The **sample autocorrelation function** (ACF) is the covariance divided by the variance, also known as the covariance for lag 0,

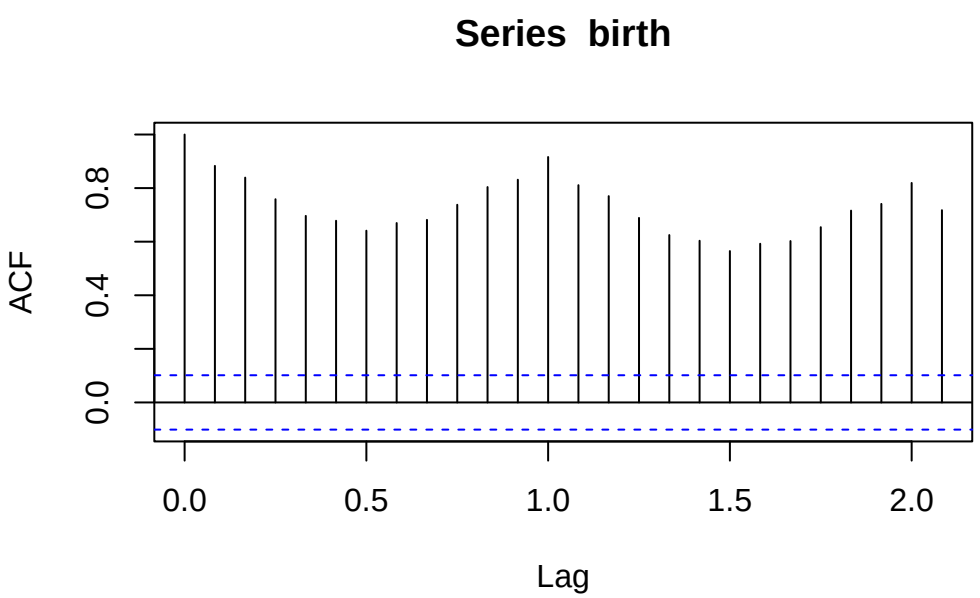
$$r_h = \frac{c_h}{c_0} \text{ for } c_0 > 0$$

There are a few useful properties of the ACF function. For any sequence of observations, $y_1, ..., y_n$,

1. $r_h = r_{-h}$
2. $r_0 = 1$ and $|r_h| \leq 1$

We expect a fairly high correlation between observations with large lags for a non-stationary series with a trend and seasonality. *This high correlation pattern typically indicates that you need to deal with trend and seasonality.*

```
acf(birth) # acf works well on ts objects (lag 1.0 is one year since it knows that one year
```

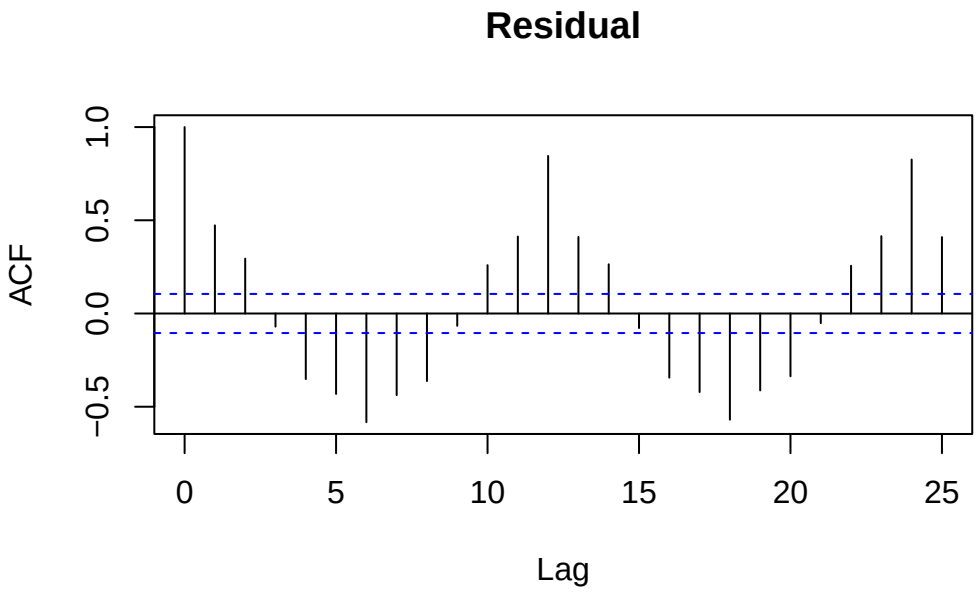


- Sample ACF for Trend: Very slow decay to zero
- Sample ACF for Trend + Seasonality: Very slow decay to zero + Periodic

We would like to see the autocorrelation of the random errors after removing the trend and the seasonality. Let's look at the sample autocorrelation of the residuals from the model with a moving average filter estimated trend and monthly averages to account for seasonality.

```
birthTS <- data.frame(
  Year = floor(as.numeric(time(birth))), # time() works on ts objects
  Month = as.numeric(cycle(birth)), # cycle() works on ts objects
  Value = as.numeric(birth)) %>%
  mutate(DecimalTime = Year + (Month-1)/12)

fltr <- c(0.5, rep(1, 23), 0.5)/24 # weights for moving average
birthTrend <- stats::filter(birth, filter = fltr, method = 'convo', sides = 2) # use the ts c

birthTS <- birthTS %>%
  mutate(Trend = birthTrend) %>%
  mutate(Residual = Value - Trend)
```

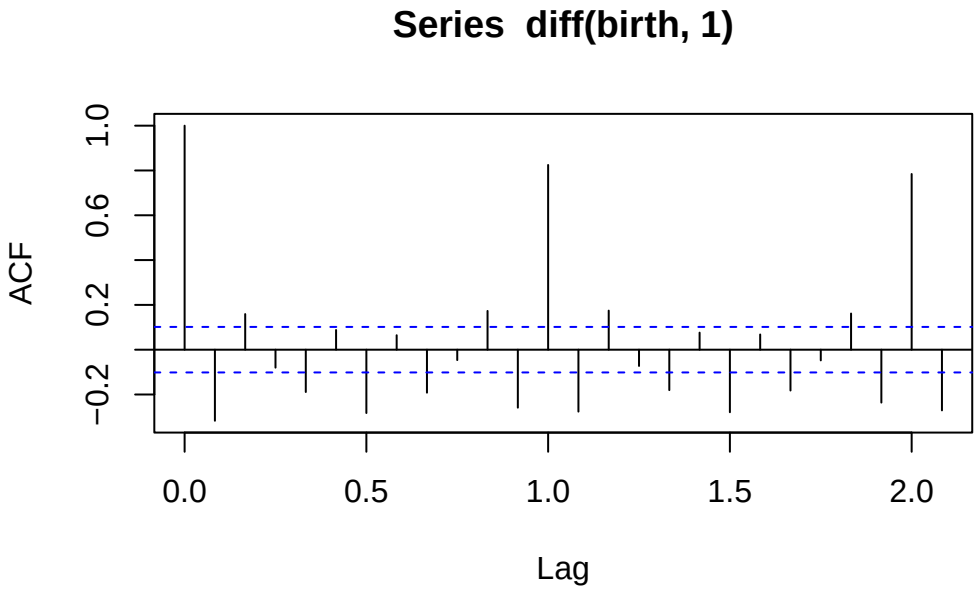
```
birthTS %>%
  dplyr::select(Residual) %>%
  dplyr::filter(complete.cases(.)) %>%
  acf() # if the data is not a ts object, lags will be in terms of the index
```



The autocorrelation has to be 1 for lag 0 because $r_0 = c_0/c_0 = 1$. Note that the lags are in terms of months here because we did not specify a `ts()` object. We see the autocorrelation slowly decreasing to zero.

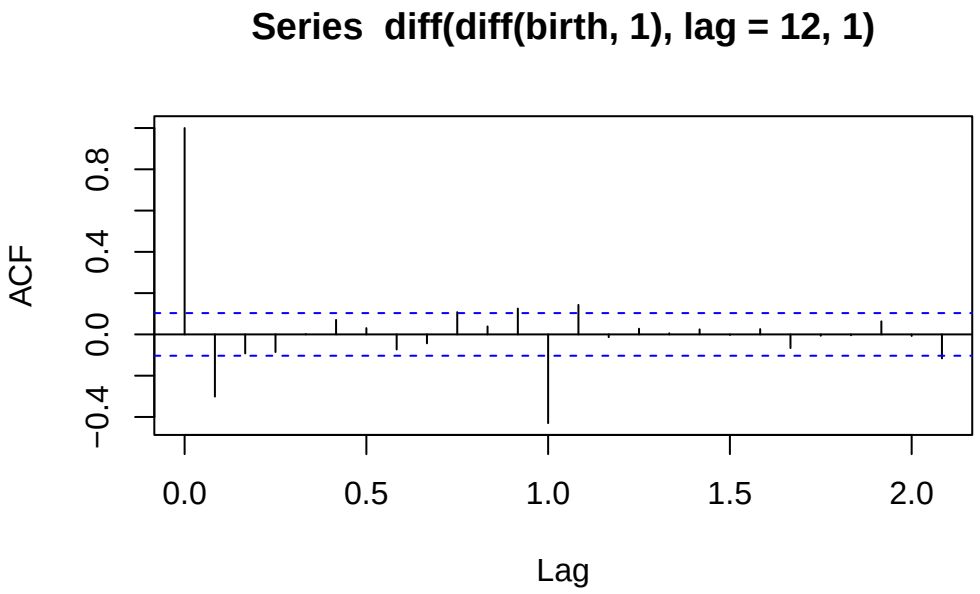
What about the ACF of birth data after differencing?

```r
acf(diff(birth, 1)) # one difference to remove trend
```



- Sample ACF for Seasonality: Periodic

```r
acf(diff(diff(birth,1), lag = 12, 1)) # differences to remove trend and seasonality
```



Note that the lags here are in terms of years (Lag = 1 on the plot refers to $h = 12$) because the data is saved as a `ts` object. In the first plot (after only first differencing), we see an

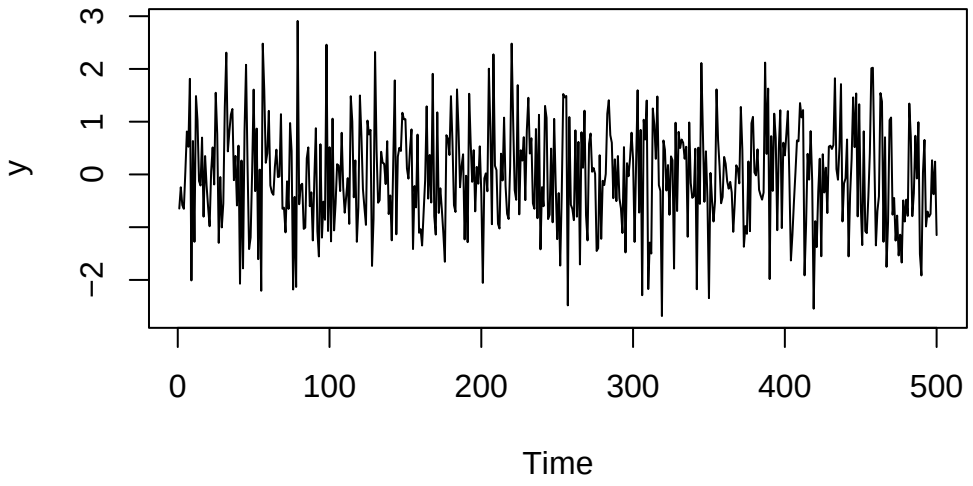
autocorrelation of about 0.5 for observations one year apart. This suggests that there may still be some seasonality to be accounted for. In the second plot, after we also did seasonal differencing for lag = 12, that decreases a bit (and becomes a bit negative).

Now, do you notice the blue, dashed horizontal lines?

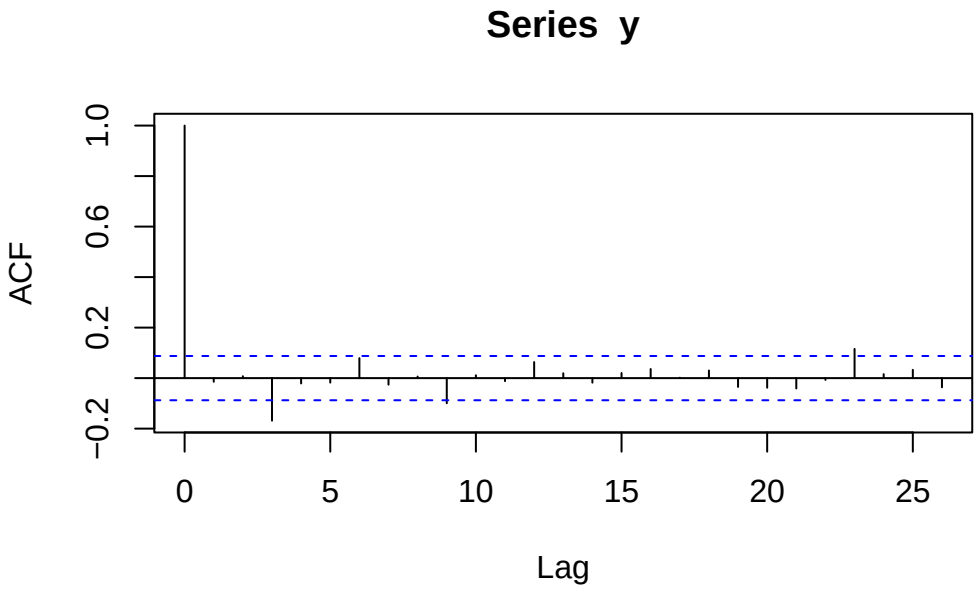
These blue horizontal lines are guide lines for us. If the random process is **white noise** such that the observations are *independent and identically distributed* with a constant mean (of 0) and constant variance σ^2, we'd expected the sample autocorrelation to be within these dashed horizontal lines ($\pm 1.96/\sqrt{n}$) roughly 95% of the time. So if the random process were independent **white noise**, we'd expect 95% of the ACF estimates for $h \neq 0$ to be within the blue lines with no systematic pattern.

See an example of Gaussian white noise below and its sample autocorrelation function.

```
y <- ts(rnorm(500))
plot(y)
```




```
acf(y)
```



- Sample ACF for White Noise: Zero except at lag 0

Many stationary time series have recognizable ACF patterns. We'll get familiar with those in a moment.

However, most time series we encounter in practice are not stationary. Thus, our first step will always be to deal with the trend and seasonality. Then we can model the (hopefully stationary) residuals with a stationary model.

5.3 Modeling the Errors

Up to this point, we have decomposed our time series into a trend component, a seasonal component, and a random component,

$$Y_t = \underbrace{f(x_t)}_{\text{trend}} + \underbrace{s(x_t)}_{\text{seasonality}} + \underbrace{\epsilon_t}_{\text{noise}}$$

With observed data, we can work with the residuals after removing the trend and seasonality,

$$e_t = y_t - \underbrace{\hat{f}(x_t)}_{\text{trend}} - \underbrace{\hat{s}(x_t)}_{\text{seasonality}}$$

which is approximately the true error process,

$$\epsilon_t = Y_t - \underbrace{f(x_t)}_{\text{trend}} - \underbrace{s(x_t)}_{\text{seasonality}}$$

For the following time series models (AR, MA, and ARMA), we will use Y_t to represent the random variable for the residuals, e_t.

In the next five sections, we'll discuss the theory of these three models. To see a real data example and the R code to fit the models, go to the Real Data Example section.

5.4 Autoregressive Models

The first model we will consider for a stationary process is an **autoregressive model**, which involves regressing current observations on values in the past. So this model regresses on itself (the 'auto' part of 'autoregressive').

5.4.1 AR(1) Model

One simple model for correlated errors is an **autoregressive order 1** or AR(1) model. The model is stated

$$Y_t = \delta + \phi_1 Y_{t-1} + W_t$$

where $W_t \overset{iid}{\sim} N(0, \sigma_w^2)$ is **independent** Gaussian white noise with mean 0 and constant variance, σ_w^2. This model is weakly stationary if $|\phi_1| < 1$.

Note: We will often think of $\delta = 0$ because our error process is often on average 0.

Properties

Under this model, the expected value (mean) of Y_t is

$$E(Y_t) = \mu = \frac{\delta}{1 - \phi_1}$$

and the variance is

$$Var(Y_t) = \frac{\sigma_w^2}{1 - \phi_1^2}$$

The correlation between observations h lags (time periods) apart is

$$\rho_h = \phi_1^h$$

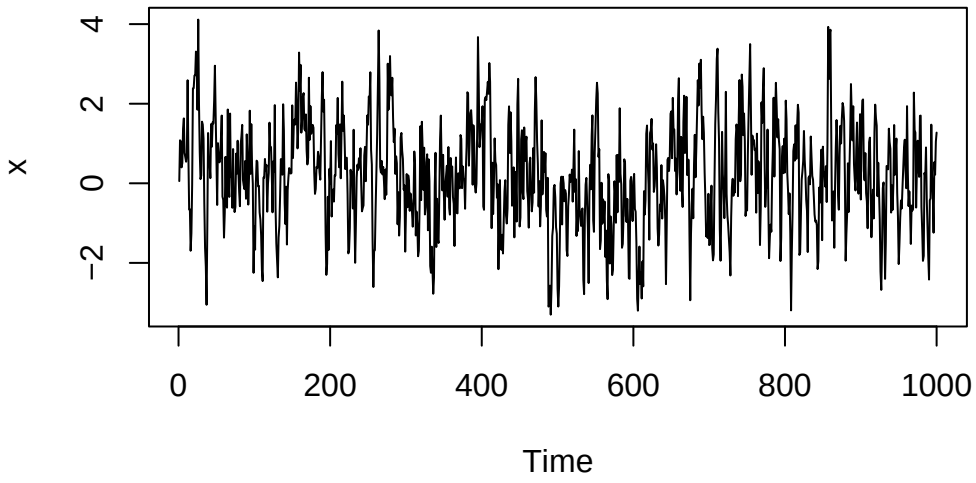
Derivations for these properties are available in the Appendix at the end of this chapter.

- Sample ACF for AR(1): Decays to zero exponentially

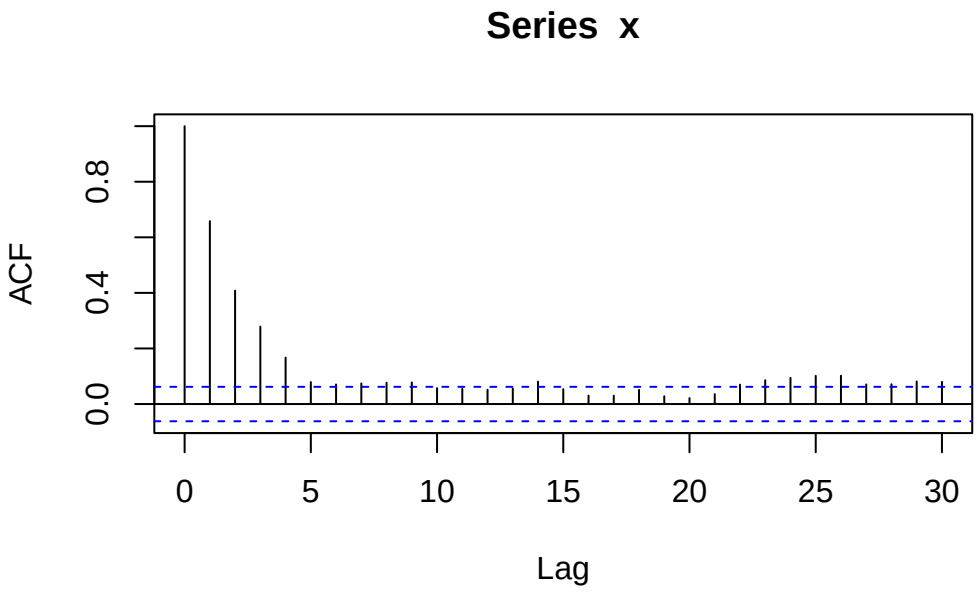
Simulated Data Example

We will generate data from an AR(1) model with $\phi_1 = 0.64$. For a positive ϕ_1, the autocorrelation exponentially decreases to 0 as the lag increases, $\rho_h = (0.64)^h$.

```
# Simulate an ar(1) process
# x = 0.05 + 0.64*x(t-1) + e
# Create the vector x
x   <- vector(length=1000)

#simulate the white noise errors
e   <- rnorm(1000)

#Set the coefficient
beta    <- 0.64

# set an initial value
x[1]    <- 0.055

 #Fill the vector x
 for(i in 2:length(x))
    {
        x[i]    <- 0.05 + beta*x[i-1] + e[i]
 }
x <- ts(x)
plot(x)
```



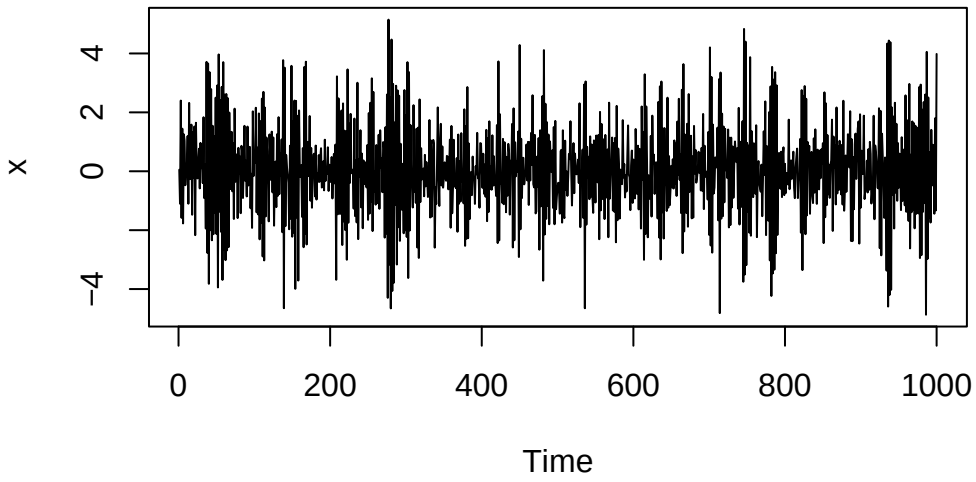

```
acf(x)
```



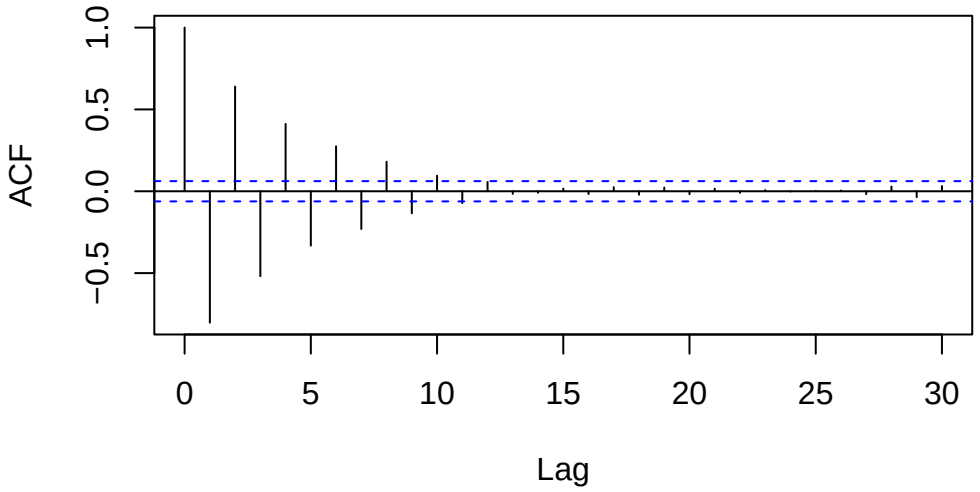
Now, we'll try $\phi_1 = -0.8$. For a negative ϕ_1, the autocorrelation exponentially decreases to 0 as the lag increases, but the signs of the autocorrelation alternate between positive and negative ($\rho_h = (-0.8)^h$).

```
# Simulate an ar(1) process
# x = 0.05 - 0.8*x(t-1) + e
# Create the vector x
x   <- vector(length=1000)
```

```r
#simulate the white noise errors
e   <- rnorm(1000)

#Set the coefficient
beta    <- -0.8

# set an initial value
x[1]    <- 0.05

#Fill the vector x
 for(i in 2:length(x))
    {
        x[i]    <- 0.05 + beta*x[i-1] + e[i]
 }
x <- ts(x)
plot(x)
```




```r
acf(x)
```



Remember that $|\phi_1| < 1$ in order for an AR(1) process to be stationary (constant mean, constant variance, autocovariance only depends on lags).

5.4.2 Random Walk

If $\phi_1 = 1$ and $\delta = 0$ (a simplifying assumption), the AR(1) model becomes a well-known process called a **random walk**. The model for a random walk is

$$Y_t = Y_{t-1} + W_t$$

where $W_t \overset{iid}{\sim} N(0, \sigma_w^2)$.

If we start with $t = 1$, then we start with random white noise,

$$Y_1 = W_1$$

The next observation is the past observation plus noise,

$$Y_2 = Y_1 + W_2 = W_1 + W_2$$

The next observation is the past observation plus noise,

$$Y_3 = Y_2 + W_3 = (W_1 + W_2) + W_3$$

And so forth,

$$Y_n = Y_{n-1} + W_n = \sum_{i=1}^{n} W_i$$

So a random walk is the sum of random white noise random variables.

Properties

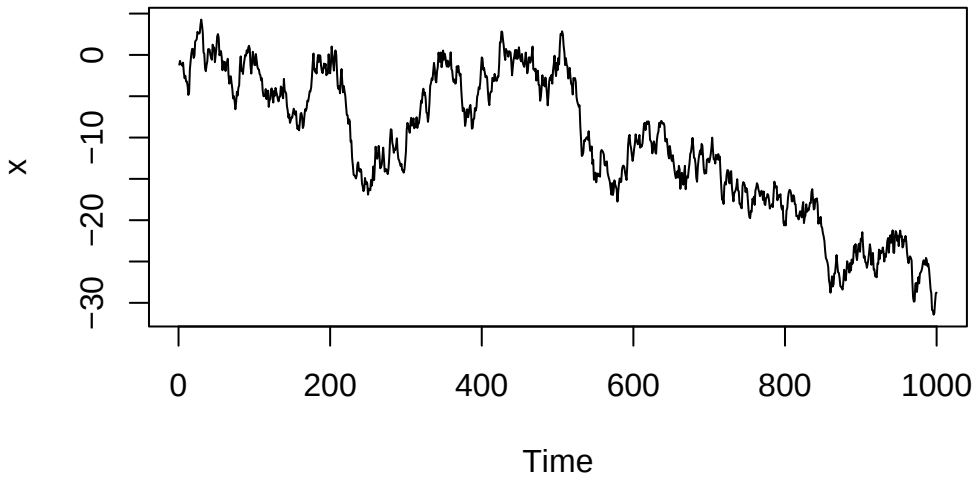
This random walk process is NOT weakly stationary because the variance is not constant as it is a function of time, t,

$$Var(Y_t) = Var(\sum_{i=1}^{t} W_i) = \sum_{i=1}^{t} Var(W_i) = t\sigma_w^2$$

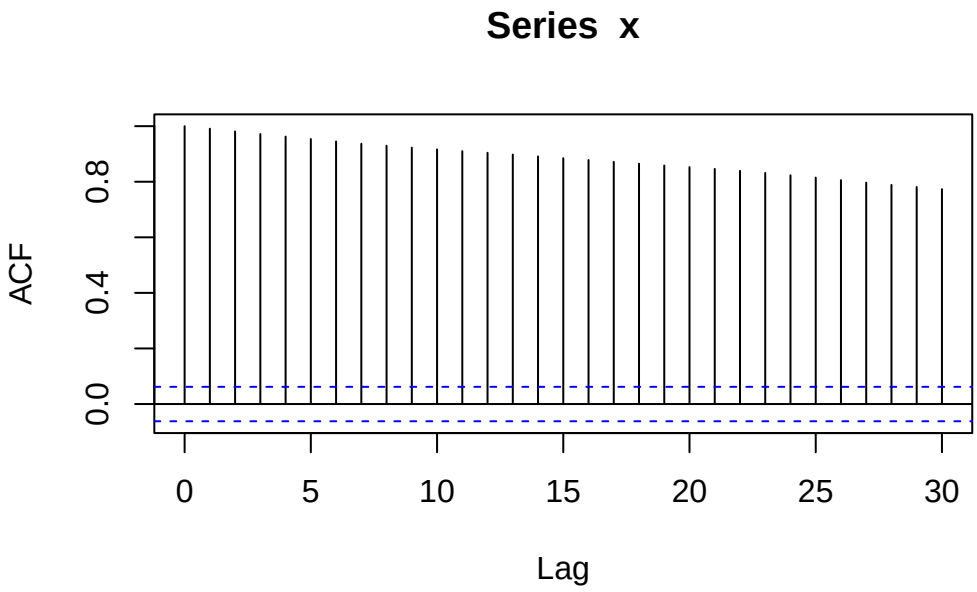
Simulated Data Example

When we generate many random walks, we'll notice greater potential variability in values later in time. Additionally, you'll see high autocorrelation at higher lags, indicating a high range of dependence. Two observations that are far in observation time are still highly correlated.

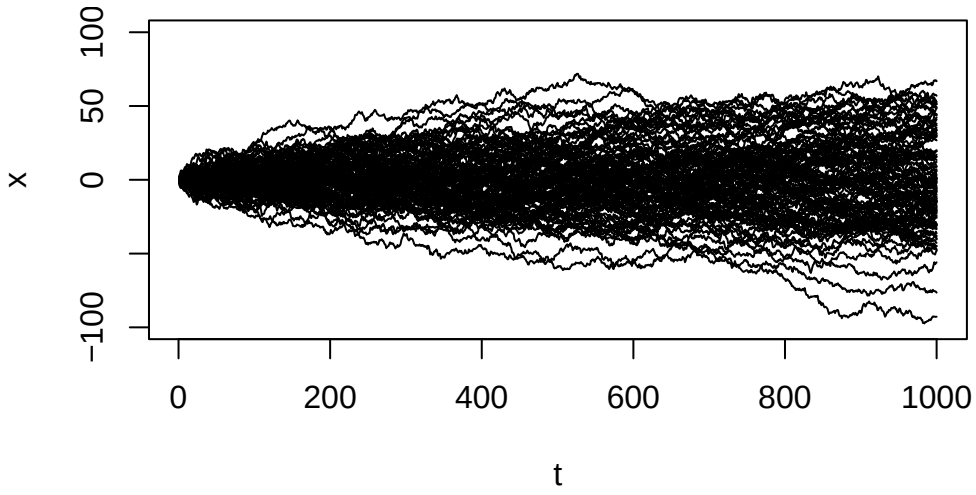
```
# Simulate one random walk process
# x = x(t-1) + e

#simulate the white noise errors
e   <- rnorm(1000)

# cumulative sum based on recursive process above
x <- cumsum(e)

x <- ts(x)
plot(x)
```




```
acf(x)
```



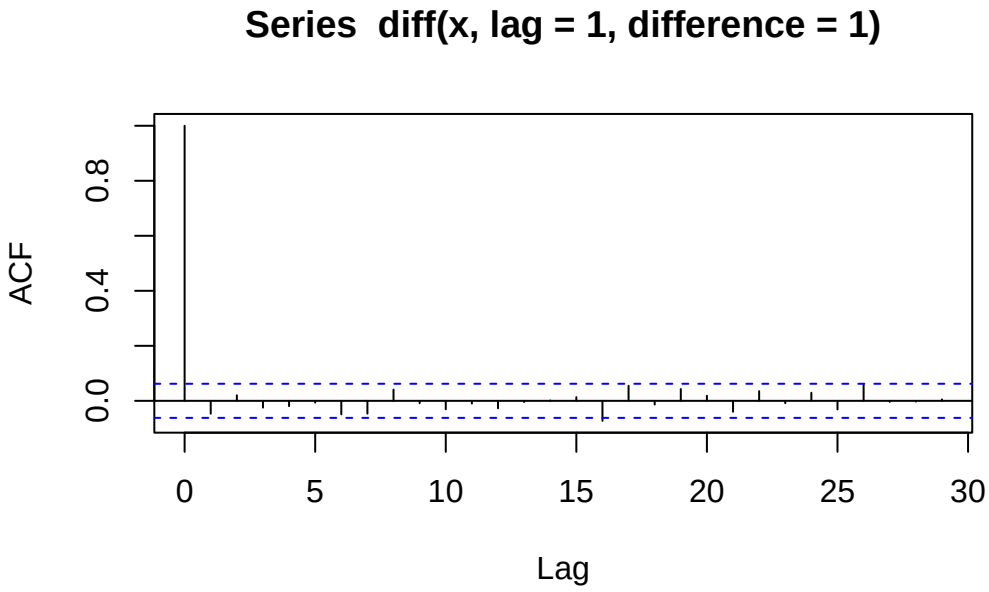

```
# Simulate many random walk processes
plot(1:1000,rep(1,1000),type='n',ylim=c(-100,100),xlab='t',ylab='x')
for(i in 1:100){
  e <- rnorm(1000)
  x <- cumsum(e)
  x <- ts(x)
  lines(ts(x))
}
```



If we were to take a first-order difference of a random walk, $Y_t - Y_{t-1}$, we could remove the trend and end up with independent white noise.

$$Y_t - Y_{t-1} = W_t$$

```
acf(diff(x, lag = 1, difference = 1))
```



5.4.3 AR(p) Model

We could generalize the idea of an AR(1) model to allow an observation to be dependent on the past p previous observations. A **autoregression process of order p** is modeled as

$$Y_t = \delta + \phi_1 Y_{t-1} + \phi_2 Y_{t-2} + \cdots + \phi_p Y_{t-p} + W_t$$

where $\{W_t\}$ is independent Gaussian white noise, $W_t \overset{iid}{\sim} N(0, \sigma_w^2)$. We will typically let $\delta = 0$, assuming we have removed the trend and seasonality before applying this model.

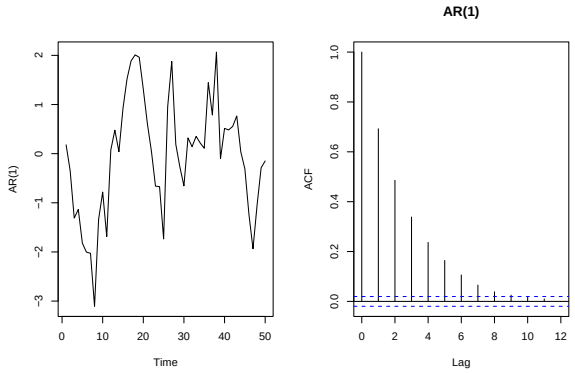
Properties

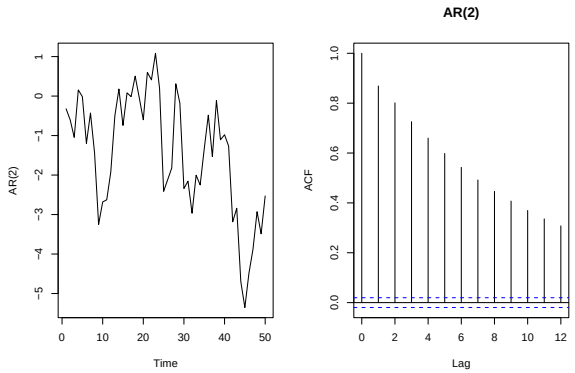
We'll look at the expected value, variance, and covariance in the AR(p) as MA(∞) section after discussing moving average models. We'll also discuss when the AR(p) model is stationary. I know you can't wait!

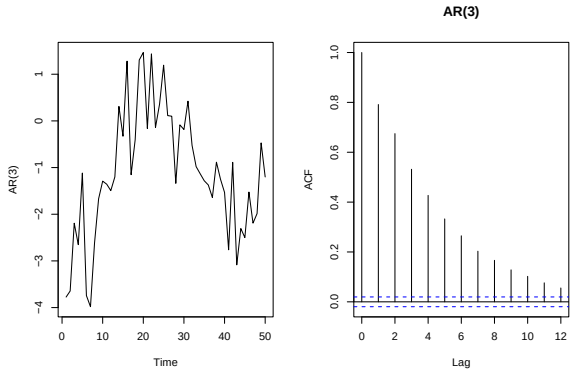
Simulated Data Example

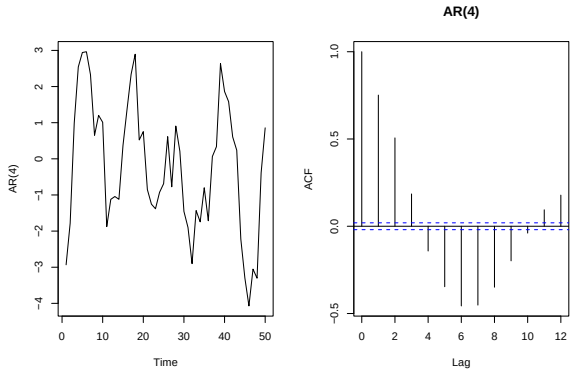
Let's see a few simulated examples and look at the autocorrelation functions.

```
set.seed(123)
## the 4 AR coefficients
ARp <- c(0.7, 0.2, -0.1, -0.3)
## empty list for storing models
AR.mods <- list()
## loop over orders of p
for (p in 1:4) {
    ## assume SD=1, so not specified
    AR.mods[[p]] <- arima.sim(n = 10000, list(ar = ARp[1:p]))
}

## set up plot region
par(mfrow = c(4, 2))
## loop over orders of p
for (p in 1:4) {
    plot.ts(AR.mods[[p]][1:50], ylab = paste("AR(", p, ")", sep = ""))
    acf(AR.mods[[p]], lag.max = 12,main=paste("AR(", p, ")", sep = ""))
}
```


AR(1)
AR(1)
Time
ACF
Lag


AR(2)
AR(2)
Time
ACF
Lag


AR(3)
AR(3)
Time
ACF
Lag


AR(4)
AR(4)
Time
ACF
Lag

- Sample ACF for AR(p): Decay to zero

5.5 Moving Average Models

In contrast to the autoregressive model, we will consider a model that considers the current outcome value to be a function of the current and past noise (rather than the past outcomes). This is a subtle difference but you'll see that a moving average (MA) model is very different than an AR model, but we'll show how they are connected.

Note: This is different from the moving average filter that we used to estimate the trend.

5.5.1 MA(1) Model

A **moving average process of order 1** or MA(1) model is a weighted sum of a current random error plus the most recent error, and can be written as

$$Y_t = \delta + W_t + \theta_1 W_{t-1}$$

where $\{W_t\}$ is independent Gaussian white noise, $W_t \overset{iid}{\sim} N(0, \sigma_w^2)$.

Like AR models, we often let $\delta = 0$.

Properties

Unlike AR(1) processes, MA(1) processes are always weakly stationary. We see the variance is constant (not a function of time),

$$Var(Y_t) = Var(W_t + \theta_1 W_{t-1}) = \sigma_w^2(1 + \theta_1^2)$$

Let's look at the autocorrelation function of an MA(1) process.

To derive the autocorrelation function, let's start with the covariance at lag 1. Plug in the model for Y_t and Y_{t-1} and use the properties to simplify the expression,

$$\begin{aligned}
\Sigma_Y(1) = Cov(Y_t, Y_{t-1}) = Cov(W_t + \theta_1 W_{t-1}, W_{t-1} + \theta_1 W_{t-2}) \\
= Cov(W_t, W_{t-1}) + Cov(W_t, \theta_1 W_{t-2}) + Cov(\theta_1 W_{t-1}, W_{t-1}) + Cov(\theta_1 W_{t-1}, \theta_1 W_{t-2}) \\
= 0 + 0 + \theta_1 Cov(W_{t-1}, W_{t-1}) + 0 \\
= \theta_1 Var(W_{t-1}) \\
= \theta_1 Var(W_t) = \theta_1 \sigma_w^2
\end{aligned}$$

because $ W_t$'s are independent of each other.

For larger lags $k > 1$,

$$\Sigma_Y(k) = Cov(Y_t, Y_{t-k}) = Cov(W_t + \theta_1 W_{t-1}, W_{t-k} + \theta_1 W_{t-k-1})$$
$$= Cov(W_t, W_{t-k}) + Cov(W_t, \theta_1 W_{t-k-1}) + Cov(\theta_1 W_{t-1}, W_{t-k}) + Cov(\theta_1 W_{t-1}, \theta_1 W_{t-k-1})$$

$$= 0 \text{ if } k > 1$$

because W_t's are independent of each other.

Now, the correlation is derived as a function of the covariance divided by the variance (which we found above),

$$\rho_1 = \frac{Cov(Y_t, Y_{t-1})}{Var(Y_t)} = \frac{\Sigma_Y(1)}{Var(Y_t)} = \frac{\theta_1 \sigma_w^2}{\sigma_w^2(1+\theta_1^2)} = \frac{\theta_1}{(1+\theta_1^2)}$$

and

$$\rho_k = \frac{Cov(Y_t, Y_{t-k})}{Var(Y_t)} = \frac{\Sigma_Y(k)}{Var(Y_t)} = 0 \text{ if } k > 1$$

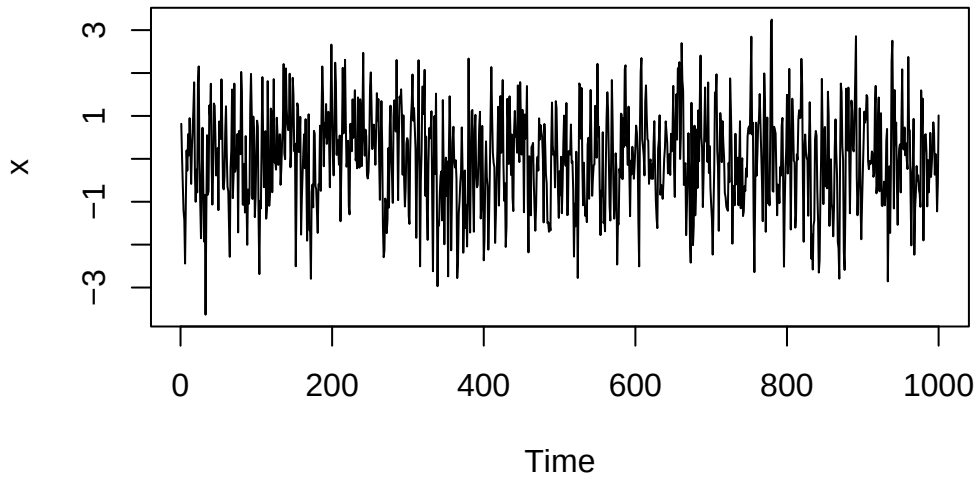
So the autocorrelation function is non-zero at lag 1 for an MA(1) process and zero otherwise. Keep this in mind as we look at sample ACF functions.

- Sample ACF for MA(1): Zero for lags > 1

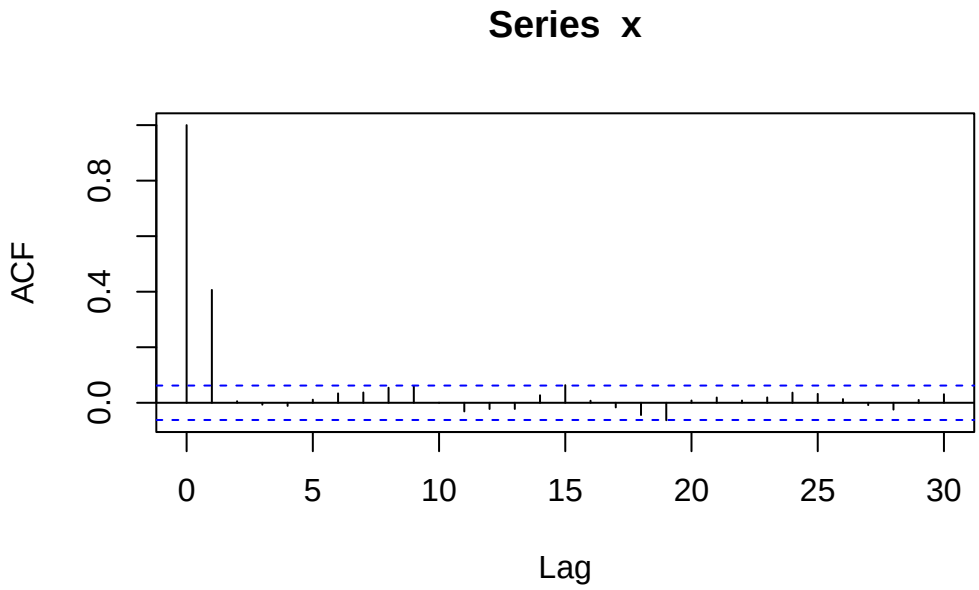
Simulated Data Example

```
# Simulate a MA1 process
# x = e + theta1 e(t-1)
# Create the vector x
x   <- vector(length=1000)
theta1 <- 0.5

#simulate the white noise errors
e   <- rnorm(1000)

x[1] <- e[1]

#Fill the vector x
 for(i in 2:length(x))
    {
```

```
        x[i]    <- e[i] + theta1*e[i-1]
 }

x <- ts(x)
plot(x)
```




```
acf(x)
```



Notice how the autocorrelation = 1 at lag 0 and then around 0.4 at lag 1. The autocorrelation estimate is in between the blue lines for the other lags, so they are practically zero.

Invertibility

No restrictions on θ_1 are needed for an MA(1) process to be stationary. However, imposing restrictions on θ_1 is generally desirable to ensure the MA process is invertible.

For an MA process to be **invertible**, we must be able to write it as an AR(∞) process that converges. We'll talk more about this soon.

We want to restrict ourselves to only invertible processes because of the non-uniqueness of the ACF. Let's imagine these two processes,

$$\begin{aligned} A: \quad & Y_t = W_t + \theta_1 W_{t-1} \\ B: \quad & Y_t = W_t + \frac{1}{\theta_1} W_{t-1} \end{aligned}$$

Let's show that they have the same ACF. We can use our derivations about the MA(1) process. For process B,

$$\rho_1 = \frac{1/\theta_1}{(1+1/\theta_1^2)} = \frac{1}{\theta_1(1+1/\theta_1^2)} = \frac{1}{\theta_1(\frac{\theta_1^2+1}{\theta_1^2})} = \frac{1}{\theta_1}\frac{\theta_1^2}{\theta_1^2+1} = \frac{\theta_1}{\theta_1^2+1}$$

which is the same autocorrelation as process A.

Now, let's **invert** process A by rewriting it for W_t as a function of Y_t,

$$W_t = Y_t - \theta_1 W_{t-1}$$

and now let's plug in the model for W_{t-1},

$$W_t = Y_t - \theta_1(Y_{t-1} - \theta_1 W_{t-1}) = Y_t - \theta_1 Y_{t-1} - \theta_1^2 W_{t-1}$$

and if you keep going, you get an infinite sum,

$$W_t = Y_t - \theta_1(Y_{t-1} - \theta_1 W_{t-1}) = Y_t - \theta_1 Y_{t-1} - \theta_1^2 Y_{t-2} - \cdots$$

or equivalently, an autoregressive model of order ∞ that is stationary if $|\theta_1| < 1$,

$$Y_t = W_t + \theta_1 Y_{t-1} + \theta_1^2 Y_{t-2} + \cdots$$

Additionally, this infinite sum only converges to a finite value when $|\theta_1| < 1$.

If you invert the second process (process B), this infinite sum will not converge if $|\theta_1| < 1$,

$$W_t = Y_t - 1/\theta_1 Y_{t-1} - 1/\theta_1^2 Y_{t-2} - \cdots$$

Thus, the first process is **invertible**, and the second is not if $|\theta_1| < 1$. When we can invert the process, notice that we have written an MA(1) process as an **autoregressive process of infinite order**.

5.5.2 MA(q) Model

A **moving average process of order q** or MA(q) model is a weighted sum of a current random error plus q most recent errors, and can be written as

$$Y_t = \delta + W_t + \theta_1 W_{t-1} + \theta_2 W_{t-2} \cdots + \theta_q W_{t-q}$$

where $\{W_t\}$ is independent Gaussian white noise, $W_t \overset{iid}{\sim} N(0, \sigma_w^2)$. Similar to AR models, we will often let $\delta = 0$.

Properties

As with MA(1), we see the variance is constant (and not a function of time),

$$Var(Y_t) = Var(W_t + \theta_1 W_{t-1} + \theta_2 W_{t-2} \cdots + \theta_q W_{t-q}) = \sigma_w^2 (1 + \sum_{i=1}^{q} \theta_i^2)$$

More generally for a MA(q) process, the autocorrelation is non-zero for the first q lags and zero for lags > q.

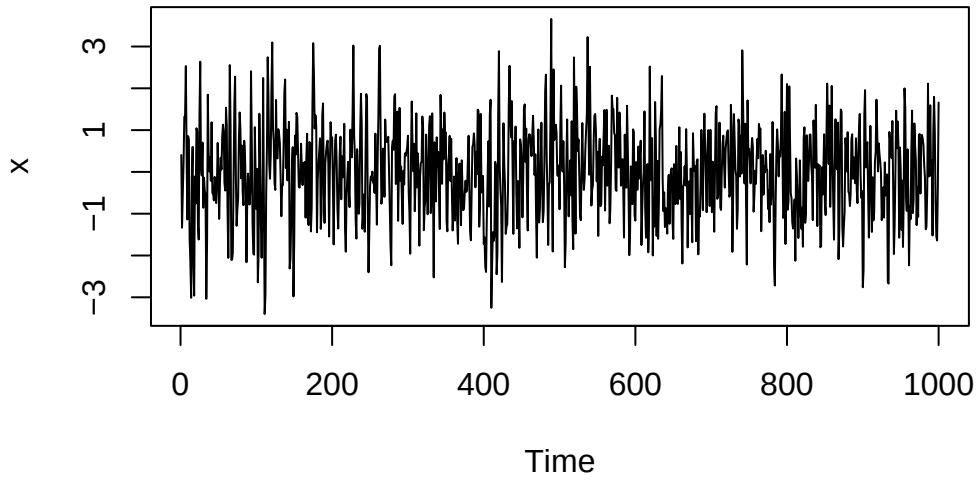
We can show this for an MA(2) process and higher-order models using the same techniques as before.

- Sample ACF for MA(q): Zero for lags > q

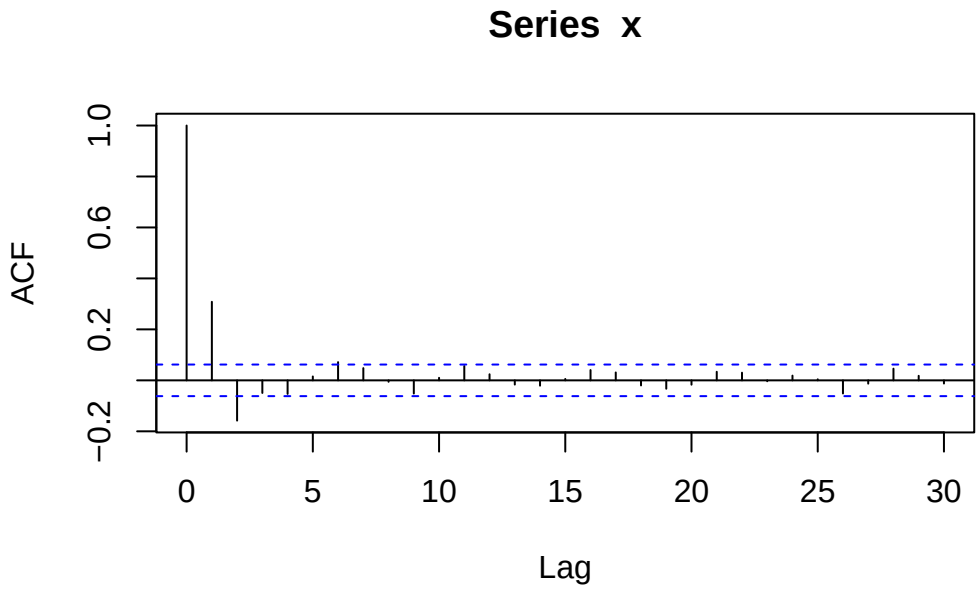
Simulated Data Example

```
# Simulate a MA2 process
# x = e + theta1 e(t-1)  + theta2 e(t-2)
# Create the vector x
x   <- vector(length=1000)
theta1 <- 0.5
theta2 <- -0.2

#simulate the white noise errors
e   <- rnorm(1000)

x[1] <- e[1]
x[2] <- e[2]
```

```r
#Fill the vector x
 for(i in 3:length(x))
    {
        x[i]    <- e[i] + theta1*e[i-1]  + theta2*e[i-2]
 }

x <- ts(x)
plot(x)
```




```r
acf(x)
```



Notice at lag = 0, the autocorrelation is 1, at lag = 1 and 2, the autocorrelation is non-zero for an MA(2) model, and for all other lags, the autocorrelation is practically zero (within blue lines).

Invertibility

To check the invertibility of an MA(q) process, we need to learn about the **backward shift operator**, denoted B, which is defined as

$$B^j Y_t = Y_{t-j}$$

The backshift operator is notation that allows us to simplify how we write down the MA(q) model.

The MA(q) model can be written as

$$\begin{aligned} Y_t &= (\theta_0 + \theta_1 B + \cdots + \theta_q B^q) W_t \\ &= \theta(B) W_t \end{aligned}$$

where $\theta(B)$ is a polynomial of order q in terms of B. It can be shown that an MA(q) process is invertible if the roots of the equation,

$\theta(B) = (\theta_0 + \theta_1 B + \cdots + \theta_q B^q) = 0$

all lie outside the unit circle, where B is regarded as a complex variable and not an operator.

Remember: the **roots** are the values of B in which $\theta(B) = 0$.

For example, a MA(1) process has a polynomial $\theta(B) = 1 + \theta_1 B$ with roots $B = -1/\theta_1$. This root is a real number, and it is outside the unit circle ($> |1|$) as long as $|\theta_1| < 1$. Rarely will you have to check this, but the software we will use restricts the values of θ_j so that the process is invertible.

5.6 AR(p) as MA(∞)

Now, we already see that we can write an MA model as an AR(∞) model. Let's write an AR model as an MA(∞) model. This is the main connection between the two models.

5.6.1 AR(1) Model

Let's return to an AR(1) model momentarily.

$$Y_t = \phi_1 Y_{t-1} + W_t$$

By successive substitution, we can rewrite this model is an infinite-order MA model,

$$\begin{aligned} Y_t &= \phi_1(\phi_1 Y_{t-2} + W_{t-1}) + W_t = \phi_1^2 Y_{t-2} + \phi_1 W_{t-1} + W_t \\ &= \phi_1^2(\phi_1 Y_{t-3} + W_{t-2}) + \phi_1 W_{t-1} + W_t = \phi_1^3 Y_{t-3} + \phi_1^2 W_{t-2} + \phi_1 W_{t-1} + W_t \\ &= W_t + \phi_1 W_{t-1} + \phi_1^2 W_{t-2} + \phi_1^3 W_{t-3} + \cdots \end{aligned}$$

This can also be shown with a backward shift operator,

$$\begin{aligned} (1 - \phi_1 B) Y_t = W_t \implies Y_t &= (1 - \phi_1 B)^{-1} W_t \\ Y_t &= (1 - \phi_1 B)^{-1} W_t \\ &= (1 + \phi_1 B + \phi_1^2 B^2 + \cdots) W_t \end{aligned}$$

This converges when $|\phi_1| < 1$ by the infinite sum rule that $\sum_{i=1}^{\infty} r^i = (1-r)^{-1}$ if $|r| < 1$.

With this format, it is clear that for an AR(1) process,

$$E(Y_t) = E(W_t + \phi_1 W_{t-1} + \phi_1^2 W_{t-2} + \cdots) = 0$$

$$Var(Y_t) = Var(W_t + \phi_1 W_{t-1} + \phi_1^2 W_{t-2} + \cdots) = \sigma_w^2(1 + \phi_1^2 + \phi_1^4 + \cdots) = \frac{\sigma_w^2}{1 - \phi_1^2} \text{ if } |\phi_1| < 1$$

5.6.2 AR(p) Model

A general AR(p) model can be written with a backward shift operator,

$$(1 - \phi_1 B - \phi_2 B^2 - \cdots - \phi_p B^p) Y_t = W_t \implies Y_t = (1 - \phi_1 B - \phi_2 B^2 - \cdots - \phi_p B^p)^{-1} W_t$$

where $(1 - \phi_1 B - \phi_2 B^2 - \cdots - \phi_p B^p)^{-1} = (1 + \beta_1 B + \beta_2 B^2 + \cdots)$ (this mathematical statement can be proved, but we won't get into that in this course). So,

$$Y_t = (1 + \beta_1 B + \beta_2 B^2 + \cdots) W_t$$

With this format, it is clear that for an AR(p) process,

$$E(Y_t) = 0$$

$$Var(Y_t) = \sigma_w^2(1 + \beta_1^2 + \beta_2^2 + \cdots) \text{ which will be finite if } \sum \beta_i^2 \text{ converges}$$

The autocovariance function is given by

$$\Sigma_Y(k) = \sigma_w^2 \sum_{i=0}^{\infty} \beta_i \beta_{i+k} \text{ where } \beta_0 = 1$$

which will converge if $\sum |\beta_i|$ converges. But, figuring out what the $beta_i$'s should be is hard. There is an easier way to do this.

5.6.3 AR(p) Estimation: Yule-Walker Equations

Let's go back to the original model statement (but let's assume $\delta = 0$),

$$Y_t = \phi_1 Y_{t-1} + \phi_2 Y_{t-2} + \cdots + \phi_p Y_{t-p} + W_t$$

Multiply that model through by Y_{t-k},

$$Y_t Y_{t-k} = \phi_1 Y_{t-1} Y_{t-k} + \phi_2 Y_{t-2} Y_{t-k} + \cdots + \phi_p Y_{t-p} Y_{t-k} + W_t Y_{t-k}$$

Then take the expectation and divide it by the variance of Y_t, $Var(Y_t)$ (assuming it is finite)

$$\frac{E(Y_t Y_{t-k})}{Var(Y_t)} = \frac{\phi_1 E(Y_{t-1} Y_{t-k})}{Var(Y_t)} + \frac{\phi_2 E(Y_{t-2} Y_{t-k})}{Var(Y_t)} + \cdots + \frac{\phi_p E(Y_{t-p} Y_{t-k})}{Var(Y_t)} + \frac{E(W_t Y_{t-k})}{Var(Y_t)}$$

Assuming the process is stationary, this simplifies to

$$\rho_k = \phi_1 \rho_{k-1} + \phi_2 \rho_{k-2} + \cdots + \phi_p \rho_{k-p} \text{ for } k = 1, 2, ...$$

If you plug in estimates of the autocorrelation function for $k = 1, ..., p$, and recognize that $\rho(-k) = \rho(k)$, and solve for these equations, you'll get the estimates of $\phi_1, ..., \phi_p$. This is a well-posed problem that can be done with matrix notation.

In practice, you will have the computer estimate these models for you. Keep reading for R examples.

- Sample ACF for AR(p): Decay to zero

5.7 ARMA Models

You might wonder why it is necessary to know the theoretical properties of the AR and MA models (expected value, variance, covariance, correlation).

This probability theory is important because it will help you choose a model for time series data when you need to choose between the models:

- Autoregressive model of order p AR(p),
- Moving Average model of order q MA(q),
- a combination of those two models, called an ARMA(p, q) and
- the values of p and/or q.

An ARMA(p, q) model is written as

$$Y_t = \delta + \phi_1 Y_{t-1} + \phi_2 Y_{t-2} + \cdots + \phi_p Y_{t-p} + W_t + \theta_1 W_{t-1} + \theta_2 W_{t-2} \cdots + \theta_q W_{t-q}$$

where the white noise $W_t \overset{iid}{\sim} N(0, \sigma_w^2)$.

Equivalently, using the backshift operator, an ARMA model can be written as

$$\phi(B) Y_t = \delta + \theta(B) W_t$$

where

$$\phi(B) = 1 - \phi_1 B - \phi_2 B^2 - \cdots - \phi_p B^p$$

and

$$\theta(B) = 1 + \theta_1 B + \theta_2 B^2 + \cdots + \theta_q B^q$$

In order for the ARMA(p,q) model

- to be a **weakly stationary process**, the roots of $\phi(B)$ must be outside the unit circle
- to be **invertible**, the roots of $\theta(B)$ must be outside the unit circle

In practice, the computer will restrict the modeling fitting to ARMA weakly stationary and invertible models. But it is important to understand this restriction if you encounter any R errors.

This combined model is useful because you can adequately model a random process with an ARMA model with fewer parameters (phi's or θ's) than a pure AR or pure MA process by itself.

Before we go any further, we need one more tool called the partial autocorrelation function.

5.7.0.1 Partial Autocorrelation

The **partial autocorrelation function** (PACF) measures the linear correlation of an observed value $\{Y_t\}$ and a lagged version of itself $\{Y_{t-k}\}$ with the linear dependence of $\{Y_{t-1}, ..., Y_{t-(k-1)}\}$ removed. In other words, the PACF is the correlation between two observations in time k lags apart after accounting for shorter lags.

$$h_k = \begin{cases} \widehat{Cor}(Y_t, Y_{t-1}) \text{ if } k = 1 \\ \widehat{Cor}(Y_t, Y_{t-k} | Y_{t-1}, ..., Y_{t-(k-1)}) \text{ if } k > 1 \end{cases}$$

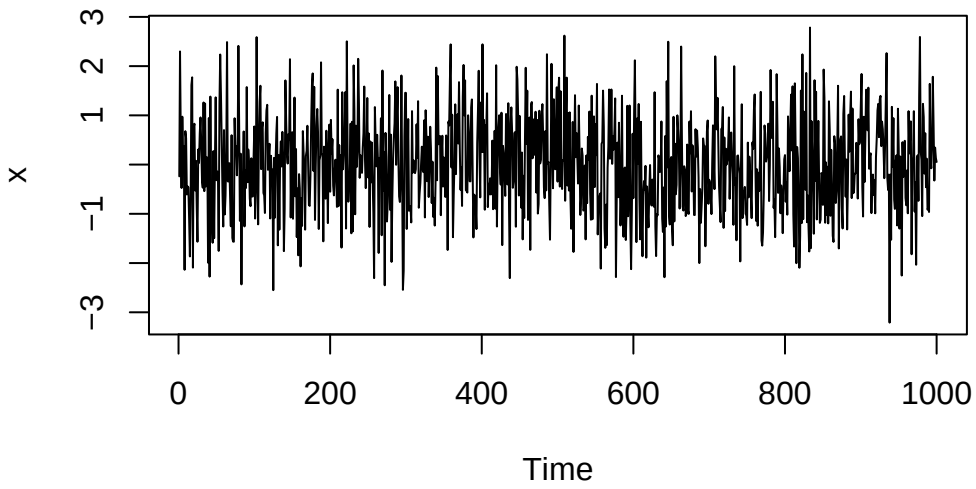
Note that the partial autocorrelation at lag 1 will be the same as the autocorrelation at lag 1. For the other lags, the partial autocorrelation measures the dependence after accounting for the relationships with observations closer in time.

Connection to Stat 155: When we interpret the slope coefficients keeping all other variables fixed, we are interpreting a relationship accounting for those other variables.

Similar to the ACF plots, the dashed horizontal lines indicate plausible values if the random process were independent white noise.

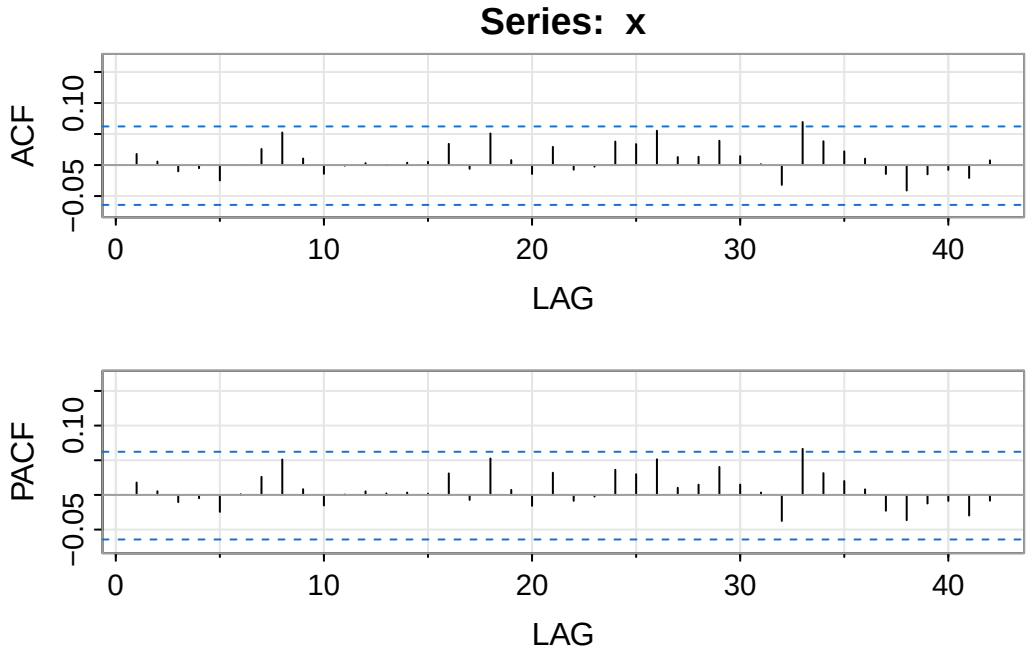
Let's look at simulated independent Gaussian white noise.

```
x <- ts(rnorm(1000))
plot(x)
```



Then, the ACF and the PACF (Partial ACF) functions.

```
acf2(x)
```



The partial autocorrelation is important to detect the value of p in an AR(p) model because it has a particular signifying pattern.

For an AR model, the theoretical PACF should be zero past the order of the model. Put another way, the number of non-zero partial autocorrelations gives the order of the AR model, the value of p.

- Sample PACF for AR(p): Zero for lags > p

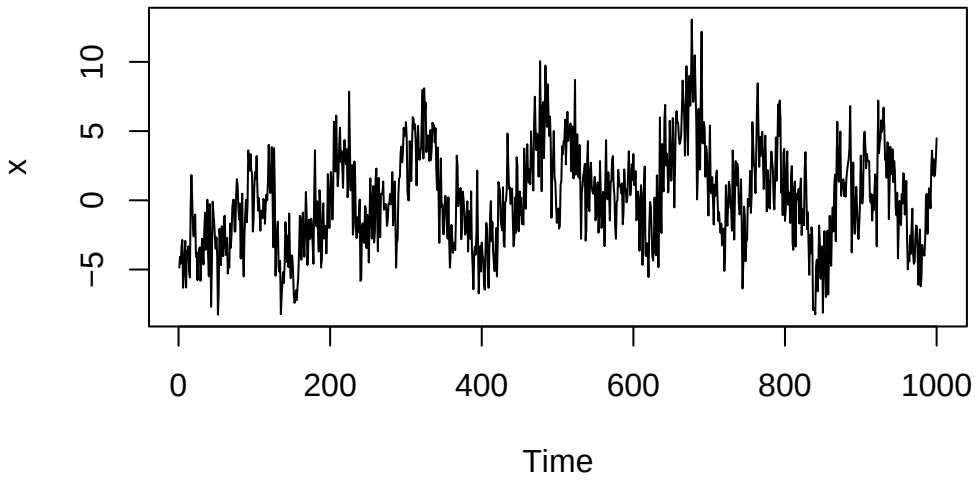
Simulated Data Examples

Let's see this in action.

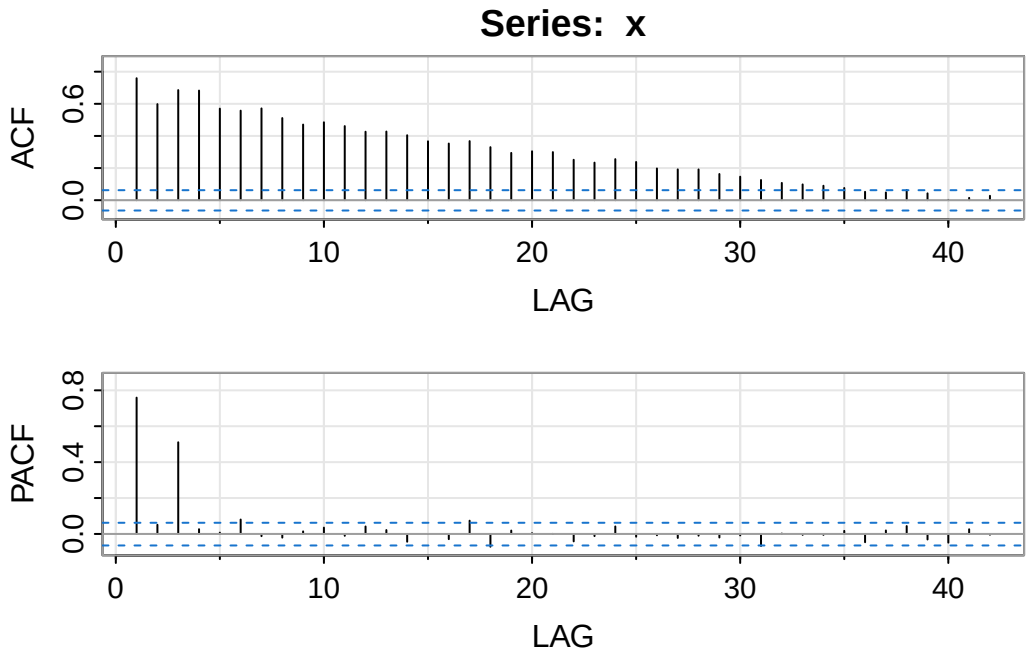
- Sample ACF for AR(p): Decays to zero
- Sample PACF for AR(p): Zero for lags > p

Play around with the values of the *phi*'s and the standard deviation to get a feel for an AR model and its associated ACF and PACF.

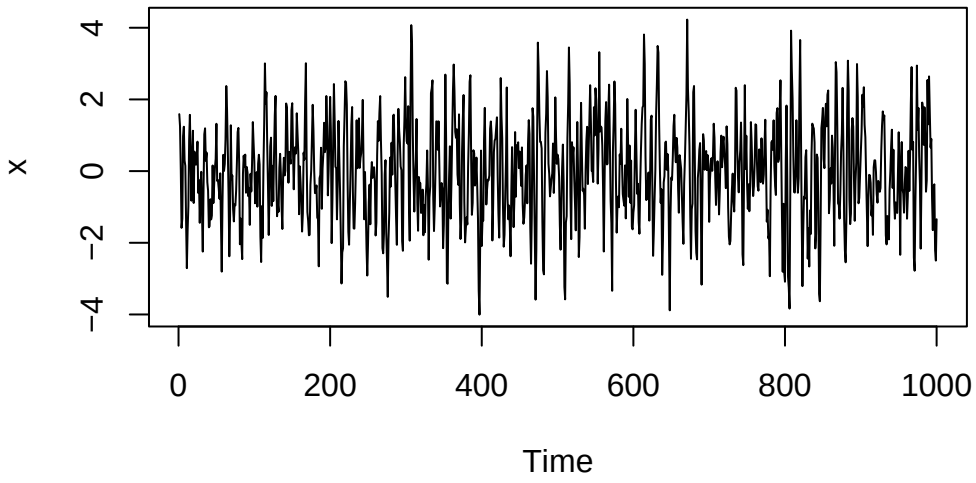
```
x <- arima.sim(n = 1000, list(ar = c(.7, -.3, .5)), sd = 2) #AR(3) model
plot(x)
```



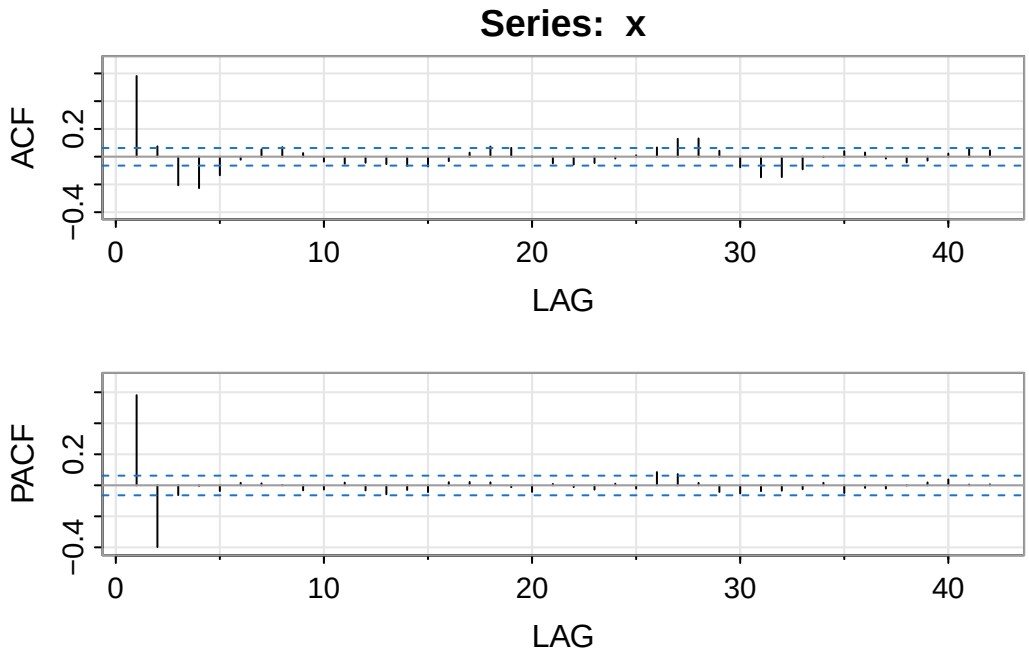

```
acf2(x)
```




```
x <- arima.sim(n = 1000, list(ar = c(.8, -.4)), sd = 1) #AR(2) model
plot(x)
```



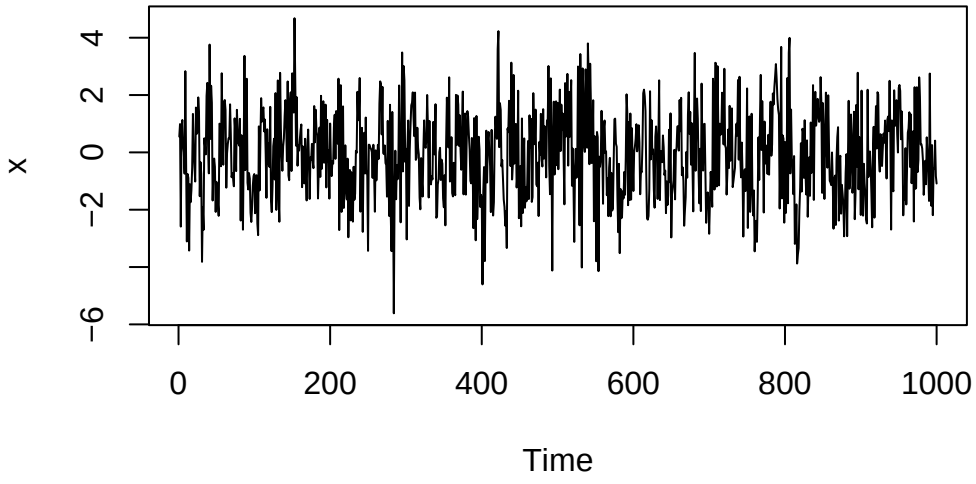

```
acf2(x)
```



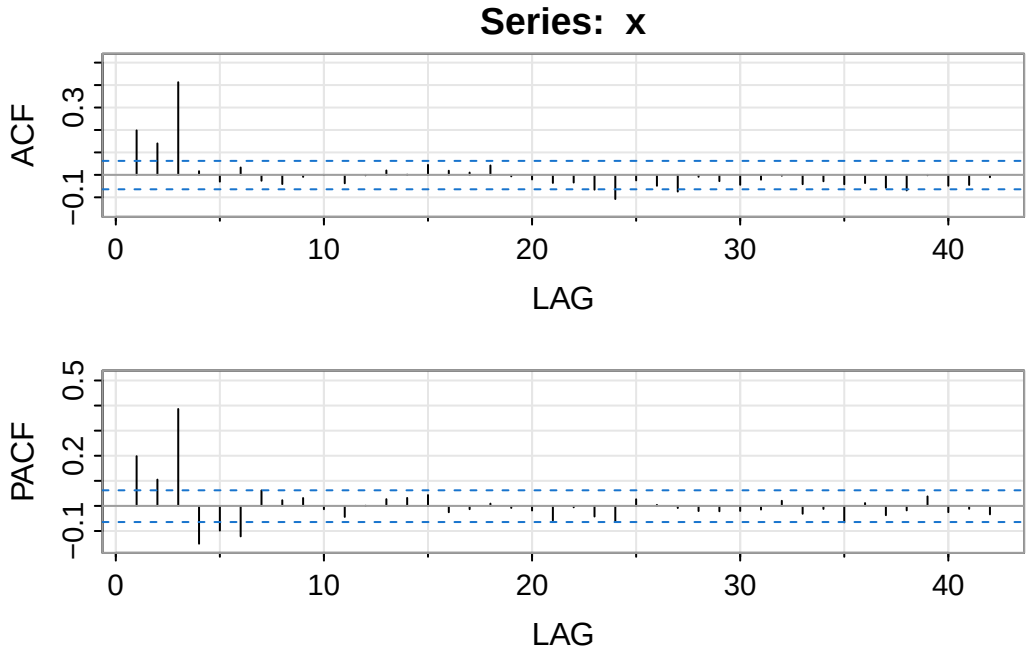
- Sample ACF for MA(q): Zero for lags > q
- Sample PACF for MA(q): Decays to zero

Play around with the values of the *theta*'s and the standard deviation to get a feel for an MA model and its associated ACF and PACF.

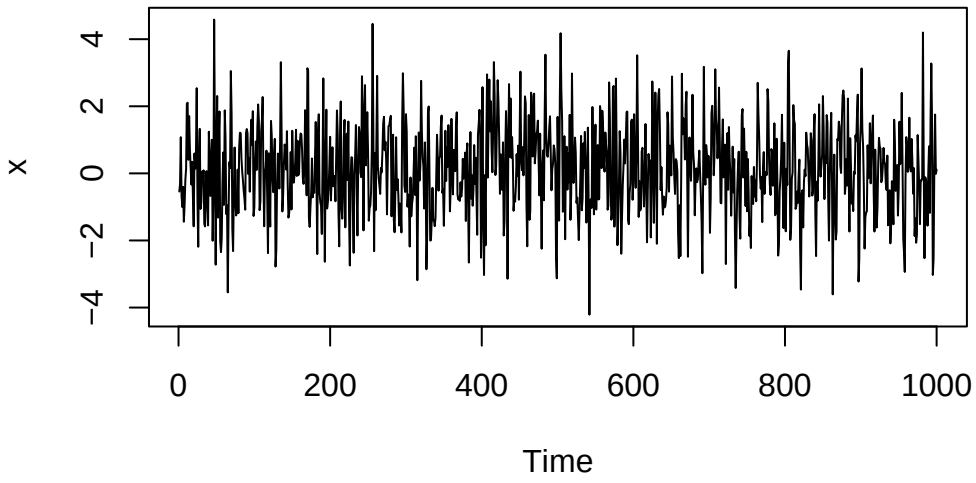
```
x <- arima.sim(n = 1000, list(ma = c(.7, -.2, .8)), sd = 1) #MA(3) model
plot(x)
```



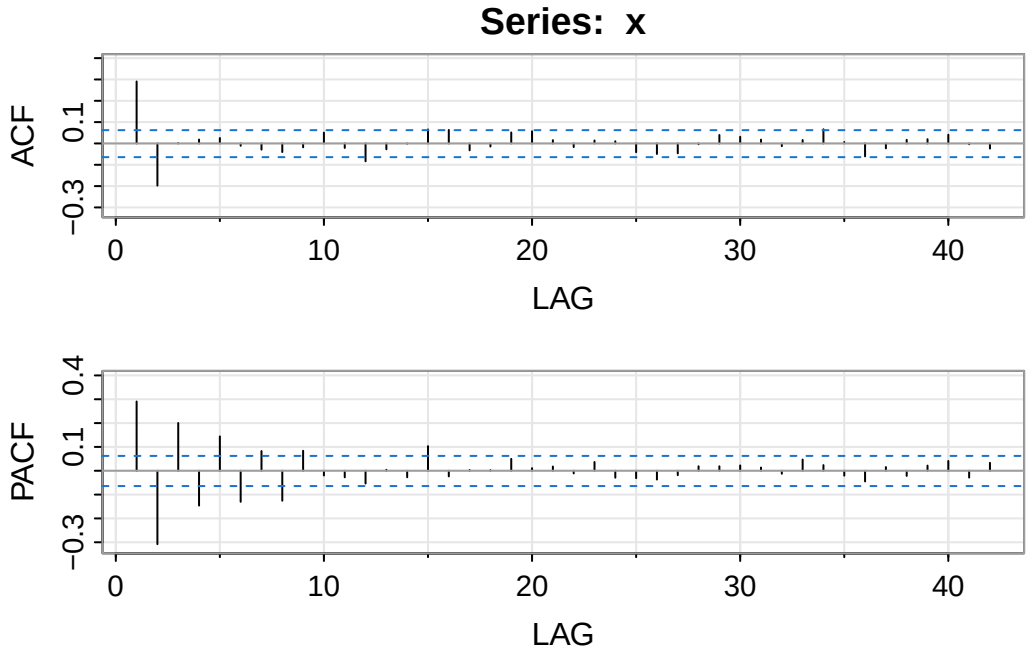

```
acf2(x)
```




```
x <- arima.sim(n = 1000, list(ma = c(.8, -.4)), sd = 1) #MA(2) model
plot(x)
```




```
acf2(x)
```



5.7.1 Model Selection

We have now learned about AR(p), MA(q), and ARMA(p,q) models. When we have observed data, we first need to estimate and remove the trend and seasonality and then choose a stationary model to account for the dependence in the errors.

Below, we have a few guidelines and tools to help you work through the modeling process.

5.7.1.1 Time Series Plot

- Look for possible long-range trends, seasonality of different orders, outliers, constant or non-constant variance.
- Model and remove the trend.
- Model and remove the seasonality.
- Investigate the outliers for possible human errors or find explanations for their extreme values.
- If the variability around the trend/seasonality is non-constant, you could transform the series with a logarithm or square root, or use a more complex model such as the ARCH model.

5.7.1.2 ACF and PACF Plots

- AR(p) models have PACF non-zero values at lags less than or equal to p and zero elsewhere. The ACF should decay or taper to zero in some fashion.
- MA(q) models have theoretical ACF with non-zero values at lags less than or equal to q. The PACF should decay or taper to zero in some fashion.
- ARMA models (p and $q > 0$) have ACFs and PACFs that taper down to 0. These are the trickiest to determine because the order will not be obvious. Try fitting a model with low orders for one or both p and q and compare models.
- If the ACF and PACF do not decay to zero but instead have values that stays close to 1 over many lags, the series is non-stationary (non-constant mean) and differencing or estimating to remove the trend will be needed. Try a first difference and look at the ACF and PACF of the difference data.
- If all of the autocorrelations are non-significant (within the horizontal lines), then the series is white noise, and you don't need to model anything. (Great!)

5.7.1.3 Diagnostics

Once we have a potential model that we are considering, we want to make sure that it fits the data well. Below are a few guidelines to check before you determine your final model.

1. Look at the significance of the coefficients. Are they significantly different from zero? If so, great. If not, they may not be necessary and you should consider changing the order of the model.
2. Look at the ACF of the residuals. Ideally, our residuals should look like white noise.

3. Look at a time series plot of the residuals and check for any non-constant variance. If you have non-constant variance in the residuals, consider taking the original data's natural log and analyzing the log instead of the raw data.

4. Look at the p-values for the Ljung-Box Test (see Real Data Example for example, R code and output). For this test, H_0 : the data are independent, and H_A : data exhibit serial correlation (not independent). The test statistic is

$$Q_h = n(n+2)\sum_{k=1}^{h}\frac{r_k^2}{n-k}$$

where h is the number of lags being tested simultaneously. If H_0 is true, Q_h follows a Chi-squared distribution with df $= h$ for very large n. So you'd like to see large p-values for each lag h to not reject the H_0.

5.7.1.4 Criteria to Choose

Which model to choose?

- Choose the 'best' model with the fewest parameters
- Pick the model with the lowest standard errors for predictions in the future
- Compare models with respect to the MSE, AIC, and BIC (the lower the better for all)
- If two models seem very similar when converted to an infinite order MA, they may be nearly the same. So it may not matter which ARMA model you end up with.

5.8 Real Data Example

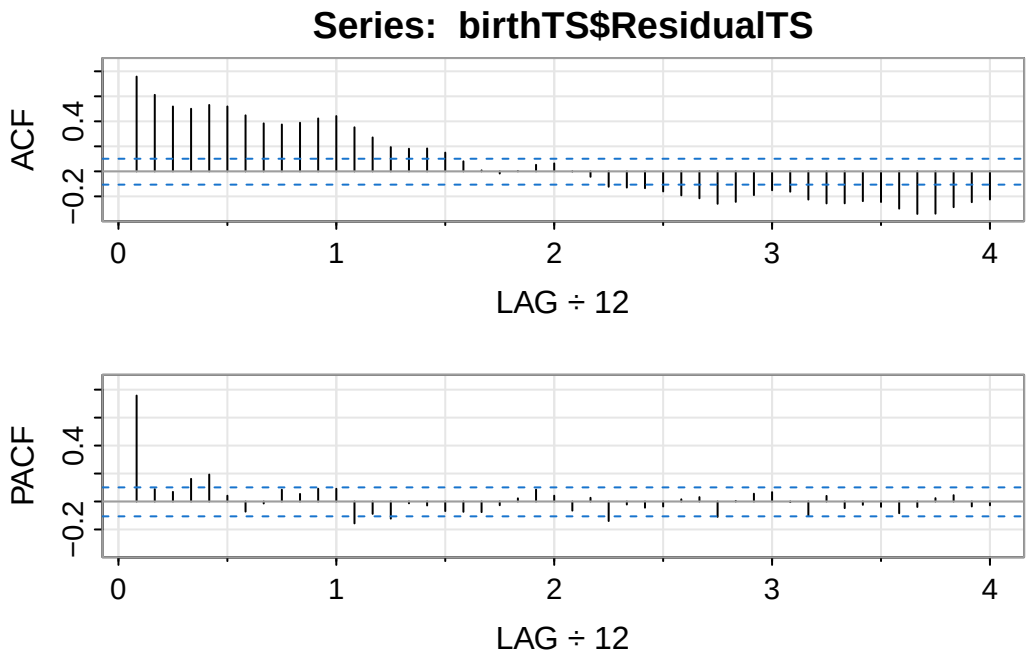
In order to fit these models to real data, we first must get a detrended stationary series (removing seasonality as well). This can be done with estimation and removing, or through differencing.

For the birth data, we tried a variety of ways of removing the trend and seasonality with differencing in the Model Components chapter. While the moving average filter worked well. Let's try a fully parametric approach by using a spline to model the trend and indicators for the seasonality,

```
library(splines)
birthTS <- data.frame(Year = time(birth), #time() works on ts objects
  Month = cycle(birth), #cycle() works on ts objects
  Value = birth)
birthTS <- birthTS %>%
  mutate(Time = Year + (Month-1)/12)

TrendSeason <- birthTS %>%
  lm(Value ~ bs(Time, degree = 2, knots = c(1965, 1972), Boundary.knots = c(1948,1981)) + fa

birthTS <- birthTS %>%
  mutate(ResidualTS = Value - predict(TrendSeason))
```

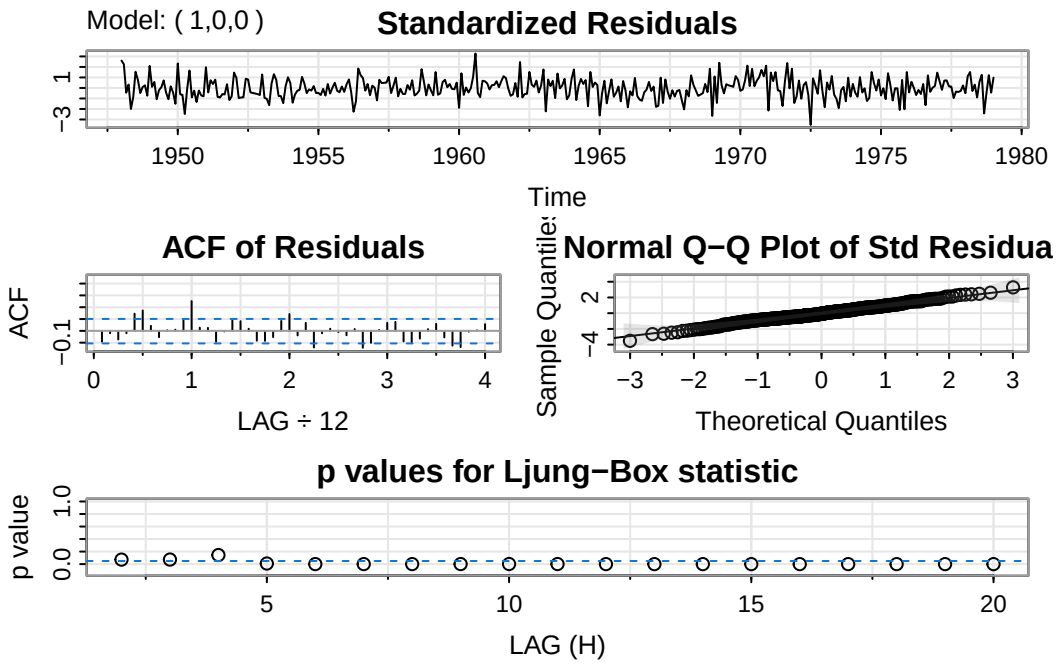
The next step is to consider the ACF and PACF to see if you can recognize any clear patterns in dropping or decaying (MA or AR).

```
acf2(birthTS$ResidualTS)
```



We note the ACF is decaying slowly to zero, and the PACF has 1 clear non-zero lag. These two graphs suggest an AR(1) model. Let's fit an AR(1) model and a few other candidate models with `sarima()` in the astsa package.

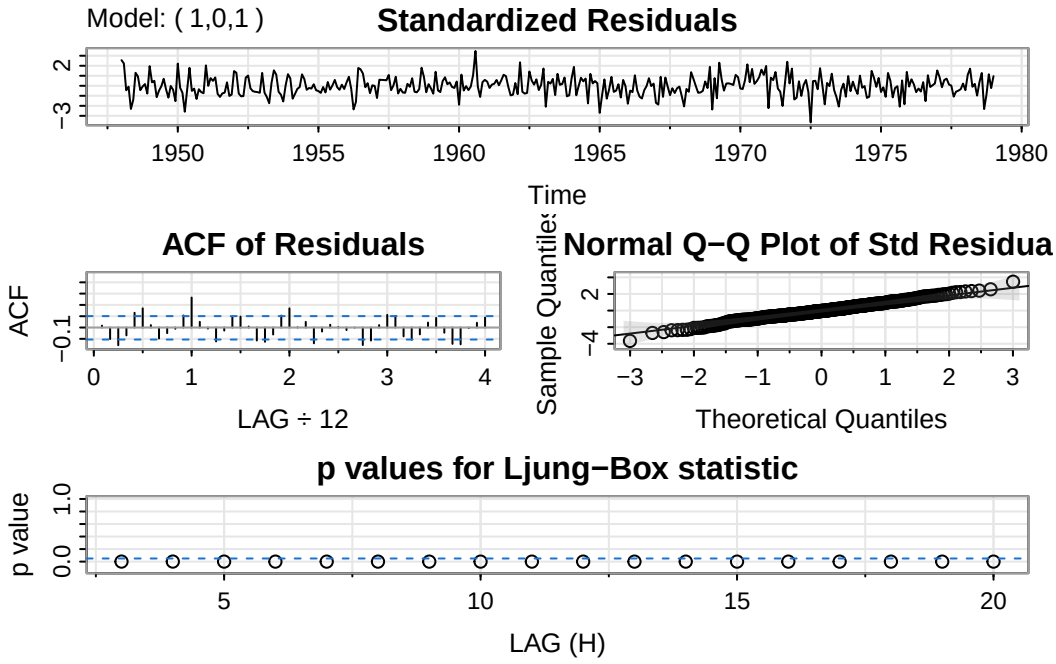
```
mod.fit1 <- sarima(birthTS$ResidualTS, p = 1,d = 0,q = 0) #AR(1)
```




```
mod.fit1$fit
```

```
Call:
arima(x = xdata, order = c(p, d, q), seasonal = list(order = c(P, D, Q), period = S),
    xreg = xmean, include.mean = FALSE, transform.pars = trans, fixed = fixed,
    optim.control = list(trace = trc, REPORT = 1, reltol = tol))

Coefficients:
         ar1    xmean
      0.7709  0.3080
s.e.  0.0334  1.5785

sigma^2 estimated as 49.61:  log likelihood = -1257.84,  aic = 2521.69
```

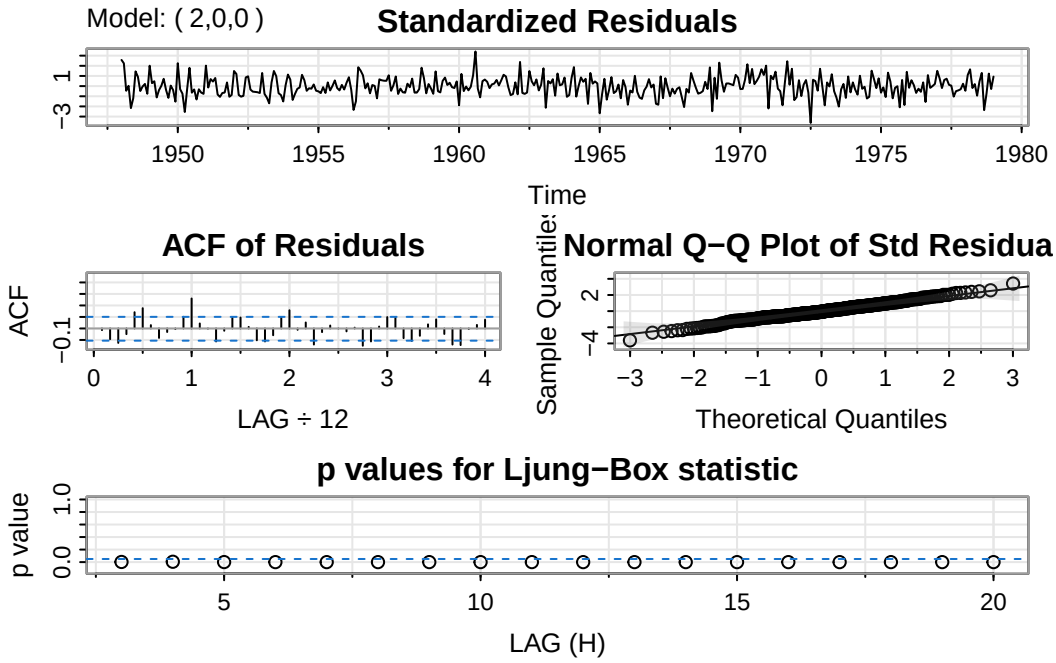
```
mod.fit2 <- sarima(birthTS$ResidualTS, p = 1,d = 0,q = 1) #ARMA(1,1)
```




```
mod.fit2$fit
```

```
Call:
arima(x = xdata, order = c(p, d, q), seasonal = list(order = c(P, D, Q), period = S),
    xreg = xmean, include.mean = FALSE, transform.pars = trans, fixed = fixed,
    optim.control = list(trace = trc, REPORT = 1, reltol = tol))

Coefficients:
         ar1      ma1   xmean
      0.8557  -0.2106  0.5044
s.e.  0.0412   0.0886  1.9553

sigma^2 estimated as 48.77:  log likelihood = -1254.71,  aic = 2517.41
```

```
mod.fit3 <- sarima(birthTS$ResidualTS, p = 2,d = 0,q = 0) #AR(2)
```




```
mod.fit3$fit
```

```
Call:
arima(x = xdata, order = c(p, d, q), seasonal = list(order = c(P, D, Q), period = S),
    xreg = xmean, include.mean = FALSE, transform.pars = trans, fixed = fixed,
    optim.control = list(trace = trc, REPORT = 1, reltol = tol))

Coefficients:
         ar1     ar2   xmean
      0.6864  0.1127  0.4171
s.e.  0.0514  0.0523  1.7847

sigma^2 estimated as 48.99:  log likelihood = -1255.54,  aic = 2519.07
```

```
#Choose a model with lowest BIC
mod.fit1$BIC
```

```
NULL
```

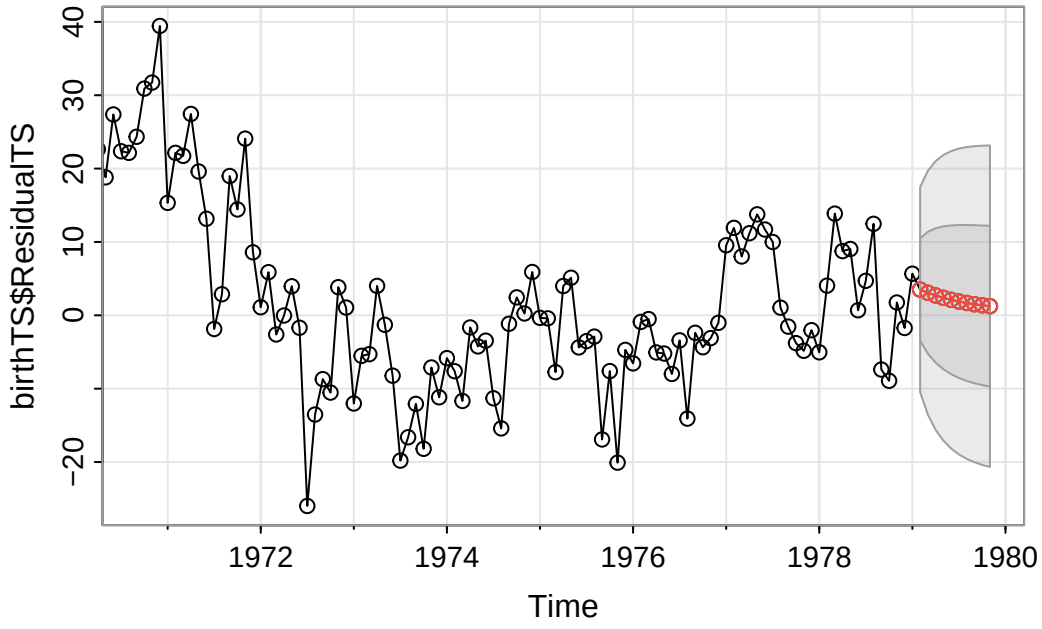
```
mod.fit2$BIC
```

```
NULL
```

```
mod.fit3$BIC
```

```
NULL
```

```
sarima.for(birthTS$ResidualTS,10, p = 1,d = 0,q = 1) #forecasts the residuals 10 time units
```




```
$pred
          Feb      Mar      Apr      May      Jun      Jul      Aug      Sep
1979 3.505241 3.072234 2.701709 2.384649 2.113339 1.881179 1.682518 1.512523
          Oct      Nov
1979 1.367058 1.242583

$se
           Feb       Mar       Apr       May       Jun       Jul       Aug
1979  6.983803  8.310959  9.161616  9.737476 10.138415 10.422217 10.625218
           Sep       Oct       Nov
1979 10.771436 10.877254 10.954089
```

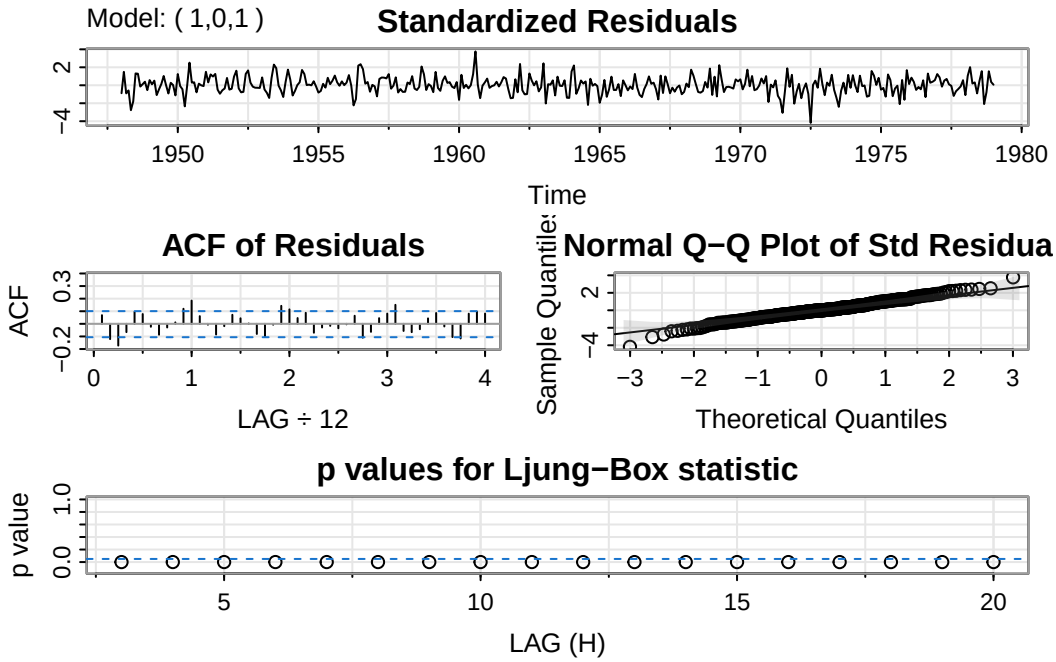
Now, you'll notice that we went through the following steps:

1. Model Trend and Remove
2. Model Seasonality and Remove
3. Model Errors

The forecasts look funny because they are only forecasts for errors. If we have taken a parametric approach to estimate the trend and the seasonality, we can incorporate the modeling and removal process in the `sarima()` function.

First, we need to create a matrix of the variables for the trend and seasonality. We can easily do that with `model.matrix()` and remove the intercept column with `[,-1]`. Then we pass that matrix to `sarima()` with the `xreg=` argument. This will be useful for us when we do forcasting.

```
X <- model.matrix(TrendSeason)[,-1] #remove the intercept

GoodMod <- sarima(birthTS$Value, p = 1,d = 0, q = 1, xreg = X)
```



Now, you should notice two things from this output:

1. There is a slightly higher ACF of residuals at lag 1.0 (at 12 months or 1 year – which is the period of the seasonality)

2. The p-values for the Ljung-Box statistic are all < 0.05. That is not ideal because it suggests that the residuals of the full model are not independent white noise. There is still dependence that we need to account for with the model.

We'll come back to fix this. Before we do that, let's talk about the function being called `sarima()` instead of `arma()`. What does the `i' stand for, and what does thes`‘ stand for? Read the next section to find out!

5.9 ARIMA and SARIMA Models

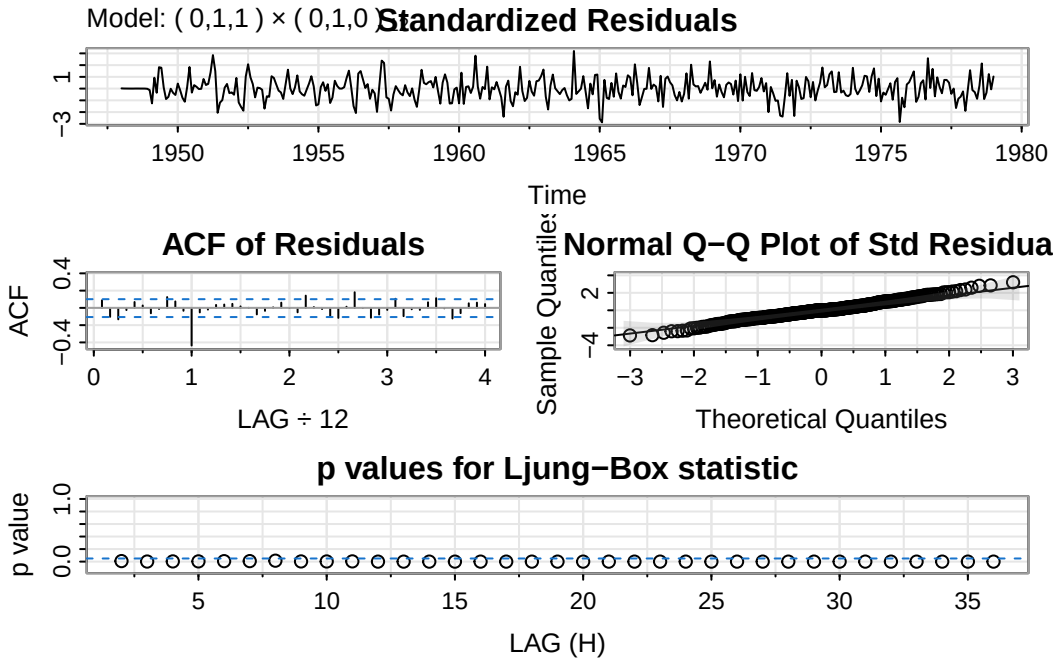
5.9.1 ARIMA Models

ARIMA models are the same as ARMA models, but the `i' stands for 'integrated', which refers to differencing. If you were differencing to remove the trend and seasonality, you could incorporate that difference in the model fitting procedure (like we did with the parametric modeling and`xreg=‘).

Below, we show fitting the model on the ORIGINAL data before removing the trend or seasonality. Then we do a first difference to remove the trend and a first-order difference of lag 12 to remove the seasonality.

The little d indicates the order of differences for lag 1, the big D indicates the order of seasonal differences, and S gives the lag for the seasonal differences. Once we know the differencing that we need to do, we incorporate the differencing into the model fitting, which will give us more accurate predictions for the future.

```
mod.fit4 <- sarima(birth, p = 0, d = 1, q = 1, D = 1, S = 12) #this model includes lag 1 dif
```




```
mod.fit4$fit
```

```
Call:
arima(x = xdata, order = c(p, d, q), seasonal = list(order = c(P, D, Q), period = S),
    include.mean = !no.constant, transform.pars = trans, fixed = fixed, optim.control = list
        REPORT = 1, reltol = tol))

Coefficients:
          ma1
      -0.5221
s.e.   0.0547

sigma^2 estimated as 71.62:  log likelihood = -1279.83,  aic = 2563.65
```

Notice, this model with differencing isn't any better than the ARMA model with the spline trend and indicator variables for the month.

5.9.2 Seasonal ARIMA Models

The `s` in the `sarima()` is seasonal. A Seasonal ARIMA model allows us to add a seasonal lag (e.g., 12) into an ARMA model. The model is written as

$$\Phi(B^S)\phi(B)Y_t = \Theta(B^S)\theta(B)W_t$$

where the non-seasonal components are:

$$\phi(B) = 1 - \phi_1 B - \phi_2 B^2 - \cdots - \phi_p B^p$$

and

$$\theta(B) = 1 + \theta_1 B + \theta_2 B^2 + \cdots + \theta_q B^q$$

and the seasonal components are:

$$\Phi(B^S) = 1 - \Phi_1 B^S - \Phi_2 B^{2S} - \cdots - \Phi_p B^{PS}$$

and

$$\Theta(B^S) = 1 + \Theta_1 B^S + \Theta_2 B^{2S} + \cdots + \Theta_q B^{QS}$$

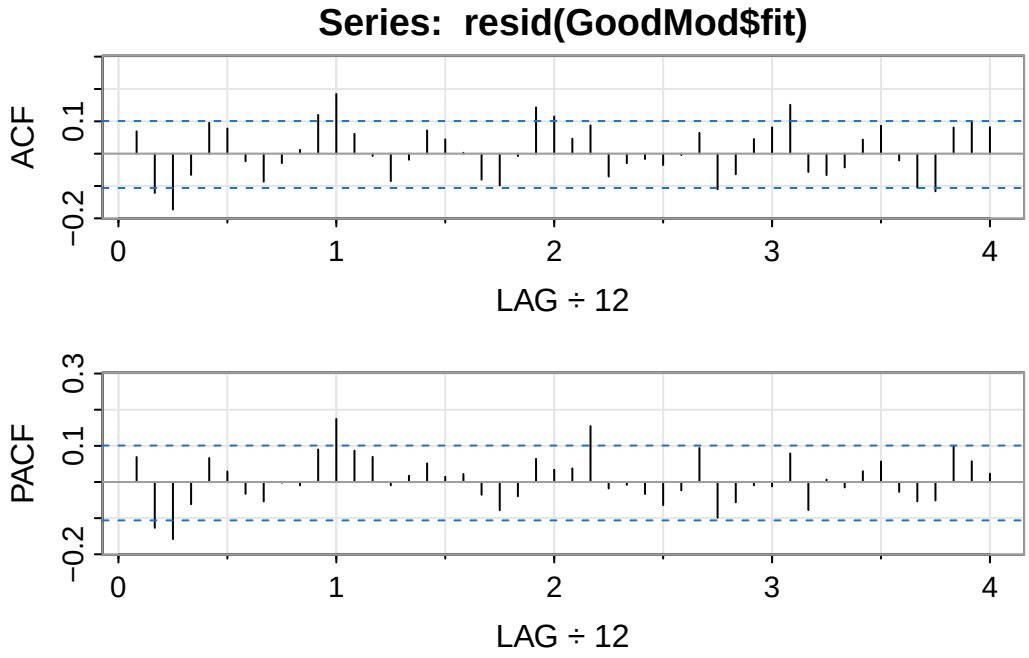
Why you might need a Seasonal ARMA?

If you see strong seasonal autocorrelation in the residuals after you fit a good ARMA model, try a seasonal ARMA.

If strong seasonal autocorrelation (non-zero values at lag S, $2S$, etc.) drops off after Q seasonal lags, you can fit a seasonal MA(Q) model.

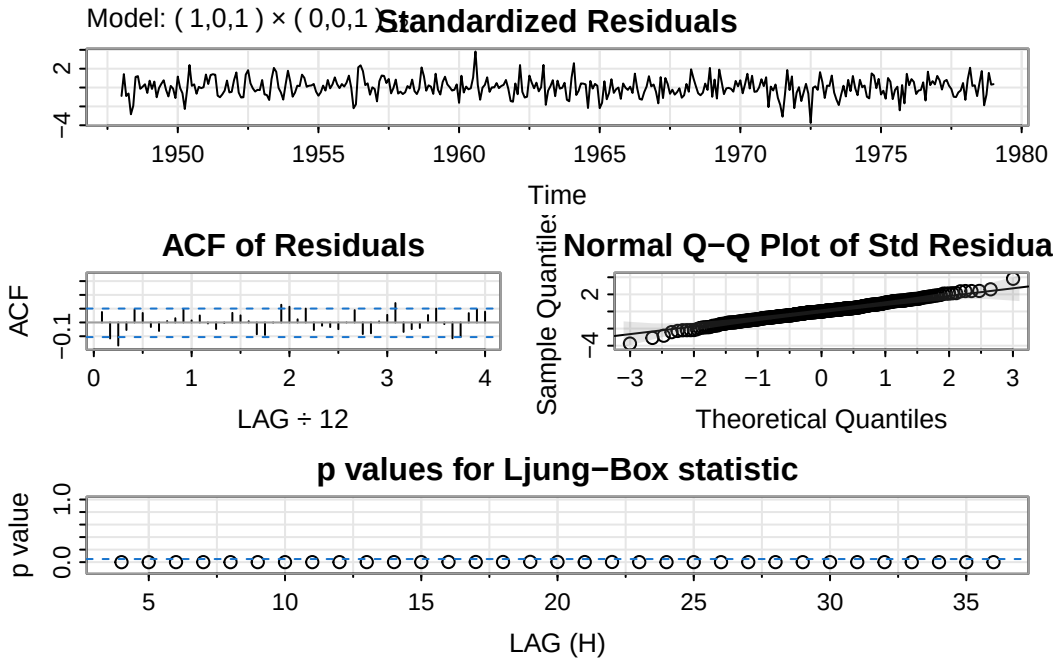
On the other hand, if you see a strong seasonal partial autocorrelation that drops off after P seasonal lags, you can fit a seasonal AR(P) model.

```
acf2(resid(GoodMod$fit)) #Notice the high ACF and PACF for lag 1 year (12 months) -- only lo
```

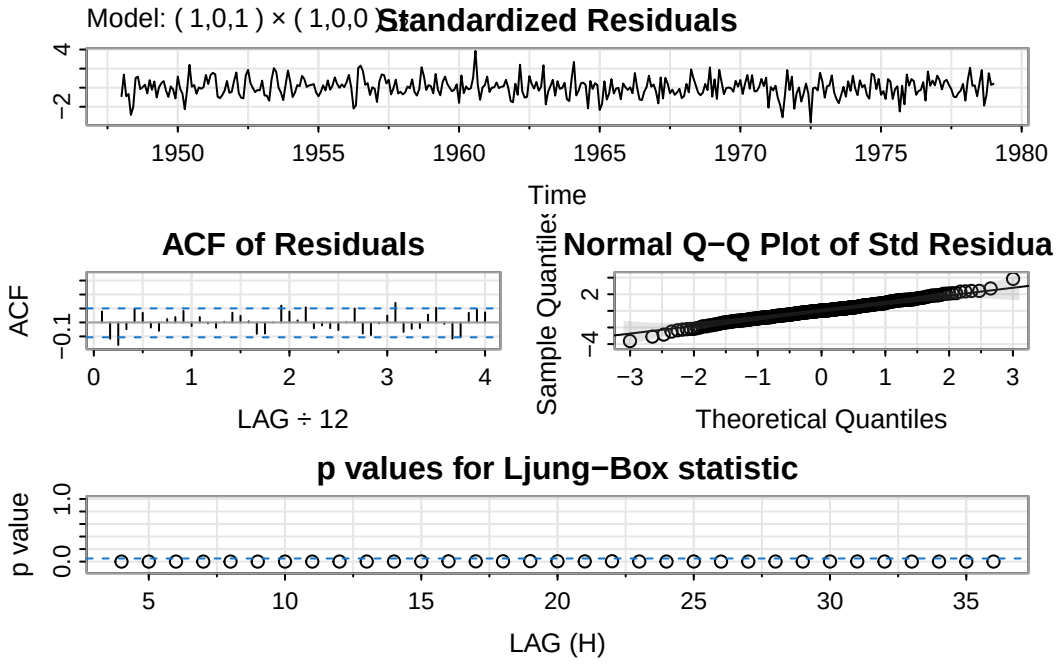


```
mod.fit5 <- sarima(birthTS$Value, p = 1,d = 0, q = 1,P = 0, D = 0, Q = 1, S = 12, xreg = X) #
```

```
Warning in arima(xdata, order = c(p, d, q), seasonal = list(order = c(P, :
possible convergence problem: optim gave code = 1
```




```
mod.fit6 <- sarima(birthTS$Value, p = 1,d = 0, q = 1,P = 1, D = 0, Q = 0, S = 12, xreg = X) #
```




```
GoodMod$BIC
```

```
NULL
```

```
mod.fit6$BIC
```

```
NULL
```

```
mod.fit5$BIC
```

```
NULL
```

While the p-values are still not ideal, the model with spline trend, indicator variables for the month, ARMA(1,1) + SeasonalAR(1) for errors has the lowest BIC, and the SARIMA coefficients are all significantly different from zero.

5.10 Forecasting

Typically, when we are working with time series, we want to make predictions for the near future based on recently observed data.

Let's think about an example. Imagine we had data $y_1, ..., y_n$ and we estimated the mean and seasonality such that $e_t = y_t - \underbrace{\hat{f}(t|y_1, ..., y_n)}_{\text{trend}} - \underbrace{\hat{S}(t|y_1, ..., y_n)}_{\text{seasonality}}$. Thus, if we knew or had a prediction of e_t, we could get a prediction of $y_t = \underbrace{\hat{f}(t|y_1, ..., y_n)}_{\text{trend}} + \underbrace{\hat{S}(t|y_1, ..., y_n)}_{\text{seasonality}} + e_t$.

Let's say we modeled e_t with an AR(2) process, then

$$e_t = \phi_1 e_{t-1} + \phi_2 e_{t-2} + W_t \qquad W_t \overset{iid}{\sim} N(0, \sigma_w^2)$$

Since we have data for $y_1, ..., y_n$, we have data for $e_1, ..., e_n$ to use to estimate ϕ_1, ϕ_2, and σ_w^2.

To get a one-step ahead prediction based on the n observed data points, e_{n+1}^n, we could plug into our model equation,

$$e_{n+1} = \phi_1 e_n + \phi_2 e_{n-1} + W_{n+1} \qquad W_t \overset{iid}{\sim} N(0, \sigma_w^2)$$

We do have the values of e_n and e_{n-1} plus estimates of ϕ_1 and ϕ_2. But we don't know w_{n+1}. However, we know that our white noise is about 0 on average. So, our prediction one step ahead based on n observed data points is

$$\hat{e}_{n+1}^n = \hat{\phi}_1 e_n + \hat{\phi}_2 e_{n-1}$$

What about two steps ahead? If we plug it into our model,

$$e_{n+2} = \phi_1 e_{n+1} + \phi_2 e_n + W_{n+2} \qquad W_t \overset{iid}{\sim} N(0, \sigma_w^2)$$

we see that we do have the value of e_n plus estimates of ϕ_1 and ϕ_2. But we don't know e_{n+1} and W_{n+2}. Similar to before, we can assume the white noise is, on average, 0, and we can plug in our predictions. So, our two-step ahead prediction based on n data points is

$$\hat{e}_{n+2}^n = \hat{\phi}_1 \hat{e}_{n+1}^n + \hat{\phi}_2 e_n$$

We could continue this process,

$$\hat{e}_{n+m}^n = \hat{\phi}_1 \hat{e}_{n+m-1}^n + \hat{\phi}_2 \hat{e}_{n+m-2}^n$$

for $m > 2$. In general, to do forecasting with an ARMA model, for $1 \leq j \leq n$ we use residuals for w_j and let $w_j = 0$ for $j > n$. Similarly, for $1 \leq j \leq n$, we use observed data for e_j and plug in the forecasts $\hat{e}_j^n$ for $j > n$.

5.10.1 Prediction Intervals

We know that our predictions will be off from the truth. But how far off?

Before we can estimate the standard deviation of our errors, we need to know one more thing about ARMA models.

5.10.1.1 ARMA to MA(∞)

With a stationary ARMA model, we can write it as an infinite MA process (similar to the AR to infinite MA process),

$$Y_t = \sum_{j=0}^{\infty} \Psi_j W_{t-j}$$

where $\Psi_0 = 1$ and $\sum_{j=1}^{\infty} |\Psi_j| < \infty$.

The R command `ARMAtoMA(ar = 0.6, ma = 0, 12)` gives the first 12 values of Ψ_j for an AR(1) model with $\phi_1 = 0.6$.

```
ARMAtoMA(ar = 0.6, ma = 0, 12)
```

```
 [1] 0.600000000 0.360000000 0.216000000 0.129600000 0.077760000 0.046656000
 [7] 0.027993600 0.016796160 0.010077696 0.006046618 0.003627971 0.002176782
```

The R command `ARMAtoMA(ar = 0.6, ma = 0.1, 12)` gives the first 12 values of Ψ_j for an ARMA(1,1) model with $\phi_1 = 0.6$ and $\theta_1 = 0.1$.

```
ARMAtoMA(ar = 0.6, ma = 0.1, 12)
```

```
 [1] 0.700000000 0.420000000 0.252000000 0.151200000 0.090720000 0.054432000
 [7] 0.032659200 0.019595520 0.011757312 0.007054387 0.004232632 0.002539579
```

5.10.1.2 Standard Errors of ARMA Errors

In order to create a prediction interval for $\hat{y}^n_{n+m}$, we need to know how big the error, $y^n_{n+m} - y_{n+m}$, may be.

It can be shown that

$$Var(\hat{y}^n_{n+m} - y_{n+m}) = \sigma^2_w \sum_{j=0}^{m-1} \Psi^2_j$$

and thus, the standard errors are

$$SE(\hat{y}^n_{n+m} - y_{n+m}) = \sqrt{\hat{\sigma}^2_w \sum_{j=0}^{m-1} \hat{\Psi}^2_j}$$

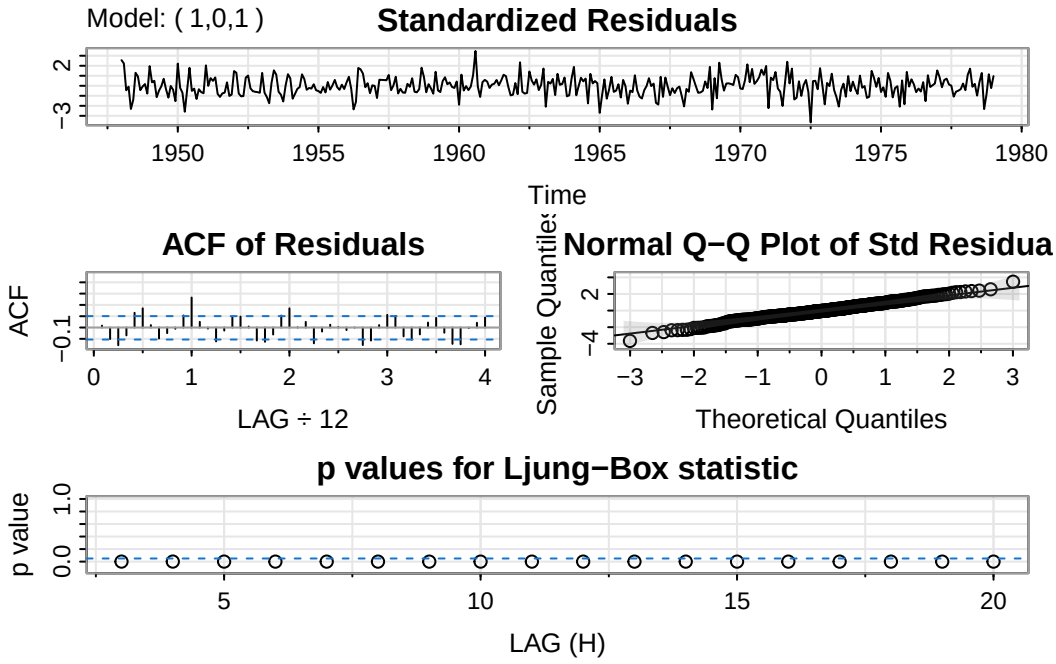
where $\hat{\sigma^2_w}$ and $\hat{\Psi}_j$ are given by the estimation process.

Let's see this play out with our birth data. We can estimate the ARMA model with `sarima()` and pull out the estimates of the sigma and the Psi_j's. Compare our hand calculations to the standard errors from `arima.for()`.

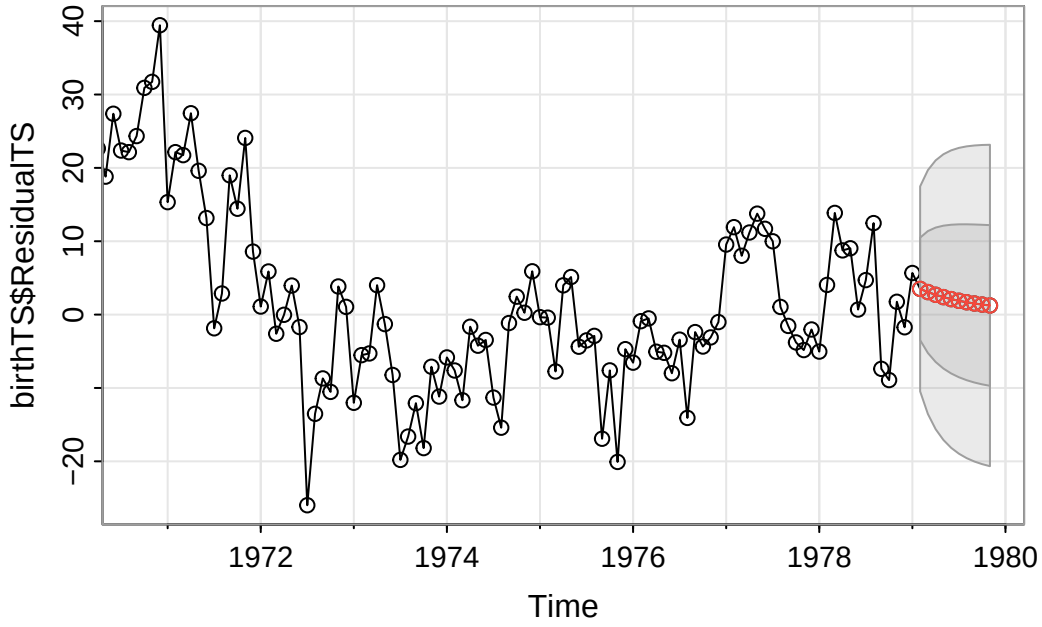
```
sarima(birthTS$ResidualTS, p = 1,0,q = 1)
```

```
initial  value 2.384529
iter   2 value 2.179452
iter   3 value 2.052027
iter   4 value 2.026011
iter   5 value 1.962581
iter   6 value 1.952577
iter   7 value 1.935739
iter   8 value 1.934205
iter   9 value 1.933599
iter  10 value 1.933531
iter  11 value 1.933495
iter  12 value 1.933483
iter  13 value 1.933478
iter  14 value 1.933470
iter  15 value 1.933456
iter  16 value 1.933451
iter  17 value 1.933448
iter  18 value 1.933444
iter  19 value 1.933443
```

```
iter  20 value 1.933440
iter  21 value 1.933436
iter  22 value 1.933433
iter  23 value 1.933433
iter  24 value 1.933433
iter  25 value 1.933433
iter  25 value 1.933433
iter  25 value 1.933433
final  value 1.933433
converged
initial  value 1.945918
iter   2 value 1.945242
iter   3 value 1.945221
iter   4 value 1.945011
iter   5 value 1.944980
iter   6 value 1.944962
iter   7 value 1.944938
iter   8 value 1.944909
iter   9 value 1.944900
iter  10 value 1.944897
iter  11 value 1.944892
iter  12 value 1.944889
iter  13 value 1.944888
iter  14 value 1.944888
iter  14 value 1.944888
iter  14 value 1.944888
final  value 1.944888
converged
<><><><><><><><><><><><><><><>

Coefficients:
      Estimate     SE t.value p.value
ar1     0.8557 0.0412 20.7562  0.0000
ma1    -0.2106 0.0886 -2.3770  0.0180
xmean   0.5044 1.9553  0.2580  0.7966

sigma^2 estimated as 48.77351 on 370 degrees of freedom

AIC = 6.749101  AICc = 6.749275  BIC = 6.791155
```




```
psi = c(1,ARMAtoMA(ar = .8727,ma = -0.2469, 9))
sigma2 = 43.5

sarima.for(birthTS$ResidualTS, n.ahead=10, p = 1, d = 0, q = 1)$se
```



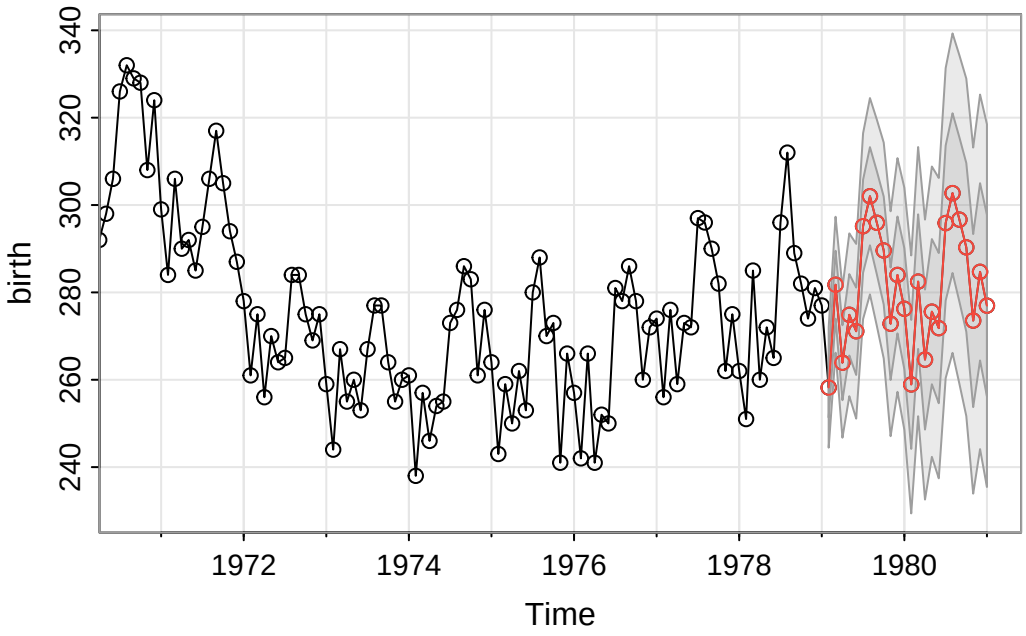
```
          Feb       Mar       Apr       May       Jun       Jul       Aug
1979 6.983803  8.310959  9.161616  9.737476 10.138415 10.422217 10.625218
          Sep       Oct       Nov
1979 10.771436 10.877254 10.954089
```

```
sqrt(sigma2*cumsum(psi^2)) #The SE for the predictions are the same
```

```
 [1]  6.595453  7.780470  8.573809  9.131903  9.535062  9.831025 10.050588
 [8] 10.214643 10.337841 10.430694
```

But if we incorporate the differencing for the trend/seasonality into the model, we'll get forecasts in the original units of y_t instead of just Y_t.

```
sarima.for(birth, n.ahead = 24, p = 0, d = 1, q = 1, P = 0, D = 1, Q = 1, S = 12)
```




```
$pred
          Jan      Feb      Mar      Apr      May      Jun      Jul      Aug
1979          258.2172 281.7558 263.9016 274.8702 271.1040 295.1588 302.0120
1980 276.2490 258.9313 282.4699 264.6157 275.5843 271.8181 295.8729 302.7261
1981 276.9632
          Sep      Oct      Nov      Dec
1979 295.9419 289.5935 272.8265 283.9909
```

```
1980 296.6561 290.3076 273.5406 284.7050
1981

$se
           Jan       Feb       Mar       Apr       May       Jun       Jul
1979            6.884911  7.781346  8.584677  9.319014  9.999569 10.636668
1980 13.857228 14.765390 15.407441 16.023785 16.617285 17.190307 17.744834
1981 20.763284
           Aug       Sep       Oct       Nov       Dec
1979 11.237707 11.808192 12.352358 12.873542 13.374432
1980 18.282549 18.804895 19.313118 19.808307 20.291414
1981
```

We could also include explanatory variables into this model to directly estimate the trend and the seasonality with `xreg` and `newxreg`. You'll use `model.matrix()[,-1]` to get the model matrix of explanatory variables to incorporate into `xreg` and create a version of that matrix with future values for `newxreg`.

Note about B-Splines: To use a spline to predict in the future, you must adjust the Boundary.knots to extend past where you want to predict for ***xreg*** *and* ***newxreg****.*

Below, I create a matrix of explanatory variables called `X` that includes the B-spline (note I extended the Boundary.knots to extend to the period I want to predict) and indicators for Month. I do this with `model.matrix()[,-1]`, removing the first column for the intercept. Then I create a matrix of those explanatory variables for times in the future, `NewX`. Based on this data set, I took the last 2 years and added 2 years to get the times for the next two years.

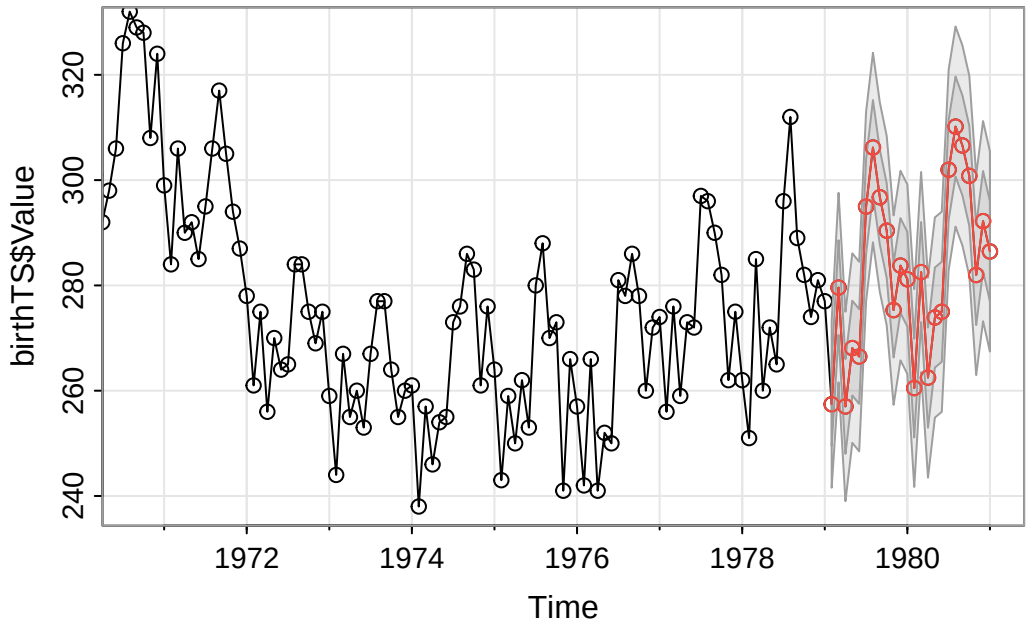
```
# One Model for Trend (B-spline - note how the Boundary Knots extend beyond the model and pre
TrendSeason <- birthTS %>%
  lm(Value ~ bs(Time, degree = 2, knots = c(1965, 1972), Boundary.knots = c(1948,1981)) + fa
```

```
X = model.matrix(TrendSeason)[,-1]

birthTS[nrow(birthTS),c('Time','Month')] #Last time observation
```

```
    Time Month
373 1979     1
```

```
newdat = data.frame(Month = c(2:12,1:12,1), Time = 1979 + (1:24)/12 )

NewX = model.matrix(~ bs(Time, degree = 2, knots = c(1965,1972), Boundary.knots = c(1948,198
```

```
sarima.for(birthTS$Value, n.ahead = 24, p = 0, d = 0, q = 1, P = 0, D = 0, Q = 1, S = 12, xre
```




```
$pred
          Jan      Feb      Mar      Apr      May      Jun      Jul      Aug
1979          257.4217 279.5911 257.0032 268.0920 266.4857 294.9724 306.1982
1980 281.1732 260.4987 282.5277 262.4848 273.9019 274.9926 301.9751 310.1671
1981 286.4452
          Sep      Oct      Nov      Dec
1979 296.7487 290.3933 275.3080 283.7623
1980 306.5751 300.8098 281.9722 292.2112
1981

$se
          Jan      Feb      Mar      Apr      May      Jun      Jul      Aug
1979          7.923840 8.999167 8.999167 8.999167 8.999167 8.999167 8.999167
1980 8.999167 9.394536 9.506053 9.506053 9.506053 9.506053 9.506053 9.506053
1981 9.506053
          Sep      Oct      Nov      Dec
1979 8.999167 8.999167 8.999167 8.999167
1980 9.506053 9.506053 9.506053 9.506053
1981
```

5.11 Appendix

5.11.1 Derivations for AR(1) Model

The model is stated

$$Y_t = \delta + \phi_1 Y_{t-1} + W_t$$

where $W_t \overset{iid}{\sim} N(0, \sigma_w^2)$ is **independent** Gaussian white noise with mean 0 and constant variance, σ_w^2.

5.11.1.1 Approach 1: Assume Stationarity

Expected Value

$$\begin{aligned} E(Y_t) = E(\delta + \phi_1 Y_{t-1} + W_t) &= E(\delta) + E(\phi_1 Y_{t-1}) + E(W_t) \\ &= \delta + \phi_1 E(Y_{t-1}) + 0 \\ &= \delta + \phi_1 E(Y_t) \end{aligned}$$

if the process is stationary (constant expected value).

If we solve for $E(Y_t)$, we get that $E(Y_t) = \mu = \frac{\delta}{1-\phi_1}$.

Variance

By the independence of current error and Y_{t-1} (as Y_{t-1} is only a function of past errors),

$$\begin{aligned} Var(Y_t) = Var(\delta + \phi_1 Y_{t-1} + W_t) &= Var(\phi_1 Y_{t-1} + W_t) \\ &= Var(\phi_1 Y_{t-1}) + Var(W_t) \\ &= \phi_1^2 Var(Y_{t-1}) + \sigma_w^2 \\ \\ &= \phi_1^2 Var(Y_t) + \sigma_w^2 \end{aligned}$$

if the process is stationary.

If we solve for $Var(Y_t)$, then we find that

$$Var(Y_t) = \frac{\sigma_w^2}{1 - \phi_1^2}$$

Since $Var(Y_t) > 0$, then $1 - \phi_1^2 > 0$ and therefore $|\phi_1| < 1$.

Covariance

Let's assume that the data have a mean of 0 (we've removed the trend and seasonality) such that $\delta = 0$ and $Y_t = \phi_1 Y_{t-1} + W_t$.

Then the $Cov(Y_t, Y_{t-k}) = E(Y_t Y_{t-k})$. For $k = 1$ (observations 1 time unit apart),

$$\begin{aligned}
\Sigma_Y(1) = Cov(Y_t, Y_{t-1}) = E(Y_t Y_{t-1}) &= E((\phi_1 Y_{t-1} + W_t) Y_{t-1}) \\
&= E(\phi_1 Y_{t-1}^2 + W_t Y_{t-1}) \\
= \phi_1 E(Y_{t-1}^2) + E(W_t) E(Y_{t-1}) &\text{ since } Cov(W_t, Y_{t-1}) = 0 \\
&= \phi_1 Var(Y_{t-1}) + 0 E(Y_{t-1}) \\
&= \phi_1 Var(Y_t)
\end{aligned}$$

Thus,

$$\rho_1 = Cor(Y_t, Y_{t-1}) = \frac{Cov(Y_t, Y_{t-1})}{Var(Y_t)} = \frac{\phi_1 Var(Y_t)}{Var(Y_t)} = \phi_1$$

For $k > 1$, multiple each side of the model by Y_{t-k} and take the expectation.

$$Y_t = \phi_1 Y_{t-1} + W_t$$

$$\begin{aligned}
Y_{t-k} Y_t &= \phi_1 Y_{t-k} Y_{t-1} + Y_{t-k} W_t \\
E(Y_{t-k} Y_t) &= E(\phi_1 Y_{t-k} Y_{t-1}) + E(Y_{t-k} W_t)
\end{aligned}$$

$$\begin{aligned}
Cov(Y_{t-k}, Y_t) &= \phi_1 Cov(Y_{t-k}, Y_{t-1}) \\
\Sigma_Y(k) &= \phi_1 \Sigma_Y(k-1)
\end{aligned}$$

If we work recursively, starting with $\Sigma_Y(1)$, we get

$$\begin{aligned}
\Sigma_Y(1) &= \phi_1 \Sigma_Y(0) \\
\Sigma_Y(2) &= \phi_1 \Sigma_Y(1) = \phi_1^2 \Sigma_Y(0) \\
\Sigma_Y(3) &= \phi_1 \Sigma_Y(2) = \phi_1^3 \Sigma_Y(0)
\end{aligned}$$

$$\Sigma_Y(k) = \phi_1^k \Sigma_Y(0)$$

So the correlation for observations k lags apart is

$$\rho_k = \frac{\Sigma_Y(k)}{Var(Y_t)} = \frac{\phi_1^k Var(Y_t)}{Var(Y_t)} = \phi_1^k$$

5.11.1.2 Approach 2: Write AR(1) as MA(∞)

$$Y_t = \delta + \phi_1 Y_{t-1} + W_t$$

where $t = ..., -3, -2, -1, 0, 1, 2, 3, ...$

By successive substitution, we can rewrite this model,

$$\begin{aligned} Y_t &= \delta + \phi_1(\delta + \phi_1 Y_{t-2} + W_{t-1}) + W_t = \delta(1+\phi_1) + \phi_1^2 Y_{t-2} + \phi_1 W_{t-1} + W_t \\ &= \delta(1+\phi_1) + \phi_1^2(\delta + \phi_1 Y_{t-3} + W_{t-2}) + \phi_1 W_{t-1} + W_t \\ &= \delta(1+\phi_1+\phi_1^2) + \phi_1^3 Y_{t-3} + \phi_1^2 W_{t-2} + \phi_1 W_{t-1} + W_t \\ &= \delta(1+\phi_1+\phi_1^2+\cdots) + W_t + \phi_1 W_{t-1} + \phi_1^2 W_{t-2} + \phi_1^3 W_{t-3} + \cdots \end{aligned}$$

Using infinite geometric series ($\sum_{i=0}^{\infty} r^i = (1-r)^{-1}$ if $|r| < 1$), the first part converges to $\frac{\delta}{1-\phi_1}$ if $|\phi_1| < 1$. Similarly, the sum of random variables converges to a finite value when $|\phi_1| < 1$.

With this format, it is clear that for an AR(1) process,

$$E(Y_t) = E(\frac{\delta}{1-\phi_1} + W_t + \phi_1 W_{t-1} + \phi_1^2 W_{t-2} + \cdots) = \frac{\delta}{1-\phi_1}$$

$$\begin{aligned} Var(Y_t) &= Var(\frac{\delta}{1-\phi_1} + W_t + \phi_1 W_{t-1} + \phi_1^2 W_{t-2} + \cdots) \\ &= \sigma_w^2(1 + \phi_1^2 + \phi_1^4 + \cdots) \\ &= \frac{\sigma_w^2}{1-\phi_1^2} \text{ if } |\phi_1| < 1 \end{aligned}$$

Let $h > 0$, then

$$Cov(Y_t, Y_{t-h}) = Cov(\frac{\delta}{1-\phi_1} + W_t + \phi_1 W_{t-1} + \phi_1^2 W_{t-2} + \cdots, \frac{\delta}{1-\phi_1} + W_{t-h} + \phi_1 W_{t-h-1} + \phi_1^2 W_{t-h-2} + \cdots)$$

$$= Cov(W_t + \phi_1 W_{t-1} + \phi_1^2 W_{t-2} + \cdots, W_{t-h} + \phi_1 W_{t-h-1} + \phi_1^2 W_{t-h-2} + \cdots)$$

$$= \sum_{j=0}^{\infty} Cov(\phi_1^{h+j} W_{t-h-j}, \phi_1^j W_{t-h-j})$$

$$= \sum_{j=0}^{\infty} \phi_1^{h+2j} Var(W_{t-h-j})$$

$$= \phi_1^h \sigma_w^2 \sum_{j=0}^{\infty} \phi_1^{2j}$$

$$= \frac{\phi_1^h \sigma_w^2}{1-\phi_1^2} \text{ if } |\phi_1| < 1$$

Thus,

$$Cor(Y_t, Y_{t-h}) = \frac{Cov(Y_t, Y_{t-h})}{SD(Y_t)SD(Y_{t-h})} = \frac{\phi_1^h \sigma_w^2}{1-\phi_1^2} / \frac{\sigma_w^2}{1-\phi_1^2}$$
$$= \phi_1^h$$

5.12 Other Time Series References

To supplement these notes, please check out the following time series textbooks for a more thorough introduction to the material.

- "Time series: a data analysis approach using R" by Robert Shunway and David Stoffer (Shunway and Stoffer 2019)
- "The Analysis of Time Series: An Introduction" by Christopher Chatfield (Chatfield and Xing 2019) is available at the Macalester Library.
- "Time Series Analysis with Applications in R" by Jonathan Cryer and Kung-Sik Chan (Cryer and Chan 2008)

6 Longitudinal Data

Although the term **longitudinal** naturally suggests that data are collected **over time**, the models and methods we will discuss broadly apply to any kind of **repeated measurement** data. That is, although repeated measurements most often take place over time, this is not the only way that measures may be taken repeatedly on the same unit. For example,

- The units may be human subjects. For each subject, reduction in diastolic blood pressure is measured on several occasions, each involving the administration of a different dose of anti-hypertensive medication. Thus, the subject is measured repeatedly with varying doses.
- The units may be trees in a forest. For each tree, measurements of the tree's diameter are made at several different points along the tree's trunk. Thus, the tree is measured repeatedly over positions along the trunk.
- The units may be pregnant female rats. Each rat gives birth to a litter of pups, and the birth weight of each pup is recorded. Thus, the rat is measured repeatedly over each of her pups. The third example is slightly different from the other two in that there is no natural **order** to the repeated measurements.

Thus, the methods will apply more broadly than the strict definition of the term **longitudinal data** indicates – the term will mean, to us, data in the form of **repeated measurements** that may well be over time but may also be over some other set of conditions. Because time is most often the measurement condition, however, many of our examples will involve repeated measurement over time. We will use the term **outcome** to denote the measurement of interest. Because units are often human or animal subjects, we use the terms **unit, individual, and subject** interchangeably.

6.1 Sources of Variation

We now consider why the values of a characteristic that we might observe vary over time.

- *Biological variation*: It is well-known that biological entities are different. However, living things of the same type tend to be similar in their characteristics; they are not the same (except perhaps in the case of genetically-identical clones). Thus, even if we focus on rats of the same genetic strain, age, and gender, we expect variation in the

possible weights of such rats that we might observe due to inherent, natural **biological variation.** This is variation due to genetics, environmental, and behavioral factors. Thus, this variation is at the **unit-level**, so we see **between unit** variation.

- *Variation due to Condition or Time* : Entities change over time and under different conditions. Suppose we consider rats over time or under various dietary conditions. In that case, we expect variation in the possible weights of such rats that we might observe due to **variation due to condition or time.** Thus, this variation is at the **observation-level** so we see **within unit** variation.
- *Measurement error*: We have discussed rat weight as though once we have a rat in hand, we may know its weight exactly. However, a scale must be used. Ideally, a scale should register the true weight of an item each time it is weighed, but because such devices are imperfect, scale measurements on the same item may vary time after time. The amount by which the measurement differs from the truth may be considered an **error**, i.e., a deviation up or down from the true value that could be observed with a perfect device. A fair or **unbiased** device does not systematically register high or low most of the time; rather, the errors may go in either direction with no pattern. Thus, this variation is typically at the **observation-level**, so we see **within unit** variation.

There are still further sources of variation that we could consider. They could be at the unit-level or the observational-level. For now, the important message is that, in considering statistical models, it is critical to be aware of different **sources of variation** that cause observations to vary.

With **cross-sectional data** (one observed point in time in units sampled from across the population - no repeated measures), we

- **cannot distinguish** between the types of variation
- **use explanatory variables** to try and explain any sources of variation (biological and other)
- use model error as a catch-all to account for any **leftover** variation (measurement or other)

With **longitudinal data**, since we have repeated measurements on units, we

- separate variation **within** units (measurement and others) from variation **between** units (biological and others)
- **use time-varying explanatory variables** to try and explain **within unit** variation
- **use time-invariant explanatory variables** to try and explain **between unit** variation
- use probability models to account for **left over within or between variation**

6.2 Data Examples

We can consider several real datasets from various applications to put things into a firmer perspective. These will not only provide us with concrete examples of longitudinal data situations but also illustrate the range of ways that data may be collected and the types of measurements that may be of interest.

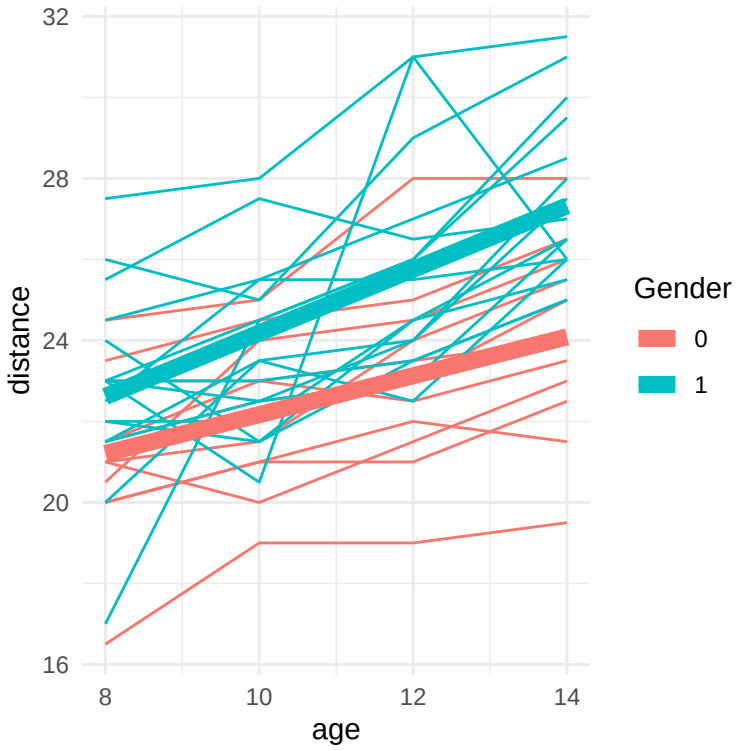
6.2.1 Example 1: The orthodontic study data of Potthoff and Roy (1964).

6.2.1.1 Data Context

A study was conducted involving 27 children, 16 boys and 11 girls. On each child, the distance (mm) from the center of the pituitary to the pterygomaxillary fissure was made at ages 8, 10, 12, and 14 years of age. In the plot below, the distance measurements are plotted against the age of each child. The trajectory for each child is connected by a solid line so that individual child patterns may be seen, and the color of the lines denotes girls (0) and boys (1).

```
dat <- read.table("./data/dental.dat", col.names=c("obsno", "id", "age", "distance", "gender"

dat %>%
  ggplot(aes(x = age, y = distance, col = factor(gender))) +
  geom_line(aes(group = id)) +
  geom_smooth(method='lm',se=FALSE,lwd=3) +
  theme_minimal() +
  scale_color_discrete('Gender')
```

```
`geom_smooth()` using formula = 'y ~ x'
```



Plots like this are often called **spaghetti plots**, for obvious reasons! Each set of connected points represented one child and their outcome measurements over time.

6.2.1.2 Research Questions

The objectives of the study were to

- Determine whether distances over time are larger for boys than for girls
- Determine whether the rate of change (i.e., slope) for distance is similar between boys and girls.

Several features are notable from the spaghetti plot of the data:

- Each child appears to have his/her own **trajectory** of distance as a function of age. For any given child, the trajectory looks roughly like a straight line, with some random fluctuations. But from child to child, the features of the trajectory (e.g., its steepness) vary. Thus, the trajectories are all similar in form but vary in their specific characteristics among children. Also, note any unusual trajectories. In particular, there is one boy whose pattern fluctuates more profoundly than those of the other children and one girl whose distance is much "lower" than the others at all time points.

- The overall trend is for the distance measurement to increase with age. The trajectories for some children exhibit strict increases with age. In contrast, others show some intermittent decreases (biologically, is this possible? or just due to measurement error?), but still with an overall increasing trend across the entire six-year period.
- The distance trajectories for boys seem, for the most part, to be "higher" than those for girls – most of the boy profiles involve larger distance measurements than those for girls. However, this is not uniformly true: some girls have larger distance measurements than boys at some ages.
- Although boys seem to have larger distance measurements, the **rate of change** of the measurements with increasing age seems similar. More precisely, the **slope** of the increasing (approximate straight-line) relationship with age seems roughly similar for boys and girls. However, for any individual boy or girl, the rate of change (slope) may be steeper or shallower than the evident "typical" rate of change.

To address the questions of interest, we need a formal way of representing the fact that each child has an individual-specific trajectory.

6.2.2 Example 2: Vitamin E diet supplement and growth of guinea pigs

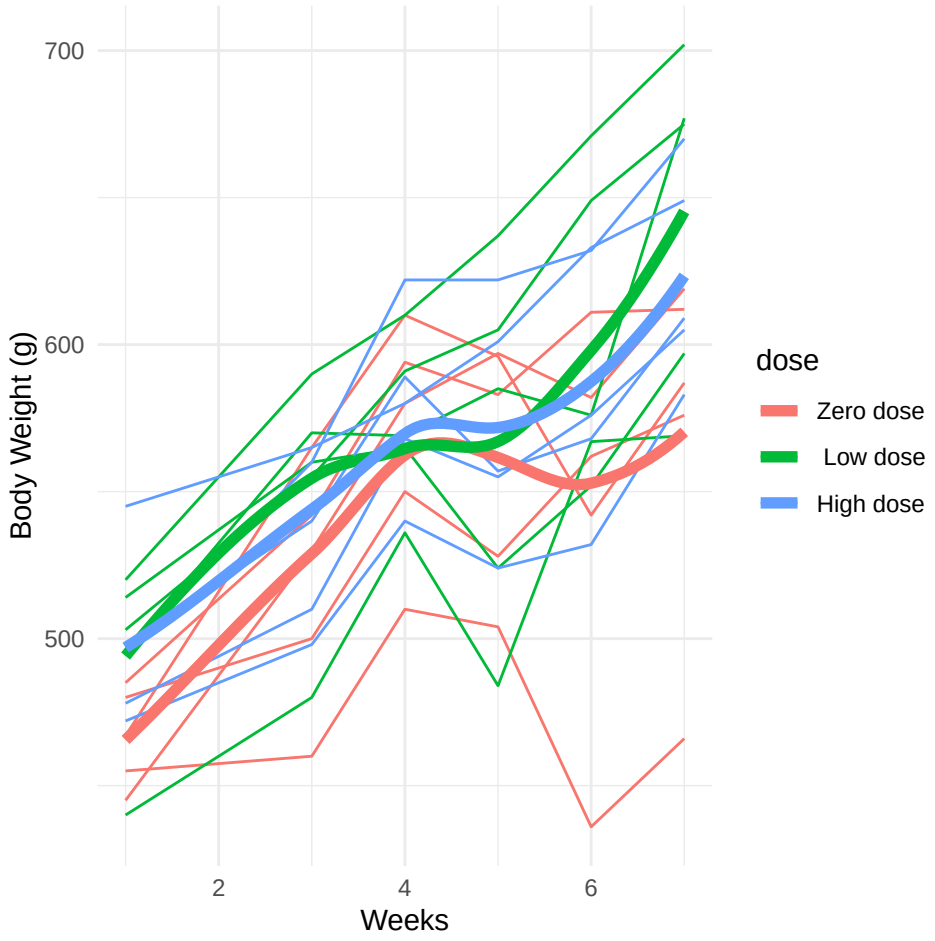
6.2.2.1 Data Context

The following data are reported by Crowder and Hand (1990, p. 27). The study concerned the effect of a vitamin E diet supplement on the growth of guinea pigs. 15 guinea pigs were given a growth-inhibiting substance at the beginning of week 1 of the study (time 0, before the first measurement), and body weight was measured at the ends of weeks 1, 3, and 4. At the beginning of week 5, the pigs were randomized into three groups of 5, and vitamin E therapy was started. One group received zero doses of vitamin E, another received a low dose, and the third received a high dose. Each guinea pig's body weight (g) was measured at the end of weeks 5, 6, and 7. The data for the three dose groups are plotted, in the plot below, with each line representing the body weight of one pig.

```
dat <- read.table("./data/diet.dat", col.names=c("id", paste("bw.",c(1,3,4,5,6,7),sep="")), "
dat <- dat %>%
  gather(Tmp,bw, bw.1:bw.7) %>%
  separate(Tmp,into=c('Var','time'), remove=TRUE)

dat$time <- as.numeric(dat$time)
dat$dose <- factor(dat$dose)
levels(dat$dose) <- c("Zero dose",' Low dose', 'High dose')
```

```r
dat %>%
  ggplot(aes(x = time, y = bw, col = dose)) +
  geom_line(aes( group = id)) +
  xlab('Weeks') +
  ylab('Body Weight (g)') +
  geom_smooth(lwd=2, se=FALSE) +
  theme_minimal()
```

```
`geom_smooth()` using method = 'loess' and formula = 'y ~ x'
```



6.2.2.2 Research Questions

The primary objective of the study was to

- Determine whether the growth patterns differed among the three groups.

As with the dental data, several features of the spaghetti plot are evident:

- For the most part, the trajectories for individual guinea pigs seem to increase overall over the study period (although note pig 1 in the zero dose group). Guinea pigs in the same dose group have different trajectories, some of which look like a straight line and others of which seem to have a "dip" at the beginning of week 5, the time at which vitamin E was added in the low and high-dose groups.
- The body weight for the zero dose group seems somewhat "lower" than those in the other dose groups.
- It is unclear whether the rate of change in body weight on average is similar or different across dose groups. It is unclear that the pattern for individual pigs or "on average" is a straight line, so the rate of change may not be constant. Because vitamin E therapy was not administered until the beginning of week 5, we might expect two "phases" in the growth pattern, before and after vitamin E, making it possibly non-linear.

6.2.3 Example 3: Epileptic seizures and chemotherapy

A common situation is where the measurements are in the form of counts. A response in the form of a **count** is by nature **discrete**–counts (usually) take only nonnegative integer values (0, 1, 2, 3,…).

6.2.3.1 Data Context

The following data were first reported by Thall and Vail (1990). A clinical trial was conducted in which 59 people with epilepsy suffering from simple or partial seizures were assigned at random to receive either the anti-epileptic drug progabide (subjects 29-59) or an inert substance (a placebo, subjects 1-28) in addition to a standard chemotherapy regimen all were taking. Because each individual might be prone to different rates of experiencing seizures, the investigators first tried to get a sense of this by recording the number of seizures suffered by each subject over the 8 weeks before administering the assigned treatment. It is common in such studies to record such baseline measurements so that the effect of treatment for each subject may be measured relative to how that subject behaved before treatment.

Following the commencement of treatment, each subject's seizures were counted for four two-week consecutive periods. The age of each subject at the start of the study was also recorded, as it was suspected that the subject's age might be associated with the effect of the treatment.

The first 10 rows of the long format data for the study are shown below.

```
require(MASS)
head(epil,10)
```

```
   y     trt base age V4 subject period      lbase        lage
1  5 placebo   11  31  0       1      1 -0.7563538  0.11420370
2  3 placebo   11  31  0       1      2 -0.7563538  0.11420370
3  3 placebo   11  31  0       1      3 -0.7563538  0.11420370
4  3 placebo   11  31  1       1      4 -0.7563538  0.11420370
5  3 placebo   11  30  0       2      1 -0.7563538  0.08141387
6  5 placebo   11  30  0       2      2 -0.7563538  0.08141387
7  3 placebo   11  30  0       2      3 -0.7563538  0.08141387
8  3 placebo   11  30  1       2      4 -0.7563538  0.08141387
9  2 placebo    6  25  0       3      1 -1.3624896 -0.10090768
10 4 placebo    6  25  0       3      2 -1.3624896 -0.10090768
```

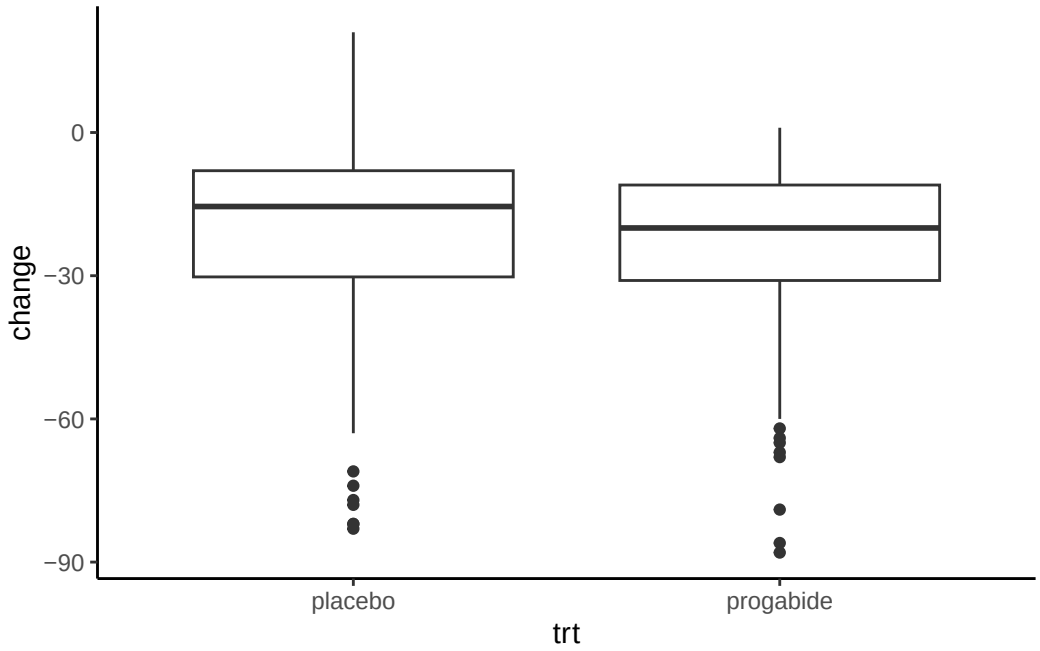
6.2.3.2 Research Questions

The primary objective of the study was to

- Determine whether progabide reduces the rate of seizures in subjects like those in the trial.

Here, we have repeated measurements (counts) on each subject over four consecutive observation periods for each subject. We would like to compare the baseline seizure counts to post-treatment counts, where the latter are observed repeatedly over time following the initiation of treatment. An appropriate analysis would best use this data feature in addressing the main objective. Below is a boxplot of the change from baseline, separated by treatment group (but ignores the repeated measures).

```
epil %>%
  mutate(change = y - base) %>%
  ggplot(aes(x = trt, y = change)) +
  geom_boxplot() +
  theme_classic()
```



Moreover, some counts are quite small; for some subjects, 0 seizures (none) were experienced in some periods. For example, subject 31 in the treatment group experienced only 0, 3, or 4 seizures over the four observation periods. Pretending that the response is continuous would be a lousy approximation of the true nature of the data! Thus, methods suitable for handling continuous data problems like the first three examples would not be appropriate for data like these.

A common approach to handling data in the form of counts is to transform them to some other scale. The motivation is to make them seem more "normally distributed" with constant variance, and the logarithm transformation is used to (hopefully) accomplish this. The desired result is that methods usually used to analyze continuous measurements may be applied.

However, the drawback of this approach is that one is no longer working with the data on the original scale of measurement, the number of seizures in this case. The statistical models the "log number of seizures," which is not particularly interesting or intuitive. New statistical methods have recently been developed to analyze discrete repeated measurements like counts on the original measurement scale.

6.2.4 Example 4: Maternal smoking and child respiratory health

Another common discrete data situation is where the response is binary; that is, the response may take on only two possible values, which usually correspond to things like

- success or failure of a treatment to elicit a desired response
- presence or absence of some condition

It would be foolish to even try and pretend such data are approximately continuous!

6.2.4.1 Data Context

The following data come from a very large public health study, the Six Cities Study, undertaken in six small American cities to investigate various public health issues. The full situation is reported in Lipsitz, Laird, and Harrington (1992). The current study focused on the association between maternal smoking and child respiratory health. Each of the 300 children was examined once a year at ages 9–12. The response of interest was "wheezing status, a measure of the child's respiratory health, which was coded as either"no" (0) or "yes" (1), where "yes" corresponds to respiratory problems. Also recorded at each examination was a code to indicate the mother's current level of smoking: 0 = none, 1 = moderate, 2 = heavy. The data for the first 5 subjects are summarized below. Missing data are denoted by a ".".

		Smoking at age				Wheezing at age			
Subject	City	9	10	11	12	9	10	11	12
1	Portage	2	2	1	1	1	0	0	0
2	Kingston	0	0	0	0	0	0	0	0
3	Portage	1	0	0	.	0	0	0	.
4	Portage	.	1	1	1	.	1	0	0
5	Kingston	1	.	1	2	0	.	0	1

A simplified version of this data is available in R.

```
require(geepack)
data(ohio)
head(ohio,10)
```

```
   resp id age smoke
1     0  0  -2     0
2     0  0  -1     0
3     0  0   0     0
4     0  0   1     0
5     0  1  -2     0
6     0  1  -1     0
7     0  1   0     0
8     0  1   1     0
9     0  2  -2     0
10    0  2  -1     0
```

6.2.4.2 Research Questions

The objective of an analysis of these data was to

- Determine how the typical "wheezing" response pattern changes with age
- Determine whether there is an association between maternal smoking severity and child respiratory status (as measured by "wheezing").

It would be pretty pointless to plot the responses as a function of age as we did in the continuous data cases – here, the only responses are 0 or 1! Inspection of subject data suggests something is happening here; for example, subject 5 did not exhibit positive wheezing until his/her mother's smoking increased in severity.

This highlights that this situation is complex: over time (measured here by the child's age), an important characteristic, maternal smoking, changes. Contrast this with the previous situations, where the main focus is to compare groups whose membership stays constant over time.

Thus, we have repeated measurements, which are binary to further complicate matters! As with the count data, one might first think about summarizing and transforming the data to allow methods for continuous data to be used; however, this would be inappropriate. As we will see later, methods for dealing with repeated binary responses and scientific questions like those above have been developed.

Another feature of these data is that some measurements are missing for some subjects. Specifically, although the intention was to collect data for each of the four ages, this information is not available for some children and their mothers at some ages; for example, subject 3 has both the mother's smoking status and wheezing indicator missing at age 12. This pattern would suggest that the mother may have failed to appear with the child for this intended examination.

A final note: In the other examples, units (children, guinea pigs, plots, patients) were assigned to treatments; thus, these may be regarded as controlled experiments, where the investigator has some control over how the factors of interest are "applied" to the units (through randomization). In contrast, in this study, the investigators did not decide which children would have mothers who smoke; instead, they could only observe the smoking behavior of the mothers and the wheezing status of their children. That is, this is an example of an observational study. Because it may be impossible or unethical to randomize subjects to potentially hazardous circumstances, studies of issues in public health and the social sciences are often observational.

As in many observational studies, an additional difficulty is the fact that the thing of interest, in this case, maternal smoking, also changes with the response over time. This leads to complicated issues of interpretation in statistical modeling that are a matter of some debate.

6.3 R: Wide V. Long Format

In R, longitudinal data could be formatted in two ways.

Wide Format: When the observations times are the same for every unit (*balanced data*), then every unit has a vector of observations of length m, and we can organize the unit's data into one row per unit, resulting in a data set with n rows and m columns that correspond to outcome values (plus other columns that will correspond to an identifier variable and other explanatory variables). See the simulated example below. The first column is an id column to identify the units from each other, the next five columns correspond to the $m = 5$ observations over time, and the next five columns correspond to a time-varying explanatory variable.

```
n = 10
m = 5

(wideD = data.frame(id = 1:n, y.1 = rnorm(n), y.2 = rnorm(n), y.3 = rnorm(n), y.4 = rnorm(n)
```

```
   id        y.1         y.2        y.3          y.4         y.5        x.1
1   1  0.4100407 -0.34321543  0.5232633 -1.052829343  1.48165146 -0.2042827
2   2 -0.8467906  0.17789644  1.0400565 -0.542453539 -0.24843630  0.9152467
3   3  1.3571810 -0.29790088  0.7761962 -1.235001897 -0.80134565 -1.0240244
4   4  1.0199928 -1.58509726  0.2117450  0.483401904  0.06145224 -1.0243878
5   5 -0.6113060 -0.94553592 -0.6712787 -0.001143426  1.39626716 -0.8845031
6   6 -0.6474944 -0.44313504  2.0020810 -0.348228908 -0.69534610  1.5344297
7   7  0.8235137  0.01242068 -1.2320169 -0.983035114 -1.83578330 -1.0199189
8   8  0.8274177  0.61786316 -1.3190720 -0.265427433 -0.69783134  0.4117536
9   9  0.6868499 -0.35821153  0.2568263  0.102128774  0.76948236 -0.8385356
10 10 -0.5295013  0.54496445  0.6888688  0.045921587 -0.01699081 -0.3105107
           x.2          x.3        x.4        x.5
1   1.57849534 -1.044084705  0.4587848  0.1526877
2   1.18647471  0.717216522 -1.3480238  0.8483097
3  -1.81019236  0.008583371  0.9924889 -1.2554282
4   1.03778747  0.020863907  0.2042637 -2.2165144
5   0.47364454 -1.353040200  0.2985420  1.2093907
6   0.20266436  0.559416375  0.6030863  0.7601944
7   0.67346316  1.099649736  0.2035936  0.1475575
8  -0.61694724  0.091468575 -0.4652250  0.7628235
9   0.47184070 -0.125490068  0.2503304  1.5399445
10  0.04164016 -0.840996844 -1.1681541  0.4670689
```

Long Format: We'll need the data in long format for most data analysis. We must use a long format when every unit's observation times differ. Imagine stacking each unit's observed

outcome vectors on top of each other. Similarly, we want to stack the observed vectors of any other explanatory variable we might have on the individuals. In contrast to wide format, it is necessary to have a variable to identify the unit and the observation time.

See below for the R code to convert a data set in wide format to one in long format. Hopefully, the variable names are given in a way that specifies the variable name and the time order of the values, such as `y.1,y.2,...y.5,x.1,x.2,...,x.5`.

In the tidyr package, `pivot_longer()` takes the wide data, the columns (`cols`) you want to make longer, and the names of two variables you want to create. The first (we use `Tmp` here) is the variable name containing the variable names of the columns you want to gather. The second (called `Value`) is the variable name that will collect the values from those columns. See below.

```
require(tidyr)

pivot_longer(wideD, cols = y.1:x.5, names_to = 'Tmp', values_to = 'Value') %>% head()
```

```
# A tibble: 6 x 3
     id Tmp    Value
  <int> <chr>  <dbl>
1     1 y.1    0.410
2     1 y.2   -0.343
3     1 y.3    0.523
4     1 y.4   -1.05
5     1 y.5    1.48
6     1 x.1   -0.204
```

Then, we want to separate the `Tmp` variable into two variables because they contain information about both the characteristic and time. We can use `separate()` to separate `Tmp` into `Var` and `Time`, it will automatically detect the `.` as a separator.

```
pivot_longer(wideD, cols = y.1:x.5, names_to = 'Tmp', values_to = 'Value') %>%
  separate(Tmp,into=c('Var','Time'), remove=TRUE)  %>%
  head()
```

```
# A tibble: 6 x 4
     id Var   Time   Value
  <int> <chr> <chr>  <dbl>
1     1 y     1      0.410
2     1 y     2     -0.343
3     1 y     3      0.523
```

```
4     1 y     4     -1.05
5     1 y     5      1.48
6     1 x     1     -0.204
```

Lastly, we want to have one variable called `x` and one variable called `y', and we can get that by pivoting the variable`Varwider into two columns with the values that come from`Value‘.

```
pivot_longer(wideD, cols = y.1:x.5, names_to = 'Tmp', values_to = 'Value') %>%
  separate(Tmp,into=c('Var','Time'), remove=TRUE) %>%
  pivot_wider(names_from = 'Var', values_from = 'Value') %>%
  head()
```

```
# A tibble: 6 x 4
     id Time       y      x
  <int> <chr>  <dbl>  <dbl>
1     1 1      0.410 -0.204
2     1 2     -0.343  1.58
3     1 3      0.523 -1.04
4     1 4     -1.05   0.459
5     1 5      1.48   0.153
6     2 1     -0.847  0.915
```

6.4 Notation

In order to

- elucidate the assumptions made under different models and methods, and
- describe the models and methods more easily,

it is convenient to think of all outcomes collected on the same unit over time or another set of conditions together so that complex relationships among them may be summarized.

Consider the random variable,

$$Y_{ij} = \text{the } j\text{th measurement taken on unit } i, \qquad i = 1, ..., n, j = 1, ..., m$$

Consider the dental study data (Example 1) to make this more concrete. Each child was measured 4 times at ages 8, 10, 12, and 14 years. Thus, we let $j = 1, ..., 4$; j index the measurement order on a child. To summarize the information on when these times occur, we might further define

$$t_{ij} = \text{ the time at which the } j \text{ measurement on unit i was taken.}$$

Here, for all children ($i = 1, ..., 27$), $t_{i1} = 8, t_{i2} = 10$, and so on for all children in the study. Thus, $t_{ij} = t_j$ for all $i = 1, ..., n$. If we ignore the gender of the children for the moment, the outcomes for the ith child, where i ranges from 1 to 27, are $Y_{i1}, ..., Y_{i4}$, taken at times $t_{i1}, ..., t_{i4}$. We may summarize the measurements for the ith child even more succinctly: define the (4×1) random vector,

$$\mathbf{Y}_i = \begin{pmatrix} Y_{i1} \\ Y_{i2} \\ Y_{i3} \\ Y_{i4} \end{pmatrix}.$$

The vector elements are random variables representing the outcomes that might be observed for child i at each time point. For this data set, the data are **balanced** because the observation times are the same for each unit, and the data are **regular** because the observation times are equally spaced apart. Most observational data is not balanced and is often irregular. We can generalize our notation to allow for this type of data by changing m to m_i, which captures the total number of measurements for the ith unit,

$$Y_{ij} = \text{the } j\text{th measurement taken on unit } i, \qquad i = 1, ..., n, j = 1, ..., m_i$$

The important message is that it is possible to represent the outcomes for the ith child in a very streamlined and convenient way. Each child i has its outcome vector, $\mathbf{Y}_i$. It often makes sense to think of the data not just as individual outcomes Y_{ij}, some from one child, some from another according to the indices, but rather as vectors corresponding to children, the units–each unit has associated with it an entire outcome vector.

We can also consider explanatory variables. If we have p explanatory variables, we'll let the value for the 1st variable, for the ith unit at the jth time, be represented as x_{ij1}. That means that for the ith unit, we can collect all of their values of explanatory variables (across time) in a matrix,

$$\mathbf{X}_i = \begin{pmatrix} x_{i11} & x_{i12} & \cdots & x_{i1p} \\ x_{i21} & x_{i22} & \cdots & x_{i2p} \\ \vdots & \vdots & \ddots & \vdots \\ x_{im1} & x_{im2} & \cdots & x_{imp} \end{pmatrix}$$

These explanatory variables might be **time-invariant** such that we have a variable like the treatment group. Alternatively, we might have explanatory variables that are **time-varying**, such as age in the values observed at each time point may change.

6.4.1 Multivariate Normal Probability Model

We first discussed this in Chapter 2, so feel free to return to Random Vectors and Matrices.

When we represent the outcomes for the ith unit as a random vector, $\mathbf{Y}_i$, it is useful to consider a multivariate model such as the **multivariate normal probability model**.

The joint probability distribution that is the extension of the univariate version to a $(m \times 1)$ random vector $\mathbf{Y}$, each of whose components is normally distributed, is given by

$$f(\mathbf{y}) = \frac{1}{(2\pi)^{m/2}} |\Sigma|^{-1/2} \exp\{-(\mathbf{y}-\mu)^T \Sigma^{-1} (\mathbf{y}-\mu)/2\}$$

- This probability density function describes the probabilities with which the random vector $\mathbf{Y}$ takes on values jointly in its m elements.
- The form is determined by the mean vector μ and covariance matrix Σ.

The form of $f(\mathbf{y})$ depends critically on the term

$$(\mathbf{y}-\mu)^T \Sigma^{-1} (\mathbf{y}-\mu)$$

The quadratic form of the term pops up in many common methods. You'll see it in generalized least squares (GLS) and the Mahalanobis distance as a **standardized sum of squares**. Read the next section if you'd like to think more deeply about this quadratic form.

6.4.1.1 Quadratic Form (Optional)

The pdf of the multivariate normal pdf depends critically on the term

$$(\mathbf{y}-\mu)^T \Sigma^{-1} (\mathbf{y}-\mu)$$

This is a **quadratic form** (in linear algebra), so it is a scalar function of the elements of $(\mathbf{y}-\mu)$ and Σ^{-1}.

If we refer to the elements of Σ^{-1} as σ^{jk}, i.e.

$$\Sigma^{-1} = \begin{pmatrix} \sigma^{11} & \cdots & \sigma^{1m} \\ \vdots & \ddots & \vdots \\ \sigma^{m1} & \cdots & \sigma^{mm} \end{pmatrix}$$

then we may write

$$(\mathbf{y}-\mu)^T \Sigma^{-1} (\mathbf{y}-\mu) = \sum_{j=1}^{m} \sum_{k=1}^{m} \sigma^{jk} (y_j - \mu_j)(y_k - \mu_k).$$

Of course, the elements σ^{jk} will be complicated functions of the elements σ_j^2, σ_{jk} of Σ, i.e. the variances of the Y_j and the covariances among them.

- This term thus depends on not only the **squared deviations** $(y_j - \mu_j)^2$ for each element in $\mathbf{y}$ (which arise in the double sum when $j = k$), but also on the crossproducts $(y_j - \mu_j)(y_k - \mu_k)$. Each contribution of these squares and cross products is standardized by values σ^{jk} that involve the variances and covariances.
- Thus, although it is quite complicated, one gets the suspicion that the quadratic form has an interpretation, albeit more complex, as a **distance measure**, just as in the univariate case.

To better understand the multivariate distribution, it is instructive to consider the special case $m = 2$, the simplest example of a multivariate normal distribution (hence the name **bivariate**).

Here,

$$\mathbf{Y} = \begin{pmatrix} Y_1 \\ Y_2 \end{pmatrix}, \mu = \begin{pmatrix} \mu_1 \\ \mu_2 \end{pmatrix}, \Sigma = \begin{pmatrix} \sigma_1^2 & \sigma_{12} \\ \sigma_{12} & \sigma_2^2 \end{pmatrix}$$

Using the inversion formula for a (2×2) matrix,

$$\Sigma^{-1} = \frac{1}{\sigma_1^2\sigma_2^2 - \sigma_{12}^2} \begin{pmatrix} \sigma_2^2 & -\sigma_{12} \\ -\sigma_{12} & \sigma_1^2 \end{pmatrix}$$

We also have that the **correlation** between Y_1 and Y_2 is given by

$$\rho_{12} = \frac{\sigma_{12}}{\sigma_1\sigma_2}.$$

Using these results, it is an algebraic exercise to show that (try it!)

$$(\mathbf{y} - \mu)^T\Sigma^{-1}(\mathbf{y} - \mu) = \frac{1}{1 - \rho_{12}^2}\left\{\frac{(y_1 - \mu_1)^2}{\sigma_1^2} + \frac{(y_2 - \mu_2)^2}{\sigma_2^2} - 2\rho_{12}\frac{(y_1 - \mu_1)}{\sigma_1}\frac{(y_2 - \mu_2)}{\sigma_2}\right\}$$

- One component is the sum of squared standardized values (z-scores)

$$\frac{(y_1 - \mu_1)^2}{\sigma_1^2} + \frac{(y_2 - \mu_2)^2}{\sigma_2^2}$$

This sum is similar to the sum of squared deviations in least squares, with the difference that each deviation is now weighted by its variance. This makes sense–because the variances of Y_1 and Y_2 differ, information on the population of Y_1 values is of a different quality than that on the population of Y_2 values. If the variance is large, the quality of information is poorer; thus, the larger the variance, the smaller the weight, so that information of higher quality receives more weight in the overall measure. Indeed, this is like a distance measure, where each contribution receives an appropriate weight.

- In addition, there is an extra term that makes it have a different form than just a sum of weighted squared deviations:

$$-2\rho_{12}\frac{(y_1-\mu_1)}{\sigma_1}\frac{(y_2-\mu_2)}{\sigma_2}$$

This term depends on the cross-product, where each deviation is again weighted by its variance. This term modifies the distance measure in a way connected with the association between Y_1 and Y_2 through their cross-product and correlation ρ_{12}. Note that the larger this correlation in magnitude (either positive or negative), the more we modify the usual sum of squared deviations.

- Note that the entire quadratic form also involves the multiplicative factor $1/(1-\rho_{12}^2)$, which is greater than 1 if $|\rho_{12}| > 0$. This factor scales the overall distance measure by the magnitude of the association.

6.5 Failure of Standard Estimation Methods

6.5.1 Ordinary Least Squares

Imagine we have n units/subjects, and each unit has m observations over time. Conditional on p explanatory variables, we typically assume a linear model for the relationship between the explanatory variables and the outcome,

$$Y_{ij} = \beta_0 + \beta_1 x_{ij1} + \cdots + \beta_p x_{ijp} + \epsilon_{ij}$$

If we are using ordinary least squares to estimate the slope parameters, we assume the errors represent independent measurement error, $\epsilon_{ij} \overset{iid}{\sim} \mathcal{N}(0, \sigma^2)$.

This can be written using vector and matrix notation with each unit having its outcome vector, $\mathbf{Y}_i$,

$$\mathbf{Y}_i = \mathbf{X}_i\beta + \epsilon_i \text{ for each unit } i$$

where $\epsilon_i \sim \mathcal{N}(0, \sigma^2 I)$ and $\mathbf{X}_i$ is the observations for the explanatory variables for subject i with m rows and $p+1$ columns (a column of 1's for the intercept plus p columns that correspond to the p variables).

If we stack all of our outcome vectors $\mathbf{Y}$ (and covariate matrices $\mathbf{X}$) on top of each other (think Long Format in R),

$$\mathbf{Y} = \mathbf{X}\beta + \epsilon$$

where $\epsilon \sim \mathcal{N}(0, \sigma^2 I)$ so we are assuming that our errors and thus our data are independent within and between each unit and have constant variance.

6.5.1.1 Estimation

To find the ordinary Least Squares (OLS) estimate $\hat{\beta}_{OLS}$, we need to minimize the sum of squared residuals, which can be written with vector and matrix notation,

$$(\mathbf{Y} - \mathbf{X}\beta)^T(\mathbf{Y} - \mathbf{X}\beta)$$

Taking the derivative with respect to $(\beta_0, \beta_1, ..., \beta_p)$, we get a set of simultaneous equations expressed in matrix notation as

$$-2\mathbf{X}^T\mathbf{Y} + 2\mathbf{X}^T\mathbf{X}\beta = \mathbf{0}$$

As long as $\mathbf{X}^T\mathbf{X}$ has an inverse, then the unique solution and our estimated parameters can be found using

$$\hat{\beta}_{OLS} = (\mathbf{X}^T\mathbf{X})^{-1}\mathbf{X}^T\mathbf{Y}$$

This estimator is unbiased in that it estimates the true value on average, $E(\hat{\beta}_{OLS}) = \beta$.

Derivation:

$$\begin{aligned} E(\hat{\beta}_{OLS}) &= (\mathbf{X}^T\mathbf{X})^{-1}\mathbf{X}^T E(\mathbf{Y}) \\ &= (\mathbf{X}^T\mathbf{X})^{-1}\mathbf{X}^T\mathbf{X}\beta \\ &= \beta \end{aligned}$$

It also has a minimum variance of all unbiased, linear estimators IF the errors are independent and have constant variance.

Derivation:

$$\begin{aligned} Cov(\hat{\beta}_{OLS}) &= (\mathbf{X}^T\mathbf{X})^{-1}\mathbf{X}^T Cov(\mathbf{Y})\{(\mathbf{X}^T\mathbf{X})^{-1}\mathbf{X}^T\}^T \\ &= (\mathbf{X}^T\mathbf{X})^{-1}\mathbf{X}^T(\sigma^2\mathbf{I})\mathbf{X}(\mathbf{X}^T\mathbf{X})^{-1} \\ &= \sigma^2(\mathbf{X}^T\mathbf{X})^{-1} \end{aligned}$$

using $Cov(\mathbf{AY}) = \mathbf{A}Cov(\mathbf{Y})\mathbf{A}^T$, which you can verify.

The OLS estimates of our coefficients are fairly good (unbiased), but the OLS standard errors are wrong unless our data are independent. Since we have correlated repeated measures, we don't have as much "information" as if they were independent observations.

6.5.2 Generalized Least Squares

If you relax the assumption of constant variance, then the covariance matrix of ϵ_i and thus $\mathbf{Y}_i$ is

$$\Sigma_i = \begin{pmatrix} \sigma_1^2 & 0 & \cdots & 0 \\ 0 & \sigma_2^2 & \cdots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \cdots & \sigma_m^2 \end{pmatrix}$$

For longitudinal data, this would allow the variability to change over time.

If you relax both the assumptions of constant variance and independence, then the covariance matrix of ϵ_i and thus $\mathbf{Y}_i$ is

$$\Sigma_i = \begin{pmatrix} \sigma_1^2 & \sigma_{12} & \cdots & \sigma_{1m} \\ \sigma_{21} & \sigma_2^2 & \cdots & \sigma_{2m} \\ \vdots & \vdots & \ddots & \vdots \\ \sigma_{m1} & \sigma_{m2} & \cdots & \sigma_m^2 \end{pmatrix}$$

For longitudinal data, this would allow the variability to change over time, and you can account for dependence between repeated measures.

If we assume individual units are independent of each other (which is typically a fair assumption), the covariance matrix for the entire vector of outcomes (for all units and their observations over time) is written as a block diagonal matrix,

$$\Sigma = \begin{pmatrix} \Sigma_1 & \mathbf{0} & \cdots & \mathbf{0} \\ \mathbf{0} & \Sigma_2 & \cdots & \mathbf{0} \\ \vdots & \vdots & \ddots & \vdots \\ \mathbf{0} & \mathbf{0} & \cdots & \Sigma_n \end{pmatrix}$$

6.5.2.1 Estimation

To find the Generalized Least Squares (GLS) estimator $\hat{\beta}$, we need to minimize the standardized sum of squared residuals,

$$(\mathbf{Y} - \mathbf{X}\beta)^T \Sigma^{-1} (\mathbf{Y} - \mathbf{X}\beta)$$

If Σ is known, taking the derivative with respect to $\beta_0, \beta_1, ..., \beta_p$, we get a set of simultaneous equations expressed in matrix notation as

$$-2\mathbf{X}^T\Sigma^{-1}\mathbf{Y} + 2\mathbf{X}^T\Sigma^{-1}\mathbf{X}\beta = \mathbf{0}$$

As long as $\mathbf{X}^T\Sigma^{-1}\mathbf{X}$ has an inverse, then the **generalized least squares** (GLS) estimator is

$$\hat{\beta}_{GLS} = (\mathbf{X}^T\Sigma^{-1}\mathbf{X})^{-1}\mathbf{X}^T\Sigma^{-1}\mathbf{Y}$$

6.5.2.1.1 Alternative Derivation of GLS

An alternative approach is to consider transforming our $\mathbf{Y}$ and $\mathbf{X}$ by pre-multiplying by the inverse of the lower triangular matrix from the Cholesky Decomposition of the covariance matrix Σ ($\mathbf{L}$), and then find the OLS estimator.

We start by transforming our data so that $\mathbf{Y}^* = \mathbf{L}^{-1}\mathbf{Y}$, $\mathbf{X}^*\mathbf{L}^{-1}\mathbf{X}$, and $\epsilon^* = \mathbf{L}^{-1}\epsilon$.

We can rewrite the model as,

$$\mathbf{L}^{-1}\mathbf{Y} = \mathbf{L}^{-1}\mathbf{X}\beta + \mathbf{L}^{-1}\epsilon$$
$$\implies \mathbf{Y}^* = \mathbf{X}^*\beta + \epsilon^*$$

Then, using OLS on these transformed vectors, we get the generalized least squares estimator,

$$\begin{aligned}
\implies \hat{\beta}_{GLS} &= (\mathbf{X}^{*T}\mathbf{X}^*)^{-1}\mathbf{X}^{*T}\mathbf{Y}^* \\
&= ((\mathbf{L}^{-1}\mathbf{X})^T\mathbf{L}^{-1}\mathbf{X})^{-1}(\mathbf{L}^{-1}\mathbf{X})^T\mathbf{L}^{-1}\mathbf{Y} \\
&= (\mathbf{X}^T(\mathbf{L}^{-1})^T\mathbf{L}^{-1}\mathbf{X})^{-1}\mathbf{X}^T(\mathbf{L}^{-1})^T\mathbf{L}^{-1}\mathbf{Y} \\
&= (\mathbf{X}^T(\mathbf{L}^T)^{-1}\mathbf{L}^{-1}\mathbf{X})^{-1}\mathbf{X}^T(\mathbf{L}^T)^{-1}\mathbf{L}^{-1}\mathbf{Y} \\
&= (\mathbf{X}^T(\mathbf{L}\mathbf{L}^T)^{-1}\mathbf{X})^{-1}\mathbf{X}^T(\mathbf{L}\mathbf{L}^T)^{-1}\mathbf{Y} \\
&= (\mathbf{X}^T\Sigma^{-1}\mathbf{X})^{-1}\mathbf{X}^T\Sigma^{-1}\mathbf{Y}
\end{aligned}$$

6.5.2.2 Estimation Issues

Main Issue: Σ is unknown!

The solution to this problem is to iterate. Estimate β and use that to estimate Σ. Repeat.

Other Issues:

- If the longitudinal data is balanced, there are $m(m+1)/2$ parameters in Σ_i to estimate. That is a lot of parameters (!)
- If the longitudinal data is unbalanced, there is no common Σ_i for every unit

The solution to these problems involves **simplifying assumptions** such as

- assuming the errors are independent (not a great solution)
- assuming the errors have constant variance (maybe not too bad, depending on the data)
- assuming there is some restriction/structure on the covariance function (best option)

We approximate the covariance matrix to get better estimates of β.

If we know the correlation structure of our data, we could use generalized least squares to get better (better than OLS) estimates of our coefficients, and the estimated standard errors would be more accurate.

The two main methods for longitudinal data that we'll learn are similar to generalized least squares in flavor.

However, we need models that are not only applicable to continuous outcomes but also binary or count outcomes. Let's first learn about the standard cross-sectional approaches to allow for binary and count outcomes (**generalized linear models**). Then, we'll be able to talk about **marginal models** and **mixed effects models.**

6.6 Generalized Linear Models

When you have outcome data that is not continuous, we can't use a least squares approach as it is only appropriate for continuous outcomes. However, we can generalize the idea of a linear model to allow for binary or count outcomes. This is called a **generalized linear model** (GLM). GLM's extend regression to situations beyond the continuous outcomes with Normal errors (Nelder and Wedderburn 1972), and they are, in fact, a broad class of models for outcomes that are continuous, discrete, binary, etc.

GLM's requires a three-part specification:

1. Distributional assumption for Y
2. Systematic component with $\mathbf{X}$
3. Link function to relate $E(Y)$ with systematic component

6.6.1 Distributional Assumption

The first assumption you need to make to fit a GLM is to assume a distribution for the outcome Y.

Many distributions you have learned in probability (normal, Bernoulli, binomial, Poisson) belong to the **exponential family** of distributions that share a general form and statistical properties. GLM's are limited to this family of distributions.

One important statistical property of the exponential family is that the variance can be written as a scaled function of the mean,

$$Var(Y) = \phi v(\mu) \quad \text{where } E(Y) = \mu$$

where $\phi > 0$ is a dispersion or scale parameter and $v(\mu)$ is a variance function of the mean.

6.6.2 Systematic Component

For a GLM, the mean or a transformed mean can be expressed as a linear combination of explanatory variables, which we'll notate as η:

$$\eta = \beta_0 + \beta_1 X_1 + \beta_2 X_2 + \cdots + \beta_p X_p$$

You'll need to decide which explanatory variables should be used to model the mean. This may include a time variable (e.g., age, time since baseline, etc.) and other unit characteristics that are time-varying or time-invariant. We'll refer to this as the **mean model**.

6.6.3 Link Function

Lastly, the chosen link function transforms the mean and links the explanatory variables to that transformed mean.

$$g(\mu) = \eta = \beta_0 + \beta_1 X_1 + \beta_2 X_2 + \cdots + \beta_p X_p$$

This link function, $g()$, allows us to use a linear function to model positive counts and binary variables.

There are **canonical link functions** for each distribution in the exponential family.

Normal (linear regression)

- $v(\mu) = 1$

- $g(\mu) = \mu$ (identity)

Bernoulli/Binomial (m=1) (logistic regression)

- $v(\mu) = \mu(1-\mu)$
- $g(\mu) = \log(\mu/(1-\mu))$ (logit)

Binomial

- $v(\mu) = m\mu(1-\mu)$
- $g(\mu) = \log(\mu/(1-\mu))$ (logit)

Poisson (poisson regression)

- $v(\mu) = \mu$
- $g(\mu) = \log(\mu)$ (log)

For the Six City Study, we can fit a model to predict whether or not a child has respiratory issues as a function of age and maternal smoking, ignoring the repeated measures on each child with the following R code. Notice we need to specify the mean model using the formula notation `resp ~ age + smoke`, the family of the distribution we assume for our outcome `family = binomial` and the link function 'link = 'logit" we use to connect the linear model to the mean. With this set of assumptions, we are fitting a **logistic regression model**.

```
summary(glm(resp ~ age + smoke, data = ohio, family = binomial(link = 'logit')))
```

```
Call:
glm(formula = resp ~ age + smoke, family = binomial(link = "logit"),
    data = ohio)

Coefficients:
            Estimate Std. Error z value Pr(>|z|)
(Intercept) -1.88373    0.08384 -22.467   <2e-16 ***
age         -0.11341    0.05408  -2.097   0.0360 *
smoke        0.27214    0.12347   2.204   0.0275 *
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

(Dispersion parameter for binomial family taken to be 1)

    Null deviance: 1829.1  on 2147  degrees of freedom
Residual deviance: 1819.9  on 2145  degrees of freedom
AIC: 1825.9
```

```
Number of Fisher Scoring iterations: 4
```

Please see my Introduction to Statistical Models Notes to refresh your memory of interpreting logistic regression models.

6.7 Marginal Models

A marginal model for longitudinal data has a four-part specification, and the first three parts are similar to a GLM.

6.7.1 Model Specification

1. A **distributional assumption** about Y_{ij}. We assume a distribution from the exponential family. We will only use this assumption to determine the mean and variance relationship.
2. A **systematic component** which is the mean model, $\eta_{ij} = \mathbf{x}_{ij}^T\beta$, which get connected to the mean through
3. a **link function**: The conditional expectation $E(Y_{ij}|x_{ij}) = \mu_{ij}$ is assumed to depend on explanatory variables through a given link function (similar to GLM),

$$g(\mu_{ij}) = \eta_{ij} = \mathbf{x}_{ij}^T\beta$$

Note: This implies that given $\mathbf{x}_{ij}$, there is no dependence of Y_{ij} on $\mathbf{x}_{ik}$ for $j \neq k$ (warning: this might not hold if Y_{ij} predicts $\mathbf{x}_{ij+1}$).

The conditional variance of each Y_{ij}, given explanatory variables, depends on the mean according to

$$Var(Y_{ij}|\mathbf{x}_{ij}) = \phi v(\mu_{ij})$$

Note: The ϕ (phi parameter) is a *dispersion parameter* and scales variance up or down.

4. **Covariance Structure**: The conditional within-subject association among repeated measures is assumed to be a function of a set of association parameters α (and the means μ_{ij}). We choose a **working correlation structure** for $Cor(\mathbf{Y}_i)$ from our familiar structures: Compound Symmetry/Exchangeable, Exponential/AR1, Gaussian, Spherical, etc. See Autocorrelation Function in Chapter 3 for examples.

Note: We may use the word "association" rather than correlation so that it applies in cases where the outcome is not continuous.

Based on the model specification, the covariance matrix for $\mathbf{Y}_i$ is

$$\mathbf{V}_i = \mathbf{A}_i^{1/2} Cor(\mathbf{Y}_i) \mathbf{A}_i^{1/2}$$

where $\mathbf{A}_i$ is a diagonal matrix with $Var(Y_{ij}|X_{ij}) = \phi v(\mu_{ij})$ along the diagonal. We use the term **working correlation structure** to acknowledge our uncertainty about the assumed model for the within-subject associations.

6.7.2 Interpretation

We typically discuss how a 1 unit increase in an explanatory variable impacts the mean, **keeping all other variables constant.**

In a marginal model, we explicitly model the **overall or marginal mean**, $E(Y_{ij}|X_{ij})$.

For a continuous outcome (with identity link function), the expected response for the value of $X_{ij} = x$ is

$$E(Y_{ij}|X_{ij} = x) = \beta_0 + \beta_1 x$$

The expected response for value of $X_{ij} = x + 1$ is

$$E(Y_{ij}|X_{ij} = x + 1) = \beta_0 + \beta_1 (x + 1)$$

The difference is

$$\begin{aligned} &E(Y_{ij}|X_{ij} = x + 1) - E(Y_{ij}|, X_{ij} = x) \\ &= \beta_0 + \beta_1 (x + 1) - (\beta_0 + \beta_1 x) = \beta_1 \end{aligned}$$

Therefore, in the context of a marginal model, β_1 has the interpretation of a population mean change.

A 1-unit increase in X leads to a β_1 increase in the overall mean response, keeping all other variables constant.

6.7.3 Estimation

To estimate the parameters, we will use **generalized estimating equations** (GEE). We want to minimize the following objective function (a standardized sum of squared residuals),

$$\sum_{i=1}^{n}(\mathbf{Y}_i - \mu_i)^T\mathbf{V}_i^{-1}(\mathbf{Y}_i - \mu_i)$$

treating $\mathbf{V}_i$ as known and where μ_i is a vector of means with elements $\mu_{ij} = g^{-1}(\mathbf{x}_{ij}^T\beta)$.

Using calculus, it can be shown that if a minimum exists, it must solve the following generalized estimating equations,

$$\sum_{i=1}^{n}\mathbf{D}_i^T\mathbf{V}_i^{-1}(\mathbf{Y}_i - \mu_i) = 0$$

where $\mathbf{D}_i = \partial\mu_i/\partial\beta$ is the gradient matrix.

Depending on the link function, there may or may not be a **closed-form solution**. If not, the solution requires an iterative algorithm.

Because GEE depends on both β, α (parameters for the working correlation matrix) and ϕ (variance scalar), the following **iterative two-stage estimation procedure** is required:

Step 1. Given current estimates of α and ϕ, V_i is estimated, and an updated estimate of β is obtained as the solution to the generalized estimating equations.

Step 2. Given the current estimate of β, updated estimates of α and ϕ are obtained from the standardized residuals

$$e_{ij} = \frac{Y_{ij} - \hat{\mu}_{ij}}{\sqrt{v(\hat{\mu}_{ij})}}$$

Step 3. We iterate between Steps 1 and 2 until convergence has been achieved (estimates for β, α, and ϕ don't change).

Starting or initial estimates of β can be obtained assuming independence.

Note: In estimating the model, we only use our assumption about the **mean model**,

$$g(\mu_{ij}) = \eta_{ij} = \mathbf{x}_{ij}^T\beta$$

and the **covariance model** of the observations,

$$\mathbf{V}_i = \mathbf{A}_i^{1/2} Cor(\mathbf{Y}_i) \mathbf{A}_i^{1/2}$$

where $\mathbf{A}_i$ is a diagonal matrix with $Var(Y_{ij}|X_{ij}) = \phi v(\mu_{ij})$ along the diagonal.

6.7.3.1 R: Three GEE R Packages

There are three packages to do this in R, gee, geepack, and geeM.

We will use geeM when we have large data sets because it is optimized for large data. The geepack package is nice as it has an anova method to help compare models. The gee package has some nice output.

The syntax for the function geem() in geeM package is

```
geem(formula, id, waves = NULL, data, family = gaussian, constr = "independence", Mv = 1)
```

- formula: Symbolic description of the model to be fitted
- id: Vector that identifies the clusters
- data: Optional dataframe
- family: Description of the error distribution and **link function**
- constr: A character string specifying the correlation structure. Allowed structures are: "independence", "exchangeable" (equal correlation), "ar1" (exponential decay), "m-dependent" (m-diagonal), "unstructured", "fixed", and "userdefined".
- Mv: for "m-dependent", the value for m.

```
require(geeM)
```

```
Loading required package: geeM
```

```
Loading required package: Matrix
```

```

Attaching package: 'Matrix'
```

```
The following objects are masked from 'package:tidyr':

    expand, pack, unpack
```

```
summary(geem(resp ~ age + smoke, id = id, data = ohio, family = binomial(link = 'logit'), co
```

```
            Estimates Model SE Robust SE    wald       p
(Intercept)   -1.8980  0.10960   0.11470 -16.550 0.00000
age           -0.1148  0.05586   0.04494  -2.554 0.01066
smoke          0.2438  0.16620   0.17980   1.356 0.17520

 Estimated Correlation Parameter:  0.3992
 Correlation Structure:  ar1
 Est. Scale Parameter:  1.017

 Number of GEE iterations: 3
 Number of Clusters:  537    Maximum Cluster Size:  4
 Number of observations with nonzero weight:  2148
```

The syntax for the function geeglm() in geepack package is

```
geeglm(formula, family = gaussian, data, id, zcor = NULL, constr, std.err = 'san.se')
```

- formula: Symbolic description of the model to be fitted
- family: Description of the error distribution and **link function**
- data: Optional dataframe
- id: Vector that identifies the clusters
- contr: A character string specifying the correlation structure. The following are permitted: “independence”, “exchangeable”, “ar1”, “unstructured” and “userdefined”
- zcor: Enter a user defined correlation structure

```
require(geepack)
summary(geeglm(resp ~ age + smoke, id = id, data = ohio, family = binomial(link = 'logit'), 
```

```
Call:
geeglm(formula = resp ~ age + smoke, family = binomial(link = "logit"),
    data = ohio, id = id, corstr = "ar1")

 Coefficients:
            Estimate  Std.err    Wald Pr(>|W|)
(Intercept) -1.90218  0.11525 272.409   <2e-16 ***
age         -0.11489  0.04539   6.407   0.0114 *
smoke        0.23448  0.18119   1.675   0.1956
```

```
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

Correlation structure = ar1
Estimated Scale Parameters:

            Estimate Std.err
(Intercept)    1.021  0.1232
  Link = identity

Estimated Correlation Parameters:
      Estimate Std.err
alpha    0.491 0.06733
Number of clusters:   537  Maximum cluster size: 4
```

The syntax for the function gee() in gee package is

```
gee(formula, id, data, family = gaussian, constr = 'independence',Mv)
```

- formula: Symbolic description of the model to be fitted
- family: Description of the error distribution and **link function**
- data: Optional dataframe
- id: Vector that identifies the clusters
- constr: Working correlation structure: "independence", "exchangeable", "AR-M", "unstructured"
- Mv: order of AR correlation (AR1: Mv = 1)

```
require(gee)
```

```
Loading required package: gee
```

```
summary(gee(resp ~ age + smoke, id = id, data = ohio, family = binomial(link = 'logit'), cors
```

```
Beginning Cgee S-function, @(#) geeformula.q 4.13 98/01/27

running glm to get initial regression estimate

(Intercept)         age       smoke
    -1.8837     -0.1134      0.2721
```

```
 GEE:  GENERALIZED LINEAR MODELS FOR DEPENDENT DATA
 gee S-function, version 4.13 modified 98/01/27 (1998)

Model:
 Link:                      Logit
 Variance to Mean Relation: Binomial
 Correlation Structure:     AR-M , M = 1

Call:
gee(formula = resp ~ age + smoke, id = id, data = ohio, family = binomial(link = "logit"),
    corstr = "AR-M", Mv = 1)

Summary of Residuals:
    Min      1Q  Median      3Q     Max
-0.1939 -0.1586 -0.1439 -0.1179  0.8821


Coefficients:
            Estimate Naive S.E. Naive z Robust S.E. Robust z
(Intercept)  -1.8982    0.10962 -17.316     0.11468  -16.552
age          -0.1148    0.05586  -2.054     0.04494   -2.554
smoke         0.2438    0.16620   1.467     0.17983    1.356

Estimated Scale Parameter:  1.017
Number of Iterations:  3

Working Correlation
        [,1]   [,2]   [,3]    [,4]
[1,] 1.00000 0.3990 0.1592 0.06352
[2,] 0.39900 1.0000 0.3990 0.15920
[3,] 0.15920 0.3990 1.0000 0.39900
[4,] 0.06352 0.1592 0.3990 1.00000
```

6.7.3.2 Properties of Estimators

We acknowledge that our working covariance matrix $\mathbf{V}_i$ only approximates the true underlying covariance matrix for $\mathbf{Y}_i$, but it is wrong.

Despite incorrectly specifying the correlation structure,

- $\widehat{\beta}$ is **consistent** (meaning that with high probability $\widehat{\beta}$ is close to the true β for very large samples) no matter whether the within-in subject associations are correctly modeled.

- In large samples, the sampling distribution of $\hat{\beta}$ is multivariate normal with mean β and the covariance is a sandwich, $Cov(\hat{\beta}) = B^{-1}MB^{-1}$ where the bread B is defined as

$$B = \sum_{i=1}^{n} \mathbf{D}_i^T \mathbf{V}_i^{-1} \mathbf{D}_i$$

and the middle (meat or cheese or filling) M is defined as

$$M = \sum_{i=1}^{n} \mathbf{D}_i^T \mathbf{V}_i^{-1} Cov(\mathbf{Y}_i) \mathbf{V}_i^{-1} \mathbf{D}_i$$

We can plug in estimates of the quantities and get

$$\widehat{Cov}(\hat{\beta}) = \hat{B}^{-1} \hat{M} \hat{B}^{-1}$$

where

$$\hat{B} = \sum_{i=1}^{n} \hat{\mathbf{D}}_i^T \hat{\mathbf{V}}_i^{-1} \hat{\mathbf{D}}_i$$

and

$$\hat{M} = \sum_{i=1}^{n} \hat{\mathbf{D}}_i^T \hat{\mathbf{V}}_i^{-1} \widehat{Cov}(Y_i) \hat{\mathbf{V}}_i^{-1} \hat{\mathbf{D}}_i$$

and $\widehat{Cov}(\mathbf{Y}_i) = (\mathbf{Y}_i - \hat{\mu}_i)(\mathbf{Y}_i - \hat{\mu}_i)^T$

This is the empirical or **sandwich robust estimator** of the standard errors. It is valid even if we are wrong about the correlation structure.

Fun fact: If we happen to choose the right correlation/covariance structure and $\mathbf{V}_i = \Sigma_i$, where $\Sigma_i = Cov(\mathbf{Y}_i)$, then $Cov(\hat{\beta}) = B^{-1}$.

Since we model the conditional expectation $E(Y_{ij}|\mathbf{x}_{ij}) = \mu_{ij}$ with

$$g(\mu_{ij}) = \mathbf{x}_{ij}^T \beta,$$

we can only make overall population interpretations regarding mean response at a given value of $\mathbf{x}_{ij}$.

We can compare two lists of $\mathbf{x}$ values, $\mathbf{x}^*$ and $\mathbf{x}$ and see how that impacts the population mean via the link function $g(\mu)$,

$$g(\mu^*) - g(\mu) = \mathbf{x}^{*T}\beta - \mathbf{x}^T\beta$$

6.7.4 Model Selection Tools and Diagnostics

The model selection tools we use will be similar to those used in Stat 155. To refresh your memory of how hypothesis testing can be used for model selection, please see my Introduction to Statistical Models Notes.

6.7.4.1 Hypothesis Testing for Coefficients (one at a time)

This section is similar to the t-tests in Stat 155.

To decide which explanatory variables should be in the model, we can test whether $\beta_k = 0$ for some k. If the slope coefficient were equal to zero, the associated variable is essentially not contributing to the model.

In marginal models (GEE), the estimated covariance matrix for $\hat{\beta}$ is given by the **robust sandwich estimator**. The standard error for $\hat{\beta}_k$ is the square rooted values of the diagonal of this matrix corresponding to $\hat{\beta}_k$. This is reported in the summary output for `Robust SE`.

To test the hypothesis $H_0 : \beta_k = 0$ vs. $H_A : \beta_k \neq 0$, we can calculate a z-statistic (referred to as a Wald statistic),

$$z = \frac{\hat{\beta}_k}{SE(\hat{\beta}_k)}$$

With GEE, the geem() function provides `wald` statistics and associated p-values for testing $H_0 : \beta_k = 0$. These p-values come from a Normal distribution because we know that is the asymptotic distribution for these estimators.

```
geemod <- geem(resp ~ age + smoke, id = id, data = ohio, family = binomial(link = 'logit'), c
summary(geemod) #look for Robust SE and wald
```

```
            Estimates Model SE Robust SE    wald        p
(Intercept)   -1.8800  0.11480   0.11390 -16.510 0.000000
age           -0.1134  0.04354   0.04386  -2.585 0.009726
smoke          0.2651  0.17700   0.17770   1.491 0.135900

 Estimated Correlation Parameter:  0.3541
 Correlation Structure:  exchangeable
 Est. Scale Parameter:  0.9999

 Number of GEE iterations: 3
 Number of Clusters:  537    Maximum Cluster Size:  4
 Number of observations with nonzero weight:  2148
```

```
geemod$beta/sqrt(diag(geemod$var)) #wald statistics by hand
```

```
(Intercept)         age       smoke
    -16.510      -2.585       1.491
```

6.7.4.2 Hypothesis Testing for Coefficients (many at one time)

This section is similar to the nested F-tests in Stat 155.

To test a more involved hypothesis $H_0 : \mathbf{L}\beta = 0$ (to test whether multiple slopes are zero or a linear combo of slopes is zero), we can calculate a Wald statistic (a squared z statistic),

$$W^2 = (\mathbf{L}\hat{\beta})^T(\mathbf{L}\widehat{Cov}(\hat{\beta})\mathbf{L}^T)^{-1}(\mathbf{L}\hat{\beta})$$

We assume the sampling distribution is approximately χ^2 with df = # of rows of $\mathbf{L}$ to calculate p-values (as long as n is large).

This model selection tool is useful when you have a categorical variable with more than two categories. You may want to check to see if we should exclude the entire variable or whether a particular category has an impact on the model.

With the data example, let's look at smoking. It is only a binary categorical variable, but let's run the hypothesis test with the example code from above. So, in this case, we let the matrix L be equal to a 1×3 matrix with 0's everywhere except in the 3rd column so that we'd test: $H_0 : \beta_2 = 0$. Even though we used a different test statistic, we ended up with the same p-value as the approach above.

```
summary(geemod)
```

```
            Estimates Model SE Robust SE    wald        p
(Intercept)   -1.8800  0.11480   0.11390 -16.510 0.000000
age           -0.1134  0.04354   0.04386  -2.585 0.009726
smoke          0.2651  0.17700   0.17770   1.491 0.135900

 Estimated Correlation Parameter:  0.3541
 Correlation Structure:  exchangeable
 Est. Scale Parameter:  0.9999

 Number of GEE iterations: 3
 Number of Clusters:  537    Maximum Cluster Size:  4
 Number of observations with nonzero weight:  2148
```

```r
b = geemod$beta
W = geemod$var

(L = matrix(c(0,0,1),nrow=1))
```

```
     [,1] [,2] [,3]
[1,]    0    0    1
```

```r
L%*%b # Lb estimate
```

```
       [,1]
[1,] 0.2651
```

```r
(se = sqrt(diag(L%*%W%*%t(L)))) # Robust SE for Lb
```

```
[1] 0.1777
```

```r
# 95% Confidence Interval (using Asymptotic Normality)
L%*%b - 1.96*se
```

```
        [,1]
[1,] -0.0833
```

```r
L%*%b + 1.96*se
```

```
       [,1]
[1,] 0.6135
```

```r
# Hypothesis Testing

w2 <- as.numeric( t(L%*%b) %*% solve(L %*% W %*% t(L))%*% (L%*%b)) ## should be approximatel
1 - pchisq(w2, df = nrow(L)) # p-value
```

```
[1] 0.1359
```

So, the p-value, the probability of getting a test statistic as or more extreme assuming the null hypothesis is true, is around 13%. We do not have enough evidence to reject the null hypothesis. This leads us to believe that smoking could be removed from the model.

Let's test whether $H_0 : \beta_1 = \beta_2 = 0$, then we let the matrix L be equal to a 2×3 matrix with 0's everywhere except in the 2nd column in row 1 and 3rd column in row 2.

```
summary(geemod)
```

```
            Estimates Model SE Robust SE    wald        p
(Intercept)   -1.8800  0.11480   0.11390 -16.510 0.000000
age           -0.1134  0.04354   0.04386  -2.585 0.009726
smoke          0.2651  0.17700   0.17770   1.491 0.135900

 Estimated Correlation Parameter:  0.3541
 Correlation Structure:  exchangeable
 Est. Scale Parameter:  0.9999

 Number of GEE iterations: 3
 Number of Clusters:  537    Maximum Cluster Size:  4
 Number of observations with nonzero weight:  2148
```

```
b = geemod$beta
W = geemod$var

(L = matrix(c(0,1,0,0,0,1),nrow=2,byrow=TRUE))
```

```
     [,1] [,2] [,3]
[1,]    0    1    0
[2,]    0    0    1
```

```
L%*%b #Lb estimate
```

```
        [,1]
[1,] -0.1134
[2,]  0.2651
```

```
(se = sqrt(diag(L%*%W%*%t(L)))) # Robust SE for Lb
```

```
[1] 0.04386 0.17775
```

```
# 95% Confidence Interval (using Asymptotic Normality)
L%*%b - 1.96*se
```

```
        [,1]
[1,] -0.1993
[2,] -0.0833
```

```
L%*%b + 1.96*se
```

```
         [,1]
[1,] -0.02743
[2,]  0.61346
```

```
# Hypothesis Testing

w2 <- as.numeric( t(L%*%b) %*% solve(L %*% W %*% t(L))%*% (L%*%b)) ## should be approximatel
1 - pchisq(w2, df = nrow(L)) #p-value
```

```
[1] 0.01092
```

We have evidence to suggest that at least one of the variables, age or smoking, has a statistically significant effect on the outcome. As we saw above, we know it must be age, but the model with age and smoking is better than a model with no explanatory variables.

6.7.4.3 Diagnostics

For any linear model with a quantitative outcome and at least one quantitative explanatory variable, we should check to see if the residuals have a pattern.

Ideally, the residual plot should have no obvious pattern, plotting the residuals on the y-axis against a quantitative x-variable. See some example code below.

```
resid = mod$y - predict(mod)
plot(predict(mod),resid)
```

If there is a pattern, then we are systematically over or under-predicting, and we can use that information to incorporate non-linearity to improve the model.

6.8 Mixed Effects

Let's return to a continuous outcome setting. Consider the mean response for a specific individual (ith individual/unit) at a specific observation time (jth time).

$$\mu_{ij} = E(Y_{ij}| \text{ Explanatory Variables, Time})$$

- If the mean is **constant** across **observation times** (horizontal trend line) and other **variables** (no X's), and the same across all individuals,

$$\mu_{ij} = \mu$$

- If the mean changes **linearly** across **observation times** and no other **variables** impact the mean, and it is the same across all individuals,

$$\mu_{ij} = \beta_0 + \beta_1 t_{ij}$$

- If the mean increases **linearly** with an **time-varying explanatory variable** and it is the same across all individuals,

$$\mu_{ij} = \beta_0 + \beta_1 x_{ij}$$

6.8.1 Individual Intercepts

In any situations listed above, individuals may have their **own intercept**, their own starting level. We can notate that by adding b_i as an individual deviation from the mean intercept,

$$\mu_{ij} = \mu + b_i$$

$$\mu_{ij} = \beta_0 + b_i + \beta_1 t_{ij}$$

$$\mu_{ij} = \beta_0 + b_i + \beta_1 x_{ij}$$

We could estimate the individual intercepts to find b_i, the difference between the overall mean intercept and an individual's intercept. If $b_i = -2$, the individual i starts -2 units lower in their outcome responses than the average.

6.8.1.1 First Approach: Separate Models

Let's use the dental data to figure out the overall mean intercept and estimate each deviation from that intercept. The first approach is to fit a separate linear model for each person and look at the intercepts.

```
dental <- read.table("./data/dental.dat", col.names=c("obsno", "id", "age", "distance", "gend
int <- sapply(unique(dental$id),function(i) lm(distance~age, data = dental[dental$id == i,])$
mean(int) #mean intercept
```

```
[1] 16.76
```

```
int-mean(int) #estimated b_i
```

```
(Intercept) (Intercept) (Intercept) (Intercept) (Intercept) (Intercept)
     0.4889     -2.5611     -2.3611      2.8889      2.8389      0.2389
(Intercept) (Intercept) (Intercept) (Intercept) (Intercept) (Intercept)
     0.1889      4.6889      1.3389     -3.2111      2.1889      0.5389
(Intercept) (Intercept) (Intercept) (Intercept) (Intercept) (Intercept)
    -1.9111     -0.7611      7.9389     -3.1111      2.1889     -1.8111
(Intercept) (Intercept) (Intercept) (Intercept) (Intercept) (Intercept)
     2.9889     -2.3611      4.4889      3.2889     -3.5111    -13.9611
(Intercept) (Intercept) (Intercept)
     2.3389     -3.2611      0.1889
```

However, this **stratification** approach also allows everyone to have their own slope for age, which we may not want to do. If there is some common growth across persons, we'd like to borrow information across all persons and fit only one model.

6.8.1.2 Second Approach: Fixed Effects Models

To estimate a model with one fixed slope for everyone that allows for individual intercepts, we fit a model often termed a **fixed effects model**. In this model, we include an indicator variable for every individual `id` to allow for individual intercepts. This can be quite difficult if we have a large sample size because we are estimating one parameter per person or unit.

```
lm(distance ~ age + factor(id), data = dental)
```

```
Call:
lm(formula = distance ~ age + factor(id), data = dental)

Coefficients:
 (Intercept)           age   factor(id)2   factor(id)3   factor(id)4
       14.11          0.66          1.62          2.37          3.50
 factor(id)5   factor(id)6   factor(id)7   factor(id)8   factor(id)9
        1.25         -0.25          1.62          2.00         -0.25
factor(id)10  factor(id)11  factor(id)12  factor(id)13  factor(id)14
       -2.88          5.00          6.37          2.00          2.87
factor(id)15  factor(id)16  factor(id)17  factor(id)18  factor(id)19
        5.25          1.62          5.00          2.37          2.50
factor(id)20  factor(id)21  factor(id)22  factor(id)23  factor(id)24
        3.75          8.12          2.25          2.87          2.87
factor(id)25  factor(id)26  factor(id)27
        3.50          4.50          1.62
```

6.8.1.3 Third Approach: Random Effects Models

An alternative way for individuals to have their own intercept is to assume a probability distribution for the $ b_i$'s such as $N(0, \sigma_b^2)$ and estimate the variability,σ_b^2. This is a **random intercept model**,

$$Y_{ij} = \beta_0 + b_i + \beta_1 x_{ij} + \epsilon_{ij}, \qquad b_i \overset{iid}{\sim} N(0, \sigma_b^2), \epsilon_{ij} \overset{iid}{\sim} N(0, \sigma_e^2),$$

where ϵ_{ij} and b_i are independent for any i and j.

```
library(lme4)
summary(lmer(distance ~ age + (1|id), data = dental))
```

```
Linear mixed model fit by REML ['lmerMod']
Formula: distance ~ age + (1 | id)
   Data: dental

REML criterion at convergence: 447

Scaled residuals:
   Min     1Q Median     3Q    Max
-3.665 -0.535 -0.013  0.487  3.722
```

```
Random effects:
 Groups   Name        Variance Std.Dev.
 id       (Intercept) 4.47     2.11
 Residual             2.05     1.43
Number of obs: 108, groups:  id, 27

Fixed effects:
            Estimate Std. Error t value
(Intercept)  16.7611     0.8024    20.9
age           0.6602     0.0616    10.7

Correlation of Fixed Effects:
    (Intr)
age -0.845
```

In the output above, you'll see information about the standardized or scaled residuals.

Below it, you'll see information on the random effects. In our case, we have a random intercept for each individual that we allowed by adding `+ (1 | id)` to the formula. The `1` indicates intercept, ' |`indicates conditional on, and`id`refers to the variable called`id‘. We assumed that $b_i \sim N(0, \sigma_b^2)$ and it has estimated $\hat{\sigma}_b^2 = 4.472$, $\hat{\sigma}_b = 2.115$. Additionally, we have assumed that our errors are independent with constant variance, $\hat{\sigma}_e^2 = 2.049$, $\hat{\sigma}_e = 1.432$.

Lastly, we have our fixed effects, the parameters we don't assume are random, β_0 and β_1. Those estimates are $\hat{\beta}_0 = 16.76$ and $\hat{\beta}_1 = 0.66$.

Now with this random effects model, the addition of a random intercept has induced correlation between observation **within an individual** because they now share an intercept.

In fact, assuming the model is true,

$$\begin{aligned} Cov(Y_{ij}, Y_{il}) &= Cov(\beta_0 + b_i + \beta_1 x_{ij} + \epsilon_{ij}, \beta_0 + b_i + \beta_1 x_{il} + \epsilon_{il}) \\ &= Cov(b_i + \epsilon_{ij}, b_i + \epsilon_{il}) \\ &= Cov(b_i, b_i) + Cov(b_i, \epsilon_{ij}) + Cov(b_i, \epsilon_{il}) + Cov(\epsilon_{ij}, \epsilon_{il}) \\ &= \sigma_b^2 + 0 + 0 + 0 \end{aligned}$$

$$\begin{aligned} Cor(Y_{ij}, Y_{il}) &= \frac{Cov(Y_{ij}, Y_{il})}{\sqrt{Var(Y_{ij}) Var(Y_{il})}} \\ &= \frac{\sigma_b^2}{\sqrt{Var(\beta_0 + b_i + \beta_1 X_{ij} + \epsilon_{ij}) Var(\beta_0 + b_i + \beta_1 X_{il} + \epsilon_{il})}} \end{aligned}$$

$$= \frac{\sigma_b^2}{\sqrt{Var(b_i + \epsilon_{ij})Var(b_i + \epsilon_{il})}}$$

$$= \frac{\sigma_b^2}{\sqrt{(\sigma_b^2 + \sigma_e^2)^2}}$$

$$= \frac{\sigma_b^2}{(\sigma_b^2 + \sigma_e^2)}$$

for $j \neq l$ and this does not depend on j or l or $j - l$. The correlation is constant for any time lags (exchangeable/compound symmetric).

6.8.2 Individual Slopes

Since we have longitudinal data, we can observe each individual's trajectory over time. Everyone has their own unique growth rate. If that growth rate can be explained with an explanatory variable, we could use an interaction term to allow for that unique growth. Otherwise, we could allow each individual to estimate their own slope.

6.8.2.1 Second Approach: Fixed Effects Models

One could use a fixed effects model and use an interaction term with the id variable to estimate each individual slope, but that is not sustainable for larger data sets.

```
lm(distance ~ age*factor(id), data = dental)
```

```
Call:
lm(formula = distance ~ age * factor(id), data = dental)

Coefficients:
     (Intercept)              age      factor(id)2      factor(id)3
        1.73e+01         3.75e-01        -3.05e+00        -2.85e+00
     factor(id)4      factor(id)5      factor(id)6      factor(id)7
        2.40e+00         2.35e+00        -2.50e-01        -3.00e-01
     factor(id)8      factor(id)9     factor(id)10     factor(id)11
        4.20e+00         8.50e-01        -3.70e+00         1.70e+00
    factor(id)12     factor(id)13     factor(id)14     factor(id)15
        5.00e-02        -2.40e+00        -1.25e+00         7.45e+00
    factor(id)16     factor(id)17     factor(id)18     factor(id)19
```

```
      -3.60e+00          1.70e+00         -2.30e+00          2.50e+00
   factor(id)20       factor(id)21       factor(id)22       factor(id)23
      -2.85e+00          4.00e+00          2.80e+00         -4.00e+00
   factor(id)24       factor(id)25       factor(id)26       factor(id)27
      -1.45e+01          1.85e+00         -3.75e+00         -3.00e-01
 age:factor(id)2    age:factor(id)3    age:factor(id)4    age:factor(id)5
       4.25e-01          4.75e-01          1.00e-01         -1.00e-01
 age:factor(id)6    age:factor(id)7    age:factor(id)8    age:factor(id)9
      -1.32e-15          1.75e-01         -2.00e-01         -1.00e-01
age:factor(id)10   age:factor(id)11   age:factor(id)12   age:factor(id)13
       7.50e-02          3.00e-01          5.75e-01          4.00e-01
age:factor(id)14   age:factor(id)15   age:factor(id)16   age:factor(id)17
       3.75e-01         -2.00e-01          4.75e-01          3.00e-01
age:factor(id)18   age:factor(id)19   age:factor(id)20   age:factor(id)21
       4.25e-01          1.47e-15          6.00e-01          3.75e-01
age:factor(id)22   age:factor(id)23   age:factor(id)24   age:factor(id)25
      -5.00e-02          6.25e-01          1.58e+00          1.50e-01
age:factor(id)26   age:factor(id)27
       7.50e-01          1.75e-01
```

6.8.2.2 Third Approach: Random Effects Models

A more efficient way to allow individuals to have their own slopes is to assume a probability distribution for the slopes (and typically intercepts) by assuming the vector of individual intercept and slope is bivariate Normal with covariance matrix G, $(b_{i0}, b_{i1}) \sim N(0, G)$. This is a **random slope and intercept model**,

$$Y_{ij} = (\beta_0 + b_{i0}) + (\beta_1 + b_{i1})x_{ij} + \epsilon_{ij}, \qquad (b_{i0}, b_{i1}) \sim N(0, G), \epsilon_{ij} \overset{iid}{\sim} N(0, \sigma_e^2),$$

where ϵ_{ij} and (b_{i0}, b_{i1}) are independent of each other for any i and j.

```
summary(lmer(distance ~ age + (age|id), data = dental))
```

```
Linear mixed model fit by REML ['lmerMod']
Formula: distance ~ age + (age | id)
   Data: dental

REML criterion at convergence: 442.6

Scaled residuals:
   Min     1Q Median     3Q    Max
```

```
-3.223 -0.494  0.007  0.472  3.916

Random effects:
 Groups   Name        Variance Std.Dev. Corr
 id       (Intercept) 5.4166   2.327
          age         0.0513   0.226    -0.61
 Residual             1.7162   1.310
Number of obs: 108, groups:  id, 27

Fixed effects:
            Estimate Std. Error t value
(Intercept)  16.7611     0.7753   21.62
age           0.6602     0.0713    9.27

Correlation of Fixed Effects:
    (Intr)
age -0.848
```

In the output above, you'll see information about the standardized or scaled residuals.

Below it, you'll see information on the random effects. In our case, we have a random intercept and a random slope for age for each individual that we allowed by adding `+ (age | id)` to the formula. We might read `(age | id)` as we want to estimate slope `|` indicates conditional on, and `id` refers to the variable called `id`. We assumed that $(b_{i0}, b_{i1}) \sim N(0, G)$. The estimated covariance matrix G is

$$\hat{G} = \begin{pmatrix} 5.41 & -0.61 * 2.32 * 0.22 \\ -0.61 * 2.32 * 0.22 & 0.05 \end{pmatrix}$$

Additionally, we have assumed that our errors are independent with constant variance, $\hat{\sigma}_e^2 = 1.72$, $\hat{\sigma}_e = 1.31$.

Lastly, we have our fixed effects, the parameters we don't assume are random, β_0 and β_1. Those estimates are $\hat{\beta}_0 = 16.76$ and $\hat{\beta}_1 = 0.66$. You'll notice that these estimates did not change much at all by adding a random intercept. This will more often happen when modeling a continuous outcome (not when modeling a binary or count outcome).

6.8.3 Multi-level or Hierarchical Model

We can write the random intercept and slope model by creating layers of models. We can write the model for the mean as the level 1 model

$$\mu_{ij} = \beta_{i0} + \beta_{i1} X_{ij} \quad \text{Level 1}$$

where the individual intercepts and slopes can be written as a level 2 model,

$$\beta_{i0} = \beta_0 + b_{i0} \quad \text{Level 2}$$

and/or

$$\beta_{i1} = \beta_1 + b_{i1} \quad \text{Level 2}$$

where $b_{i0} \sim N(0, \tau_0^2)$, $b_{i1} \sim N(0, \tau_1^2)$, $Cov(\delta_{i0}, \delta_{i1}) = \tau_{01}$.

This is often called a **multi-level or hierarchical model** in that the model is written with more than one level of models.

Combine the levels by plugging in the second level into the first level.

$$Y_{ij} = \beta_0 + b_{i0} + (\beta_1 + b_{i1}) x_{ij} + \epsilon_{ij}$$

$$= (\beta_0 + \beta_1 x_{ij}) + (b_{i1} x_{ij} + b_{i0}) + \epsilon_{ij}$$

$$= \text{Fixed effects} + \text{Random effects} + \text{Error}$$

This composite model is often called a **mixed effects model** because there are fixed-parameter effects and random parameter effects.

6.8.4 Mixed Effects Model

In general, we can write a mixed-effects model for our outcome vector,

$$\mathbf{Y}_i = \mathbf{X}_i \beta + \mathbf{Z}_i \mathbf{b}_i + \epsilon_i$$

$$= \underbrace{\text{Fixed effects}}_{\text{mean model}} + \underbrace{\text{Random effects}}_{\text{between unit variation}} + \underbrace{textError}_{\text{within unit variation}}$$

where $\mathbf{X}_i$ is the design matrix for the fixed effects, $\mathbf{Z}_i$ is the design matrix for the random effects (a subset of columns of $\mathbf{X}_i$), $\epsilon_i \sim N(0, \Sigma)$, and $\mathbf{b}_i \sim N(0, \mathbf{G})$.

$$E(\mathbf{Y}_i) = E(\mathbf{X}_i \beta + \mathbf{Z}_i \mathbf{b}_i + \epsilon_i)$$

$$= \mathbf{X}_i\beta$$

$$\begin{aligned} Cov(\mathbf{Y}_i) &= Cov(\mathbf{X}_i\beta + \mathbf{Z}_i\mathbf{b}_i + \epsilon_i) \\ &= Cov(\mathbf{Z}_i\mathbf{b}_i + \epsilon_i) \end{aligned}$$

Assuming $\mathbf{b}_i$ and ϵ_i are independent, then

$$\begin{aligned} \mathbf{V}_i = Cov(\mathbf{Y}_i) &= Cov(\mathbf{Z}_i\mathbf{b}_i) + Cov(\epsilon_i) \\ &= \mathbf{Z}_i Cov(\mathbf{b}_i)\mathbf{Z}_i^T + \Sigma \\ &= \mathbf{Z}_i\mathbf{G}\mathbf{Z}_i^T + \Sigma \end{aligned}$$

Since $\mathbf{b}_i$ and ϵ_i are almost always assumed Normal, then

$$\mathbf{Y}_i \sim \mathcal{N}(\mathbf{X}_i\beta, \mathbf{Z}_i\mathbf{G}\mathbf{Z}_i^T + \Sigma)$$

6.8.5 History

The origins of mixed effects models go back to **R.A. Fisher** work under the analysis of variance (**ANOVA**) paradigm in the 1910-1930's.

- Earliest mixed effects model: **random intercept model**
- Work continued in the ANOVA framework until 1970's
- Mixed effects models in a linear model framework are based on the ANOVA paradigm
- Seminal paper by **Harville** in 1977
- Highly cited mixed models paper by **Laird and Ware** in 1982

6.8.6 Interpretation

We typically discuss how a 1 unit increase in a variable in X impacts the mean, **keeping all other variables constant.**

With random effects, this means interpreting the change within a unit or subject.

The expected response, given subject's random effects, is

$$E(\mathbf{Y}_i|\mathbf{b}_i) = \mathbf{X}_i\beta + \mathbf{Z}_i\mathbf{b}_i$$

For the sake of simplicity, let's consider the random intercept model,

$$E(Y_{ij}|b_i) = \beta_0 + b_i + \beta_1 X_{ij}$$

The expected response for value of $X_{ij} = x$ is

$$E(Y_{ij}|b_i, X_{ij} = x) = \beta_0 + b_i + \beta_1 x$$

The expected response for value of $X_{ij} = x + 1$ is

$$E(Y_{ij}|b_i, X_{ij} = x + 1) = \beta_0 + b_i + \beta_1 (x + 1)$$

The difference is in the expected response for individual i with random effect b_i is,

$$\begin{aligned} &E(Y_{ij}|b_i, X_i = x + 1) - E(Y_{ij}|b_i, X_i = x) \\ &= \beta_0 + b_i + \beta_1(x + 1) - (\beta_0 + b_i + \beta_1 x) = \beta_1 \end{aligned}$$

This means we can give a subject-specific interpretation!

We can also consider the overall or marginal mean, $E(Y_{ij})$.

First, a definition of conditional expectation in the discrete setting,

$$E(Y|B = b) = \sum_b b \cdot P(Y = y|B = b)$$

Then we can show that

$$E(Y) = E(E(Y|B = b))$$

The iterated expectation also holds for continuous random variables.

We can also consider the **overall or marginal mean**, $E(Y_{ij})$.

$$E(Y_{ij}) = E(E(Y_{ij}|b_i)) = \beta_0 + \beta_1 X_{ij}$$

The expected response for value of $X_{ij} = x$ is

$$E(Y_{ij}|X_{ij} = x) = \beta_0 + \beta_1 x$$

The expected response for value of $X_{ij} = x + 1$ is

$$E(Y_{ij}|X_{ij} = x + 1) = \beta_0 + \beta_1(x + 1)$$

The difference is

$$E(Y_{ij}|X_{ij} = x + 1) - E(Y_{ij}|, X_{ij} = x)$$
$$= \beta_0 + \beta_1(x + 1) - (\beta_0 + \beta_1 x) = \beta_1$$

Therefore, in the context of a linear mixed effects model, β_1 has the interpretation of the subject-specific change as well as the population mean change.

A 1-unit increase in X leads to a β_1 increase in the subject's mean response, keeping all other variables constant.

A 1-unit increase in X leads to a β_1 increase in the overall mean response, keeping all other variables constant.

6.8.7 Estimation

6.8.7.1 Maximum Likelihood Estimation

Mixed Effects Models require specifying the entire distribution. We assume probability models for the random effects and the errors, and thus we could use **maximum likelihood estimation** to find estimates for our slopes.

If we specify the REML argument in `lmer` as REML = FALSE, then the `lmer` function will maximize the log-likelihood function (based on Normal assumptions for the errors and random effects),

$$l = -\frac{m * n}{2}\log(2\pi) - \frac{1}{2}\sum_{i=1}^{n}\log|\mathbf{V}_i|$$
$$-\frac{1}{2}\{\sum_{i=1}^{n}(\mathbf{y}_i - \mathbf{X}_i\beta)^T\mathbf{V}_i^{-1}(\mathbf{y}_i - \mathbf{X}_i\beta)\}$$

where $\mathbf{V}_i$ is the composite covariance matrix.

- Estimates of *beta*'s are **unbiased** if the model is correct (random effects and mean model).
- All parameter estimates are **consistent** if the model is correct.
- All parameter estimates are **asymptotically unbiased**; as $n \to \infty$, they become unbiased (if model is correct).
- Estimates are **asymptotically Normal**; as $n \to \infty$, the sampling distribution becomes more Normal.
- Estimates of variance of random effects are **biased** with finite n.

6.8.7.2 Restricted Maximum Likelihood Estimation

Patterson and Thompson (1971) and Harville (1974) provided technical solutions to the issues of maximum likelihood. They suggested maximizing the likelihood of observing the sample residuals rather than the sample data. This is referred to as **restricted maximum likelihood** or REML and is implemented in `lmer` by default when REML = TRUE.

REML Algorithm:

- Estimate fixed effects using OLS.
- Write down the likelihood of residuals in terms of residuals and variance parameters.
- Then maximize likelihood with respect to variance parameters.

Another REML Algorithm:

- Split the likelihood into one part about the mean and one part about the variance.
- First, maximize the variance part to get estimates of the variance parameters
- Then maximize the part about the mean using the estimated variance.

After some complicated mathematics, you ultimately end up maximizing the following with respect the parameters of $\mathbf{V}_i$

$$l = -\frac{m*n}{2}\log(2\pi) - \frac{1}{2}\sum_{i=1}^{n}\log|\mathbf{V}_i|$$
$$-\frac{1}{2}\left\{\sum_{i=1}^{n}(\mathbf{y}_i - \mathbf{X}_i\widehat{\beta})^T\mathbf{V}_i^{-1}(\mathbf{y}_i - \mathbf{X}_i\widehat{\beta})\right\}$$
$$-\frac{1}{2}\log|\sum_{i=1}^{n}\mathbf{X}_i^T\mathbf{V}_i^{-1}\mathbf{X}_i|$$

and $\widehat{\beta} = (\sum_{i=1}^{n}\mathbf{X}_i^T\mathbf{V}_i\mathbf{X}_i)^{-1}(\sum_{i=1}^{n}\mathbf{X}_i^T\mathbf{V}_i\mathbf{Y}_i)$ are the GLS estimates.

Advantages of REML

- **Less bias** in variance estimators.
- $\widehat{\beta}$ is still **unbiased** as long as the model is correct.

Disadvantages of REML

- You can only compare models that differ in their variance components with REML.
- You **cannot** compare models with differing fixed effects because you didn't maximize the full information (use ML for this).

6.8.8 Model Selection

6.8.8.1 Hypothesis Testing for Fixed Effects (one at a time)

To decide which fixed effects should be in the model, we can test whether $\beta_k = 0$.

The estimated covariance matrix for $\hat{\beta}$ is

$$\widehat{Cov}(\hat{\beta}) = \left\{ \sum_{i=1}^{n} \mathbf{X}_i^T \hat{\mathbf{V}}_i \mathbf{X}_i \right\}^{-1}$$

The standard error for $\hat{\beta}_k$ is the square rooted values of the diagonal of this matrix corresponding to $\hat{\beta}_k$. This is what is reported in the summary output.

```
mod.randomslope <- lmer(distance~age + (age|id), data = dental)
lme4::fixef(mod.randomslope)
```

```
(Intercept)         age
    16.7611      0.6602
```

```
vcov(mod.randomslope)
```

```
2 x 2 Matrix of class "dpoMatrix"
            (Intercept)       age
(Intercept)     0.60105 -0.046855
age            -0.04685  0.005077
```

```
summary(mod.randomslope)
```

```
Linear mixed model fit by REML ['lmerMod']
Formula: distance ~ age + (age | id)
   Data: dental

REML criterion at convergence: 442.6

Scaled residuals:
   Min     1Q Median     3Q    Max
-3.223 -0.494  0.007  0.472  3.916

Random effects:
```

```
 Groups   Name        Variance Std.Dev. Corr
 id       (Intercept) 5.4166   2.327
          age         0.0513   0.226    -0.61
 Residual             1.7162   1.310
Number of obs: 108, groups:  id, 27

Fixed effects:
            Estimate Std. Error t value
(Intercept)  16.7611     0.7753   21.62
age           0.6602     0.0713    9.27

Correlation of Fixed Effects:
    (Intr)
age -0.848
```

To test the hypothesis $H_0 : \beta_k = 0$ vs. $H_A : \beta_k \neq 0$, we can calculate a z-statistic,

$$z = \frac{\hat{\beta}_k}{SE(\hat{\beta}_k)}$$

There is debate about the appropriate distribution of this statistic, which is why p-values are not reported in the output. However, if you have an estimate that is large relative to the standard error, that indicates that it is significantly different from zero.

To test a more involved hypothesis $H_0 : \mathbf{L}\beta = 0$ (for multiple rows), we can calculate a Wald statistic,

$$W^2 = (\mathbf{L}\hat{\beta})^T(\mathbf{L}\hat{Cov}(\hat{\beta})\mathbf{L}^T)^{-1}(\mathbf{L}\hat{\beta})$$

Then we assume the sampling distribution is approximately χ^2 with df = # of rows of $\mathbf{L}$ to calculate p-values (as long as n is large). Below is some example R code.

```r
b = fixef(mod1)
W = vcov(mod1)

L = matrix(c(0,0,0,1),nrow=1)

L%*%b
(se = sqrt(diag(L%*%W%*%t(L)))) ##Robust SE's for Lb

##95% Confidence Interval (using Asymptotic Normality)
L%*%b - 1.96*se
```

```
L%*%b + 1.96*se

##Hypothesis Testing
w2 <- as.numeric( t(L%*%b) %*% solve(L %*% W %*% t(L))%*% (L%*%b)) ## should be approximatel
1 - pchisq(w2, df = nrow(L)) #p-value
```

Let's consider the data on the guinea pigs. In particular, we fit a model with a random intercept and dose, time, and the interaction between dose and time. If we want to know if the interaction term is necessary, we need to test whether both of their slopes for the interaction term are equal to 0.

```
pigs <- read.table("./data/diet.dat", col.names=c("id", paste("bw.",c(1,3,4,5,6,7),sep="")),
pigs <- pigs %>%
  gather(Tmp,bw, bw.1:bw.7) %>%
  separate(Tmp,into=c('Var','time'), remove=TRUE)

pigs$time <- as.numeric(pigs$time)
pigs$dose <- factor(pigs$dose)
levels(pigs$dose) <- c("Zero dose",' Low dose', 'High dose')

mod.pigs <- lmer(bw ~ dose*time + (1|id), data = pigs)
summary(mod.pigs)
```

```
Linear mixed model fit by REML ['lmerMod']
Formula: bw ~ dose * time + (1 | id)
   Data: pigs

REML criterion at convergence: 843.8

Scaled residuals:
    Min      1Q  Median      3Q     Max
-2.5621 -0.4107  0.0169  0.6843  2.1099

Random effects:
 Groups   Name        Variance Std.Dev.
 id       (Intercept) 1347     36.7
 Residual              703     26.5
Number of obs: 90, groups:  id, 15

Fixed effects:
                  Estimate Std. Error t value
```

```
(Intercept)          469.69      20.15   23.31
dose Low dose          4.84      28.50    0.17
doseHigh dose         11.15      28.50    0.39
time                  16.03       2.45    6.53
dose Low dose:time     6.53       3.47    1.88
doseHigh dose:time     3.60       3.47    1.04

Correlation of Fixed Effects:
            (Intr) dsLwds dsHghd time   dsLds:
doseLowdose -0.707
doseHighdos -0.707  0.500
time        -0.528  0.373  0.373
dosLowds:tm  0.373 -0.528 -0.264 -0.707
dosHghds:tm  0.373 -0.264 -0.528 -0.707  0.500
```

```
b = lme4::fixef(mod.pigs)
W = vcov(mod.pigs)
(L <- matrix(c(rep(0,4),1,0,rep(0,5),1),nrow=2,byrow=TRUE))
```

```
     [,1] [,2] [,3] [,4] [,5] [,6]
[1,]    0    0    0    0    1    0
[2,]    0    0    0    0    0    1
```

```
w2 <-as.numeric( t(L%*%b) %*% solve(L %*% W %*% t(L))%*% (L%*%b))
1- pchisq(w2, df = nrow(L)) #We don't have enough evidence to reject null (thus, don't need
```

```
[1] 0.1695
```

Lastly, another option is a **likelihood ratio test**. Suppose we have two models with the same random effects and covariance models.

- Full model: M_f has p columns in $\mathbf{X}_i$ and thus p fixed effects $\beta_0, ..., \beta_{p-1}$.
- Nested model: M_n has k columns such that $k < p$ and $p - k$ fixed effects $\beta_l = 0$.

If we use maximum likelihood estimation, it makes sense to **compare the likelihood of two models**.

H_0 : nested model is true, H_A : full model is true

If we take the ratio of the likelihoods from the nested model and full model and plug in the maximum likelihood estimators, then we have another statistic.

$$D = -2\log\left(\frac{L_n(\hat{\beta},\hat{\mathbf{V}})}{L_f(\hat{\beta},\hat{\mathbf{V}})}\right) = -2\log(L_n(\hat{\beta},\hat{\mathbf{V}})) + 2\log(L_f(\hat{\beta},\hat{\mathbf{V}}))$$

The sampling distribution of this statistic is approximately **chi-squared** with degrees of freedom equal to the **difference in the number of parameters between the two models**.

```
mod.pigsML <- lmer(bw ~ dose*time + (1|id), data = pigs, REML=FALSE)
mod.pigsML2 <- lmer(bw ~ dose+time + (1|id), data = pigs, REML=FALSE)

anova(mod.pigsML,mod.pigsML2)
```

```
Data: pigs
Models:
mod.pigsML2: bw ~ dose + time + (1 | id)
mod.pigsML: bw ~ dose * time + (1 | id)
            npar AIC BIC logLik -2*log(L) Chisq Df Pr(>Chisq)
mod.pigsML2    6 892 907   -440       880
mod.pigsML     8 893 913   -438       877  3.61  2       0.16
```

```
(Dstat <- -2*logLik(mod.pigsML2)+2*logLik(mod.pigsML))
```

```
'log Lik.' 3.609 (df=6)
```

```
1-pchisq(Dstat,df = 2) #df = difference in number of parameters
```

```
'log Lik.' 0.1645 (df=6)
```

6.8.8.2 Information Criteria for Choosing Fixed Effects

To use BIC to choose models, you must use maximum likelihood estimation instead of REML.

```
mod.pigsML <- lmer(bw ~ dose*time + (1|id), data = pigs, REML=FALSE)
summary(mod.pigsML)
```

```
Linear mixed model fit by maximum likelihood  ['lmerMod']
Formula: bw ~ dose * time + (1 | id)
   Data: pigs

      AIC       BIC    logLik -2*log(L)  df.resid
    892.9     912.9    -438.4     876.9        82

Scaled residuals:
   Min     1Q Median     3Q    Max
-2.656 -0.405  0.034  0.684  2.167

Random effects:
 Groups   Name        Variance Std.Dev.
 id       (Intercept) 1059     32.5
 Residual              675     26.0
Number of obs: 90, groups:  id, 15

Fixed effects:
                   Estimate Std. Error t value
(Intercept)          469.69      18.52   25.36
dose Low dose          4.84      26.19    0.18
doseHigh dose         11.15      26.19    0.43
time                  16.03       2.40    6.66
dose Low dose:time     6.53       3.40    1.92
doseHigh dose:time     3.60       3.40    1.06

Correlation of Fixed Effects:
            (Intr) dsLwds dsHghd time   dsLds:
doseLowdose -0.707
doseHighdos -0.707  0.500
time        -0.563  0.398  0.398
dosLowds:tm  0.398 -0.563 -0.281 -0.707
dosHghds:tm  0.398 -0.281 -0.563 -0.707  0.500
```

```
BIC(mod.pigsML)
```

```
[1] 912.9
```

```
mod.pigsML2 <- lmer(bw ~ dose+time + (1|id), data = pigs, REML=FALSE)
summary(mod.pigsML2)
```

```
Linear mixed model fit by maximum likelihood  ['lmerMod']
```

```
Formula: bw ~ dose + time + (1 | id)
   Data: pigs

      AIC       BIC    logLik -2*log(L)  df.resid
    892.5     907.5    -440.2     880.5        84

Scaled residuals:
    Min      1Q  Median      3Q     Max
-2.8167 -0.4878 -0.0196  0.6569  2.1617

Random effects:
 Groups   Name        Variance Std.Dev.
 id       (Intercept) 1053     32.5
 Residual              708     26.6
Number of obs: 90, groups:  id, 15

Fixed effects:
             Estimate Std. Error t value
(Intercept)    455.05      16.50   27.58
dose Low dose   33.13      21.65    1.53
doseHigh dose   26.77      21.65    1.24
time            19.40       1.42   13.64

Correlation of Fixed Effects:
            (Intr) dsLwds dsHghd
doseLowdose -0.656
doseHighdos -0.656  0.500
time        -0.374  0.000  0.000
```

```
BIC(mod.pigsML2) #This model has a lower BIC
```

```
[1] 907.5
```

6.8.8.3 Hypothesis Testing for Random Effects

The main way to compare models with different random effects is to compare two models with the same fixed effects. Then use a likelihood ratio test with ML to compare the two models' fit. Again, the null hypothesis is that the smaller model (less complex model – fewer parameters) is the true model. If we see a higher log-likelihood with the more complex model, we may get a small p-value suggesting that we reject the null hypothesis in favor of the more complex model.

Below, I compare two models fit to the guinea pig data. They have the same fixed effects, but I allowed one to have a random slope and intercept while the other only has a random slope. By comparing these two models, we find that the one with a random slope and intercept better fits the data. Thus we reject the null hypothesis that the model with the random intercept is true in favor of the model with the random slope and intercept.

```
mod.pigsML2 <- lmer(bw ~ dose+time + (1|id), data = pigs, REML=FALSE)
mod.pigsML3 <- lmer(bw ~ dose+time + (time|id), data = pigs, REML=FALSE)

anova(mod.pigsML2, mod.pigsML3)
```

```
Data: pigs
Models:
mod.pigsML2: bw ~ dose + time + (1 | id)
mod.pigsML3: bw ~ dose + time + (time | id)
            npar AIC BIC logLik -2*log(L) Chisq Df Pr(>Chisq)
mod.pigsML2    6 892 907   -440       880
mod.pigsML3    8 879 899   -432       863  17.3  2    0.00017 ***
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

6.8.9 Predicting Random Effects

The best predictor of the random effects is

$$E(\mathbf{b}_i|\mathbf{Y}_i) = \mathbf{G}\mathbf{Z}_i^T\mathbf{V}_i^{-1}(\mathbf{Y}_i - \mathbf{X}_i\hat{\beta})$$

This is the best linear unbiased predictor (BLUP).

Therefore,

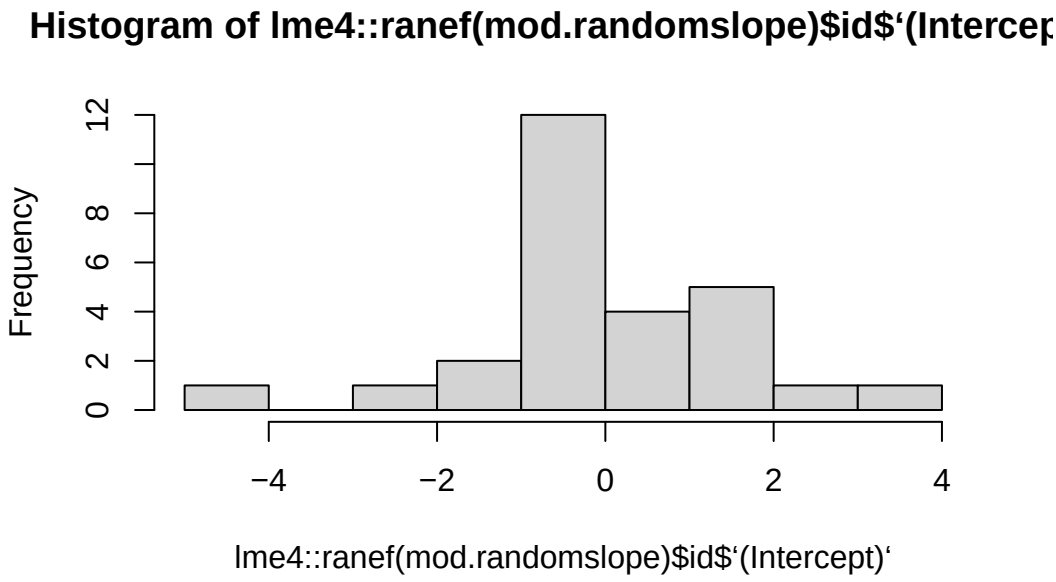
$$\hat{\mathbf{b}}_i = \hat{\mathbf{G}}\mathbf{Z}_i^T\hat{\mathbf{V}}_i^{-1}(\mathbf{Y}_i - \mathbf{X}_i\hat{\beta})$$

which is called the empirical BLUP. We can get this with R's `ranef()` function.

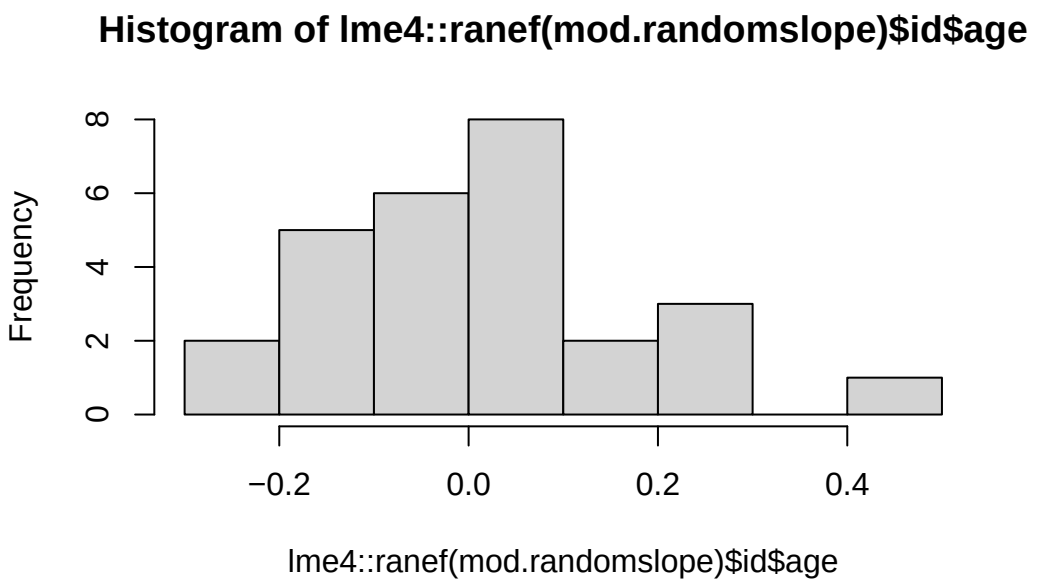
```
lme4::ranef(mod.randomslope)$id
```

```
   (Intercept)       age
1      -0.4859 -0.178215
2      -1.0120  0.009873
3      -0.7730  0.050657
4       1.0694 -0.029879
5       0.5170 -0.167976
6      -0.6205 -0.186561
7      -0.1877 -0.068856
8       1.2507 -0.174430
9      -0.2908 -0.218052
10     -2.2816 -0.250575
11      1.2178  0.083180
12      1.0516  0.215684
13     -0.7276  0.014519
14     -0.1740  0.035857
15      3.0010 -0.065931
16     -1.1769  0.025619
17      1.2178  0.083180
18     -0.6081  0.034911
19      0.8605 -0.094755
20     -0.4446  0.135924
21      2.6535  0.211123
22      0.8907 -0.118846
23     -0.9982  0.114586
24     -4.1305  0.413754
25      0.9045 -0.014133
26     -0.5352  0.208199
27     -0.1877 -0.068856
```

```
hist(lme4::ranef(mod.randomslope)$id$`(Intercept)`)
```




```
hist(lme4::ranef(mod.randomslope)$id$age)
```



We can see here that the predicted random effects are not necessarily Normally distributed (which was an assumption we made in fitting the model).

6.8.10 Predicting Outcomes

Thus our best prediction of $\mathbf{Y}_i$ is

$$\hat{\mathbf{Y}}_i = \mathbf{X}_i\hat{\beta} + \mathbf{Z}_i\hat{\mathbf{b}}_i$$

which can be rewritten as the weighted average of the population marginal mean, $\mathbf{X}_i\hat{\beta}$ and the individual response $\mathbf{Y}_i$,

$$\hat{\mathbf{Y}}_i = (\hat{\Sigma}\hat{\mathbf{V}}_i^{-1})\mathbf{X}_i\hat{\beta} + (I - \hat{\Sigma}\hat{\mathbf{V}}_i^{-1})\mathbf{Y}_i$$

```
predict(mod.randomslope)
```

```
    1     2     3     4     5     6     7     8     9    10    11    12    13
20.13 21.09 22.06 23.02 21.11 22.45 23.79 25.13 21.67 23.10 24.52 25.94 22.87
   14    15    16    17    18    19    20    21    22    23    24    25    26
24.13 25.39 26.65 21.22 22.20 23.18 24.17 19.93 20.88 21.82 22.77 21.30 22.49
   27    28    29    30    31    32    33    34    35    36    37    38    39
23.67 24.85 21.90 22.87 23.84 24.81 20.01 20.89 21.78 22.66 17.76 18.58 19.39
   40    41    42    43    44    45    46    47    48    49    50    51    52
20.21 23.93 25.41 26.90 28.39 24.82 26.57 28.32 30.07 21.43 22.78 24.13 25.48
   53    54    55    56    57    58    59    60    61    62    63    64    65
22.16 23.55 24.94 26.33 24.52 25.70 26.89 28.08 21.07 22.44 23.81 25.19 23.93
   66    67    68    69    70    71    72    73    74    75    76    77    78
25.41 26.90 28.39 21.71 23.10 24.49 25.88 22.15 23.28 24.41 25.54 22.69 24.28
   79    80    81    82    83    84    85    86    87    88    89    90    91
25.87 27.46 26.39 28.13 29.87 31.61 21.98 23.07 24.15 25.23 21.96 23.51 25.06
   92    93    94    95    96    97    98    99   100   101   102   103   104
26.61 21.22 23.37 25.52 27.67 22.83 24.13 25.42 26.71 23.17 24.91 26.65 28.38
  105   106   107   108
21.30 22.49 23.67 24.85
```

6.8.11 Generalized Linear Mixed Effects Models

We can generalize the linear mixed effects model by using a link function, $g()$, to connect the linear model (with random effects)

$$g(E(\mathbf{Y}_i)) = \mathbf{X}_i\beta + \mathbf{Z}_i\mathbf{b}_i$$

where

$$\mathbf{b}_i \sim N(0, G)$$

Similar to GLM's, we assume an outcome distribution from the exponential family that determined the relationship between the mean and the variance,

$$Var(Y_{ij}|\mathbf{b}_i) = \phi v(E(Y_{ij}|\mathbf{b}_i))$$

```
summary(glmer(resp ~ age + smoke + (1|id), data = ohio, family = binomial))
```

```
Generalized linear mixed model fit by maximum likelihood (Laplace
  Approximation) [glmerMod]
 Family: binomial  ( logit )
Formula: resp ~ age + smoke + (1 | id)
   Data: ohio

      AIC       BIC    logLik -2*log(L)  df.resid
   1597.9    1620.6    -794.9    1589.9      2144

Scaled residuals:
   Min     1Q Median     3Q    Max
-1.403 -0.180 -0.158 -0.132  2.518

Random effects:
 Groups Name        Variance Std.Dev.
 id     (Intercept) 5.49     2.34
Number of obs: 2148, groups:  id, 537

Fixed effects:
            Estimate Std. Error z value Pr(>|z|)
(Intercept)   -3.374      0.275  -12.27   <2e-16 ***
age           -0.177      0.068   -2.60   0.0093 **
smoke          0.415      0.287    1.45   0.1484
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

Correlation of Fixed Effects:
      (Intr) age
age    0.227
smoke -0.419 -0.010
```

7 Spatial Data

Compared to time series and longitudinal data, spatial data is indexed by space (in 2 or 3 dimensions).

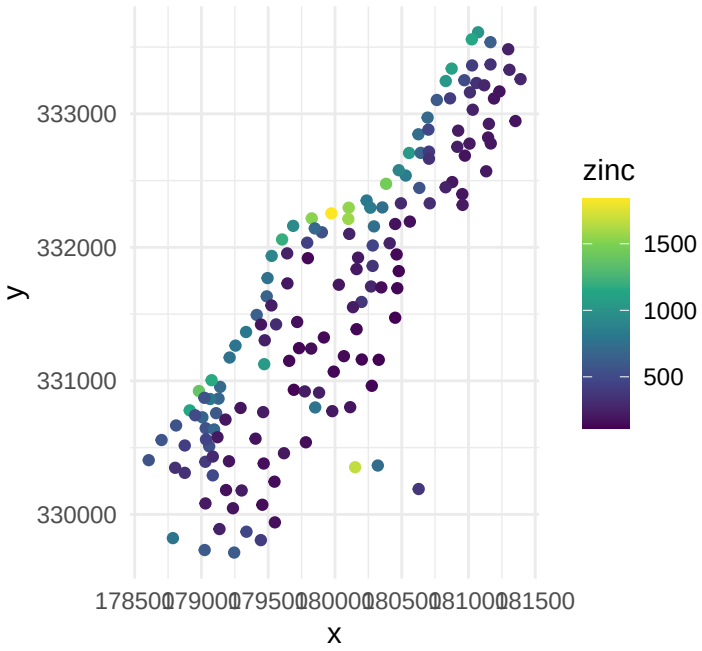
Typically, we have **point-referenced** or **geostatistical** data where our outcome is $Y(s)$ where $s \in \mathbb{R}^d$ and s varies continuously. s may be a point on the globe referenced by its longitude and latitude or a point in another coordinate system. We are typically interested in the relationships between the outcome and explanatory variables and making predictions at locations where we do not have data. We will use that points closer to each other in space are more likely to be similar in value in our endeavors.

Below, we have mapped the value of zinc concentration (coded by color) at 155 spatial locations in a flood plain of the Meuse River in the Netherlands. We might be interested in explaining the variation in zinc concentrations in terms of the distance to the river, flooding frequency, soil type, land use, etc. After building a model to predict the mean zinc concentration, we could use that model to help us understand the current landscape and to make predictions. Remember that to make predictions, we have to observe these characteristics at other spatial locations.

```
require(sp)
```

```
Loading required package: sp
```

```
require(ggmap)
data(meuse)

ggplot(meuse, aes(x = x, y = y, color = zinc)) +
  geom_point() +
  scale_color_viridis_c() +
  coord_equal() +
  theme_minimal()
```



We may not be able to collect data at that fine granularity of spatial location due to a lack of data or to protect the confidentiality of individuals. Instead, we may have **areal** or **lattice** or **discrete** data such that we have aggregate data that summarizes observations within a spatial boundary such as a county or state (or country or within a square grid). In this circumstance, we think that spatial areas are similar if they are close (share a boundary, centers are close to each other, etc.) We must consider correlation based on factors other than longitude and latitude.

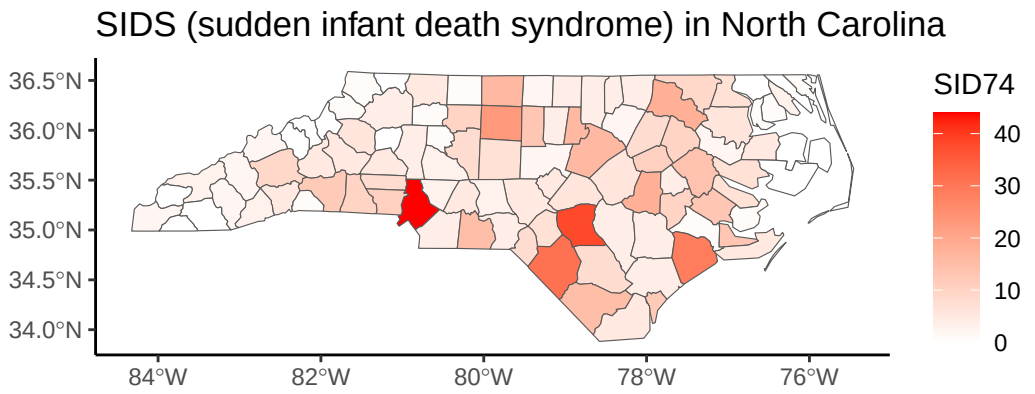
Below, we have mapped the rate of sudden infant death syndrome (SIDS) for countries in North Carolina in 1974. We might be interested in explaining the variation in country SIDS rates in terms of population size, birth rate, and other factors that might explain county-level differences. After building a model to predict the mean SIDS rate, we could use that model to help us understand the current public health landscape, and we can use it to make predictions in the future.

```
library(sf)
nc <- st_read(system.file("shape/nc.shp", package="sf"))
```

```
Reading layer `nc' from data source
  `/Library/Frameworks/R.framework/Versions/4.5-arm64/Resources/library/sf/shape/nc.shp'
  using driver `ESRI Shapefile'
Simple feature collection with 100 features and 14 fields
Geometry type: MULTIPOLYGON
Dimension:     XY
```

```
Bounding box:  xmin: -84.32385 ymin: 33.88199 xmax: -75.45698 ymax: 36.58965
Geodetic CRS:  NAD27
```

```
st_crs(nc) <- "+proj=longlat +datum=NAD27"
row.names(nc) <- as.character(nc$FIPSNO)

ggplot(nc, aes(fill = SID74)) +
  geom_sf() +
  scale_fill_gradient(low='white',high='red') +
  labs(title = "SIDS (sudden infant death syndrome) in North Carolina") +
  theme_classic()
```

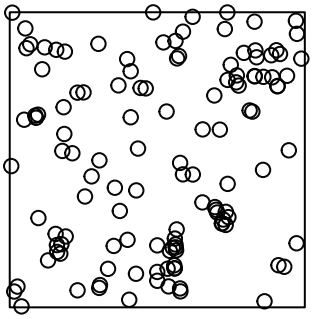


In some disciplines, we may be most interested in the locations themselves and study the **point patterns** or **point processes** to try and determine any structure in the locations.

The data below record the locations of 126 pine saplings in a Finnish forest, including their heights and diameters. We might be interested if there are any patterns in the location of the pines. Is it uniform? Is there clustering? Are there optimal distances between pines such that they'll only grow if they are far enough away from each other (repelling each other)?

```
require(spatstat)
data(finpines)
plot(unmark(finpines), main="Finnish pines: locations")
```

Finnish pines: locations



7.1 Coordinate Reference Systems (CRS)

At the heart of every spatial visualization is a set of locations. One way to describe a location is in terms of coordinates and a coordinate reference system (known as CRS).

There are three main components to a CRS: ellipsoid, datum, and a projection.

7.1.1 Ellipsoid

While you might have learned that the Earth is a sphere, it is actually closer to an ellipsoid with a bulge at the equator. Additionally, the surface is irregular and not smooth. To define a CRS, we first need to choose a mathematical model represent a smooth approximation to the shape of the Earth. The common ellipsoid models are known as WGS84 and GRS80. See the illustration below of one ellipsoid model (shown in black) as compared to Earth's true irregular surface (shown in red).

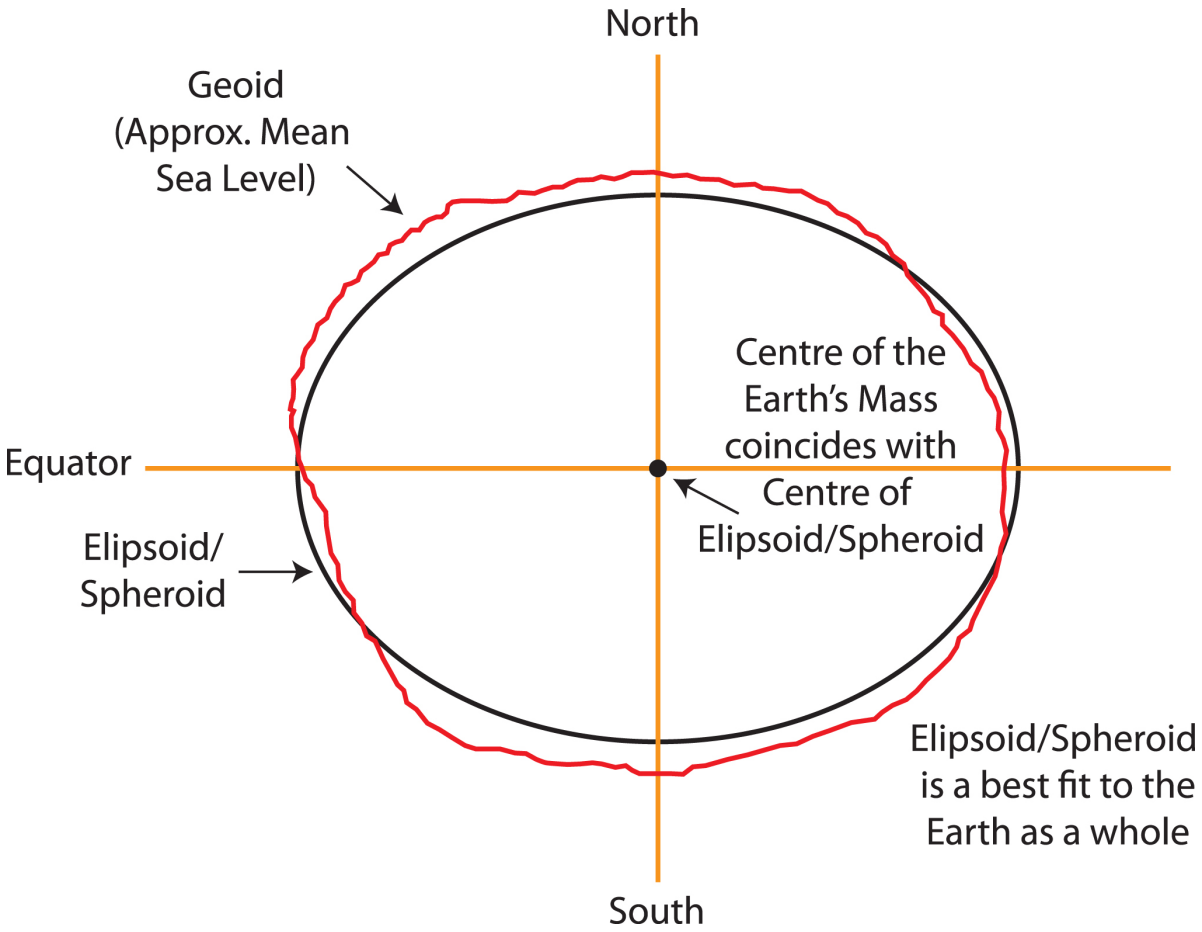


Figure 7.1: Illustration of ellipsoid model and Earth's irregular surface, centered to have an overall best fit. Source: www.icsm.gov.au

7.1.2 Datum

Each ellipsoid model has different ways to position it self relative to Earth depending on the center or origin. Each potential position and reference frame for representing the position of locations on Earth is called a datum.

For example, two different datum for the same ellipsoid model can provide a more accurate fit or approximation of the Earth's surface depending on the region of interest (South America v. North America). For example, the NAD83 datum is a good fit for the GRS80 ellipsoid in North America, but SIRGAS2000 is a better fit for the GRS80 ellipsoid in South America. The illustration below shows one datum in which the center of the ellipsoid does not coincide with the center of Earth's mass. With this position of the ellipsoid, we gain a better fit for the southern half of the Earth.

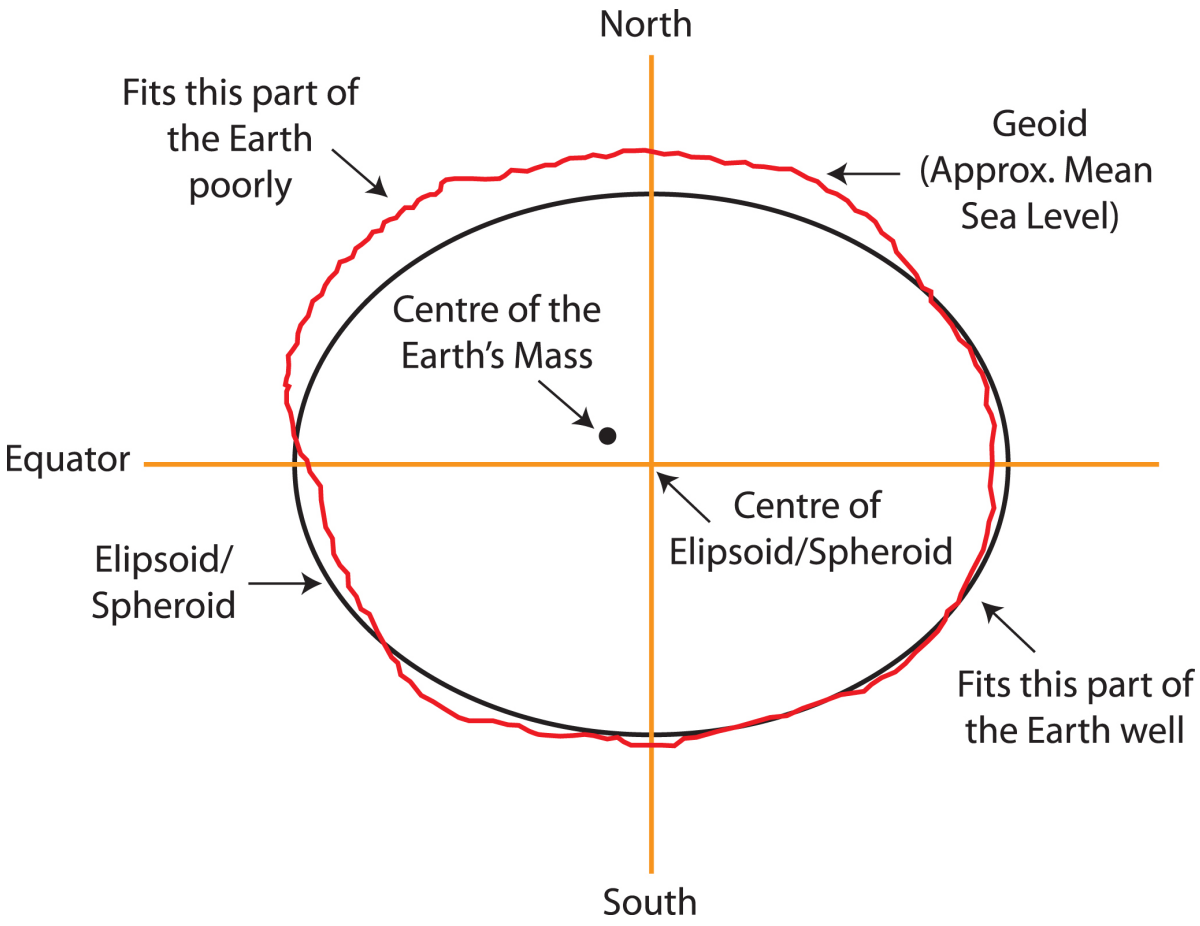


Figure 7.2: Illustration of ellipsoid model and Earth's irregular surface for a datum that better fits southern part (bottom right) of the Earth. Source: www.icsm.gov.au

It is useful to know that the Global Positioning System (GPS) uses the WGS84 ellipsoid model and a datum by the same name, which provides an overall best fit of the Earth.

If you have longitude and latitude coordinates for a location, it is important to know what datum and ellipsoid were used to define those positions.

Note: In practice, the horizontal distance between WGS84 and NAD83 coordinates is about 3-4 feet in the US, which may not be significant for most applications.

The 3 most common datum/ellipsoids used in the U.S.:

WGS84 (EPSG: 4326)

```
+init=epsg:4326 +proj=longlat +ellps=WGS84
+datum=WGS84 +no_defs +towgs84=0,0,0
```

```
## CRS used by Google Earth and the U.S. Department of Defense for all their mapping.
## Tends to be used for global reference systems.
## GPS satellites broadcast the predicted WGS84 orbits.
```

NAD83 (EPSG: 4269)

```
+init=epsg:4269 +proj=longlat +ellps=GRS80 +datum=NAD83
+no_defs +towgs84=0,0,0
##Most commonly used by U.S. federal agencies.
## Aligned with WGS84 at creation, but has since drifted.
## Although WGS84 and NAD83 are not equivalent, for most applications they are very similar.
```

NAD27 (EPSG: 4267)

```
+init=epsg:4267 +proj=longlat +ellps=clrk66 +datum=NAD27
+no_defs
+nadgrids=@conus,@alaska,@ntv2_0.gsb,@ntv1_can.dat
## Has been replaced by NAD83, but is still encountered!
```

For more resources related to EPSG, go to https://epsg.io/ and https://spatialreference.org/.

7.1.3 Projection

Lastly, we must **project** the 3D ellipse on a 2D surface to make a map with Easting and Northing coordinates. Flattening a round object without distortion is impossible, resulting in trade-offs between area, direction, shape, and distance. For example, distance and direction are trade-offs because both features can not be simultaneously preserved. No "best" projection exists, but some are better suited to different applications.

For a good overview of common projection methods, see https://pubs.usgs.gov/gip/70047422/report.pdf.

One of the most commonly used projection is the Mercator projection which is a cylindrical map projection from the 1500's. It became popular for navigation because it represented north as up and south as down everywhere and preserves local directions and shape. However, it inflates the size of regions far from the equator. Thus, Greenland, Antarctica, Canada, and Russia appear large relative to their actual land mass as compared to Central Africa. See the illustration below to compare the area/shape of the countries with the Mercator projection of the world (light blue) with the true areas/shapes (dark blue).

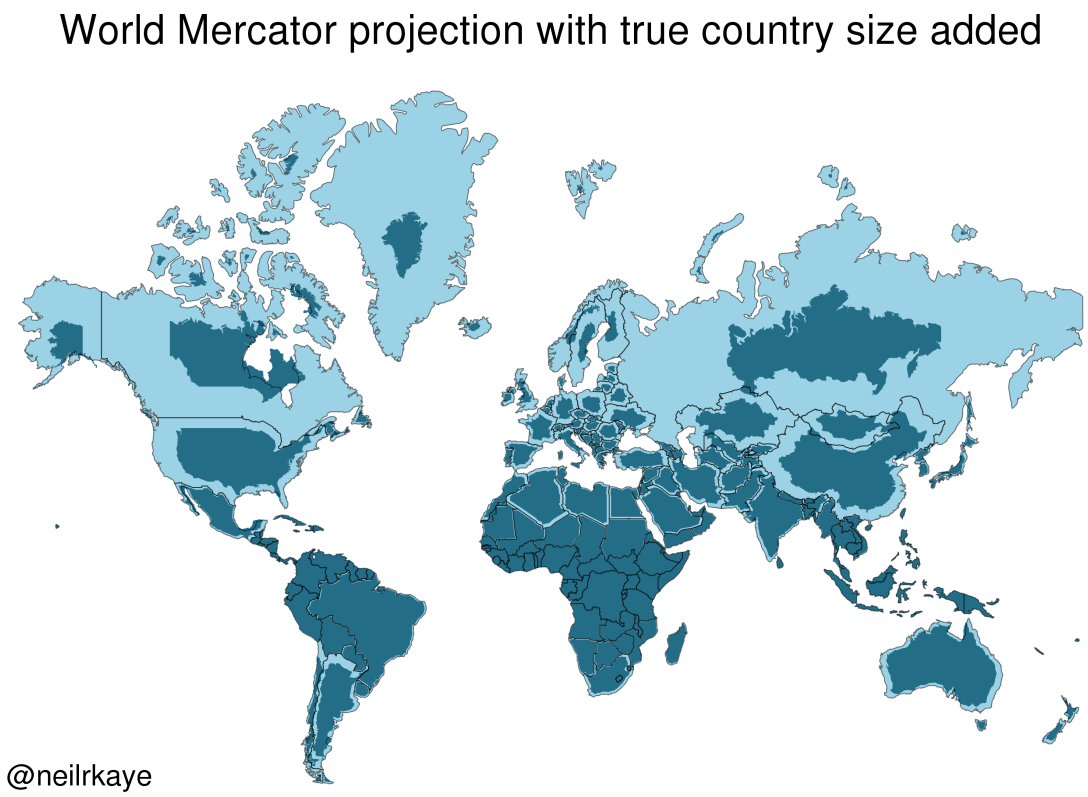
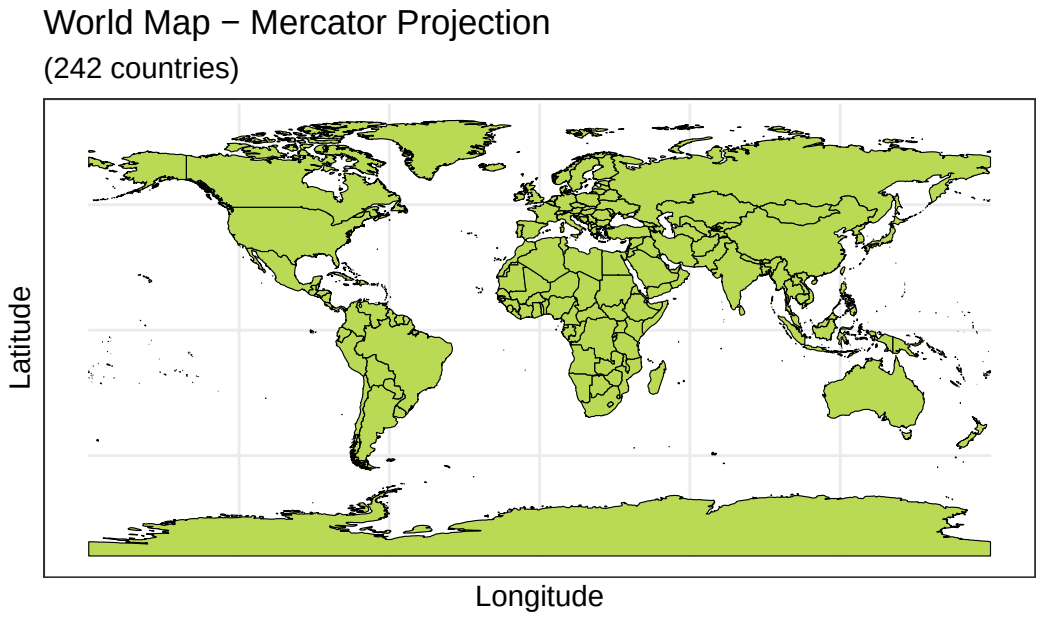


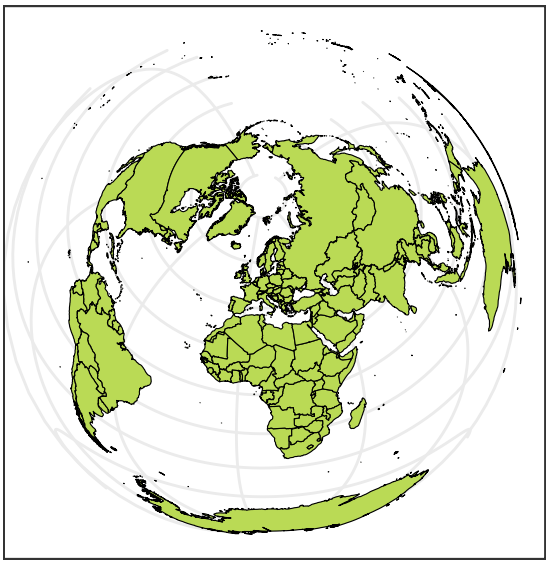
Figure 7.3: Source: (**neilrkaye?**)

Below you can see four different world projections. Take note of what is lost in terms of angle, area, or distance in these projections.

```
library(rnaturalearth)
world <- ne_countries(scale = "medium", returnclass = "sf")

# Basic Map w/ labels
ggplot(data = world) +
  geom_sf(color = "black", fill = "#bada55") +
  labs(x = "Longitude", y = "Latitude", title = "World Map - Mercator Projection", subtitle =
  theme_bw()
```



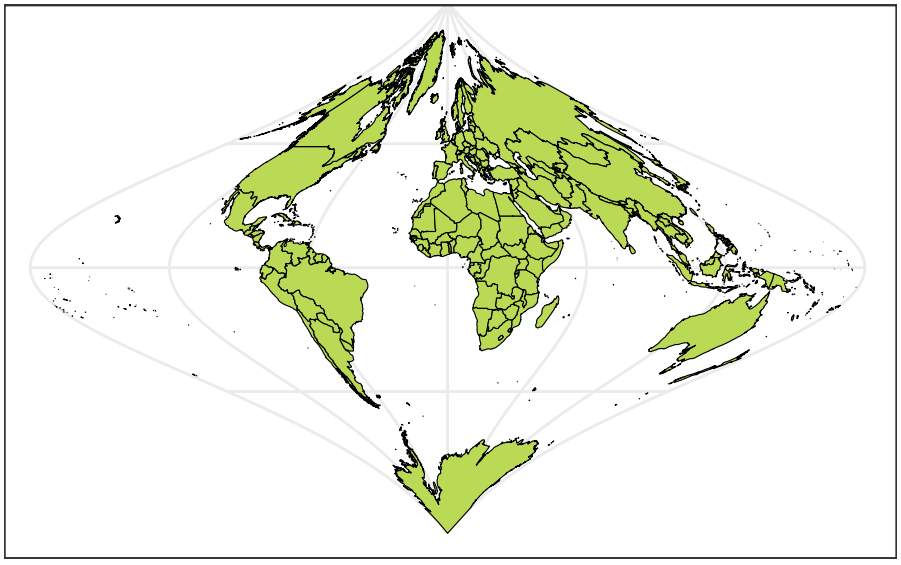

```
ggplot(data = world) +
    geom_sf(color = "black", fill = "#bada55") +
    coord_sf(crs = "+proj=laea +lat_0=52 +lon_0=10 +x_0=4321000 +y_0=3210000 +ellps=GRS80 +u
  labs(title = "Lambert Azimuthal Equal-Area Projection", subtitle = "Correctly represents a
  theme_bw()
```



```
ggplot(data = world) +
    geom_sf(color = "black", fill = "#bada55") +
    coord_sf(crs = "+proj=fouc") +
  labs(title = "Foucaut Projection", subtitle = "Correctly represents area, lots of distortio
  theme_bw()
```

Foucaut Projection

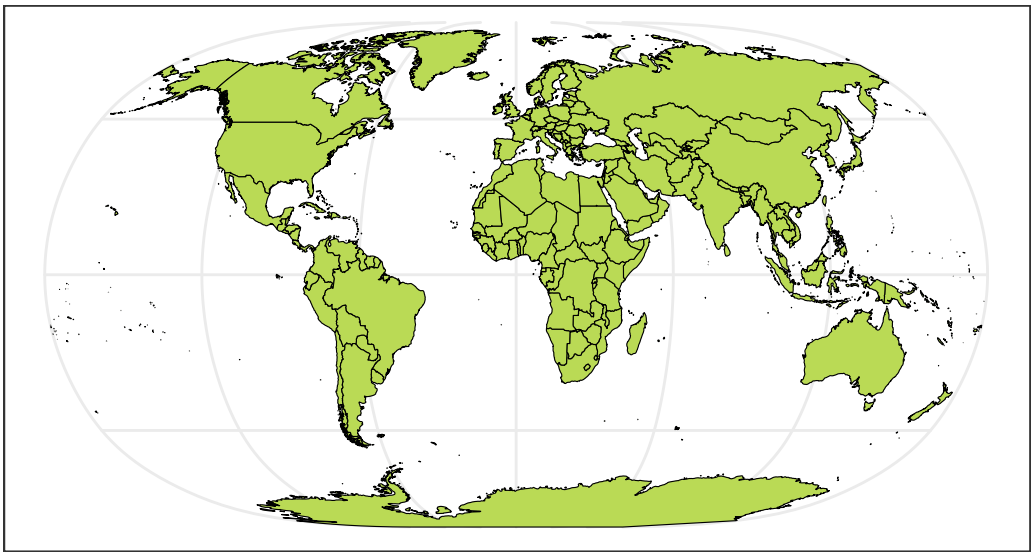
Correctly represents area, lots of distortion in high latitudes



```
ggplot(data = world) +
    geom_sf(color = "black", fill = "#bada55") +
    coord_sf(crs = "+proj=natearth2") +
    labs(title = "Natural Earth II Projection", subtitle = "Represents globe shape, distorted
  theme_bw()
```

Natural Earth II Projection

Represents globe shape, distorted at high latitudes



It is important to consider what ellipsoid, datum, and projection your locations have been recorded and mapped using.

7.2 Data Models

7.2.1 Vector

Vector data represents the world as a set of spatial geometries that are defined in terms of location coordinates (with a specified CRS) with non-spatial attributes or properties.

The three basic geometries are

- Points: Locations defined based on a (x, y) coordinates.
- Lines: A set of ordered points connected by straight lines.
- Polygons: A set of ordered points connected by straight lines, first and last point are the same.

For example, city locations can be represented with points, roads and rivers can be represented by lines, and geo-political boundaries and lakes can be represented by polygons.

Hundreds of file formats exist to store spatial vector data. A text file (such as .csv) can store the coordinates in two columns (x,y) in addition to a group id (needed for lines and polygons) plus attributes or properties in additional columns. Note that text files do not store the CRS. However, shapefiles (.shp) developed by ESRI is one of the most widely supported

spatial vector file format (that includes the CRS). Additionally, GeoJSON (.geojson) and KML (.kml) are additional popular formats.

7.2.2 Raster

Raster data represents the world using a continuous grid of cells where each cell has a single value. These values could be continuous such as elevation or precipitation or categorical such as land cover or soil type.

Typically regular cells are square in shape but they can be rotated and sheared. Rectilinear and curvilinear shapes are also possible, depending on the spatial region of interest and CRS.

Be aware that high resolution raster data involves a large number of small cells. This can slow down the computations and visualizations.

Many raster file formats exist. One of the most popular is GeoTIFF (.tif or .tiff). More complex raster formats include NetCDF (.nc) and HDF (.hdf). To work with raster data in R, you'll use the raster, terra, and the stars packages. If you are interested in learning more, check out https://r-spatial.github.io/stars/.

7.3 Working with Spatial Data in R

The technology available in R is rapidly evolving and improving. In this set of notes, I've highlighted some basics for working with spatial data in R, but I list some good resources below.

- https://cran.r-project.org/web/views/Spatial.html (last updated in 2023)
- https://r-spatial.org/book/ (last updated in 2023)
- https://r-spatial.github.io/sf/index.html (last updated in 2023)
- https://r.geocompx.org/ (last updated in 2023)
- http://www.nickeubank.com/gis-in-r/ (last updated in 2023)
- https://cengel.github.io/R-spatial/ (last updated in 2019)

7.3.1 R Packages

The following R packages support spatial data classes (data sets that are indexed with geometries):

- `sf`: generic support for working with spatial data
- `geojsonsf`: read in geojson files

The following R packages contain cultural and physical boundaries, and raster maps:

- `maps`: polygon maps of the world
- `USAboundaries`: contemporary US state, county, and Congressional district boundaries, as well as zip code tabulation area centroids
- `rnaturalearth`: hold and facilitate interaction with Natural Earth map data

The following R packages support geostatistical/point-referenced data analysis:

- `gstat`: classical geostatistics
- `geoR`: model-based geostatistics
- `RandomFields`: stochastic processes
- `akima`: interpolation

The following R packages support regional/areal data analysis:

- `spdep`: spatial dependence
- `spgwr`: geographically weighted regression

The following R packages support point patterns/processes data analysis:

- `spatstat`: parametric modeling, diagnostics
- `splancs`: non-parametric, space-time

7.3.2 Read in data to R

For each file format, we need use a different function to read in the data. See the examples below for reading in GeoJSON, csv, and shapefiles.

```
# Read in GeoJSON file
hex_spatial <- geojsonsf::geojson_sf("data/us_states_hexgrid.geojson")

# Read in CSV File
pop_growth <- readr::read_csv('data/apportionment.csv') %>% janitor::clean_names()
```

```
Rows: 684 Columns: 10
-- Column specification --------------------------------------------------------
Delimiter: ","
chr (2): Name, Geography Type
dbl (5): Year, Percent Change in Resident Population, Resident Population De...
num (3): Resident Population, Resident Population Density, Average Apportion...

i Use `spec()` to retrieve the full column specification for this data.
i Specify the column types or set `show_col_types = FALSE` to quiet this message.
```

```
# Read in Shapefiles
mn_cities <- sf::read_sf('data/shp_loc_pop_centers') #shp file/folder
```

```
Warning in CPL_read_ogr(dsn, layer, query, as.character(options), quiet, :
automatically selected the first layer in a data source containing more than
one.
```

```
mn_water <- sf::read_sf('data/shp_water_lakes_rivers') #shp file/folder
```

7.3.3 Data classes in R

When data is read it, an R data object is created of a default class. Notice the classes of the R objects we read in. Also, notice that an object may have multiple classes, which indicate the type of structure it has and how functions may interact with the object.

```
class(hex_spatial)
```

```
[1] "sf"         "data.frame"
```

```
class(pop_growth)
```

```
[1] "spec_tbl_df" "tbl_df"      "tbl"         "data.frame"
```

```
class(mn_cities)
```

```
[1] "sf"         "tbl_df"     "tbl"        "data.frame"
```

```
class(mn_water)
```

```
[1] "sf"         "tbl_df"     "tbl"        "data.frame"
```

You also may encounter classes such as `SpatialPoints`, `SpatialLines`, and `SpatialPolygons`, or `Spatial*DataFrame` from the `sp` package. The community is moving away from using older `sp` classes to `sf` classes. It is useful for you to know that the older versions exist but stick with the `sf` classes.

- `sfc` objects are modern, general versions of the spatial geometries from the `sp` package with a bbox, CRS, and many geometries available.
- `sf` objects are data.frame-like objects with a geometry column of class `sfc`

7.3.4 Convert data class types

We can convert objects between these data classes with the following functions:

- `fortify(x)`: `sp` object x to `data.frame`
- `st_as_sf(x )`: `sp` object x to `sf`
- `st_as_sf(x, coords = c("long", "lat"))`: `data.frame` x to `sf` as points
- To convert a `data.frame` with columns of `long`, `lat`, and `group` containing polygon geometry information, you can use:

```
st_as_sf(x, coords = c("long", "lat")) %>%
group_by(group) %>%
  summarise(geometry = st_combine(geometry)) %>%
  st_cast("POLYGON")
```

7.3.5 Static Visualizations

In general, if you have geometries (points, polygons, lines, etc.) that you want to plot, you can use `geom_sf()` with `ggplot()`. See https://ggplot2.tidyverse.org/reference/ggsf.html for more details.

7.3.5.1 Plotting points

If you have x and y coordinates (longitude and latitude over a small region), we can use our typical plotting tools in `ggplot2` package using the x and y coordinates as the x and y values in `geom_point()`. Then you can color the points according to a covariate or outcome variable.

If you have longitude and latitude over the globe or a larger region, you must project those coordinates onto a 2D surface. You can do this using the sf package and `st_transform()` after specifying the CRS (documentation: https://www.rdocumentation.org/packages/sf/versions/0.8-0/topics/st_transform). Then we can use `geom_sf()`.

We'll walk through create a map of MN with different layers of information (city point locations, county polygon boundaries, rivers as lines and polygons, and a raster elevation map). To add all of this information on one map, we need to ensure that the CRS is the same for all spatial datasets.

```
#check CRS
st_crs(mn_cities)
```

```
Coordinate Reference System:
  User input: NAD83 / UTM zone 15N
  wkt:
PROJCRS["NAD83 / UTM zone 15N",
    BASEGEOGCRS["NAD83",
        DATUM["North American Datum 1983",
            ELLIPSOID["GRS 1980",6378137,298.257222101,
                LENGTHUNIT["metre",1]]],
        PRIMEM["Greenwich",0,
            ANGLEUNIT["degree",0.0174532925199433]],
        ID["EPSG",4269]],
    CONVERSION["UTM zone 15N",
        METHOD["Transverse Mercator",
            ID["EPSG",9807]],
        PARAMETER["Latitude of natural origin",0,
            ANGLEUNIT["Degree",0.0174532925199433],
            ID["EPSG",8801]],
        PARAMETER["Longitude of natural origin",-93,
            ANGLEUNIT["Degree",0.0174532925199433],
            ID["EPSG",8802]],
        PARAMETER["Scale factor at natural origin",0.9996,
            SCALEUNIT["unity",1],
            ID["EPSG",8805]],
        PARAMETER["False easting",500000,
            LENGTHUNIT["metre",1],
            ID["EPSG",8806]],
        PARAMETER["False northing",0,
            LENGTHUNIT["metre",1],
            ID["EPSG",8807]]],
    CS[Cartesian,2],
        AXIS["(E)",east,
            ORDER[1],
            LENGTHUNIT["metre",1]],
        AXIS["(N)",north,
            ORDER[2],
            LENGTHUNIT["metre",1]],
    ID["EPSG",26915]]
```

```
#check CRS
st_crs(mn_water)
```

```
Coordinate Reference System:
```

```
  User input: NAD83 / UTM zone 15N
  wkt:
PROJCRS["NAD83 / UTM zone 15N",
    BASEGEOGCRS["NAD83",
        DATUM["North American Datum 1983",
            ELLIPSOID["GRS 1980",6378137,298.257222101,
                LENGTHUNIT["metre",1]]],
        PRIMEM["Greenwich",0,
            ANGLEUNIT["degree",0.0174532925199433]],
        ID["EPSG",4269]],
    CONVERSION["UTM zone 15N",
        METHOD["Transverse Mercator",
            ID["EPSG",9807]],
        PARAMETER["Latitude of natural origin",0,
            ANGLEUNIT["Degree",0.0174532925199433],
            ID["EPSG",8801]],
        PARAMETER["Longitude of natural origin",-93,
            ANGLEUNIT["Degree",0.0174532925199433],
            ID["EPSG",8802]],
        PARAMETER["Scale factor at natural origin",0.9996,
            SCALEUNIT["unity",1],
            ID["EPSG",8805]],
        PARAMETER["False easting",500000,
            LENGTHUNIT["metre",1],
            ID["EPSG",8806]],
        PARAMETER["False northing",0,
            LENGTHUNIT["metre",1],
            ID["EPSG",8807]]],
    CS[Cartesian,2],
        AXIS["(E)",east,
            ORDER[1],
            LENGTHUNIT["metre",1]],
        AXIS["(N)",north,
            ORDER[2],
            LENGTHUNIT["metre",1]],
    ID["EPSG",26915]]
```

```r
#transform CRS of water to the same of the cities
mn_water <- mn_water %>%
  st_transform(crs = st_crs(mn_cities))
```

```r
# install.packages("remotes")
# remotes::install_github("ropensci/USAboundaries")
# remotes::install_github("ropensci/USAboundariesData")

# Load country boundaries data as sf object
mn_counties <- USAboundaries::us_counties(resolution = "high", states = "Minnesota")

# Remove duplicate column names
names_counties <- names(mn_counties)
names(mn_counties)[names_counties == 'state_name'] <- c("state_name1", "state_name2")

# Check CRS
st_crs(mn_counties)
```

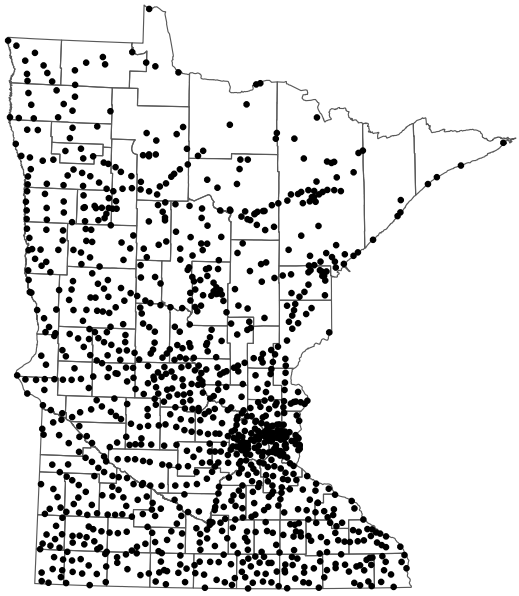
```
Coordinate Reference System:
  User input: EPSG:4326
  wkt:
GEOGCRS["WGS 84",
    DATUM["World Geodetic System 1984",
        ELLIPSOID["WGS 84",6378137,298.257223563,
            LENGTHUNIT["metre",1]]],
    PRIMEM["Greenwich",0,
        ANGLEUNIT["degree",0.0174532925199433]],
    CS[ellipsoidal,2],
        AXIS["geodetic latitude (Lat)",north,
            ORDER[1],
            ANGLEUNIT["degree",0.0174532925199433]],
        AXIS["geodetic longitude (Lon)",east,
            ORDER[2],
            ANGLEUNIT["degree",0.0174532925199433]],
    USAGE[
        SCOPE["Horizontal component of 3D system."],
        AREA["World."],
        BBOX[-90,-180,90,180]],
    ID["EPSG",4326]]
```

```r
# Transform the CRS of county data to the more local CRS of the cities
mn_counties <- mn_counties %>%
  st_transform(crs = st_crs(mn_cities))

st_crs(mn_counties)
```

```
Coordinate Reference System:
  User input: NAD83 / UTM zone 15N
  wkt:
PROJCRS["NAD83 / UTM zone 15N",
    BASEGEOGCRS["NAD83",
        DATUM["North American Datum 1983",
            ELLIPSOID["GRS 1980",6378137,298.257222101,
                LENGTHUNIT["metre",1]]],
        PRIMEM["Greenwich",0,
            ANGLEUNIT["degree",0.0174532925199433]],
        ID["EPSG",4269]],
    CONVERSION["UTM zone 15N",
        METHOD["Transverse Mercator",
            ID["EPSG",9807]],
        PARAMETER["Latitude of natural origin",0,
            ANGLEUNIT["Degree",0.0174532925199433],
            ID["EPSG",8801]],
        PARAMETER["Longitude of natural origin",-93,
            ANGLEUNIT["Degree",0.0174532925199433],
            ID["EPSG",8802]],
        PARAMETER["Scale factor at natural origin",0.9996,
            SCALEUNIT["unity",1],
            ID["EPSG",8805]],
        PARAMETER["False easting",500000,
            LENGTHUNIT["metre",1],
            ID["EPSG",8806]],
        PARAMETER["False northing",0,
            LENGTHUNIT["metre",1],
            ID["EPSG",8807]]],
    CS[Cartesian,2],
        AXIS["(E)",east,
            ORDER[1],
            LENGTHUNIT["metre",1]],
        AXIS["(N)",north,
            ORDER[2],
            LENGTHUNIT["metre",1]],
    ID["EPSG",26915]]
```

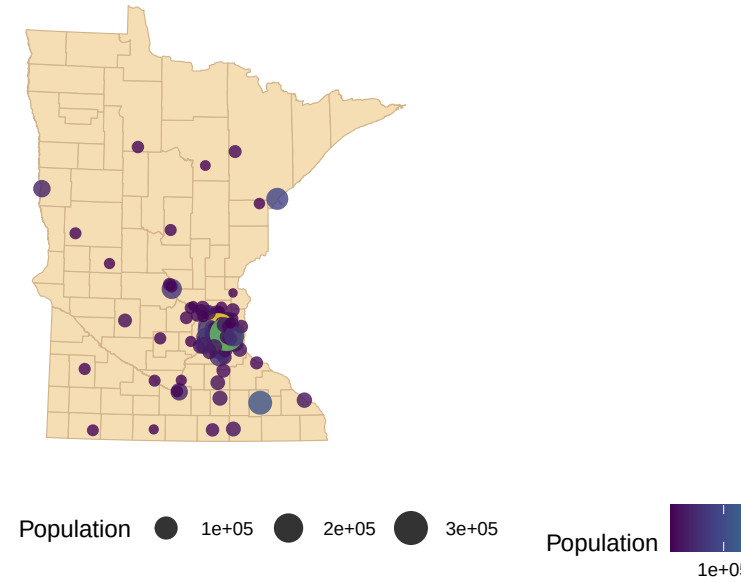
```r
ggplot() + # plot frame
  geom_sf(data = mn_cities, size = 0.5) + # city point layer
  ggthemes::theme_map()
```

```
ggplot() + # plot frame
  geom_sf(data = mn_counties, fill = NA) + # county boundary layer
  geom_sf(data = mn_cities, size = 0.5) + # city point layer
  ggthemes::theme_map()
```



```
ggplot() +
  geom_sf(data = mn_counties, fill = 'wheat', color = "tan") +
```

```
  geom_sf(data = mn_cities %>% filter(Population >= 10000), mapping = aes(color = Population
  scale_color_viridis_c() + #continuous (gradient) color scale
  labs(title = "Minnesota Cities with Population >= 10,000") +
  ggthemes::theme_map() + theme(legend.position = "bottom")  #move legend
```



7.3.5.2 Plotting polygons

If you have areal data, you'll need shapefiles with boundaries for those polygons. City, state, or federal governments often provide these. We can read them into R with `st_read()` in the sf package. Once we have that stored object, we can view shapefile metadata using the `st_geometry_type()`, `st_crs()`, and `st_bbox()`. These tell you about the type of geometry stored about the shapes, the CRS, and the bounding box that determines the study area of interest.

```
#read in shapefile information
shapefile <- st_read(shapefile)
```

If we have data that are points that we will aggregate to a polygon level, then we could use code such as the one below to join together the summaries at an average longitude and latitude coordinate with the shapefiles by whether the longitude and latitude intersect with the polygon.

```
#join the shapefile and our data summaries with a common polygon I.D. variable
fulldat <- left_join(shapefile, dat)
```

```
hex_growth <- hex_spatial %>%
  mutate(name = str_replace(google_name,' \\(United States\\)',''),
         abbr = iso3166_2) %>%
  left_join(pop_growth, by = 'name')
```

If we have data that are points that we will aggregate to a polygon level, then we could use code such as the one below to join together the summaries at an average longitude and latitude coordinate with the shapefiles by whether the longitude and latitude intersect with the polygon.

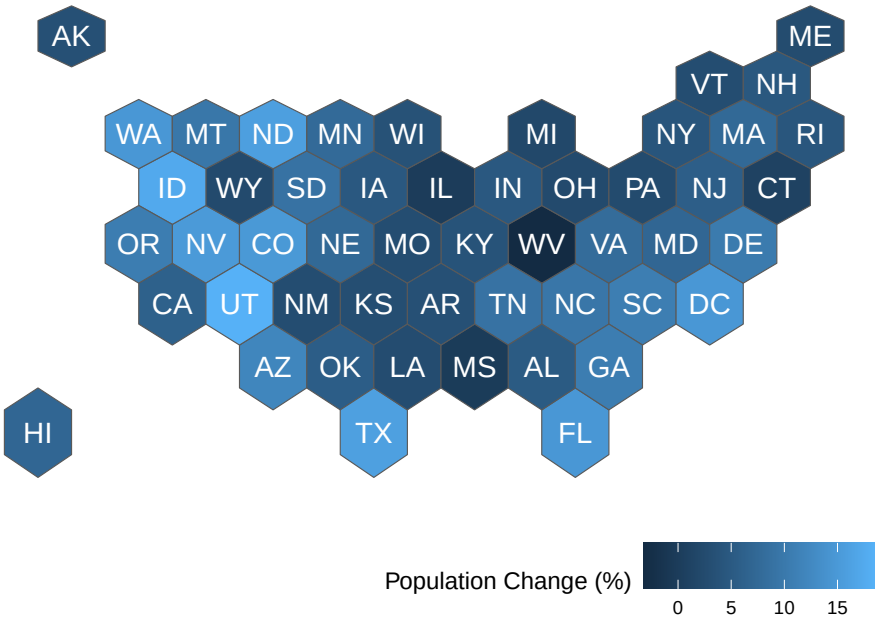
```
# make our data frame a spatial data frame
dat <- st_as_sf(originaldatsum, coords = c('Longitude','Latitude'))
#copy the coordinate reference info from the shapefile onto our new data frame
st_crs(dat) <- st_crs(shapefile)

#join the shapefile and our data summaries
fulldat <- st_join(shapefile, dat, join = st_intersects)
```

If you are working with U.S. counties, states, or global countries, the shapefiles are already available in the map package. You'll need to use the I.D. variable to join this data with your polygon-level data.

Once we have an sf object with attributes (variables) and geometry (polygons), we can use `geom_sf(aes(fill = attribute))` to plot and color the polygons in the ggplot2 context with respect to an outcome variable.

```
hex_growth%>%  # start with sf object
  filter(year == 2020) %>% #filter to focus on data from 2020
  ggplot() +
  geom_sf(aes(fill = percent_change_in_resident_population)) + # plot the sf geometry (polyg
  geom_sf_text( aes(label = abbr), color = 'white') + # add text labels to the sf geometry re
  labs(fill = 'Population Change (%)') + # Change legend label
  ggthemes::theme_map() + theme(legend.position = 'bottom', legend.justification = 'right') #
```

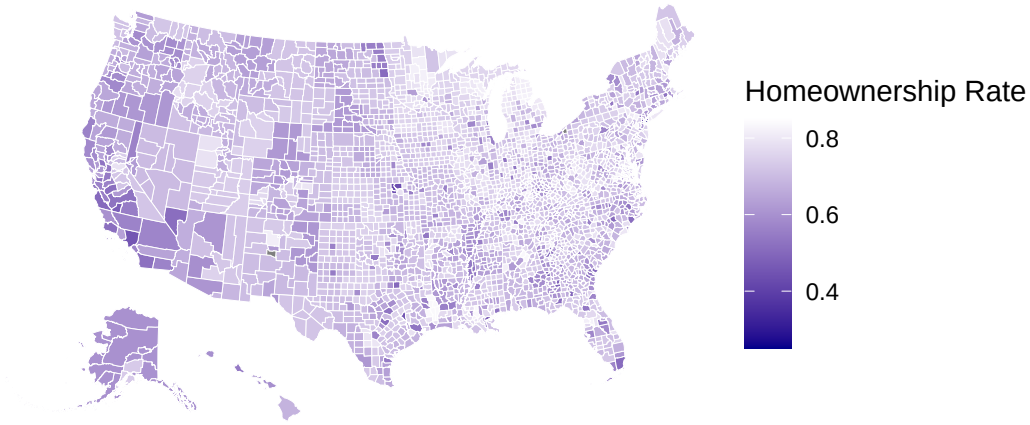
```
Warning in st_point_on_surface.sfc(sf::st_zm(x)): st_point_on_surface may not
give correct results for longitude/latitude data
```




```
# install.packages('devtools')
# devtools::install_github("UrbanInstitute/urbnmapr")
library(urbnmapr) # projection with Alaska and Hawaii close to lower 48 states
get_urbn_map(map = "counties", sf = TRUE) %>%
    left_join(countydata) %>%
    ggplot() + geom_sf(aes(fill = horate), color = 'white',linewidth = 0.01) +
    labs(fill = 'Homeownership Rate') +
    scale_fill_gradient(high='white',low = 'darkblue',limits=c(0.25,0.85)) +
    theme_void()
```

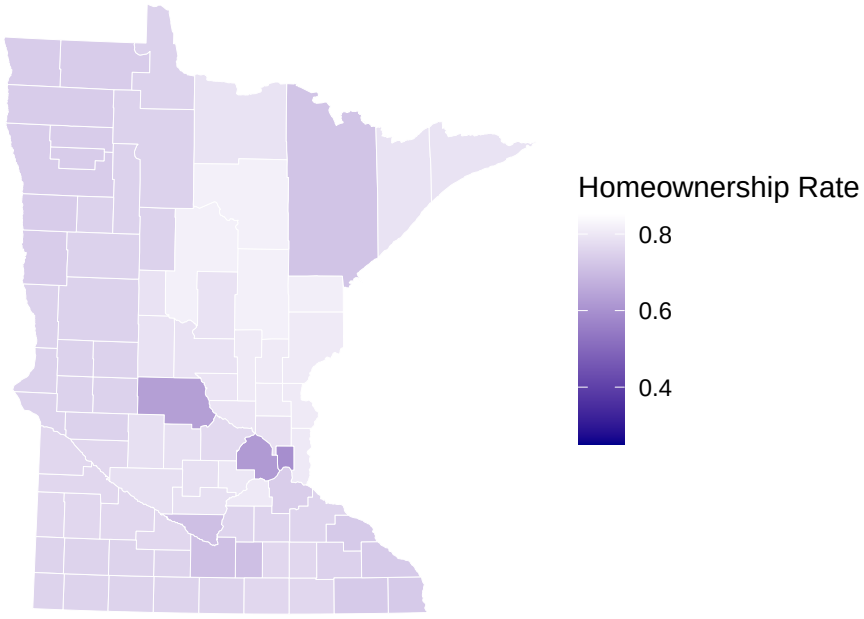
```
Joining with `by = join_by(county_fips)`
old-style crs object detected; please recreate object with a recent
sf::st_crs()
```

```
Warning in CPL_crs_from_input(x): GDAL Message 1: CRS EPSG:2163 is deprecated.
Its non-deprecated replacement EPSG:9311 will be used instead. To use the
original CRS, set the OSR_USE_NON_DEPRECATED configuration option to NO.
```




```
# Subset to M.N.
get_urbn_map(map = "counties", sf = TRUE) %>%
    left_join(countydata) %>%
    filter(stringr::str_detect(state_abbv,'MN')) %>%
    ggplot() + geom_sf(aes(fill = horate), color = 'white',linewidth = 0.05) +
    labs(fill = 'Homeownership Rate') +
    coord_sf(crs=26915) +
  scale_fill_gradient(high='white',low = 'darkblue',limits=c(0.25,0.85)) +
  theme_void()
```

```
Joining with `by = join_by(county_fips)`
old-style crs object detected; please recreate object with a recent
sf::st_crs()
```



7.3.5.3 Plotting Raster

To include raster images in the visualization, we need to obtain/load raster data. Below shows code to get the elevation raster image for MN.

Then we need to convert the raster to a data.frame to plot using `geom_raster` as an additional layer to `ggplot()`.

```
elevation <- elevatr::get_elev_raster(mn_counties, z = 5, clip = 'bbox')
```

```
Mosaicing & Projecting
```

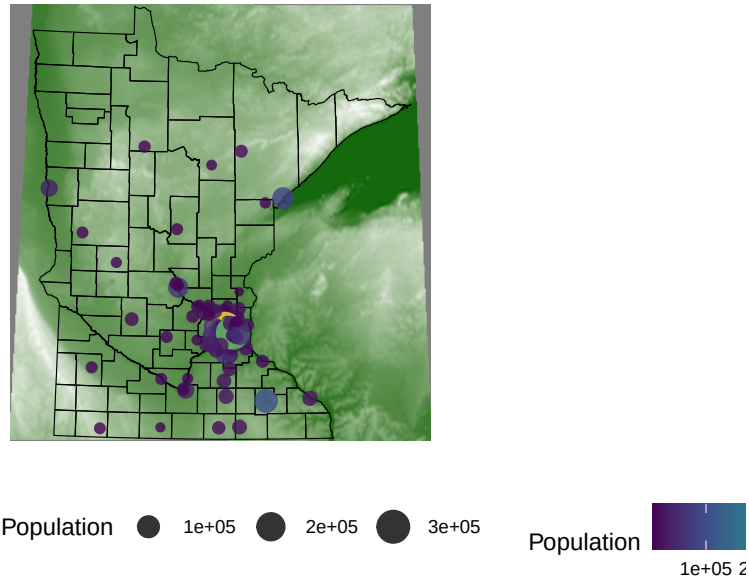
```
Clipping DEM to bbox
```

```
Note: Elevation units are in meters.
```

```
raster::crs(elevation) <- sf::st_crs(mn_counties)

# Convert to Data Frame for plotting
elev_df <- elevation %>% terra::as.data.frame(xy = TRUE)
names(elev_df) <-c('x','y','value')
```

```
ggplot() +
  geom_raster(data = elev_df, aes(x = x,y = y,fill = value)) + # adding the elevation as firs
  geom_sf(data = mn_counties, fill = NA, color = "black") +
  geom_sf(data = mn_cities %>% filter(Population >= 10000), mapping = aes(color = Population
  scale_color_viridis_c() + #continuous (gradient) color scale
  scale_fill_gradient(low = 'darkgreen',high = 'white', guide = FALSE) +
  labs(title = "Minnesota Cities with Population >= 10,000") +
  ggthemes::theme_map() + theme(legend.position = "bottom")  #move legend
```

```
Warning: The `guide` argument in `scale_*()` cannot be `FALSE`. This was deprecated in
ggplot2 3.3.4.
i Please use "none" instead.
```



To demonstrate multiple layers on one visualization, let's zoom into the Twin Cities and add waterways and rivers.

```
Seven_countyarea <- st_bbox(mn_counties %>% filter(name %in% c("Anoka", "Hennepin", "Ramsey"

elevation <- elevatr::get_elev_raster(mn_counties %>% st_crop(Seven_countyarea), z = 9, clip
```

```
Warning: attribute variables are assumed to be spatially constant throughout
all geometries
```

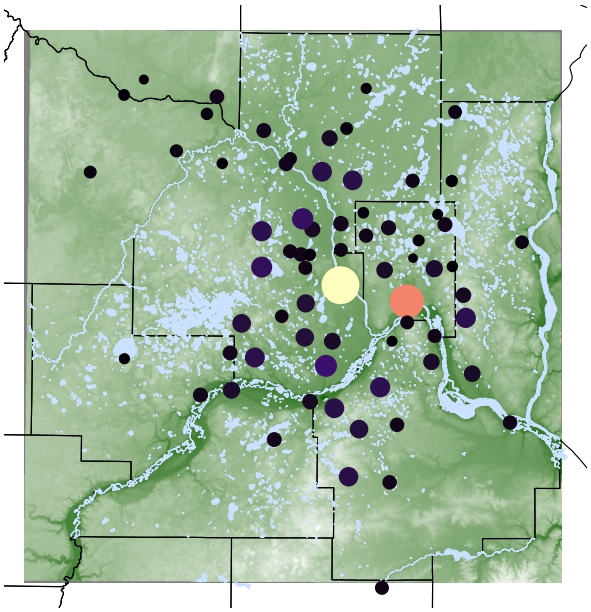
Mosaicing & Projecting

Clipping DEM to bbox

Note: Elevation units are in meters.

```
raster::crs(elevation) <- sf::st_crs(mn_counties)

#Convert to Data Frame for plotting
elev_df <- elevation %>% terra::as.data.frame(xy = TRUE)
names(elev_df) <-c('x','y','value')


ggplot() +
  geom_raster(data = elev_df, aes(x = x,y = y,fill = value)) +
  geom_sf(data = mn_counties, fill = NA, color = "black") + # county boundary layer
  geom_sf(data = mn_water, fill = 'lightsteelblue1',color = 'lightsteelblue1') + # added a ri
  geom_sf(data = mn_cities %>% filter(Population >= 10000), mapping = aes(color = Population
  coord_sf(xlim = Seven_countyarea[c(1,3)],ylim = Seven_countyarea[c(2,4)]) + # crop map to
  scale_color_viridis_c(option = 'magma') + #continuous (gradient) color scale
  scale_fill_gradient(low = 'darkgreen',high = 'white') + #continuous (gradient) fill scale
  labs(title = "Twin Cities with Population >= 10,000") +
  ggthemes::theme_map() + theme(legend.position = "none")  #remove legend
```

Twin Cities with Population >= 10,000



7.3.6 More R Resources

- sf package: https://r-spatial.github.io/sf/
- sf Cheat sheet: https://github.com/rstudio/cheatsheets/blob/main/sf.pdf

7.4 Point Processes (optional)

A **point pattern/process** data set gives the locations of objects/events occurring in a study region. These points could represent trees, animal nests, earthquake epicenters, crimes, cases of influenza, galaxies, etc. We assume that a point's occurrence or non-occurrence at a location is random.

The observed points may have extra information called **marks** attached to them. These marks represent an attribute or characteristic of the point. These could be categorical or continuous.

7.4.1 Poisson Point Processes

The underlying probability model we assume determines the number of events in a small area will be a Poisson model with parameter $\lambda(s)$, the **intensity** in a fixed area. This intensity may be constant (uniform) across locations or vary from location to location (inhomogeneous).

For a homogeneous Poisson process, we model the number of points in any region A, $N(A)$, to be Poisson with

$$E(N(A)) = \lambda \cdot \text{Area}(A)$$

such that λ is constant across space. Given $N(A) = n$, the N events form an iid sample from the uniform distribution on A. Under this model, for any two disjoint regions A and B, the random variables $N(A)$ and $N(B)$ are independent.

If we observe n events in a region A, the estimator $\hat{\lambda} = n/\text{Area(A)}$ is unbiased if the model is correct.

This homogeneous Poisson model is known as **complete spatial randomness (CSR)**. If points deviate from it, we might be able to detect this with a hypothesis test. We'll return to this when we discuss **distances to neighbors**.

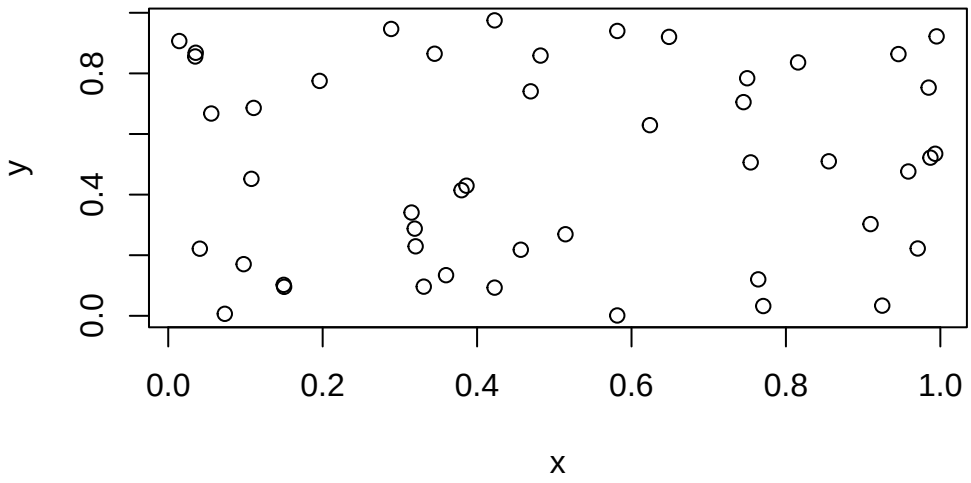
To simulate data from a CSR process in a square [0,1]x[0,1], we could

- Generate the number of events from a Poisson distribution with λ

```
n <- rpois(1, lambda = 50)
```

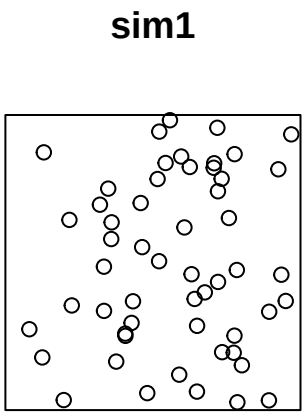
- Fix n and generate locations from the uniform

```
x <- runif(n,0,1)
y <- runif(n,0,1)
plot(x,y)
```



- Alternatively, generate the data with `rpoispp`.

```
sim1 <- rpoispp(lambda = 50)
plot(sim1)
```



If the intensity is not constant across space (inhomogeneous Poisson process), then we define intensity as

$$\lambda(s) = \lim_{|ds| \to 0} \frac{E(N(ds))}{|ds|}$$

where ds is a small area element and $N(A)$ has a Poisson distribution with mean

$$\mu(A) = \int_A \lambda(s) ds$$

When working with point process data, we generally want to estimate and/or model $\lambda(s)$ in a region A and determine if $\lambda(s) = \lambda$ for $s \in A$.

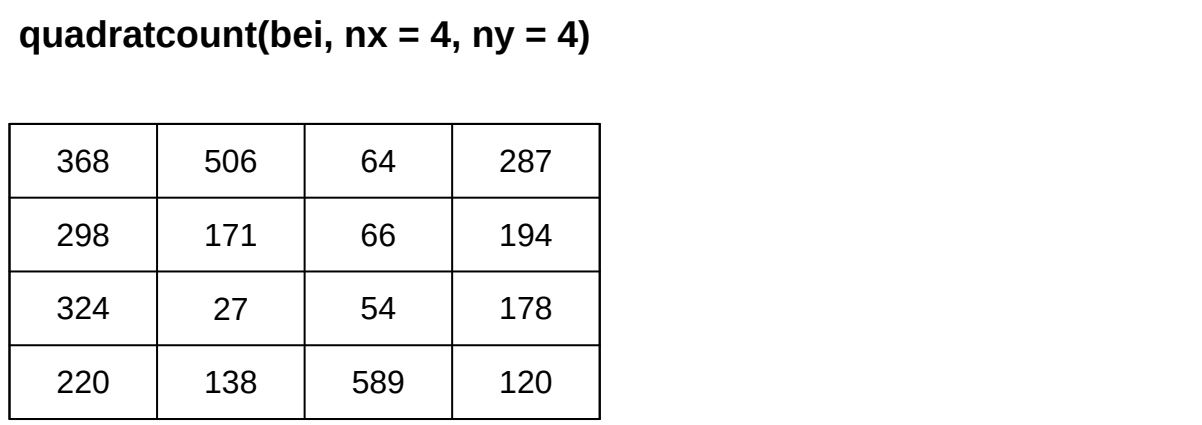
7.4.2 Non-Parametric Intensity Estimation

7.4.2.1 Quadrat Estimation

One way to estimate the intensity $\lambda(s)$ requires dividing the study area into sub-regions (also known as quadrats). Then, the estimated density is computed for each quadrat by dividing the number of points in each quadrat by the quadrat's area. Quadrats can take on many shapes, such as hexagons and triangles or the typically square quadrats.

If the points have uniform/constant intensity (CSR), the quadrat counts should be Poisson random numbers with constant mean. Using a χ^2 goodness-of-fit test statistic, we can test H_0 : CSR,

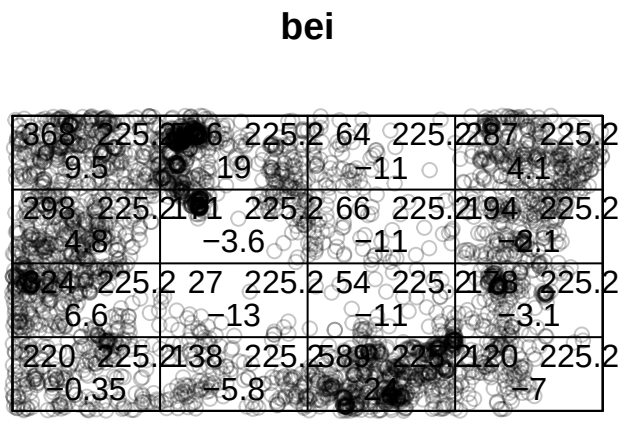
```
plot(quadratcount(bei, nx=4, ny=4))
```




```
(T <- quadrat.test(bei, nx=4, ny=4))
```

```
    Chi-squared test of CSR using quadrat counts

data:  bei
X2 = 1754.6, df = 15, p-value < 2.2e-16
alternative hypothesis: two.sided

Quadrats: 4 by 4 grid of tiles
```

```
plot(bei)
plot(T, add=TRUE)
```



The choice of quadrat numbers and shape can influence the estimated density and must be chosen carefully. If the quadrats are too small, you risk having many quadrats with no points, which may prove uninformative (and can cause issues if you are trying to run a χ^2 test). If very large quadrat sizes are used, you risk missing subtle changes in spatial density.

You should wonder why if the density is not uniform across the space. Is there a covariate (characteristic of the space that could be collected at every point in space) that could help explain the difference in the intensity? For example, perhaps the elevation of an area could be impacting the intensity of trees that thrive in the area. Or there could be a west/east or north/south pattern such that the longitude or latitude impacts the intensity.

If there is a covariate, we can convert the covariate across the continuous spatial field into discretized areas. We can then plot the relationship between the estimated point density within the quadrat and the covariate regions to assess any dependence between variables. If there is a clear relationship, the covariate levels can define new sub-regions within which the density can be computed. This is the idea of **normalizing** data by some underlying covariate.

The quadrat analysis approach has its advantages in that it is easy to compute and interpret; however, it does suffer from the modifiable areal unit problem (MAUP) as the relationship

observed can change depending on the size and shape of the quadrats chosen. Another density-based approach that will be explored next (less susceptible to the MAUP) is the kernel density estimation process.

7.4.2.2 Kernel Density Estimation

The **kernel density** approach is an extension of the quadrat method. Like the quadrat density, the kernel approach computes a localized density for subsets of the study area. Still, unlike its quadrat density counterpart, the sub-regions overlap, providing a moving sub-region window. A kernel defines this moving window. The kernel density approach generates a grid of density values whose cell size is smaller than the kernel window's. Each cell is assigned the density value computed for the kernel window centered on that cell.

A kernel not only defines the shape and size of the window, but it can also weight the points following a well-defined kernel function. The simplest function is a basic kernel where each point in the window is assigned equal weight (uniform kernel). Some popular kernel functions assign weights to points inversely proportional to their distances to the kernel window center. A few such kernel functions follow a Gaussian or cubic distribution function. These functions tend to produce a smoother density map.

$$\hat{\lambda}_{\mathbf{H}}(\mathbf{s}) = \frac{1}{n}\sum_{i=1}^{n} K_{\mathbf{H}}(\mathbf{s} - \mathbf{s}_i)$$

where $\mathbf{s}_1, ..., \mathbf{s}_n$ are the locations of the observed points (typically specified by longitude and latitude),

$$K_{\mathbf{H}}(\mathbf{x}) = |\mathbf{H}|^{-1/2} K(\mathbf{H}^{-1/2}\mathbf{x})$$

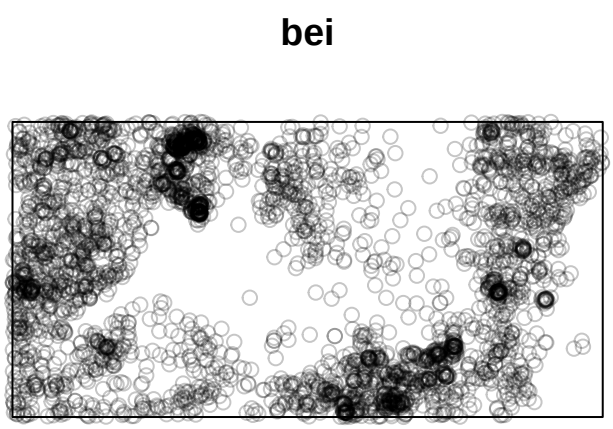
$\mathbf{H}$ is the bandwidth matrix, and K is the kernel function, typically assumed multivariate Gaussian. Still, it could be another symmetric function that decays to zero as it moves away from the center.

To incorporate covariates into kernel density estimation, we can estimate a **normalized density** as a function of a covariate; we notate this as $\rho(Z(s))$ where $Z(s)$ is a continuous spatial covariate. There are three ways to estimate this: ratio, re-weight, or transform. We will not delve into the differences between these methods but note that there is more than one way to estimate ρ. This is a **non-parametric** way to estimate the intensity.

7.4.2.3 Data Example and R Code

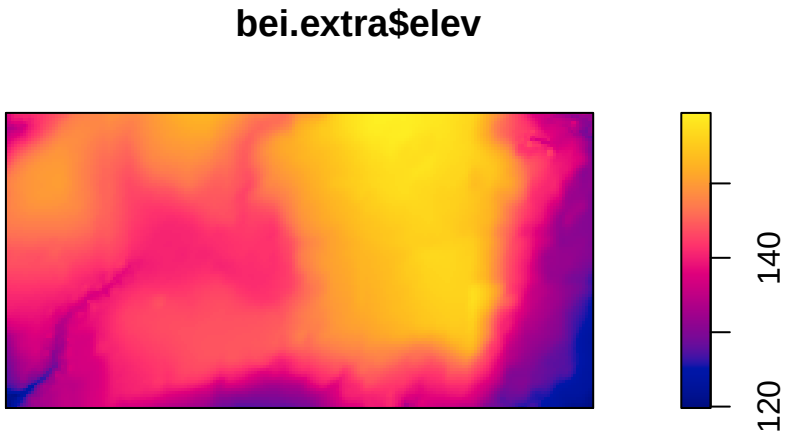
Below is a point pattern giving the locations of 3605 trees in a tropical rain forest.

```
library(spatstat)
plot(bei)
```

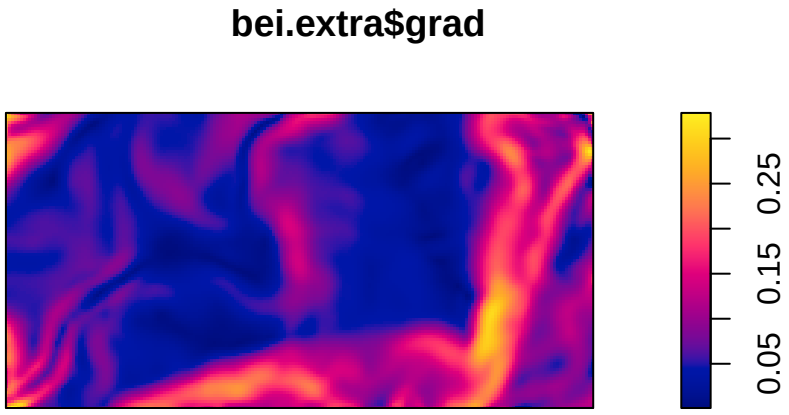


Below are two pixel images of covariates, the elevation and slope (gradient) of the elevation in the study region.

```
plot(bei.extra$elev)
```

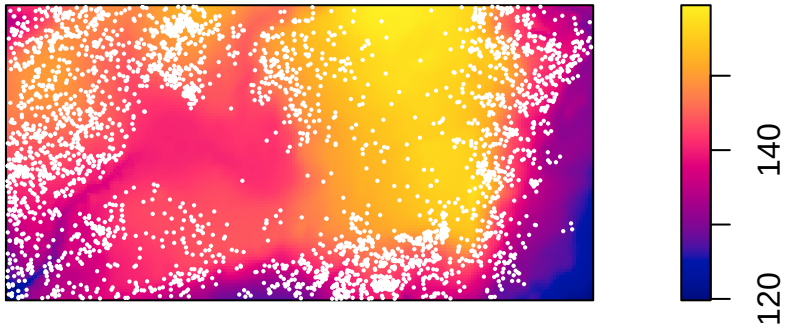



```
plot(bei.extra$grad)
```

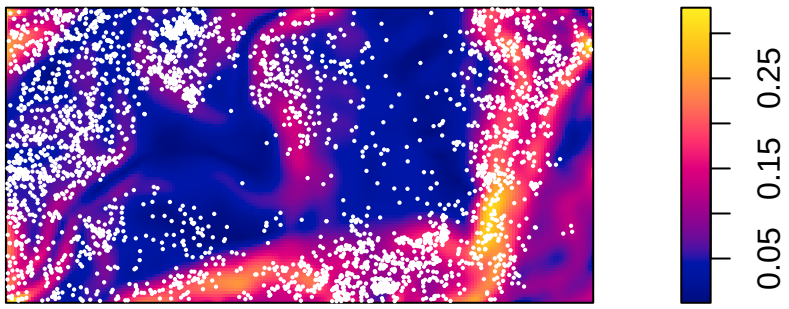


Let's plot the points on top of the covariates to see if we can see any relationships.

```
plot(bei.extra$elev, main = "")
plot(bei, add = TRUE, cex = 0.3, pch = 16, cols = "white")
```

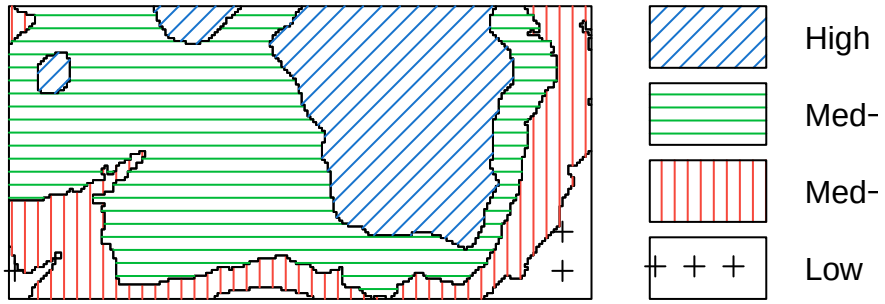



```
plot(bei.extra$grad, main = "")
plot(bei, add = TRUE, cex = 0.3, pch = 16, cols = "white")
```

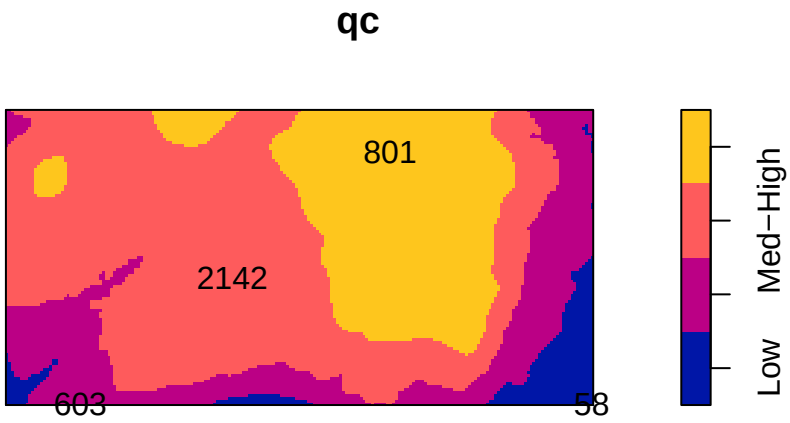


We could convert the image above of the elevation to a tesselation, count the number of points in each region using quadratcount, and plot the quadrat counts.

```
elev <- bei.extra$elev
Z <- cut(elev, 4, labels=c("Low", "Med-Low", "Med-High", "High"))
textureplot(Z, main = "")
```




```
Y <- tess(image = Z)
qc <- quadratcount(bei, tess = Y)
plot(qc)
```

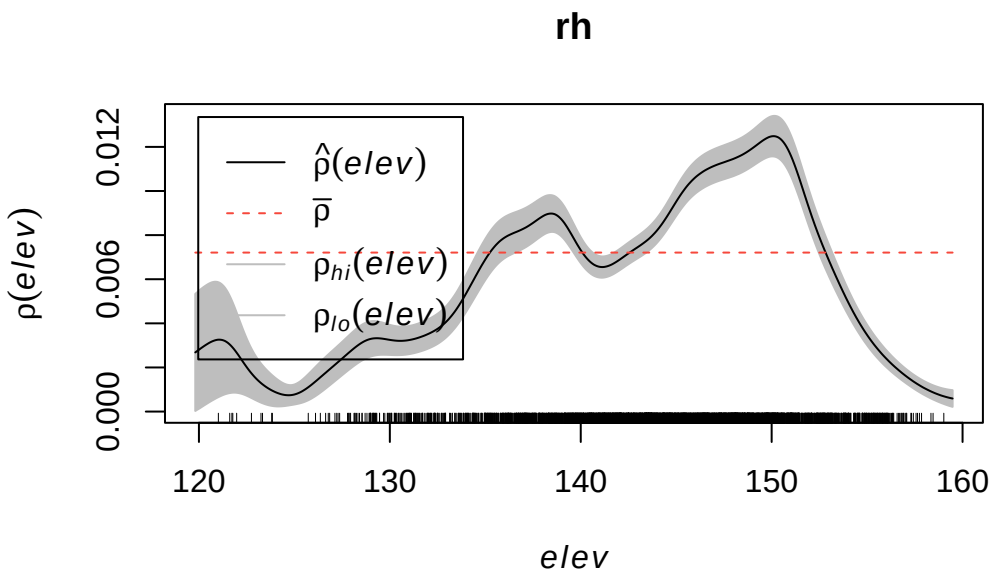



```
intensity(qc)
```

```
tile
        Low     Med-Low    Med-High        High
0.002259007 0.006372523 0.008562862 0.005843516
```

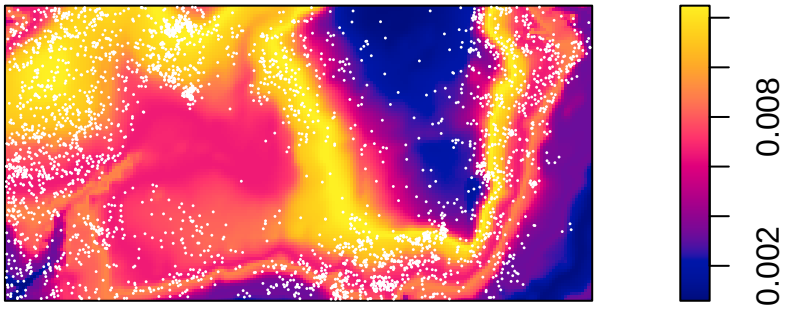
Using a non-parametric kernel density estimation, we can estimate the intensity as a function of the elevation.

```
rh <- rhohat(bei, elev)
plot(rh)
```



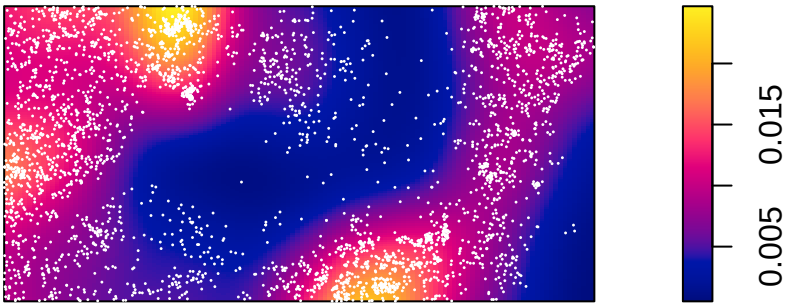
Then predict the intensity based on this function.

```
prh <- predict(rh)
plot(prh, main = "")
plot(bei, add = TRUE, cols = "white", cex = .2, pch = 16)
```

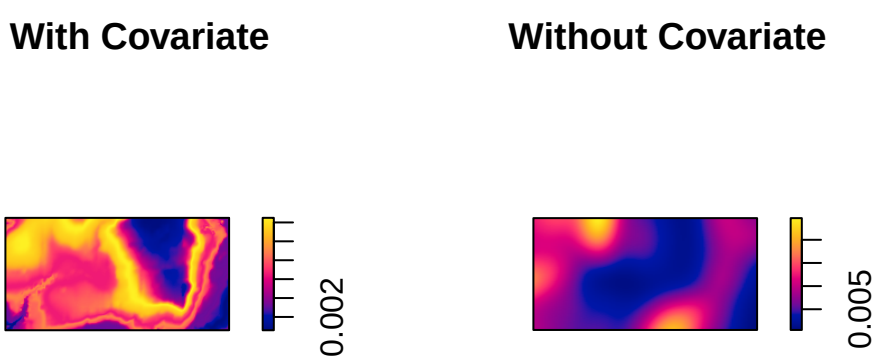


Contrast this to a simple kernel density estimate without a covariate:

```
dbei <- density(bei)
plot(dbei, main = "")
plot(bei, add = TRUE, cols = "white", cex = .2, pch = 16)
```




```
par(mfrow=c(1,2))
plot(prh, main = "With Covariate")
plot(dbei, main = "Without Covariate")
```



7.4.3 Parametric Intensity Estimation

Beyond kernel density estimates, we may want to model this intensity with a parametric model. We'll use a log-link function between the linear model and our intensity.

We assume $log(\lambda(s)) = \beta_0$ for a uniform Poisson process.

```
ppm(bei ~ 1)
```

```
Stationary Poisson process
Fitted to point pattern dataset 'bei'
Intensity: 0.007208
             Estimate       S.E.    CI95.lo    CI95.hi Ztest      Zval
log(lambda) -4.932564 0.01665742 -4.965212 -4.899916   *** -296.1182
```

We may think that the x coordinate linearly impacts the intensity, $log(\lambda(s)) = \beta_0 + \beta_1 x$.

```
ppm(bei ~ x)
```

```
Nonstationary Poisson process
Fitted to point pattern dataset 'bei'

Log intensity:  ~x

Fitted trend coefficients:
  (Intercept)             x
-4.5577338857 -0.0008031298

                 Estimate         S.E.       CI95.lo      CI95.hi Ztest
(Intercept) -4.5577338857 3.040310e-02 -4.6173228598 -4.498144912   ***
x           -0.0008031298 5.863308e-05 -0.0009180485 -0.000688211   ***
                  Zval
(Intercept) -149.91019
x            -13.69755
```

We may think that the x and y coordinates linearly impact the intensity, $log(\lambda(s)) = \beta_0 + \beta_1 x + \beta_2 y$.

```
ppm(bei ~ x + y)
```

```
Nonstationary Poisson process
Fitted to point pattern dataset 'bei'

Log intensity:  ~x + y

Fitted trend coefficients:
  (Intercept)             x             y
-4.7245290274 -0.0008031288  0.0006496090

                 Estimate         S.E.       CI95.lo       CI95.hi Ztest
(Intercept) -4.7245290274 4.305915e-02 -4.8089234185 -4.6401346364   ***
x           -0.0008031288 5.863311e-05 -0.0009180476 -0.0006882100   ***
y            0.0006496090 1.157132e-04  0.0004228153  0.0008764027   ***
                  Zval
(Intercept) -109.721827
x            -13.697530
y              5.613957
```

We could use a variety of models based solely on the x and y coordinates of their location:

```
(mod <- ppm(bei ~ polynom(x,y,2))) #quadratic relationship
```

```
Nonstationary Poisson process
Fitted to point pattern dataset 'bei'

Log intensity:  ~x + y + I(x^2) + I(x * y) + I(y^2)

Fitted trend coefficients:
  (Intercept)             x             y        I(x^2)      I(x * y)
-4.275762e+00 -1.609187e-03 -4.895166e-03  1.625968e-06 -2.836387e-06
       I(y^2)
 1.331331e-05

                 Estimate         S.E.       CI95.lo       CI95.hi Ztest
(Intercept) -4.275762e+00 7.811138e-02 -4.428857e+00 -4.122666e+00   ***
x           -1.609187e-03 2.440907e-04 -2.087596e-03 -1.130778e-03   ***
y           -4.895166e-03 4.838993e-04 -5.843591e-03 -3.946741e-03   ***
I(x^2)       1.625968e-06 2.197200e-07  1.195325e-06  2.056611e-06   ***
I(x * y)    -2.836387e-06 3.511163e-07 -3.524562e-06 -2.148212e-06   ***
I(y^2)       1.331331e-05 8.487506e-07  1.164979e-05  1.497683e-05   ***
                  Zval
(Intercept) -54.739290
x            -6.592577
y           -10.116084
I(x^2)        7.400185
I(x * y)     -8.078197
I(y^2)       15.685769
```

```
(mod1 <- ppm(bei ~ I( x > 50))) #threshold
```

```
Nonstationary Poisson process
Fitted to point pattern dataset 'bei'

Log intensity:  ~I(x > 50)

Fitted trend coefficients:
  (Intercept) I(x > 50)TRUE
   -4.3790111    -0.5960561
```

```
                Estimate       S.E.    CI95.lo    CI95.hi Ztest      Zval
(Intercept)   -4.3790111 0.05471757 -4.4862556 -4.2717666   *** -80.02935
I(x > 50)TRUE -0.5960561 0.05744408 -0.7086444 -0.4834677   *** -10.37628
```

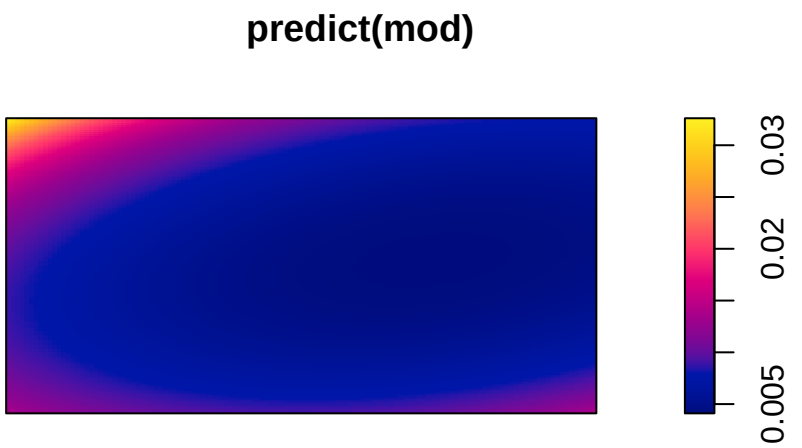
```
require(splines)
```

```
Loading required package: splines
```

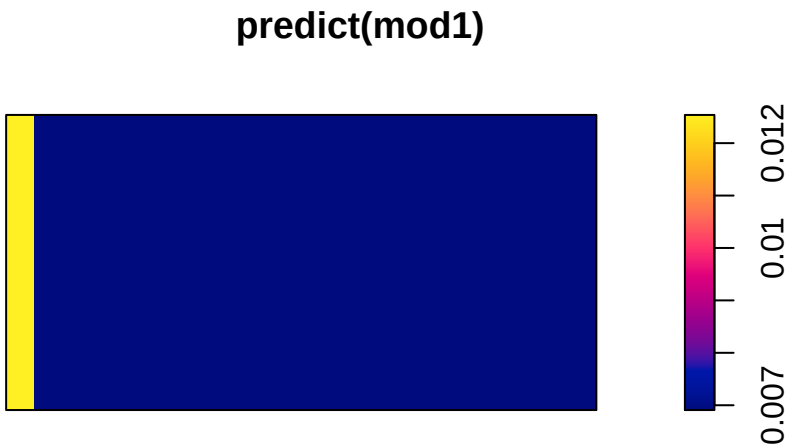
```
(mod2 <- ppm(bei ~ bs(x) + bs(y))) #B-spline
```

```
Nonstationary Poisson process
Fitted to point pattern dataset 'bei'

Log intensity:  ~bs(x) + bs(y)

Fitted trend coefficients:
(Intercept)      bs(x)1      bs(x)2      bs(x)3      bs(y)1      bs(y)2
-3.49662617 -2.04025568 -0.23163941 -1.36342361 -1.79118390 -0.57815937
     bs(y)3
-0.05630791

               Estimate       S.E.    CI95.lo     CI95.hi Ztest        Zval
(Intercept) -3.49662617 0.07468060 -3.6429974 -3.35025488   *** -46.8210784
bs(x)1      -2.04025568 0.17102665 -2.3754617 -1.70504961   *** -11.9294608
bs(x)2      -0.23163941 0.12980360 -0.4860498  0.02277098        -1.7845376
bs(x)3      -1.36342361 0.10227353 -1.5638760 -1.16297118   *** -13.3311487
bs(y)1      -1.79118390 0.18113345 -2.1461989 -1.43616886   ***  -9.8887526
bs(y)2      -0.57815937 0.11815893 -0.8097466 -0.34657213   ***  -4.8930655
bs(y)3      -0.05630791 0.08835629 -0.2294831  0.11686724        -0.6372824
```

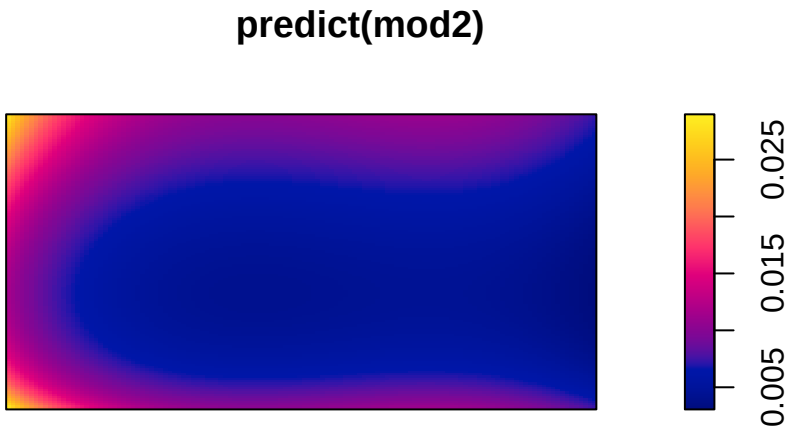
```
plot(predict(mod))
```




```
plot(predict(mod1))
```




```
plot(predict(mod2))
```



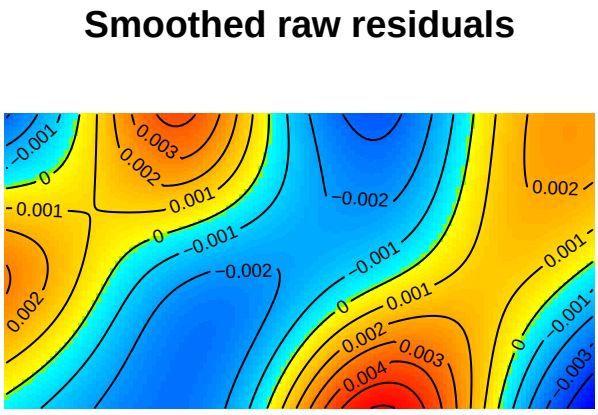
We can compare models using a likelihood ratio test.

```
hom <- ppm(bei ~ 1)

anova(hom, mod, test = 'Chi')
```

```
Analysis of Deviance Table

Model 1: ~1      Poisson
Model 2: ~x + y + I(x^2) + I(x * y) + I(y^2)     Poisson
  Npar Df Deviance  Pr(>Chi)
1    1
2    6  5   604.15 < 2.2e-16 ***
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

Also, considering the residuals, we should see if there is a pattern where our model has errors in predicting the intensity.

```
diagnose.ppm(mod, which="smooth")
```




```
Model diagnostics (raw residuals)
Diagnostics available:
    smoothed residual field
range of smoothed field =  [-0.004988, 0.006086]
```

As we saw with the kernel density estimation, the elevation plays a role in the intensity of trees. Let's add that to our model,

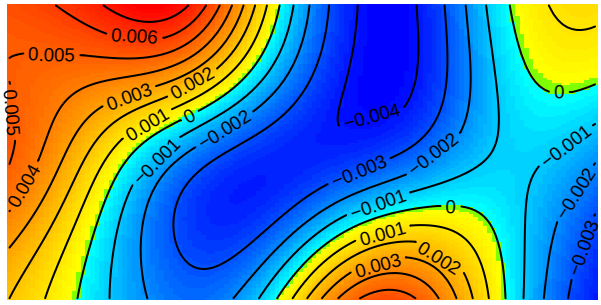
$$\log(\lambda(s)) = \beta_0 + \beta_1 Elevation(s)$$

```
(mod2 <- ppm(bei ~ elev))
```

```
Nonstationary Poisson process
Fitted to point pattern dataset 'bei'

Log intensity:  ~elev

Fitted trend coefficients:
(Intercept)        elev
-5.63919077  0.00488995

               Estimate        S.E.       CI95.lo      CI95.hi Ztest       Zval
(Intercept) -5.63919077 0.304565582 -6.2361283457 -5.042253203   *** -18.515522
elev         0.00488995 0.002102236  0.0007696438  0.009010256     *   2.326071
```

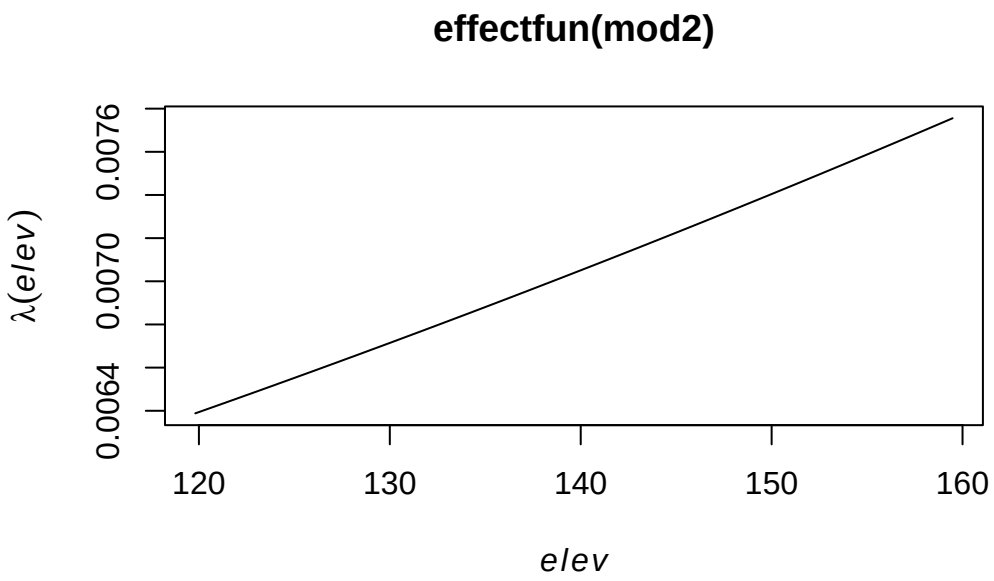
```
diagnose.ppm(mod2, which="smooth")
```

Smoothed raw residuals




```
Model diagnostics (raw residuals)
Diagnostics available:
    smoothed residual field
range of smoothed field =  [-0.004413, 0.007798]
```

```
plot(effectfun(mod2))
```



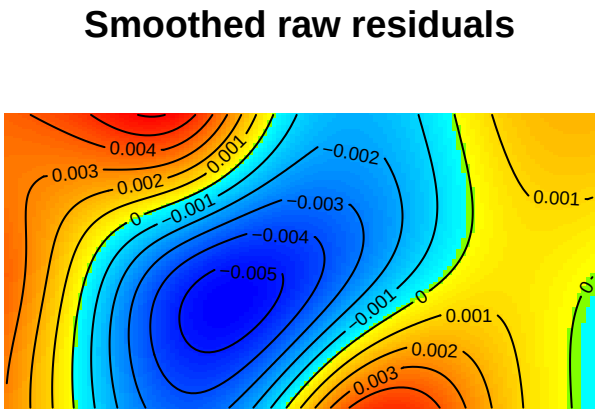
But the relationship may not be linear, so let's try a quadratic relationship with elevation,

$$\log(\lambda(s)) = \beta_0 + \beta_1 Elevation(s) + \beta_2 Elevation^2(s)$$

```
(mod2 <- ppm(bei ~ polynom(elev,2)))
```

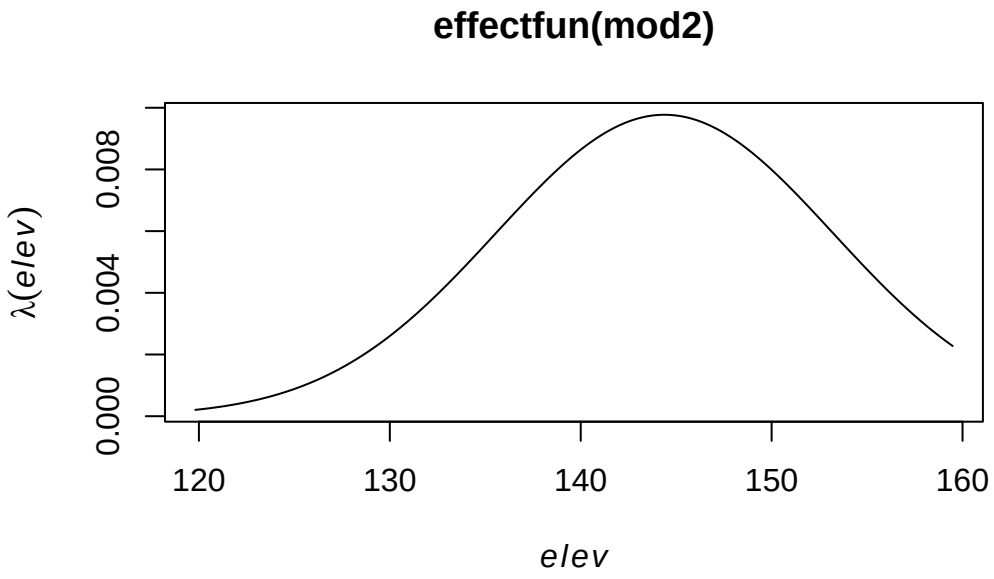
```
Nonstationary Poisson process
Fitted to point pattern dataset 'bei'

Log intensity:  ~elev + I(elev^2)

Fitted trend coefficients:
  (Intercept)          elev     I(elev^2)
-1.379706e+02  1.847007e+00 -6.396003e-03

                 Estimate         S.E.       CI95.lo      CI95.hi Ztest
(Intercept) -1.379706e+02 6.7047209548 -1.511116e+02 -124.8295945   ***
elev         1.847007e+00 0.0927883205  1.665145e+00    2.0288686   ***
I(elev^2)   -6.396003e-03 0.0003207726 -7.024705e-03   -0.0057673   ***
                 Zval
(Intercept) -20.57813
elev         19.90560
I(elev^2)   -19.93937
```

```
diagnose.ppm(mod2, which="smooth")
```




```
Model diagnostics (raw residuals)
Diagnostics available:
    smoothed residual field
range of smoothed field =  [-0.005641, 0.006091]
```

```
plot(effectfun(mod2))
```



7.4.4 Detecting Interaction Effects

Classical tests about the interaction effects between points are based on derived distances.

- Pairwise distances: $d_{ij} = ||s_i - s_j||$ (`pairdist`)

```
pairdist(bei)[1:10,1:10] #distance matrix for each pair of points
```

```
          [,1]       [,2]       [,3]       [,4]       [,5]       [,6]
 [1,]    0.0000 1025.97671 1008.73600 1012.76608 969.054054 964.316302
 [2,] 1025.9767    0.00000   19.03786   13.26084  56.922755  61.762205
 [3,] 1008.7360   19.03786    0.00000   10.01249  40.429692  45.845065
 [4,] 1012.7661   13.26084   10.01249    0.00000  43.729281  48.509381
 [5,]  969.0541   56.92275   40.42969   43.72928   0.000000   5.920304
 [6,]  964.3163   61.76221   45.84507   48.50938   5.920304   0.000000
 [7,]  973.0970   54.02814   40.31724   40.88239  11.601724  11.412712
 [8,]  959.1238   71.28324   59.21976   58.54955  26.109385  21.202123
 [9,]  961.3980   73.63457   63.88153   61.61980  35.327751  31.005806
[10,]  928.8108  158.87769  154.57400  149.57837 128.720472 123.704850
          [,7]      [,8]      [,9]     [,10]
 [1,] 973.09701 959.12378 961.39801 928.81077
 [2,]  54.02814  71.28324  73.63457 158.87769
 [3,]  40.31724  59.21976  63.88153 154.57400
 [4,]  40.88239  58.54955  61.61980 149.57837
 [5,]  11.60172  26.10939  35.32775 128.72047
 [6,]  11.41271  21.20212  31.00581 123.70485
 [7,]   0.00000  19.34580  26.45695 120.34168
 [8,]  19.34580   0.00000  10.50952 102.61822
 [9,]  26.45695  10.50952   0.00000  93.90575
[10,] 120.34168 102.61822  93.90575   0.00000
```

- Nearest neighbor distances: $t_i = \min_{j \neq i} d_{ij}$ (`nndist`)

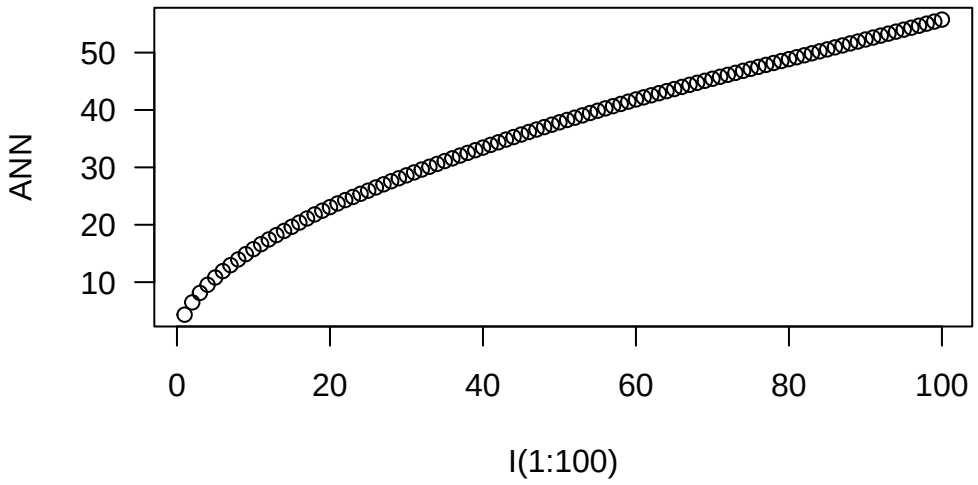
```
mean(nndist(bei)) #average first nearest neighbor distance
```

```
[1] 4.329677
```

```
mean(nndist(bei,k=2)) #average second nearest neighbor distance
```

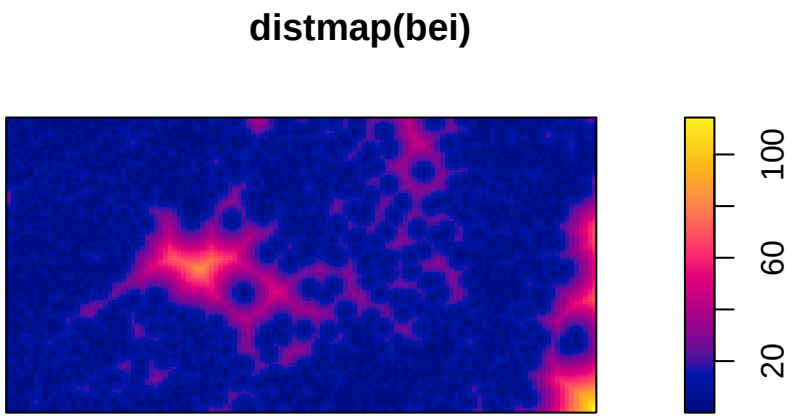
```
[1] 6.473149
```

```
ANN <- apply(nndist(bei, k=1:100), 2, mean) #Mean for 1,...,100 nearest neighbors
plot(ANN ~ I(1:100), type="b", main=NULL, las=1)
```



- Empty space distances: $d(u) = \min_i ||u - s_i||$, the distance from a fixed reference location u in the window to the nearest data point (`distmap`)

```
plot(distmap(bei)) #map of the distances to the nearest observed point
```



7.4.4.1 F function

Consider the empty-space distance.

- Fix the observation window A. The distribution of $d(u)$, the minimum distance to an observed point, depends on A, which is hard to derive in closed form.

Consider the CDF,

$$F_u(r) = P(d(u) \leq r)$$

$$= P(\text{ at least one point within radius } r \text{ of } u)$$
$$= 1 - P(\text{ no points within radius } r \text{ of } u)$$

For a homogeneous Poisson process on R^2,

$$F(r) = 1 - e^{-\lambda \pi r^2}$$

- Consider the process defined on R^2 and we only observe it on A. The observed distances are biased relative to the "true" distances, which results in "edge effects".
- If you define a set of locations $u_1, ..., u_m$ for estimating $F(r)$ under an assumption of statitionary, the estimator

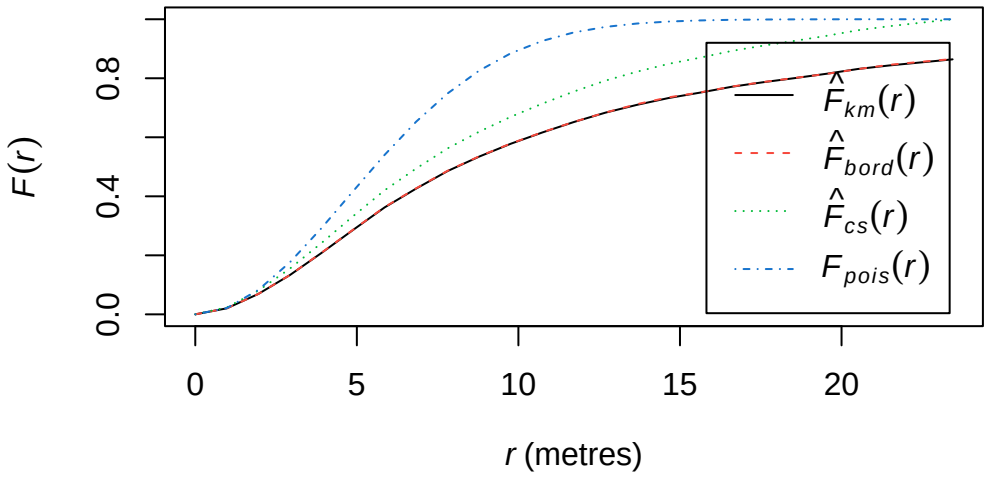
$$\hat{F}(r) = \frac{1}{m} \sum_{j=1}^{m} I(d(u_j) \leq r)$$

is biased due to the edge effects.

- There are a variety of corrections.
  - One example is to not consider u's within distance r of the boundary (border correction).
  - Another example is considering it a censored data problem and using a spatial variant of the Kaplan-Meier correction.

Compare the $\hat{F}(r)$ to the $F(r)$ under a homogenous Poisson process.

```
plot(Fest(bei)) #km is kaplan meier correction, bord is the border correction (reduced sample
```



In this plot, we observe fewer short distances than expected if it were a completely random Poisson point process (F_{pois}) and thus many longer distances to observed points. Thus, more spots don't have many points (larger distances to observed points), meaning points are more clustered than expected from a uniform distribution across the space.

7.4.4.2 G function

Working instead with the nearest-neighbor distances from points themselves ($t_i = \min_{j \neq i} d_{ij}$), we define

$$G(r) = P(t_i \leq r)$$

for an arbitrary observed point s_i. We can estimate it using our observed $ t_i$'s.

$$\hat{G}(r) = \frac{1}{n} \sum_{i=1}^{n} I(t_i \leq r)$$

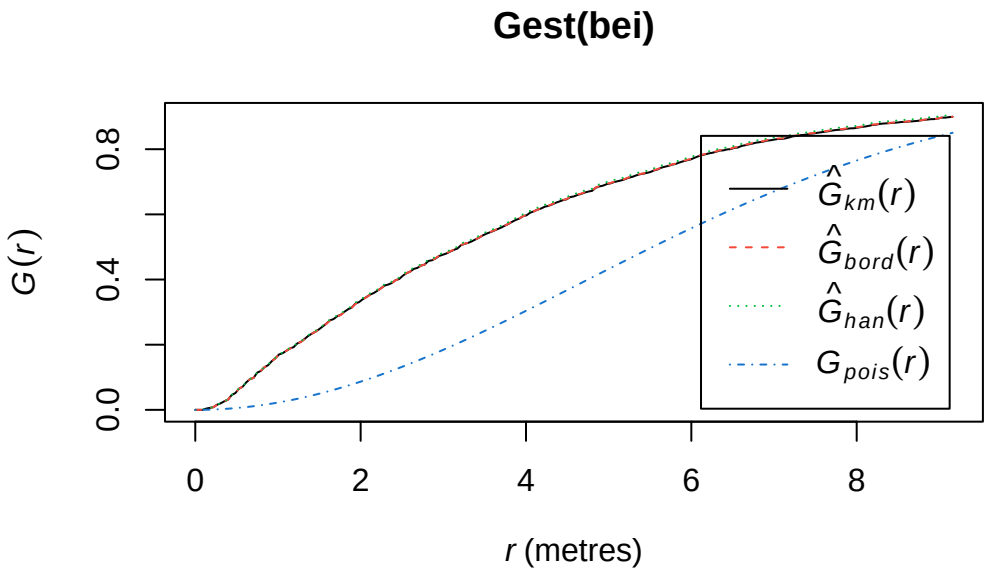
As before, we may or may not make an edge correction.

For a homogeneous Poisson process on R^2,

$$G(r) = 1 - e^{-\lambda \pi r^2}$$

Compare the $\hat{G}(r)$ to the $G(r)$ under a homogenous Poisson process.

```
plot(Gest(bei))
```



Here we observe that we see more short nearest-neighbor distances than expected if it were a completely random Poisson point process. Thus, the points are more clustered than expected from a uniform distribution across the space.

7.4.4.3 K function

Another approach is to consider **Ripley's K function**, which summarizes the distance between points. It divides the mean of the total number of points at different distance lags from each point by the area event density. In other words, it is concerned with the number of events falling within a circle of radius r of an event.

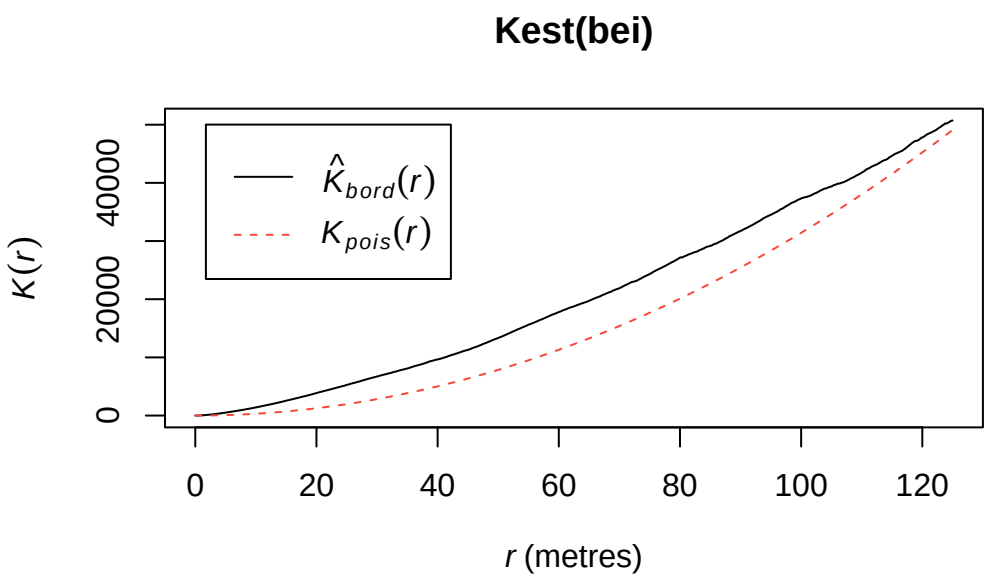
$$K(r) = \frac{1}{\lambda} E(\text{ points within distance r of an arbitrary point})$$

Under CSR (uniform intensity), we'd expect $K(r) = \pi r^2$ because that is the area of the circle. Points that cluster more than you'd expect corresponds to $K(r) > \pi r^2$ and spatial repulsion corresponds to $K(r) < \pi r^2$.

Below, we have an estimate of the K function (black solid) along with the predicted K function if H_0 were true (red dashed). For this set of trees, the points are more clustered than you'd expect for a homogenous Poisson process.

```
plot(Kest(bei))
```

```
number of data points exceeds 3000 - computing border correction estimate only
```



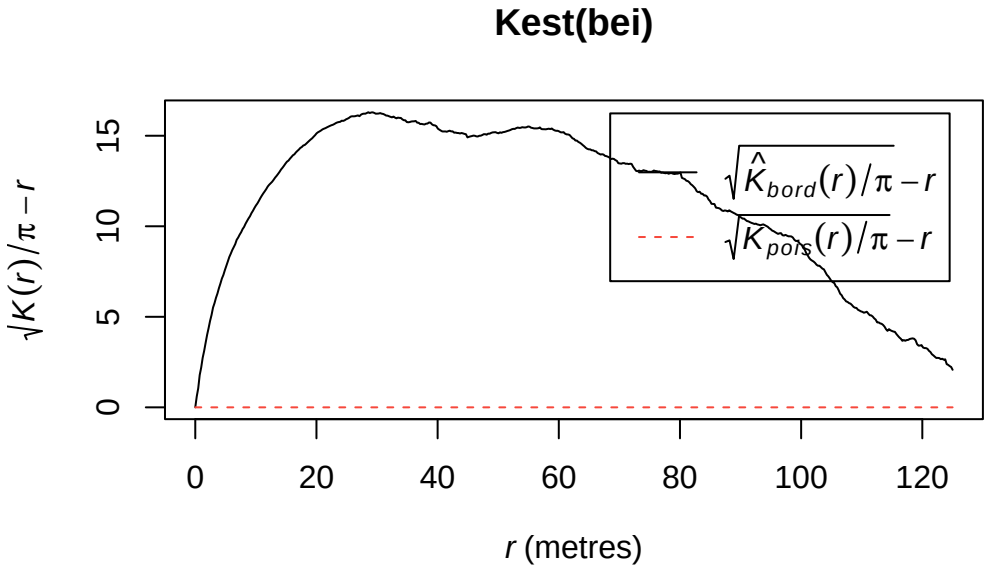
7.4.4.4 L function

It is hard to see differences between the estimated K and expected functions. An alternative is to transform the values so that the expected values are horizontal. The transformation is calculated as follows:

$$L(r) = \sqrt{K(r)/\pi} - d$$

```
plot(Kest(bei),sqrt(./pi) - r ~ r)
```

```
number of data points exceeds 3000 - computing border correction estimate only
```



7.4.4.5 Pair Correlation Function

The second-order intensity function is

$$\lambda_2(s_1, s_2) = \lim_{|ds_1|,|ds_2|\to 0} \frac{E[N(ds_1)N(ds_2)]}{|ds_1||ds_2|}$$

where $s_1 \neq s_2$. If $\lambda(s)$ is constant and

$$\lambda_2(s_1, s_2) = \lambda_2(s_1 - s_2) = \lambda_2(s_2 - s_1)$$

then we call N second-order stationary.

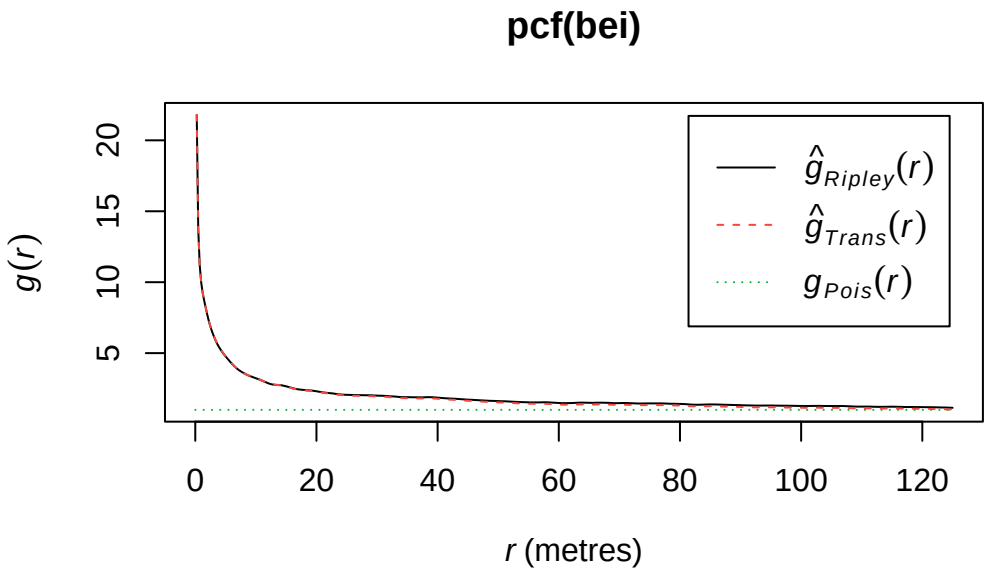
We can define g as the **pair correlation function** (PCF),

$$g(s_1, s_2) = \frac{\lambda_2(s_1, s_2)}{\lambda(s_1)\lambda(s_2)}$$

In R, if we assume stationarity and isotropy (direction of distance is not important), we can estimate $g(r)$ where $r = ||s_1 - s_2||$. If a process has an isotropic pair correlation function, then our K function can be defined as

$$K(r) = 2\pi \int_0^r ug(u)du, r > 0$$

```
plot(pcf(bei))
```



If $g(r) = 1$, then we have a CSR process. If $g(r) > 1$, it implies a cluster process and if $g(r) < 1$, it implies an inhibition process. In particular, if $g(r) = 0$ for $r_i < r$, it implies a hard core process (no point pairs within this distance).

7.4.4.6 Envelopes

We want to test H_0: CSR (constant intensity) to understand whether points are closer or further away from each other than we'd expect with a Poisson process.

We can compare the estimated function to the theoretical for any of these 3 (4) functions, but that doesn't tell us how far our estimate might be from the truth under random variability. So we can simulate under our null hypothesis (CSR), say B times, and for each time, we estimate the function. Then we create a single global test:

$$D_i = \max_r |\hat{H}_i(r) - H(r)|, \quad i = 1, ..., B$$

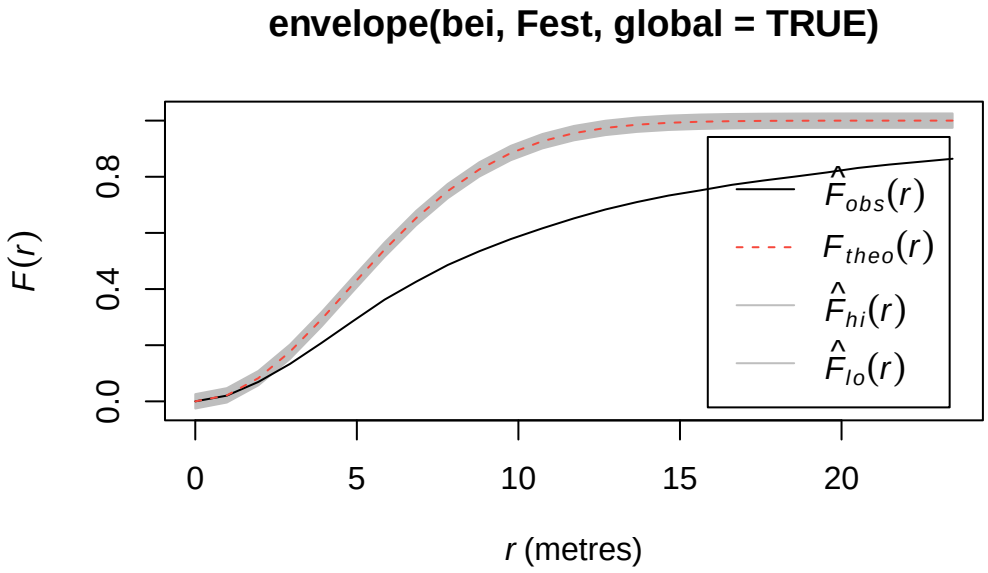
where $H(r)$ is the theoretical value under CSR. Then we define D_{crit} as the kth largest among the D_i. The "global" envelopes are then.

$$L(r) = H(r) - D_{crit}$$
$$U(r) = H(r) + D_{crit}$$

Then the test that rejects $\hat{H}_1$ (estimate from our observations) ever wanders outside $(L(r), U(r))$ has size $\alpha = 1 - k/B$.

```
#See Example Code Below: Runs Slowly
plot(envelope(bei, Fest, global=TRUE))
```

```
Generating 99 simulations of CSR  ...
1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15, 16, 17, 18, 19, 20,
21, 22, 23, 24, 25, 26, 27, 28, 29, 30, 31, 32, 33, 34, 35, 36, 37, 38, 39, 40,
41, 42, 43, 44, 45, 46, 47, 48, 49, 50, 51, 52, 53, 54, 55, 56, 57, 58, 59, 60,
61, 62, 63, 64, 65, 66, 67, 68, 69, 70, 71, 72, 73, 74, 75, 76, 77, 78, 79, 80,
81, 82, 83, 84, 85, 86, 87, 88, 89, 90, 91, 92, 93, 94, 95, 96, 97, 98,
99.

Done.
```




```
#plot(envelope(bei, Gest, global=TRUE))
#plot(envelope(bei, Kest, global=TRUE))
#plot(envelope(bei, Lest, global=TRUE))
#plot(envelope(bei, pcf, global=TRUE))
```

7.4.5 Cluster Poisson Processes

A Poisson cluster process is defined by

1. Parent events form Poisson process with intensity λ.
2. Each parent produces a random number M of children, iid for each parent according to a discrete distribution p_m.
3. The positions of the children relative to parents are iid according to a bivariate pdf.

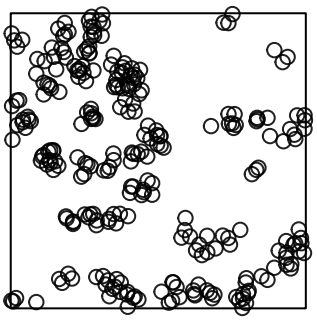
By convention, the point pattern consists of only the children, not the parents.

7.4.5.1 Matern Process

- Homogeneous Poisson parent process with intensity λ.
- Poisson distributed number of children, mean $= \mu$
- Children uniformly distributed on disc of radius, r, centered at the parent location

Let's simulate some data from this process.

```
win <- owin(c(0, 100), c(0, 100))
clust1 <- rMatClust(50/10000, r = 4, mu = 5, win = win)
plot(clust1)
```



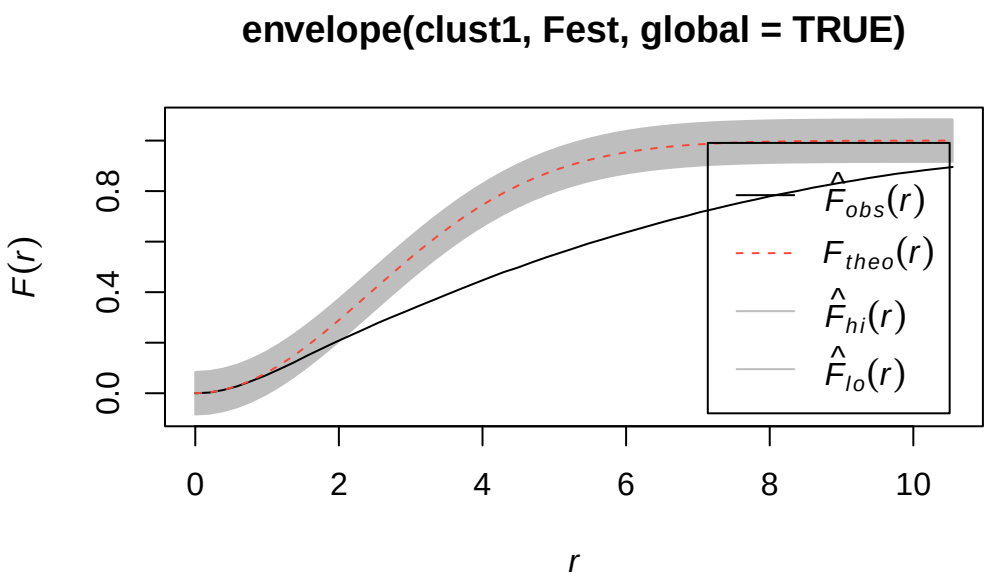

```
quadrat.test(clust1)
```

```
    Chi-squared test of CSR using quadrat counts

data:  clust1
X2 = 106.86, df = 24, p-value = 3.969e-12
alternative hypothesis: two.sided
```

```
Quadrats: 5 by 5 grid of tiles
```

```
plot(envelope(clust1, Fest, global=TRUE))
```

```
Generating 99 simulations of CSR  ...
1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15, 16, 17, 18, 19, 20,
21, 22, 23, 24, 25, 26, 27, 28, 29, 30, 31, 32, 33, 34, 35, 36, 37, 38, 39, 40,
41, 42, 43, 44, 45, 46, 47, 48, 49, 50, 51, 52, 53, 54, 55, 56, 57, 58, 59, 60,
61, 62, 63, 64, 65, 66, 67, 68, 69, 70, 71, 72, 73, 74, 75, 76, 77, 78, 79, 80,
81, 82, 83, 84, 85, 86, 87, 88, 89, 90, 91, 92, 93, 94, 95, 96, 97, 98,
99.

Done.
```

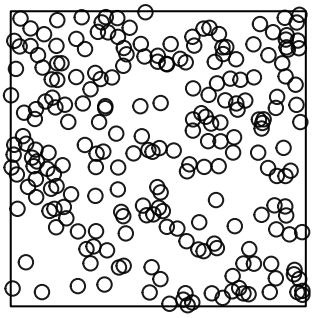



```
#plot(envelope(clust1, Gest, global=TRUE))
#plot(envelope(clust1, Kest, global=TRUE))
#plot(envelope(clust1, Lest, global=TRUE))
```

If we increase the radius of space for the children, we won't notice that it is clustered.

```
clust2 <- rMatClust(50/10000, r = 40, mu = 5, win = win)
plot(clust2)
```

clust2

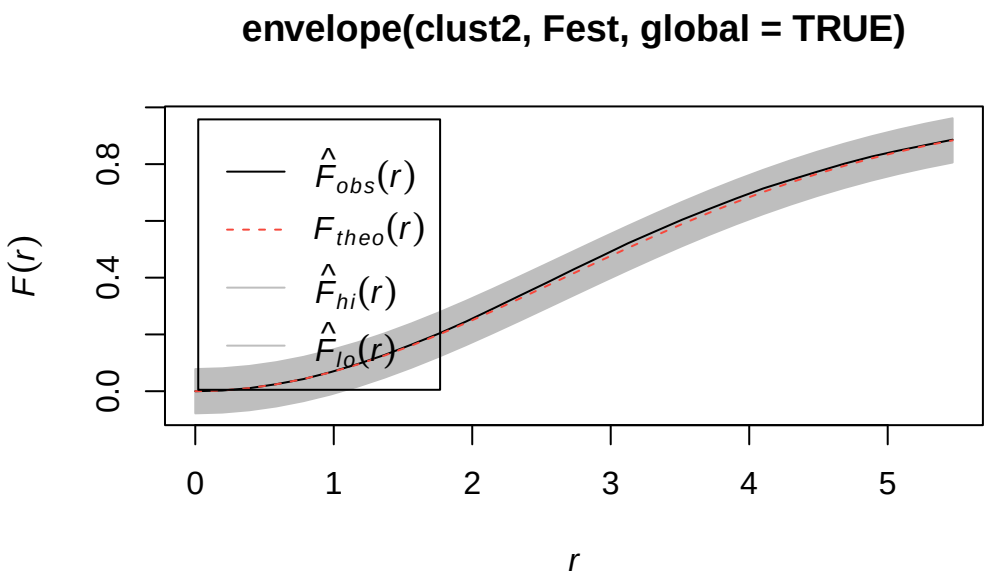


```
quadrat.test(clust2)
```

```
    Chi-squared test of CSR using quadrat counts

data:  clust2
X2 = 33.555, df = 24, p-value = 0.1858
alternative hypothesis: two.sided

Quadrats: 5 by 5 grid of tiles
```

```
plot(envelope(clust2, Fest, global=TRUE))
```

```
Generating 99 simulations of CSR  ...
1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15, 16, 17, 18, 19, 20,
21, 22, 23, 24, 25, 26, 27, 28, 29, 30, 31, 32, 33, 34, 35, 36, 37, 38, 39, 40,
41, 42, 43, 44, 45, 46, 47, 48, 49, 50, 51, 52, 53, 54, 55, 56, 57, 58, 59, 60,
61, 62, 63, 64, 65, 66, 67, 68, 69, 70, 71, 72, 73, 74, 75, 76, 77, 78, 79, 80,
81, 82, 83, 84, 85, 86, 87, 88, 89, 90, 91, 92, 93, 94, 95, 96, 97, 98,
99.

Done.
```



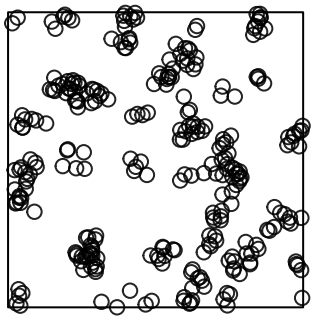

```
#plot(envelope(clust2, Gest, global=TRUE))
#plot(envelope(clust2, Kest, global=TRUE))
#plot(envelope(clust2, Lest, global=TRUE))
```

7.4.5.2 Thomas Process

- Homogeneous Poisson parent process
- Poisson distributed number of children
- Locations of children according to an isotropic bivariate normal distribution with variance σ^2

```
clust3 <- rThomas(50/10000, scale = 2, mu = 5, win = win)
plot(clust3)
```

clust3

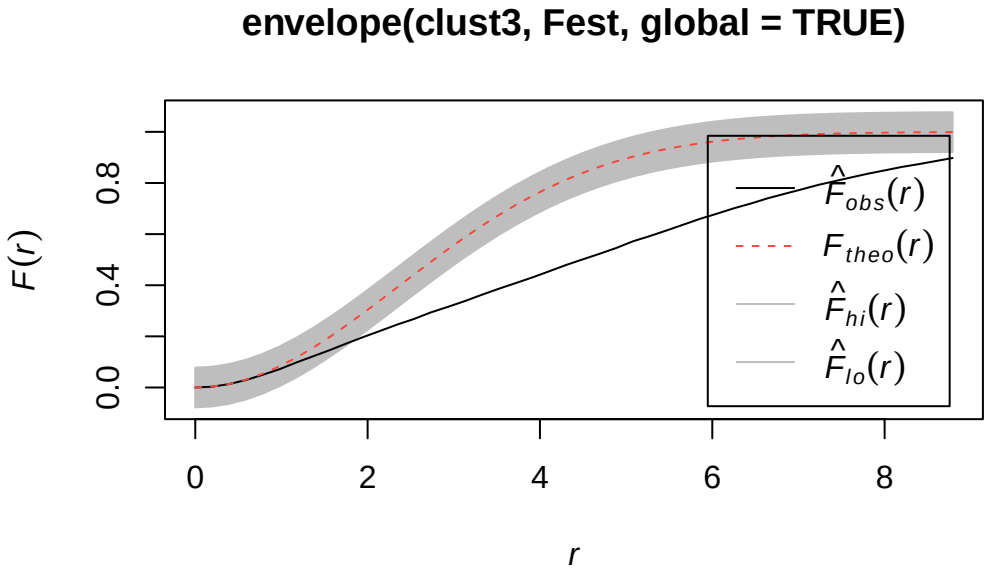


```
quadrat.test(clust3)
```

```
    Chi-squared test of CSR using quadrat counts

data:  clust3
X2 = 81.271, df = 24, p-value = 7.624e-08
alternative hypothesis: two.sided

Quadrats: 5 by 5 grid of tiles
```

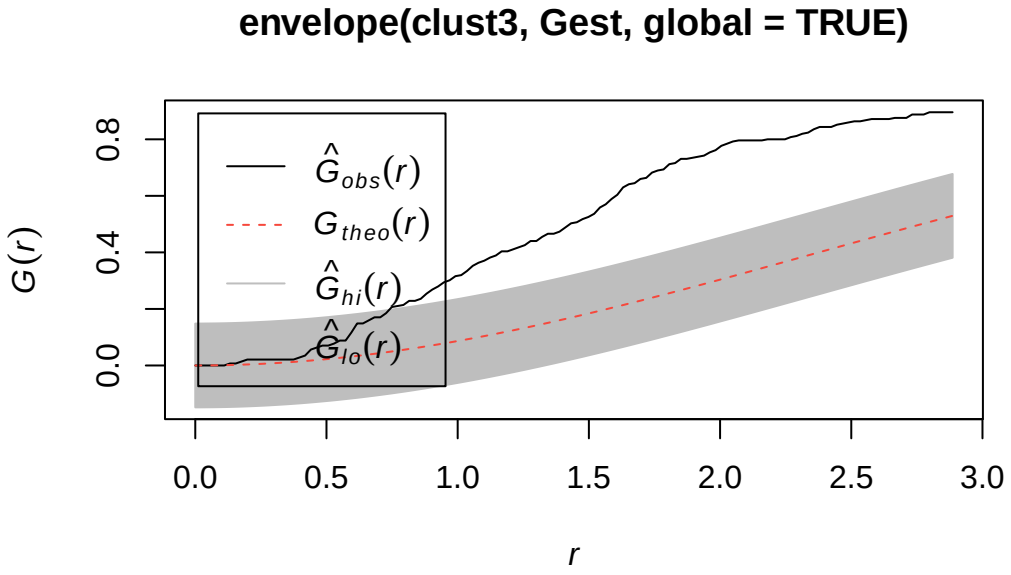
```
plot(envelope(clust3, Fest, global=TRUE))
```

```
Generating 99 simulations of CSR  ...
1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15, 16, 17, 18, 19, 20,
21, 22, 23, 24, 25, 26, 27, 28, 29, 30, 31, 32, 33, 34, 35, 36, 37, 38, 39, 40,
41, 42, 43, 44, 45, 46, 47, 48, 49, 50, 51, 52, 53, 54, 55, 56, 57, 58, 59, 60,
61, 62, 63, 64, 65, 66, 67, 68, 69, 70, 71, 72, 73, 74, 75, 76, 77, 78, 79, 80,
81, 82, 83, 84, 85, 86, 87, 88, 89, 90, 91, 92, 93, 94, 95, 96, 97, 98,
99.

Done.
```



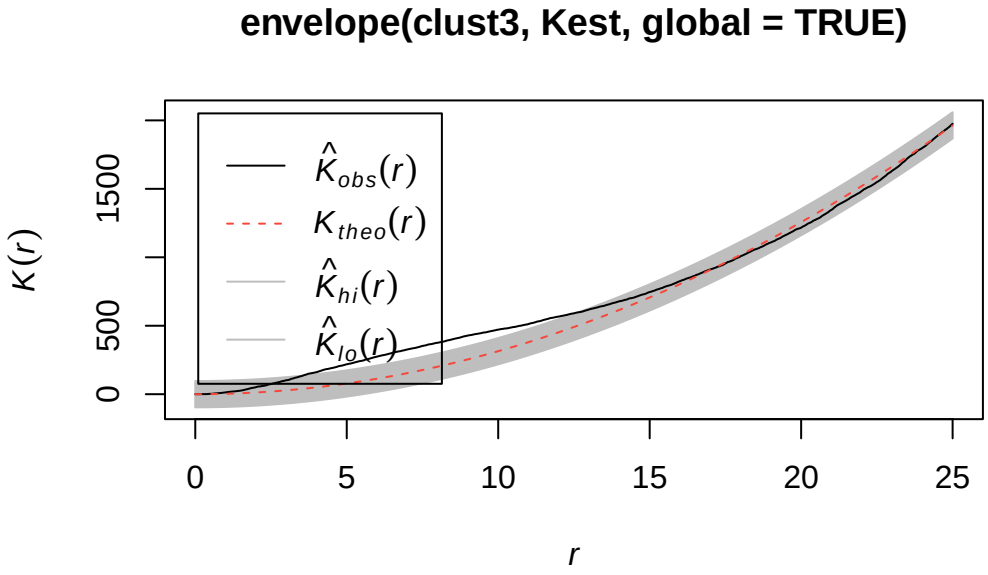

```
plot(envelope(clust3, Gest, global=TRUE))
```

```
Generating 99 simulations of CSR  ...
1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15, 16, 17, 18, 19, 20,
21, 22, 23, 24, 25, 26, 27, 28, 29, 30, 31, 32, 33, 34, 35, 36, 37, 38, 39, 40,
41, 42, 43, 44, 45, 46, 47, 48, 49, 50, 51, 52, 53, 54, 55, 56, 57, 58, 59, 60,
61, 62, 63, 64, 65, 66, 67, 68, 69, 70, 71, 72, 73, 74, 75, 76, 77, 78, 79, 80,
81, 82, 83, 84, 85, 86, 87, 88, 89, 90, 91, 92, 93, 94, 95, 96, 97, 98,
99.

Done.
```



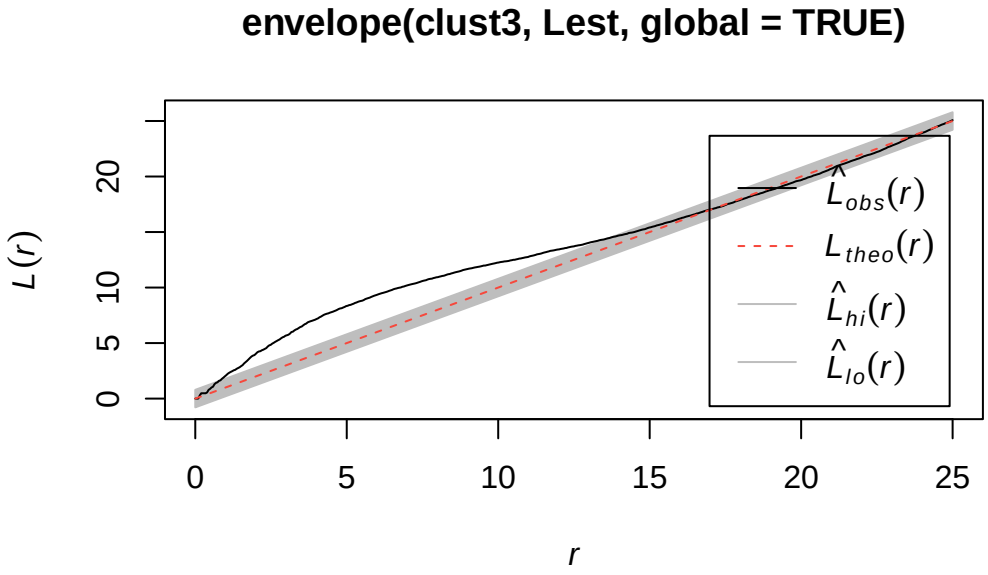

```
plot(envelope(clust3, Kest, global=TRUE))
```

```
Generating 99 simulations of CSR  ...
1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15, 16, 17, 18, 19, 20,
21, 22, 23, 24, 25, 26, 27, 28, 29, 30, 31, 32, 33, 34, 35, 36, 37, 38, 39, 40,
41, 42, 43, 44, 45, 46, 47, 48, 49, 50, 51, 52, 53, 54, 55, 56, 57, 58, 59, 60,
61, 62, 63, 64, 65, 66, 67, 68, 69, 70, 71, 72, 73, 74, 75, 76, 77, 78, 79, 80,
81, 82, 83, 84, 85, 86, 87, 88, 89, 90, 91, 92, 93, 94, 95, 96, 97, 98,
99.

Done.
```




```
plot(envelope(clust3, Lest, global=TRUE))
```

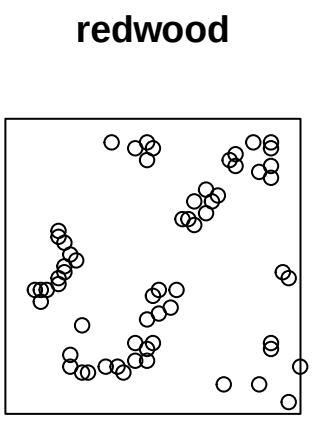
```
Generating 99 simulations of CSR  ...
1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15, 16, 17, 18, 19, 20,
21, 22, 23, 24, 25, 26, 27, 28, 29, 30, 31, 32, 33, 34, 35, 36, 37, 38, 39, 40,
41, 42, 43, 44, 45, 46, 47, 48, 49, 50, 51, 52, 53, 54, 55, 56, 57, 58, 59, 60,
61, 62, 63, 64, 65, 66, 67, 68, 69, 70, 71, 72, 73, 74, 75, 76, 77, 78, 79, 80,
81, 82, 83, 84, 85, 86, 87, 88, 89, 90, 91, 92, 93, 94, 95, 96, 97, 98,
99.

Done.
```



7.4.5.3 Fitting Cluster Poisson Process Models

If you believe the data are clustered, you can fit a model that accounts for any trends in $\lambda(s)$ and clustering in points in the generation process. Two potential models you can fit are the Thomas and the Matern model.

```
plot(redwood)
```




```
summary(kppm(redwood, ~1, "Thomas"))
```

```
Stationary cluster point process model
Fitted to point pattern dataset 'redwood'
```

```
Fitted by minimum contrast
    Summary statistic: K-function
Minimum contrast fit (object of class "minconfit")
Model: Thomas process
Fitted by matching theoretical K function to redwood

Internal parameters fitted by minimum contrast ($par):
       kappa       sigma2
23.548568483  0.002213841

Fitted cluster parameters:
      kappa       scale
23.54856848  0.04705148
Mean cluster size:  2.632856 points

Converged successfully after 105 function evaluations

Starting values of parameters:
       kappa       sigma2
62.000000000  0.006173033
Domain of integration: [ 0 , 0.25 ]
Exponents: p= 2, q= 0.25

---------- TREND  -----
Point process model
Fitted to data: X
Fitting method: maximum likelihood (Berman-Turner approximation)
Model was fitted using glm()
Algorithm converged
Call:
ppm.ppp(Q = X, trend = trend, rename.intercept = FALSE, covariates = covariates,
    covfunargs = covfunargs, use.gam = use.gam, forcefit = TRUE,
    improve.type = ppm.improve.type, improve.args = ppm.improve.args,
    nd = nd, eps = eps)
Edge correction: "border"
    [border correction distance r = 0 ]
--------------------------------------------------------------------------------
Quadrature scheme (Berman-Turner) = data + dummy + weights

Data pattern:
Planar point pattern:  62 points
Average intensity 62 points per square unit
Window: rectangle = [0, 1] x [-1, 0] units
```

```
                    (1 x 1 units)
Window area = 1 square unit

Dummy quadrature points:
     32 x 32 grid of dummy points, plus 4 corner points
     dummy spacing: 0.03125 units

Original dummy parameters: =
Planar point pattern:  1028 points
Average intensity 1030 points per square unit
Window: rectangle = [0, 1] x [-1, 0] units
                    (1 x 1 units)
Window area = 1 square unit
Quadrature weights:
     (counting weights based on 32 x 32 array of rectangular tiles)
All weights:
    range: [0.000326, 0.000977] total: 1
Weights on data points:
    range: [0.000326, 0.000488] total: 0.0277
Weights on dummy points:
    range: [0.000326, 0.000977] total: 0.972
--------------------------------------------------------------------------------
FITTED :

Stationary Poisson process

---- Intensity: ----


Uniform intensity:
[1] 62

            Estimate      S.E.  CI95.lo CI95.hi Ztest     Zval
(Intercept) 4.127134 0.1270001 3.878219 4.37605   *** 32.49709

----------- gory details -----

Fitted regular parameters (theta):
(Intercept)
   4.127134

Fitted exp(theta):
(Intercept)
```

```
         62

----------- CLUSTER  -----------
Model: Thomas process

Fitted cluster parameters:
      kappa       scale
23.54856848  0.04705148
Mean cluster size:  2.632856 points

Final standard error and CI
(allowing for correlation of cluster process):
            Estimate      S.E.  CI95.lo  CI95.hi Ztest     Zval
(Intercept) 4.127134 0.2329338 3.670593 4.583676   *** 17.71806

----------- cluster strength indices ----------
Sibling probability 0.6041858
Count overdispersion index (on original window): 3.408838
Cluster strength: 1.526438

Spatial persistence index (over window): 0

Bound on distance from Poisson process (over window): 1
     = min (1, 115.0878, 6030.864, 5306724, 76.60044)

Bound on distance from MIXED Poisson process (over window): 1

Intensity of parents of nonempty clusters: 21.85607
Mean number of offspring in a nonempty cluster: 2.836741
Intensity of parents of clusters of more than one offspring point: 17.39995
Ratio of parents to parents-plus-offspring: 0.2752655 (where 1 = Poisson
process)
Probability that a typical point belongs to a nontrivial cluster: 0.9281271
```

```
AIC(kppm(redwood, ~1, "Thomas", method='palm'))
```

```
[1] -2477.982
```

```
summary(kppm(redwood, ~x, "Thomas"))
```

```
Inhomogeneous cluster point process model
```

```
Fitted to point pattern dataset 'redwood'
Fitted by minimum contrast
    Summary statistic: inhomogeneous K-function
Minimum contrast fit (object of class "minconfit")
Model: Thomas process
Fitted by matching theoretical K function to redwood

Internal parameters fitted by minimum contrast ($par):
       kappa       sigma2
22.917939455  0.002148329

Fitted cluster parameters:
      kappa       scale
22.91793945  0.04635007
Mean cluster size:  [pixel image]

Converged successfully after 85 function evaluations

Starting values of parameters:
       kappa       sigma2
62.000000000  0.006173033
Domain of integration: [ 0 , 0.25 ]
Exponents: p= 2, q= 0.25

---------- TREND  -----
Point process model
Fitted to data: X
Fitting method: maximum likelihood (Berman-Turner approximation)
Model was fitted using glm()
Algorithm converged
Call:
ppm.ppp(Q = X, trend = trend, rename.intercept = FALSE, covariates = covariates,
    covfunargs = covfunargs, use.gam = use.gam, forcefit = TRUE,
    improve.type = ppm.improve.type, improve.args = ppm.improve.args,
    nd = nd, eps = eps)
Edge correction: "border"
    [border correction distance r = 0 ]
--------------------------------------------------------------------------------
Quadrature scheme (Berman-Turner) = data + dummy + weights

Data pattern:
Planar point pattern:  62 points
Average intensity 62 points per square unit
```

```
Window: rectangle = [0, 1] x [-1, 0] units
                    (1 x 1 units)
Window area = 1 square unit

Dummy quadrature points:
     32 x 32 grid of dummy points, plus 4 corner points
     dummy spacing: 0.03125 units

Original dummy parameters: =
Planar point pattern:  1028 points
Average intensity 1030 points per square unit
Window: rectangle = [0, 1] x [-1, 0] units
                    (1 x 1 units)
Window area = 1 square unit
Quadrature weights:
     (counting weights based on 32 x 32 array of rectangular tiles)
All weights:
    range: [0.000326, 0.000977] total: 1
Weights on data points:
    range: [0.000326, 0.000488] total: 0.0277
Weights on dummy points:
    range: [0.000326, 0.000977] total: 0.972
--------------------------------------------------------------------------------
FITTED :

Nonstationary Poisson process

---- Intensity: ----

Log intensity: ~x

Fitted trend coefficients:
(Intercept)           x
  3.9745791   0.2976994

             Estimate      S.E.    CI95.lo  CI95.hi Ztest       Zval
(Intercept) 3.9745791 0.2639734  3.4572007 4.491958   *** 15.0567391
x           0.2976994 0.4409396 -0.5665264 1.161925        0.6751478

----------- gory details -----

Fitted regular parameters (theta):
(Intercept)           x
```

```
  3.9745791   0.2976994

Fitted exp(theta):
(Intercept)           x
  53.227709    1.346757

----------- CLUSTER  -----------
Model: Thomas process

Fitted cluster parameters:
      kappa       scale
22.91793945  0.04635007
Mean cluster size:  [pixel image]

Final standard error and CI
(allowing for correlation of cluster process):
             Estimate      S.E.   CI95.lo  CI95.hi Ztest      Zval
(Intercept) 3.9745791 0.4641811  3.064801 4.884357   *** 8.5625609
x           0.2976994 0.7850129 -1.240898 1.836296       0.3792287

----------- cluster strength indices ----------
Mean sibling probability 0.6177764
Count overdispersion index (on original window): 3.494409
Cluster strength: 1.616269

Spatial persistence index (over window): 0

Bound on distance from Poisson process (over window): 1
     = min (1, 118.5459, 6380.467, 5790004, 78.82096)
```

```
AIC(kppm(redwood, ~x, "Thomas", method='palm'))
```

```
[1] -2465.114
```

```
summary(kppm(redwood, ~1, "MatClust"))
```

```
Stationary cluster point process model
Fitted to point pattern dataset 'redwood'
Fitted by minimum contrast
    Summary statistic: K-function
Minimum contrast fit (object of class "minconfit")
```

```
Model: Matern cluster process
Fitted by matching theoretical K function to redwood

Internal parameters fitted by minimum contrast ($par):
      kappa           R
24.55865127  0.08653577

Fitted cluster parameters:
      kappa       scale
24.55865127  0.08653577
Mean cluster size:  2.524569 points

Converged successfully after 57 function evaluations

Starting values of parameters:
      kappa           R
62.00000000  0.07856865
Domain of integration: [ 0 , 0.25 ]
Exponents: p= 2, q= 0.25

---------- TREND  -----
Point process model
Fitted to data: X
Fitting method: maximum likelihood (Berman-Turner approximation)
Model was fitted using glm()
Algorithm converged
Call:
ppm.ppp(Q = X, trend = trend, rename.intercept = FALSE, covariates = covariates,
    covfunargs = covfunargs, use.gam = use.gam, forcefit = TRUE,
    improve.type = ppm.improve.type, improve.args = ppm.improve.args,
    nd = nd, eps = eps)
Edge correction: "border"
    [border correction distance r = 0 ]
--------------------------------------------------------------------------------
Quadrature scheme (Berman-Turner) = data + dummy + weights

Data pattern:
Planar point pattern:  62 points
Average intensity 62 points per square unit
Window: rectangle = [0, 1] x [-1, 0] units
                    (1 x 1 units)
Window area = 1 square unit
```

```
Dummy quadrature points:
     32 x 32 grid of dummy points, plus 4 corner points
     dummy spacing: 0.03125 units

Original dummy parameters: =
Planar point pattern:  1028 points
Average intensity 1030 points per square unit
Window: rectangle = [0, 1] x [-1, 0] units
                    (1 x 1 units)
Window area = 1 square unit
Quadrature weights:
     (counting weights based on 32 x 32 array of rectangular tiles)
All weights:
    range: [0.000326, 0.000977] total: 1
Weights on data points:
    range: [0.000326, 0.000488] total: 0.0277
Weights on dummy points:
    range: [0.000326, 0.000977] total: 0.972
--------------------------------------------------------------------------------
FITTED :

Stationary Poisson process

---- Intensity: ----


Uniform intensity:
[1] 62

            Estimate      S.E.  CI95.lo CI95.hi Ztest     Zval
(Intercept) 4.127134 0.1270001 3.878219 4.37605   *** 32.49709

----------- gory details -----

Fitted regular parameters (theta):
(Intercept)
   4.127134

Fitted exp(theta):
(Intercept)
         62

----------- CLUSTER  -----------
```

```
Model: Matern cluster process

Fitted cluster parameters:
      kappa       scale
24.55865127  0.08653577
Mean cluster size:  2.524569 points

Final standard error and CI
(allowing for correlation of cluster process):
            Estimate      S.E.  CI95.lo  CI95.hi Ztest     Zval
(Intercept) 4.127134 0.2303018 3.675751 4.578518   *** 17.92054

---------- cluster strength indices ----------
Sibling probability 0.6338109
Count overdispersion index (on original window): 3.32639
Cluster strength: 1.73083

Spatial persistence index (over window): 0

Bound on distance from Poisson process (over window): 1
     = min (1, 114.0685, 6809.832, 895130.7, 81.56782)

Bound on distance from MIXED Poisson process (over window): 1

Intensity of parents of nonempty clusters: 22.59168
Mean number of offspring in a nonempty cluster: 2.744373
Intensity of parents of clusters of more than one offspring point: 17.62592
Ratio of parents to parents-plus-offspring: 0.2837227 (where 1 = Poisson
process)
Probability that a typical point belongs to a nontrivial cluster: 0.9199071
```

```
AIC(kppm(redwood, ~1, "MatClust",method='palm'))
```

```
[1] -2476.282
```

```
summary(kppm(redwood, ~x, "MatClust"))
```

```
Inhomogeneous cluster point process model
Fitted to point pattern dataset 'redwood'
Fitted by minimum contrast
    Summary statistic: inhomogeneous K-function
```

```
Minimum contrast fit (object of class "minconfit")
Model: Matern cluster process
Fitted by matching theoretical K function to redwood

Internal parameters fitted by minimum contrast ($par):
      kappa           R
23.89160402  0.08523814

Fitted cluster parameters:
      kappa       scale
23.89160402  0.08523814
Mean cluster size:  [pixel image]

Converged successfully after 55 function evaluations

Starting values of parameters:
      kappa           R
62.00000000  0.07856865
Domain of integration: [ 0 , 0.25 ]
Exponents: p= 2, q= 0.25

---------- TREND  -----
Point process model
Fitted to data: X
Fitting method: maximum likelihood (Berman-Turner approximation)
Model was fitted using glm()
Algorithm converged
Call:
ppm.ppp(Q = X, trend = trend, rename.intercept = FALSE, covariates = covariates,
    covfunargs = covfunargs, use.gam = use.gam, forcefit = TRUE,
    improve.type = ppm.improve.type, improve.args = ppm.improve.args,
    nd = nd, eps = eps)
Edge correction: "border"
    [border correction distance r = 0 ]
--------------------------------------------------------------------------------
Quadrature scheme (Berman-Turner) = data + dummy + weights

Data pattern:
Planar point pattern:  62 points
Average intensity 62 points per square unit
Window: rectangle = [0, 1] x [-1, 0] units
                    (1 x 1 units)
Window area = 1 square unit
```

```
Dummy quadrature points:
     32 x 32 grid of dummy points, plus 4 corner points
     dummy spacing: 0.03125 units

Original dummy parameters: =
Planar point pattern:  1028 points
Average intensity 1030 points per square unit
Window: rectangle = [0, 1] x [-1, 0] units
                    (1 x 1 units)
Window area = 1 square unit
Quadrature weights:
     (counting weights based on 32 x 32 array of rectangular tiles)
All weights:
    range: [0.000326, 0.000977] total: 1
Weights on data points:
    range: [0.000326, 0.000488] total: 0.0277
Weights on dummy points:
    range: [0.000326, 0.000977] total: 0.972
--------------------------------------------------------------------------------
FITTED :

Nonstationary Poisson process

---- Intensity: ----

Log intensity: ~x

Fitted trend coefficients:
(Intercept)           x
  3.9745791   0.2976994

             Estimate      S.E.    CI95.lo  CI95.hi Ztest       Zval
(Intercept) 3.9745791 0.2639734  3.4572007 4.491958   *** 15.0567391
x           0.2976994 0.4409396 -0.5665264 1.161925        0.6751478

----------- gory details -----

Fitted regular parameters (theta):
(Intercept)           x
  3.9745791   0.2976994

Fitted exp(theta):
```

```
(Intercept)           x
  53.227709    1.346757

----------- CLUSTER  -----------
Model: Matern cluster process

Fitted cluster parameters:
      kappa       scale
23.89160402  0.08523814
Mean cluster size:  [pixel image]

Final standard error and CI
(allowing for correlation of cluster process):
             Estimate      S.E.   CI95.lo CI95.hi Ztest      Zval
(Intercept) 3.9745791 0.4599425  3.073108 4.87605   *** 8.6414699
x           0.2976994 0.7783921 -1.227921 1.82332       0.3824544

----------- cluster strength indices ----------
Mean sibling probability 0.647109
Count overdispersion index (on original window): 3.409944
Cluster strength: 1.833736

Spatial persistence index (over window): 0

Bound on distance from Poisson process (over window): 1
     = min (1, 117.8056, 7209.547, 977197.8, 83.95629)
```

```
AIC(kppm(redwood, ~x, "MatClust",method='palm'))
```

```
[1] -2463.417
```

We can then interpret the estimates for that process to tell us about the clustering.

INTERPRETATIONS

- theta: parameters for intensity trend
- kappa: average number of clusters per unit area
- scale: standard deviation of the distance of a point from its cluster center.
- mu: mean number of points per cluster

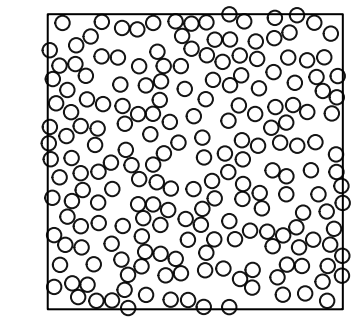
7.4.6 Inhibition Poisson Processes

If points "repel" each other, we need to account for that also.

7.4.6.1 Simple Sequential Inhibition

Each new point is generated uniformly in the window (space of interest) and independently of preceding points. If the new point lies closer than r units from an existing point, it is rejected, generating another random point. The process terminates when no further points can be added.

```
plot(rSSI(r = 0.05, n = 200))
```



7.4.6.2 Matern I Inhibition Model

Matern's Model I first generates a homogeneous Poisson process Y (intensity = ρ). Any pairs of points that lie closer than a distance r from each other are deleted. Thus, pairs of close neighbors annihilate each other.

The probability an arbitrary point survives is $e^{-\pi\rho r^2}$ so the intensity of the final point pattern is $\lambda = \rho e^{-\pi\rho r^2}$.

7.4.7 Other Point Process Models

- Markov Point Processes: a large class of models that allow for interaction between points (attraction or repulsion)
- Hard Core Gibbs Point Processes: a subclass of Markov Point Processes that allow for interaction between points; no interaction is a Poisson point process
- Strauss Processes: a subclass of Markov Point Processes that allow for the repulsion between pairs of points

- Cox Processes: a doubly stochastic Poisson process in which the intensity is a random process and conditional on the intensity, the events are an inhomogeneous Poisson process.

For more information on how to fit spatial point processes in R, see http://www3.uji.es/~mateu/badturn.pdf.

7.5 Point Referenced Data (optional)

A spatial stochastic process is a spatially indexed collection of random variables,

$$\{Y(s) : s \in D \subset \mathbb{R}^d\}$$

where d will typically be 2 or 3.

A realization from a stochastic process is sometimes called a **sample path**. We typically observe a vector,

$$\mathbf{Y} \equiv Y(s_1), ..., Y(s_n)$$

7.5.1 Gaussian Process

A Gaussian process (G.P.) has finite-dimensional distributions that are all multivariate normal, and we define it using a mean and covariance function,

$$\mu(s) = E(Y(s))$$
$$C(s_1, s_2) = Cov(Y(s_1), Y(s_2))$$

As with time series and longitudinal data, we must consider the covariance of our data. Most covariance models that we will consider in this course require that our data come from a stationary, isotropic process such that the mean is constant across space, variance is constant across space, and the covariance is only a function of the distance between locations (disregarding the direction) such that

$$E(Y(s)) = E(Y(s+h)) = \mu$$
$$Cov(Y(s), Y(s+h)) = Cov(Y(0), Y(h)) = C(||h||)$$

7.5.2 Covariance Models

A covariance model that is often used in spatial data is called the **Matern class** of covariance functions,

$$C(t) = \frac{\sigma^2}{2^{\nu-1}\Gamma(\nu)}(t\rho)^{\nu}\mathcal{K}_{\nu}(d/\rho)$$

where $\mathcal{K}_{\nu}$ is a modified Bessel function of order ν. ν is a smoothness parameter and if we let $\nu = 1/2$, we get the exponential covariance.

7.5.3 Variograms, Semivariograms

Beyond the standard definition of stationarity, there is another form of **stationary (intrinsic stationarity),** which is when

$$Var(Y(s+h) - Y(s)) \text{ depends only on h}$$

When this is true, we call

$$\gamma(h) = \frac{1}{2}Var(Y(s+h) - Y(s))$$

the **semivariogram** and $2\gamma(h)$ the variogram.

If a covariance function is stationary, it will be intrinsic stationary and

$$\gamma(h) = C(0) - C(h) = C(0)(1 - \rho(h))$$

where $C(0)$ is the variance (referred to as the **sill**).

First, to estimate the semivariogram, fit a trend so that you have a stationary process in your residuals.

- Let $H_1, ..., H_k$ partition the space of possible lags, with h_u being a representative spatial lag/distance in H_u. Then use your residuals to estimate the empirical semivariogram.

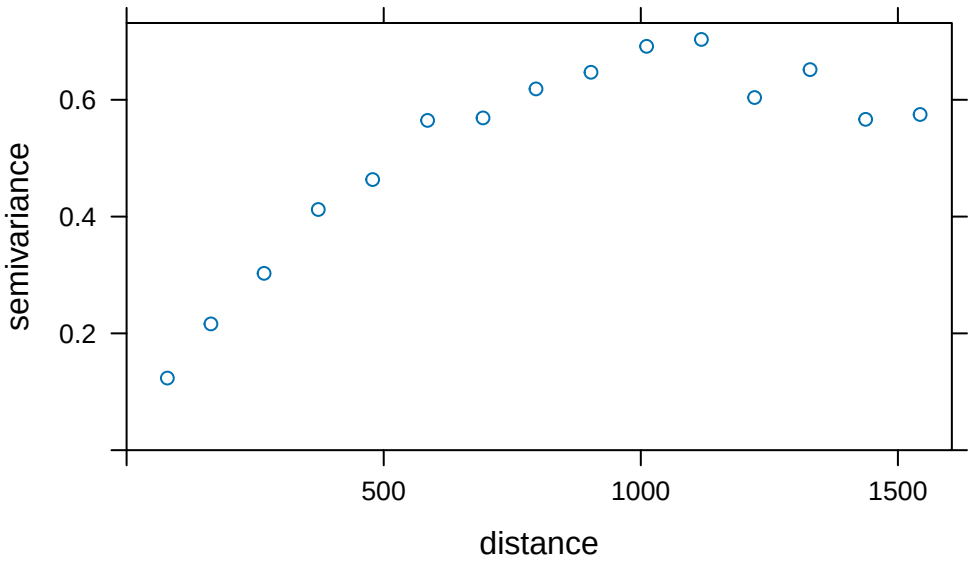
$$\hat{\gamma}(h_u) = \frac{1}{2 \cdot |\{s_i - s_j \in H_u\}|} \sum_{\{s_i - s_j \in H_u\}} (e(s_i) - e(s_j))^2$$

- This is a non-parametric estimate of the semivariogram.

```
library(gstat)
```

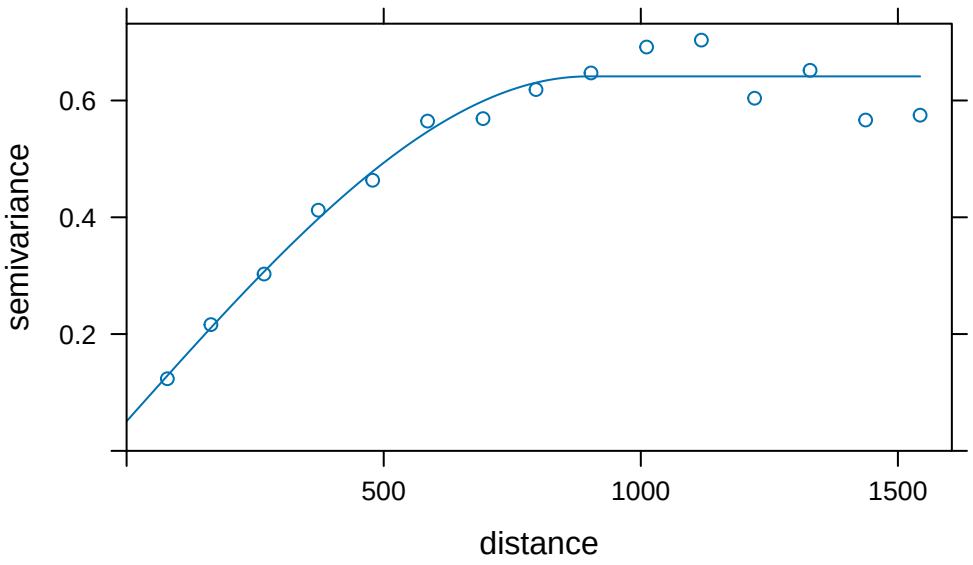
```
Attaching package: 'gstat'

The following object is masked from 'package:spatstat.explore':

    idw
```

```
coordinates(meuse) = ~x+y

estimatedVar <- variogram(log(zinc) ~ 1, data = meuse)
plot(estimatedVar)
```

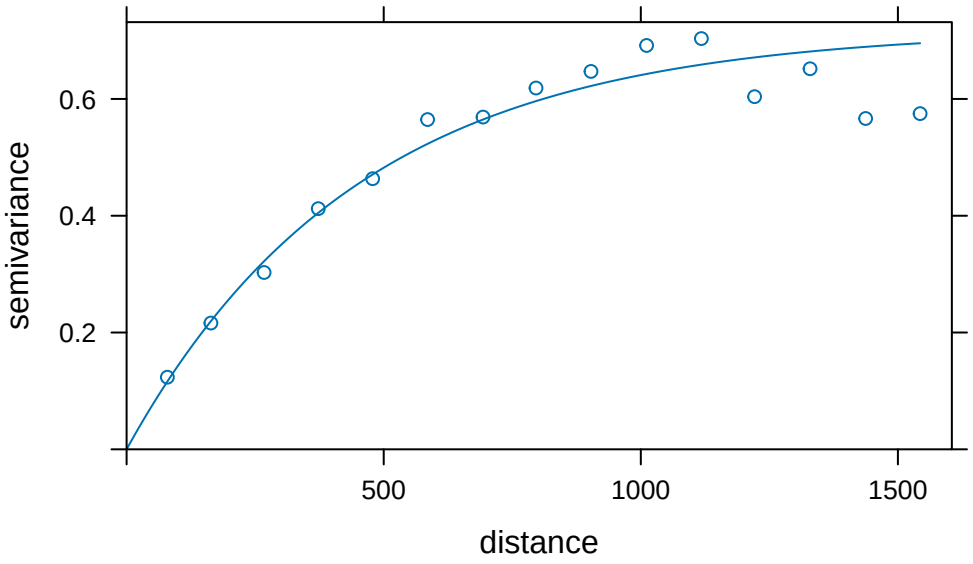


- If we notice a particular pattern in the points, we could try and fit a curve based on correlation models.

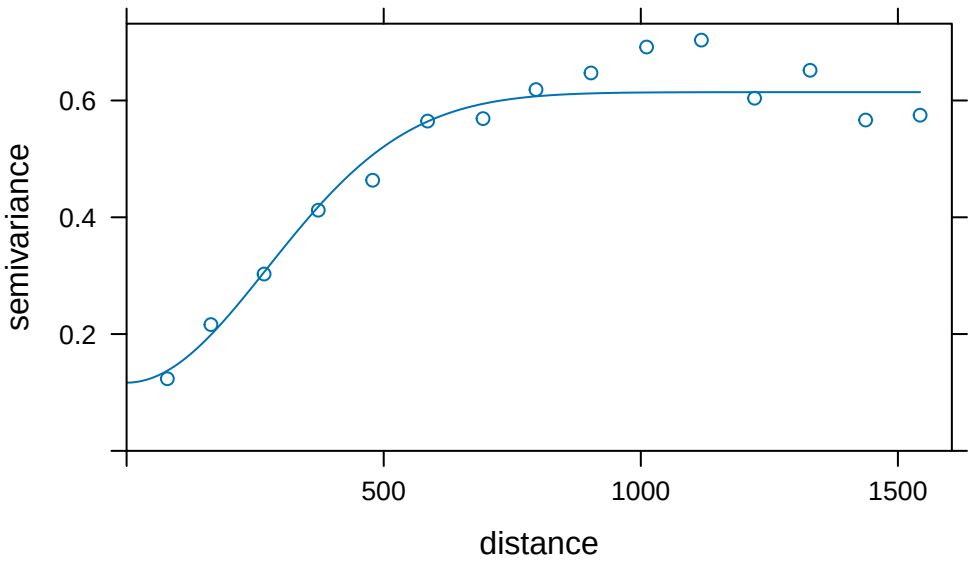
```
Var.fit1 <- fit.variogram(estimatedVar, model = vgm(1, "Sph", 900, 1))
plot(estimatedVar,Var.fit1)
```



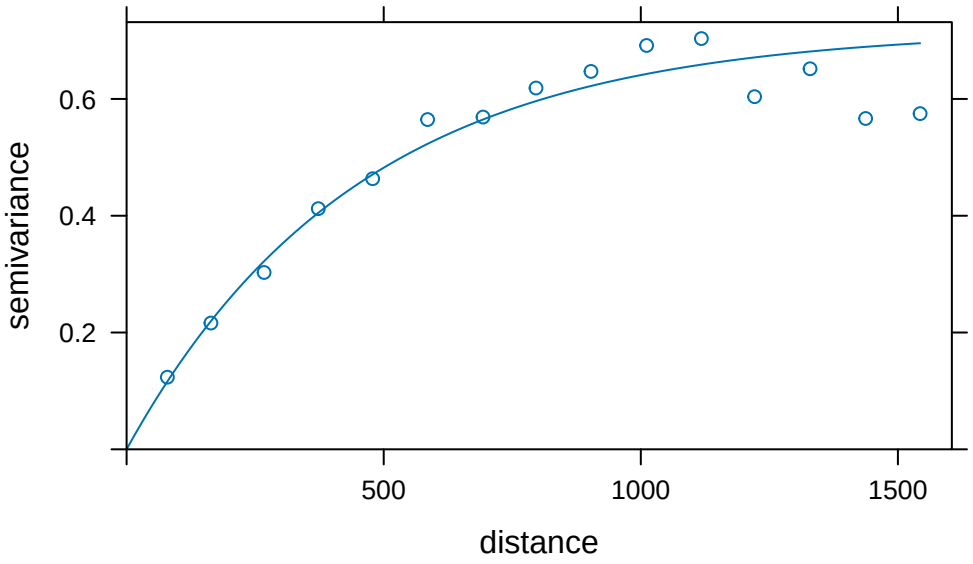

```
Var.fit2 <- fit.variogram(estimatedVar, model = vgm(1, "Exp", 900, 1))
plot(estimatedVar,Var.fit2)
```




```
Var.fit3 <- fit.variogram(estimatedVar, model = vgm(1, "Gau", 900, 1))
plot(estimatedVar,Var.fit3)
```




```
Var.fit4 <- fit.variogram(estimatedVar, model = vgm(1, "Mat", 900, 1))
plot(estimatedVar,Var.fit4)
```




```
g = gstat::gstat(formula = log(zinc) ~ 1, model = Var.fit4, data = meuse)
```

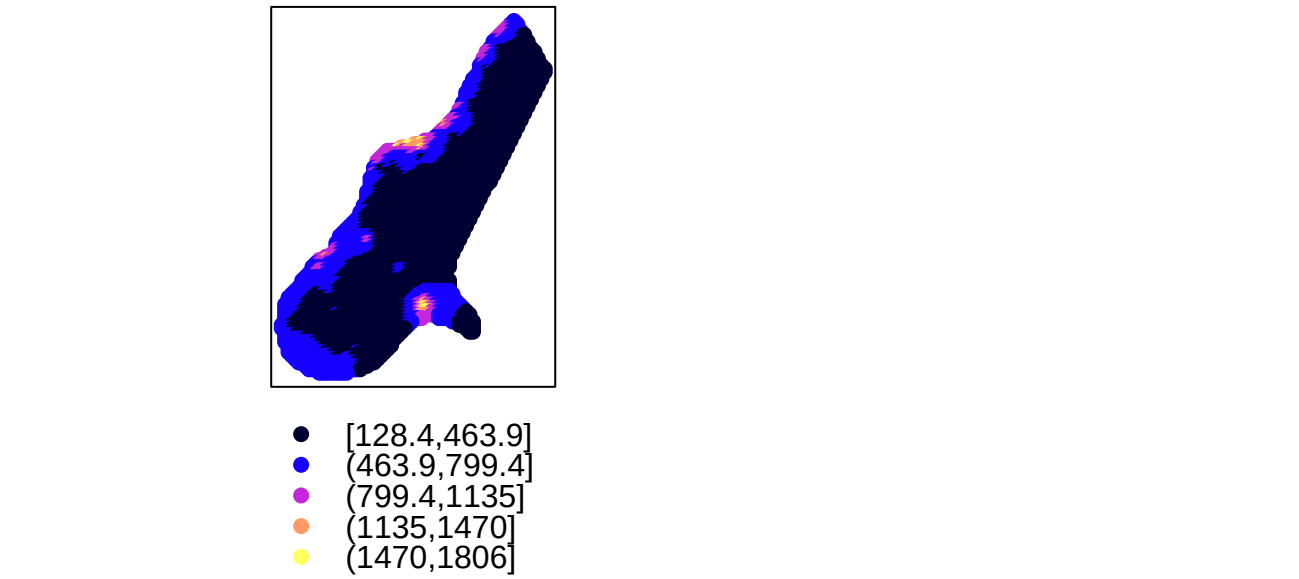
7.5.4 Kriging

Section is under construction.

```
data(meuse)
data(meuse.grid)
coordinates(meuse) = ~x+y
coordinates(meuse.grid) = ~x+y


zinc.idw <- idw(zinc~1, meuse, meuse.grid)
```

```
[inverse distance weighted interpolation]
```

```
spplot(zinc.idw["var1.pred"], main = "zinc inverse distance weighted interpolations")
```

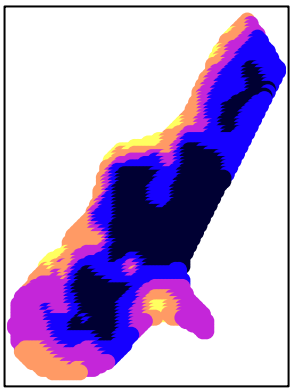



```
zinc.ord <- predict(g, meuse.grid)
```

```
[using ordinary kriging]
```

```
spplot(zinc.ord["var1.pred"], main = "zinc ordinary kriging interpolations")
```

zinc ordinary kriging interpolations

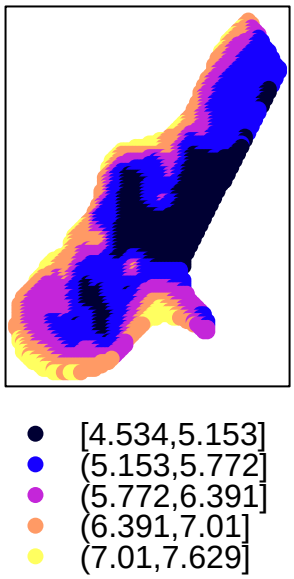



```
g = gstat::gstat(formula = log(zinc) ~ sqrt(dist), model = Var.fit4, data = meuse)
zinc.uni <- predict(g, meuse.grid)
```

```
[using universal kriging]
```

```
spplot(zinc.uni["var1.pred"], main = "zinc universal kriging interpolations")
```

zinc universal kriging interpolations



7.6 Areal Data

Areal data can often be thought of as a "coarser-resolution" version of other data types, such as

- average/aggregation of point reference data (a geostatistical field)
- a count of points within a boundary from a point process.

For areal data, we will explore relationships between aggregate summaries of observations within boundaries while specifying spatial dependence regarding notions of neighborhoods and spatial proximity. The boundary of areas can be considered polygons determined by a closed sequence of ordered coordinates connected by straight line segments.

7.6.1 Polygons

Let's look at an example to understand what we mean by polygons. Say we are interested in the county-level data in North Carolina. We can read in a series of shapefiles and specify a CRS (ellipsoid, datum, and projection) to project longitude and latitude onto a 2d surface.

```
library(spdep)
```

```
Loading required package: spData
```

```
To access larger datasets in this package, install the spDataLarge
package with: `install.packages('spDataLarge',
repos='https://nowosad.github.io/drat/', type='source')`


Attaching package: 'spData'

The following object is masked _by_ '.GlobalEnv':

    world
```

```
library(sf)
nc <- st_read(system.file("shape/nc.shp", package="sf"), quiet=TRUE)
st_crs(nc) <- "+proj=longlat +datum=NAD27"
row.names(nc) <- as.character(nc$FIPSNO)
```

This data set includes some county-level information about births. Let's look at the first 6 rows. This data set is not just a typical data frame but has geometric information. In particular, the geometry field is a list of multi polygons, a series of coordinates that can be connected to create polygons. They might be multi polygons in case one county may consist of two or more closed polygons (separated by a body of water, such as an island).

```
head(nc)
```

```
Simple feature collection with 6 features and 14 fields
Geometry type: MULTIPOLYGON
Dimension:     XY
Bounding box:  xmin: -81.74107 ymin: 36.07282 xmax: -75.77316 ymax: 36.58965
Geodetic CRS:  +proj=longlat +datum=NAD27
       AREA PERIMETER CNTY_ CNTY_ID        NAME  FIPS FIPSNO CRESS_ID BIR74
37009 0.114     1.442  1825    1825        Ashe 37009  37009        5  1091
37005 0.061     1.231  1827    1827   Alleghany 37005  37005        3   487
37171 0.143     1.630  1828    1828       Surry 37171  37171       86  3188
37053 0.070     2.968  1831    1831   Currituck 37053  37053       27   508
37131 0.153     2.206  1832    1832 Northampton 37131  37131       66  1421
37091 0.097     1.670  1833    1833    Hertford 37091  37091       46  1452
      SID74 NWBIR74 BIR79 SID79 NWBIR79                       geometry
37009     1      10  1364     0      19 MULTIPOLYGON (((-81.47276 3...
37005     0      10   542     3      12 MULTIPOLYGON (((-81.23989 3...
37171     5     208  3616     6     260 MULTIPOLYGON (((-80.45634 3...
37053     1     123   830     2     145 MULTIPOLYGON (((-76.00897 3...
```

```
37131     9    1066  1606     3    1197 MULTIPOLYGON (((-77.21767 3...
37091     7     954  1838     5    1237 MULTIPOLYGON (((-76.74506 3...
```

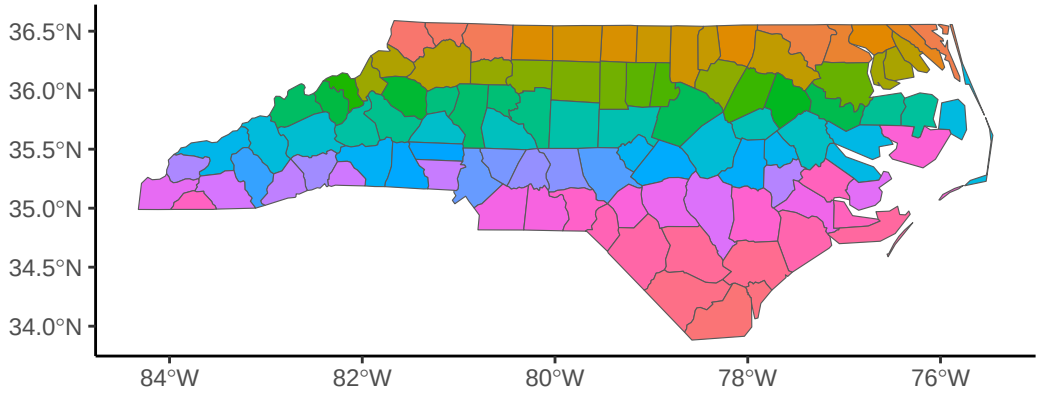
```
st_geometry(nc)
```

```
Geometry set for 100 features
Geometry type: MULTIPOLYGON
Dimension:     XY
Bounding box:  xmin: -84.32385 ymin: 33.88199 xmax: -75.45698 ymax: 36.58965
Geodetic CRS:  +proj=longlat +datum=NAD27
First 5 geometries:

MULTIPOLYGON (((-81.47276 36.23436, -81.54084 3...

MULTIPOLYGON (((-81.23989 36.36536, -81.24069 3...

MULTIPOLYGON (((-80.45634 36.24256, -80.47639 3...

MULTIPOLYGON (((-76.00897 36.3196, -76.01735 36...

MULTIPOLYGON (((-77.21767 36.24098, -77.23461 3...
```

If we plot these polygons and fill them with their own color, we can see each boundary, including some counties that include islands (multiple polygons).

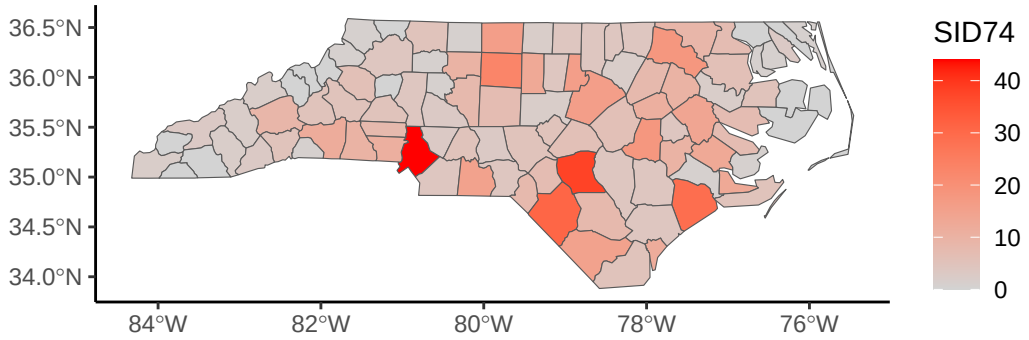
```
ggplot(nc) +
  geom_sf(aes(fill = factor(CNTY_ID))) +
  guides(fill = FALSE) + theme_classic()
```

```
Warning: The `<scale>` argument of `guides()` cannot be `FALSE`. Use "none" instead as
of ggplot2 3.3.4.
```



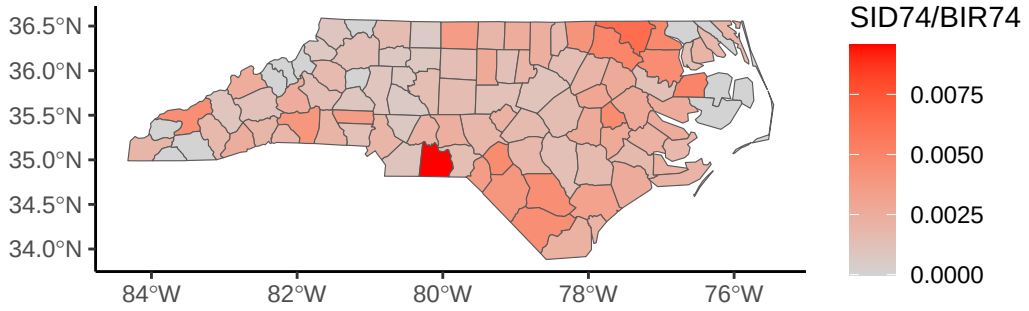
If we'd like to visualize an outcome spatially, we can change the fill to correspond to the value of another variable and shade the color on a spectrum. Below are cases of Sudden Infant Death (SID) in 1974 at a county level in North Carolina.

```
ggplot(nc) +
  geom_sf(aes(fill = SID74)) +
  scale_fill_gradient(high= 'red', low ='lightgrey') +
  theme_classic()
```



Here are the cases of SID relative to the birth rate in 1974 at a county level in North Carolina. This considers that more metropolitan areas will have higher birth rates and, thus, higher SID rates.

```
ggplot(nc) +
  geom_sf(aes(fill = SID74/BIR74)) +
  scale_fill_gradient(high= 'red', low ='lightgrey') +
  theme_classic()
```

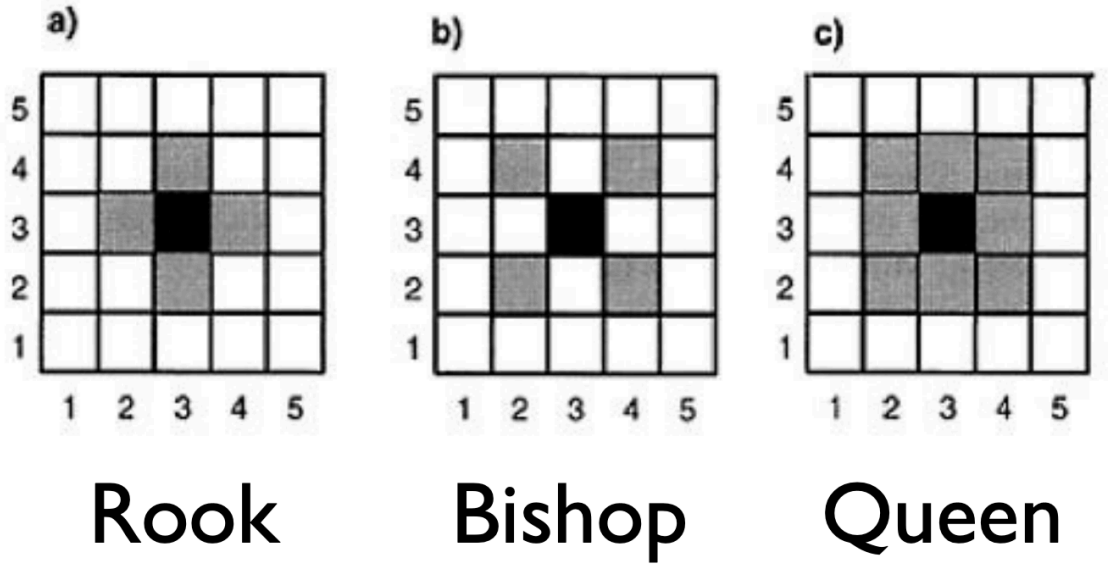


Now, we might wonder why some counties have a higher incidence of SID. Do we have other county-level information that can help us explain those differences? Are there geographical similarities and differences that might explain why two counties closer together might have similar rates?

To answer these questions, we must first consider what it means for two polygons or areas to be "close." We'll use the language of two areas being close if they are neighbors. But, we need to define a neighborhood structure on our geometry of polygons.

7.6.2 Neighborhood Structure

For data on a regular lattice, it's fairly straightforward to think about what a neighborhood means. You'll often see terminology referring to Chess moves.



There are many ways to define the neighborhood structure if we have irregular lattices.

- **Queen**: If two polygons touch at all, even at one point, such as a corner, they are neighbors.
- **Rook**: If two polygons share an edge (more than one point), they are neighbors.
- **K Nearest Neighbors**: If we calculate the distance between the centers of the polygons, also known as centroids, we can define a neighborhood based on K nearest polygons, distance based on the centers.
- **Distance Nearest Neighbors**: If we calculate the distance between the centers of the polygons, also known as centroids, we can define a neighborhood based on a minimum and maximum distance for the nearest polygons, distance based on the centers.

As you see in the visualizations, these different ways of defining a neighborhood lead to nice and not-so-nice properties.

- **Nice Properties**

  - Each polygon has at least one neighbor
  - Neighbors make sense in the data context in that those neighbors share attributes that might make them more correlated

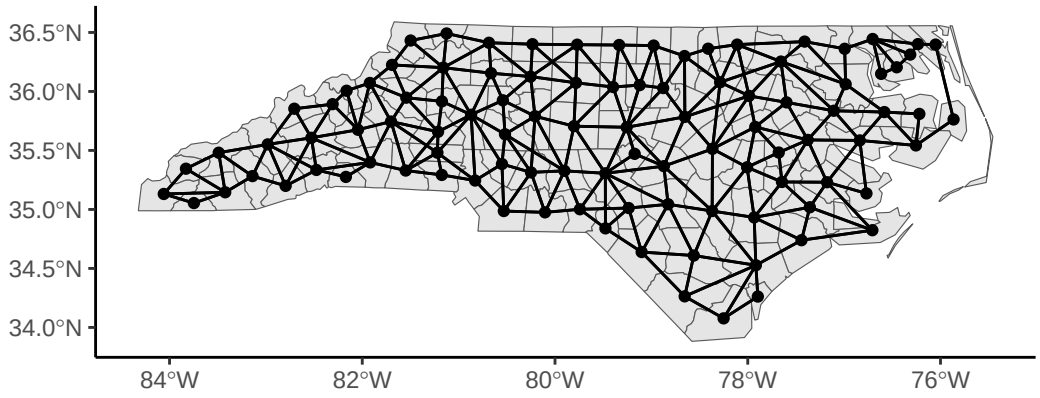
- **Not So Nice Properties**
  - Polygons have no neighbors; which means that we are assuming they aren't spatially correlated with any other polygon
  - Polygons have too many neighbors, including some that wouldn't be spatially correlated based on the data context

Of course, you can also define the neighborhoods manually to incorporate your knowledge of the data context. The approaches defined above are great ways to start thinking about neighboring polygons or areas and the correlation structure in the data.

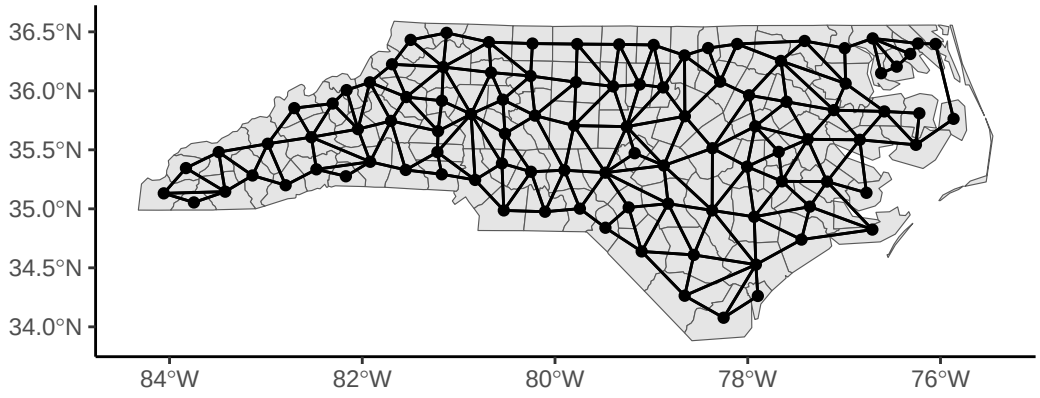
```
# Centroid of Polygons
centroids <- st_centroid(st_geometry(nc), of_largest_polygon=TRUE)


#Neighbor information for nc data
Queen <- poly2nb(nc, queen = TRUE)
Rook <-  poly2nb(nc, queen = FALSE)

KNN <- knn2nb(knearneigh(centroids, k = 3)) #k: number of neighbors
KNNDist <- dnearneigh(centroids ,d1 = 0,d2 = 40) #d1: min distance, d2: max distanc
```

```
Warning in dnearneigh(centroids, d1 = 0, d2 = 40): neighbour object has 7
sub-graphs
```

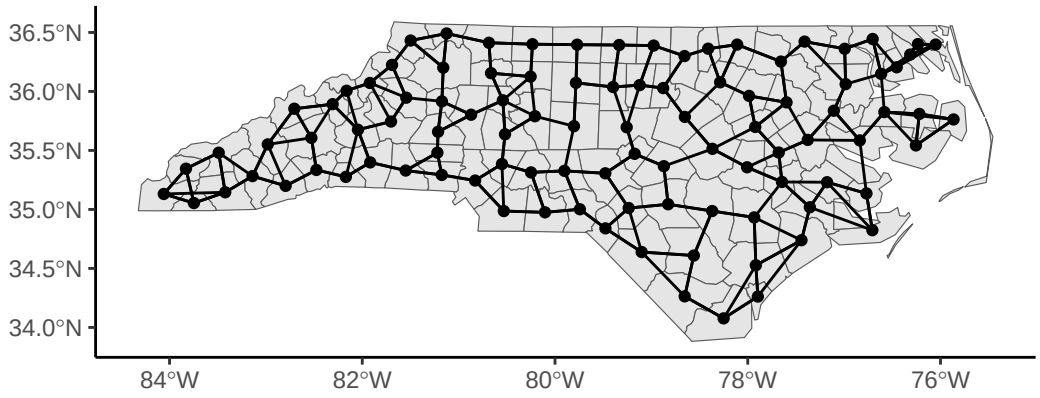
```
# Network lines from Neighbors
nb_Q_net <- nb2lines(nb = Queen, coords = centroids, as_sf = TRUE)
nb_R_net <- nb2lines(nb = Rook, coords = centroids, as_sf = TRUE)
nb_KNN_net <- nb2lines(nb = KNN, coords = centroids, as_sf = TRUE)
nb_KNNDist_net <- nb2lines(nb = KNNDist, coords = centroids, as_sf = TRUE)


# Plots of Neighbor Networks
nc %>%
ggplot() +
  geom_sf() +
  geom_sf(data = centroids) +
  geom_sf(data = nb_Q_net) +
  theme_classic()
```



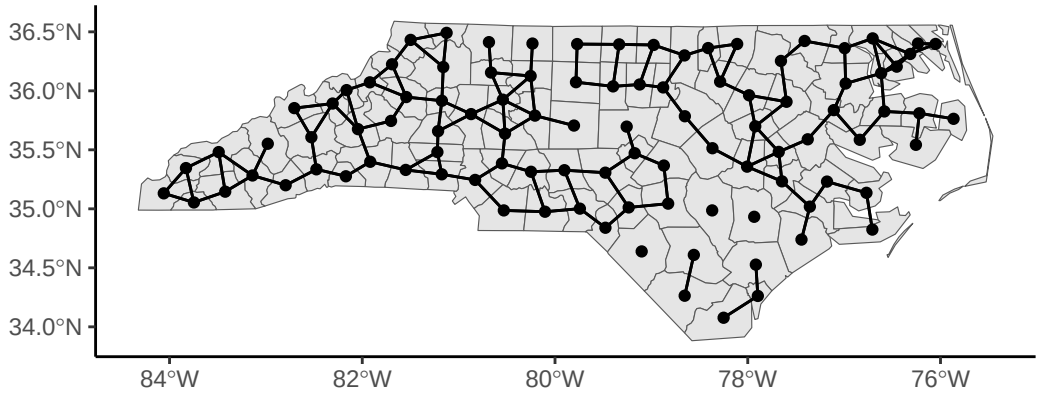

```
nc %>%
ggplot() +
  geom_sf() +
  geom_sf(data = centroids) +
  geom_sf(data = nb_R_net) +
  theme_classic()
```




```
nc %>%
ggplot() +
  geom_sf() +
  geom_sf(data = centroids) +
  geom_sf(data = nb_KNN_net) +
  theme_classic()
```




```
nc %>%
ggplot() +
  geom_sf() +
  geom_sf(data = centroids) +
  geom_sf(data = nb_KNNDist_net) +
  theme_classic()
```



Typically, we codify this neighborhood structure with a spatial proximity or weighting matrix, W. This $n \times n$ matrix has values, w_{ij}, between 0 and 1 that reflect whether or not the i area is a neighbor of the j area (0: not neighbor) and the strength of the influence of i on j (>0 if there is an influence or 1 if neighbor). These are called spatial weights in ArcGIS. We could restrict ourselves to binary values of w_{ij} such that 1 indicates a neighbor and 0 indicates not a neighbor.

Note: These relationships are often symmetric but not always. For example, an urban county may influence a nearby rural one more than vice versa. To incorporate this data context, we'd have to update this W matrix manually.

Considering the approaches above, Queen and Rook should give a symmetric W, but Nearest Neighbors may not give a symmetric W based on the algorithm of defining neighbors.

```
# Neighbor Information to Spatial Proximity or Weighting Matrix
## style = 'B' forces W to be a binary matrix
## style = 'W' standardizes the rows (taking into account how many neighbors an area has)
## zero.policy = TRUE allows for polygons with no neighbors

W <- nb2mat(Queen, style='B', zero.policy = TRUE)

W[1:10,1:10]
```

```
      37009 37005 37171 37053 37131 37091 37029 37073 37185 37169
37009     0     1     0     0     0     0     0     0     0     0
37005     1     0     1     0     0     0     0     0     0     0
37171     0     1     0     0     0     0     0     0     0     1
37053     0     0     0     0     0     0     1     0     0     0
37131     0     0     0     0     0     1     0     0     1     0
37091     0     0     0     0     1     0     0     1     0     0
37029     0     0     0     1     0     0     0     1     0     0
37073     0     0     0     0     0     1     1     0     0     0
37185     0     0     0     0     1     0     0     0     0     0
37169     0     0     1     0     0     0     0     0     0     0
```

7.6.3 Neighborhood-based Correlation

Now that we've defined a neighborhood structure using W, we can estimate a neighborhood-based correlation called Moran's I. This is a measure of spatial autocorrelation using the information in W. We define Moran's I as

$$I = \frac{n \sum_i \sum_j w_{ij}(Y_i - \bar{Y})(Y_j - \bar{Y})}{\sum_{i,j} w_{ij} \sum_i (Y_i - \bar{Y})^2}$$

Under H_0: Y_i are independent and identically distributed, then $\frac{I+1/(n-1)}{\sqrt{Var(I)}} \to N(0,1)$

- Test heavily depends on the form of W
- Observations may not be independent or identically distributed due to a spatial trend or non-constant variance, so we'll need to deal with that first.
- There is a lack of consensus about how areas with no neighbors should be treated in this context.

```
spdep::moran.test(nc$SID79, nb2listw(Queen, style='B'), randomisation=FALSE, alternative = '
```

```
    Moran I test under normality

data:  nc$SID79
weights: nb2listw(Queen, style = "B")

Moran I statistic standard deviate = 1.9892, p-value = 0.04668
alternative hypothesis: two.sided
sample estimates:
Moran I statistic       Expectation          Variance
      0.113074290      -0.010101010       0.003834515
```

```
spdep::moran.test(nc$SID79, nb2listw(Queen, style='B'), alternative = 'two.sided') # Using ra
```

```
    Moran I test under randomisation

data:  nc$SID79
weights: nb2listw(Queen, style = "B")

Moran I statistic standard deviate = 2.065, p-value = 0.03892
alternative hypothesis: two.sided
sample estimates:
Moran I statistic       Expectation          Variance
      0.113074290      -0.010101010       0.003557876
```

You can calculate a local version of Moran's I. For each region i,

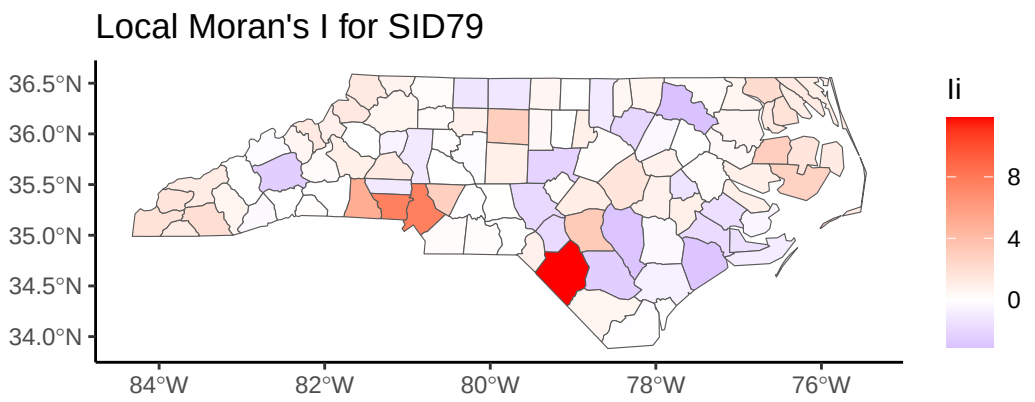
$$I_i = \frac{n(Y_i - \bar{Y}) \sum_j w_{ij}(Y_j - \bar{Y})}{\sum_j (Y_j - \bar{Y})^2}$$

such that the global version is proportional to the sum of the local Moran's I values:

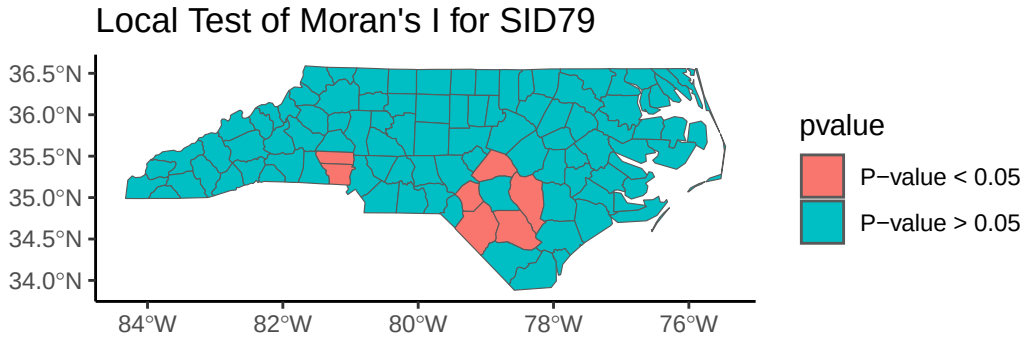
$$I = \frac{1}{\sum_{i \neq j} w_{ij}} \sum_i I_i$$

```
local_i <- spdep::localmoran(nc$SID79, nb2listw(Queen, style='B'), alternative = 'two.sided')

nc %>% bind_cols(local_i) %>%
  ggplot() +
  geom_sf(aes(fill = Ii)) +
```

```
  scale_fill_gradient2(mid = 'white', high= 'red', low ='blue') +
  labs(title='Local Moran\'s I for SID79') +
  theme_classic()
```




```
nc %>% bind_cols(local_i) %>%
  mutate(pvalue = cut(`Pr(z != E(Ii))`,breaks=c(-Inf,0.05,Inf), labels = c('P-value < 0.05',
  ggplot() +
  geom_sf(aes(fill = pvalue)) +
  labs(title='Local Test of Moran\'s I for SID79') +
  theme_classic()
```



7.6.4 Spatial Models

To account for spatial trends and spatial autocorrelation, we follow a series of steps:

1. Account for deterministic factors: Fit a regression model to model the mean with OLS, then create a visual map of the residuals

$$Y_i = x_i^T \beta + \epsilon_i$$

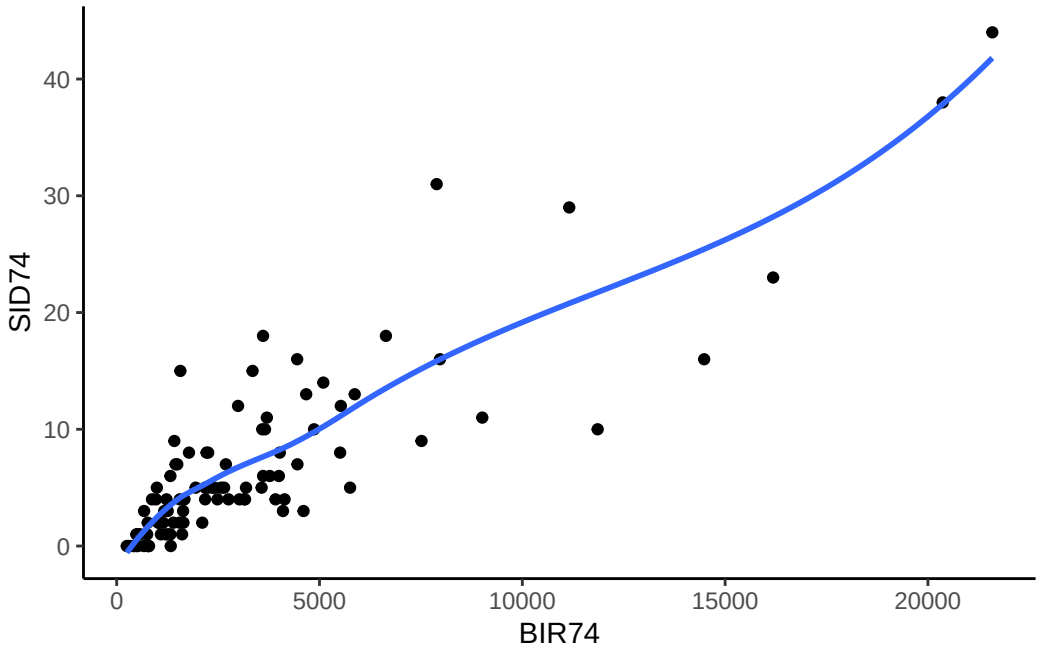
where i indexes spatial region

2. Test the residuals after detrending with Moran's I
3. If there is spatial autocorrelation as measured by Moran's I, fit an autoregressive model.

For example, if we want to predict the SID county rate in North Carolina as a function of the birth rate, we could fit a linear model first, then map the residuals and estimate the spatial autocorrelation of the residuals.

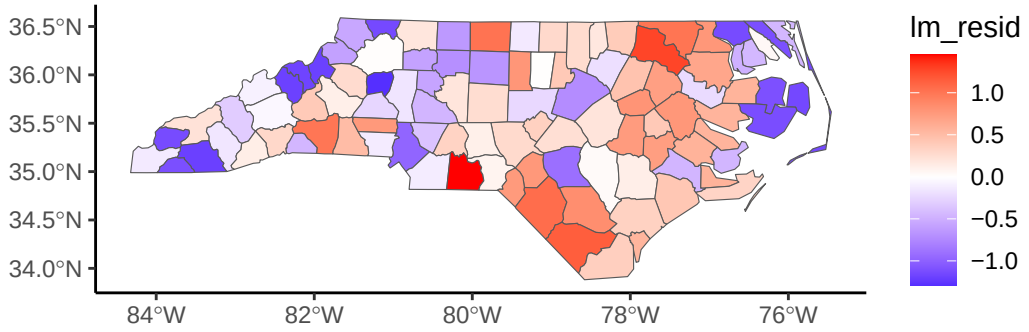
```
nc %>%
  ggplot(aes(y = SID74, x = BIR74)) +
  geom_point() + geom_smooth(se=FALSE) + theme_classic()
```

```
`geom_smooth()` using method = 'loess' and formula = 'y ~ x'
```




```
nc_lm <- lm(formula = log(SID74+1) ~ BIR74, data = nc)
nc$lm_resid <- resid(nc_lm)

nc %>%
  ggplot()+
```

```
  geom_sf(aes(fill = lm_resid)) +
  scale_fill_gradient2(mid = 'white', high= 'red', low ='blue') +
  theme_classic()
```




```
#Examine Extreme Residuals (higher or lower than what we'd expect)
nc %>%
  filter(abs(scale(lm_resid)) > 2)
```

```
Warning: Using one column matrices in `filter()` was deprecated in dplyr 1.1.0.
i Please use one dimensional logical vectors instead.

Simple feature collection with 1 feature and 15 fields
Geometry type: MULTIPOLYGON
Dimension:     XY
Bounding box:  xmin: -80.32528 ymin: 34.80792 xmax: -79.85371 ymax: 35.20452
Geodetic CRS:  +proj=longlat +datum=NAD27
       AREA PERIMETER CNTY_ CNTY_ID  NAME  FIPS FIPSNO CRESS_ID BIR74 SID74
37007 0.138     1.621  2096    2096 Anson 37007  37007        4  1570    15
      NWBIR74 BIR79 SID79 NWBIR79                       geometry lm_resid
37007     952  1875     4    1161 MULTIPOLYGON (((-79.91995 3... 1.448938
```

```
spdep::moran.test(nc$lm_resid, nb2listw(Queen, style='B'), alternative = 'two.sided') # Using
```

```
    Moran I test under randomisation

data:  nc$lm_resid
weights: nb2listw(Queen, style = "B")

Moran I statistic standard deviate = 4.0515, p-value = 5.089e-05
```

```
alternative hypothesis: two.sided
sample estimates:
Moran I statistic       Expectation          Variance
      0.241674505      -0.010101010       0.003861889
```

7.6.4.1 Spatial Autoregressive Models

Simultaneous Autoregressive Model (SAR)

We build autoregressive models for the data based on the proximity matrix W,

$$\mathbf{Y} = \lambda \mathbf{W}\mathbf{Y} + \mathbf{X}\beta + \epsilon, \qquad \epsilon \sim N(0, \sigma^2 I)$$

and typically, we assume Y are Gaussian if the outcome is continuous and use maximum likelihood estimation to estimate the parameters of this model.

This can be rewritten as

$$\mathbf{Y} \sim N(\mathbf{X}\beta, \sigma^2[(I - \lambda \mathbf{W})^T (I - \lambda \mathbf{W})]^{-1})$$

where the proximity matrix $\mathbf{W}$ should be weighted so that the rows sum to 1 ('style = 'W").

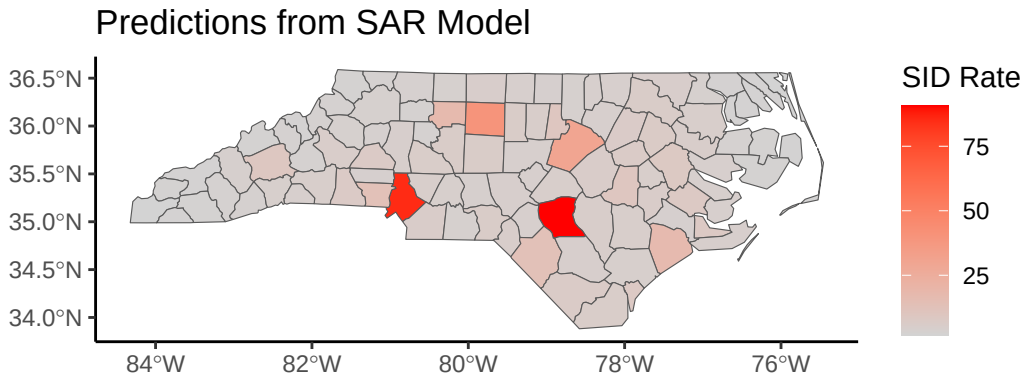
```
library(spatialreg)
```

```
Loading required package: Matrix


Attaching package: 'Matrix'

The following objects are masked from 'package:tidyr':

    expand, pack, unpack


Attaching package: 'spatialreg'

The following objects are masked from 'package:spdep':

    get.ClusterOption, get.coresOption, get.mcOption,
    get.VerboseOption, get.ZeroPolicyOption, set.ClusterOption,
    set.coresOption, set.mcOption, set.VerboseOption,
    set.ZeroPolicyOption
```

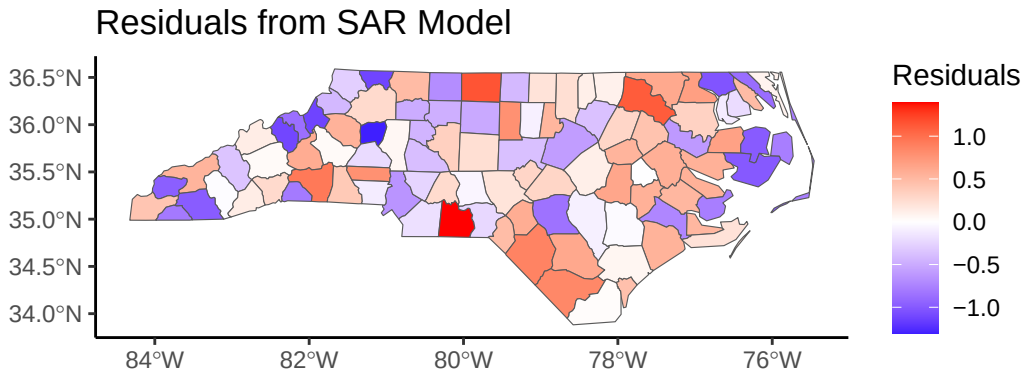
```r
# Convert Neighborhood Information to List (with weighting so that rows sum to 1)
listW <- nb2listw(Queen, style = 'W')

# Fit SAR Model
nc_sar <- spautolm(formula = log(SID74+1) ~ BIR74, data = nc, listw = listW, family = "SAR")

summary(nc_sar)
```

```
Call: spautolm(formula = log(SID74 + 1) ~ BIR74, data = nc, listw = listW,
    family = "SAR")

Residuals:
      Min        1Q    Median        3Q       Max
-1.298005 -0.429960  0.077978  0.492145  1.386400

Coefficients:
              Estimate Std. Error z value  Pr(>|z|)
(Intercept) 1.0882e+00 1.2639e-01  8.6097 < 2.2e-16
BIR74       1.5451e-04 1.6083e-05  9.6069 < 2.2e-16

Lambda: 0.48703 LR test value: 16.581 p-value: 4.661e-05
Numerical Hessian standard error of lambda: 0.10493

Log likelihood: -92.78273
ML residual variance (sigma squared): 0.3529, (sigma: 0.59406)
Number of observations: 100
Number of parameters estimated: 4
AIC: 193.57
```

```r
nc$sar_resid <- resid(nc_sar)
nc$sar_pred <- exp(fitted(nc_sar))

nc %>%
  ggplot()+
  geom_sf(aes(fill = sar_pred)) +
  labs(title='Predictions from SAR Model',fill = 'SID Rate') +
  scale_fill_gradient(low = 'lightgrey', high='red')+
  theme_classic()
```




```
nc %>%
  ggplot()+
  geom_sf(aes(fill = sar_resid)) +
  labs(title='Residuals from SAR Model',fill = 'Residuals') +
  scale_fill_gradient2(mid = 'white', high= 'red', low ='blue')+
  theme_classic()
```




```
#RMSE
sqrt(mean(nc$sar_resid^2))
```

```
[1] 0.5940562
```

```
#Extreme Residuals (higher or lower than what we'd expect)
nc %>%
  filter(abs(scale(sar_resid)) > 3)
```

```
Simple feature collection with 0 features and 17 fields
Bounding box:  xmin: NA ymin: NA xmax: NA ymax: NA
```

```
Geodetic CRS:  +proj=longlat +datum=NAD27
 [1] AREA      PERIMETER CNTY_     CNTY_ID   NAME      FIPS      FIPSNO
 [8] CRESS_ID  BIR74     SID74     NWBIR74   BIR79     SID79     NWBIR79
[15] geometry  lm_resid  sar_resid sar_pred
<0 rows> (or 0-length row.names)
```

```
# Double check the residuals after the SAR model are independent
spdep::moran.test(nc$sar_resid, nb2listw(Queen, style='W'), alternative = 'two.sided') # Usi
```

```
    Moran I test under randomisation

data:  nc$sar_resid
weights: nb2listw(Queen, style = "W")

Moran I statistic standard deviate = -0.40597, p-value = 0.6848
alternative hypothesis: two.sided
sample estimates:
Moran I statistic       Expectation          Variance
     -0.036666245      -0.010101010       0.004281978
```

Conditional Autoregressive Model (CAR)

An alternative but similar model says that

$$E(Y_i|Y_j, j \neq i) = x_i^T\beta + \sum_{j=1}^{n} c_{ij}(Y_j - x_j^T\beta)$$

$$Var(Y_i|Y_j, j \neq i) = \tau_i^2 = \tau^2/w_{i.}$$

where $w_{i.} = \sum_{j=1}^{n} w_{ij}$, c_{ij} is nonzero only if $Y_j \in N_i$, where N_i is the neighborhood of Y_i. c_{ij} is typically $\lambda w_{ij}/w_{i.}$.

This can be rewritten as

$$\mathbf{Y} \sim N(\mathbf{X}\beta, \sigma^2(I - \lambda\mathbf{W})^{-1})$$

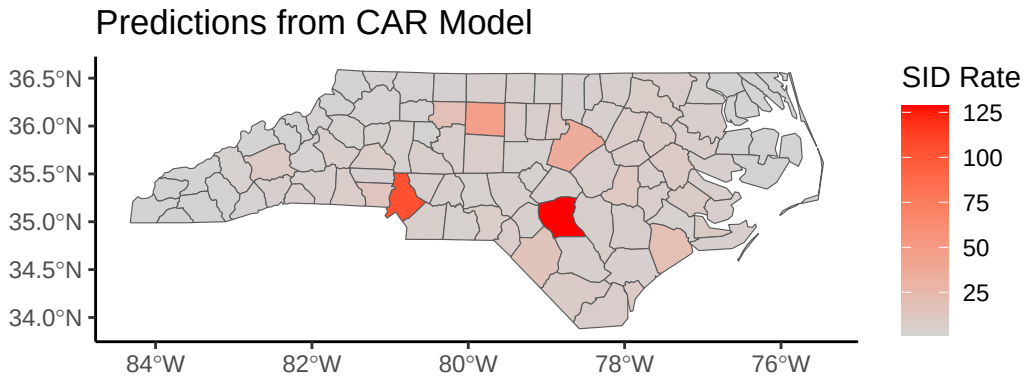
```
listW = nb2listw(Queen, style = 'W')
nc_car = spautolm(formula = log(SID74+1) ~ BIR74, data = nc, listw = listW, family = "CAR")
```

```
Warning in spautolm(formula = log(SID74 + 1) ~ BIR74, data = nc, listw = listW,
: Non-symmetric spatial weights in CAR model
```

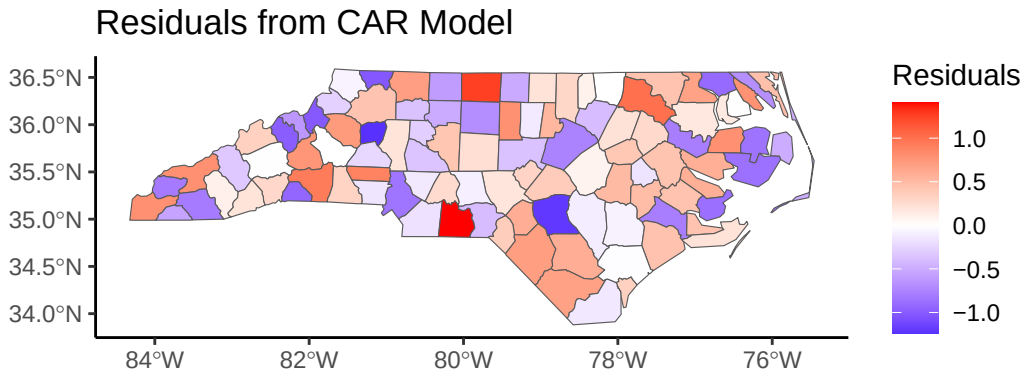
```
summary(nc_car)
```

```
Call: spautolm(formula = log(SID74 + 1) ~ BIR74, data = nc, listw = listW,
    family = "CAR")

Residuals:
     Min       1Q   Median       3Q      Max
-1.23287 -0.39827  0.10560  0.43790  1.40183

Coefficients:
              Estimate Std. Error z value  Pr(>|z|)
(Intercept) 8.9713e-01 1.3265e-01  6.7634 1.348e-11
BIR74       1.6749e-04 1.6149e-05 10.3716 < 2.2e-16

Lambda: 0.77612 LR test value: 16.605 p-value: 4.6027e-05
Numerical Hessian standard error of lambda: 0.34138

Log likelihood: -92.77079
ML residual variance (sigma squared): 0.34097, (sigma: 0.58393)
Number of observations: 100
Number of parameters estimated: 4
AIC: 193.54
```

```
nc$car_resid <- resid(nc_car)
nc$car_pred <- exp(fitted(nc_car))

nc %>%
  ggplot()+
  geom_sf(aes(fill = car_pred)) +
  labs(title='Predictions from CAR Model',fill = 'SID Rate') +
  scale_fill_gradient(low = 'lightgrey', high='red')+
  theme_classic()
```




```
nc %>%
  ggplot()+
  geom_sf(aes(fill = car_resid)) +
  labs(title='Residuals from CAR Model',fill = 'Residuals') +
  scale_fill_gradient2(mid = 'white', high= 'red', low ='blue')+
  theme_classic()
```




```
#RMSE
sqrt(mean(nc$car_resid^2))
```

```
[1] 0.5882464
```

```
#Extreme Residuals (higher or lower than what we'd expect)
nc %>%
  filter(abs(scale(car_resid)) > 3)
```

```
Simple feature collection with 0 features and 19 fields
Bounding box:  xmin: NA ymin: NA xmax: NA ymax: NA
```

```
Geodetic CRS:  +proj=longlat +datum=NAD27
 [1] AREA      PERIMETER CNTY_     CNTY_ID   NAME      FIPS      FIPSNO
 [8] CRESS_ID  BIR74     SID74     NWBIR74   BIR79     SID79     NWBIR79
[15] geometry  lm_resid  sar_resid sar_pred  car_resid car_pred
<0 rows> (or 0-length row.names)
```

```
# Double check the residuals after the SAR model are independent
spdep::moran.test(nc$car_resid, nb2listw(Queen, style='W'), alternative = 'two.sided') # Usi
```

```
    Moran I test under randomisation

data:  nc$car_resid
weights: nb2listw(Queen, style = "W")

Moran I statistic standard deviate = -3.2381, p-value = 0.001203
alternative hypothesis: two.sided
sample estimates:
Moran I statistic       Expectation          Variance
     -0.222000699      -0.010101010       0.004282266
```

Typically, the CAR model and its variations (iCAR, BYM, etc.) are fit in a Bayesian context. The details of the Bayesian estimation are beyond the scope of this course. See `CARBayes` package for more details.

Note: If you have count data within an area, transform it with a `log()` and model with a Gaussian model.

Reference about SAR and CAR Models and Covariance: https://doi.org/10.1016/j.spasta.2018.04.006

See the `CARBayes` package for more information.

7.6.4.2 Spatial Mixed Effects Models

While the SAR and CAR models are similar to autoregressive models for time series, we could also allow coefficients to differ across space, similar to a mixed effects model that we used for longitudinal data.

$$\mathbf{Y} = \mathbf{X}\beta + \mathbf{b} + \epsilon, \qquad \epsilon \sim N(0, \sigma^2 I)$$

where the random effects $\mathbf{b} \sim N(0, G)$ and G can be assumed to be a covariance matrix based on spatial correlation such as the Matern correlation structure.

```r
library(spaMM)
```

```
Registered S3 methods overwritten by 'registry':
  method               from
  print.registry_field proxy
  print.registry_entry proxy

spaMM (Rousset & Ferdy, 2014, version 4.6.1) is loaded.
Type 'help(spaMM)' for a short introduction,
'news(package='spaMM')' for news,
and 'citation('spaMM')' for proper citation.
Further infos, slides, etc. at https://gitlab.mbb.univ-montp2.fr/francois/spamm-ref.


Attaching package: 'spaMM'

The following objects are masked from 'package:spatstat.model':

    Poisson, pseudoR2, response
```

```r
nc2 <- cbind(nc, st_coordinates(st_centroid(nc))) %>%
  rename(x = X, y = Y) %>% # Rename coordinates to x and y for spaMM
  as.data.frame()
```

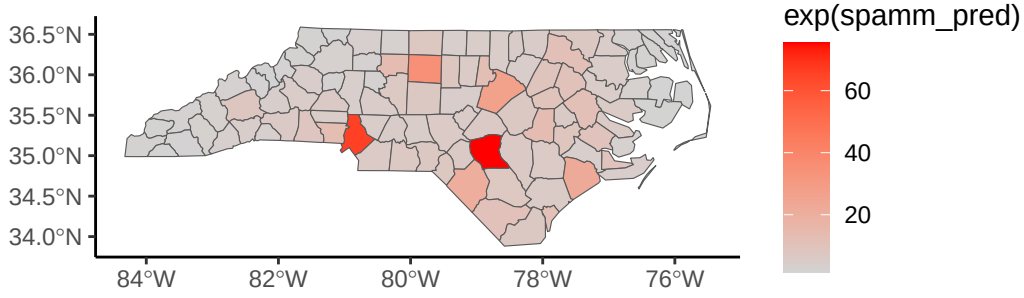
```
Warning: st_centroid assumes attributes are constant over geometries
```

```r
spamm <- fitme(log(SID74+1) ~ BIR74 + Matern(1 | x+y), data = nc2, fixed = list(nu = 0.5))

summary(spamm)
```

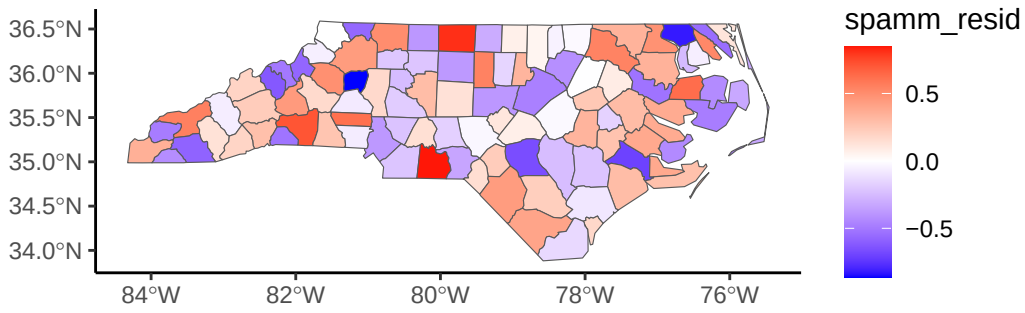
```
formula: log(SID74 + 1) ~ BIR74 + Matern(1 | x + y)
ML: Estimation of corrPars, lambda and phi by ML.
    Estimation of fixed effects by ML.
Estimation of lambda and phi by 'outer' ML, maximizing logL.
family: gaussian( link = identity )
 ------------ Fixed effects (beta) ------------
             Estimate  Cond. SE t-value
(Intercept) 0.9579409 0.2921880   3.279
BIR74       0.0001473 0.0000156   9.446
```

```
 --------------- Random effects ---------------
Family: gaussian( link = identity )
                    --- Correlation parameters:
     1.nu     1.rho
0.5000000 0.6325215
           --- Variance parameters ('lambda'):
lambda = var(u) for u ~ Gaussian;
   x + y  :  0.3342
# of obs: 100; # of groups: x + y, 100
 -------------- Residual variance  ------------
phi estimate was 0.214876
 ------------- Likelihood values  -------------
                         logLik
logL       (p_v(h)): -89.61781
```

```
spamm.map <- cbind(nc, spamm_pred = predict(spamm)) %>%
  mutate(spamm_resid = log(SID74+1) - spamm_pred)


spamm.map %>%
  ggplot(aes(fill = exp(spamm_pred))) +
  geom_sf() +
  scale_fill_gradient(low = 'lightgrey', high='red')+
  theme_classic()
```




```
spamm.map %>%
  ggplot(aes(fill = spamm_resid)) +
  geom_sf() +
  scale_fill_gradient2(mid = 'white', high= 'red', low ='blue') +
  theme_classic()
```




```
## RMSE
sqrt(mean(spamm.map$spamm_resid^2))
```

```
[1] 0.3854601
```

```
# Extreme Residuals (higher or lower than what we'd expect)
spamm.map %>%
  filter(abs(scale(spamm_resid)) > 3)
```

```
Simple feature collection with 0 features and 21 fields
Bounding box:  xmin: NA ymin: NA xmax: NA ymax: NA
Geodetic CRS:  +proj=longlat +datum=NAD27
 [1] AREA        PERIMETER   CNTY_       CNTY_ID     NAME        FIPS
 [7] FIPSNO      CRESS_ID    BIR74       SID74       NWBIR74     BIR79
[13] SID79       NWBIR79     lm_resid    sar_resid   sar_pred    car_resid
[19] car_pred    spamm_pred  geometry    spamm_resid
<0 rows> (or 0-length row.names)
```

```
# Double check the residuals after the  model are independent
spdep::moran.test(spamm.map$spamm_resid, nb2listw(Queen, style='W'), alternative = 'two.sided')
```

```
    Moran I test under randomisation

data:  spamm.map$spamm_resid
weights: nb2listw(Queen, style = "W")

Moran I statistic standard deviate = -2.3794, p-value = 0.01734
alternative hypothesis: two.sided
sample estimates:
Moran I statistic       Expectation          Variance
     -0.165848405      -0.010101010       0.004284663
```

7.6.4.3 Geographically (Spatially) Weighted Regression

Lastly, we present a non-parametric approach to spatial correlation called Geographically Weighted Regression (GWR). The general model is

$$Y_i = \beta_{i0} + \sum_{k=1}^{p} \beta_{ik} x_{ik} + \epsilon_i$$

where Y_i is the outcome at location i, x_{ik} is the value of the kth explanatory variable at location i, and β_{ik} is a local regression coefficient for the kth explanatory variable.

GWR allows the coefficients to vary continuously over a study region, and a set of coefficients can be estimated at any location. This is a non-parametric approach in that we do not specify the structure of how the coefficients vary, but rather use a "local regression" approach in 2 dimensions.

Each set of regression coefficients is estimated by weighted least squares (weighting points closer in space than those further away),

$$\widehat{\beta}_i = (\mathbf{X}^T \mathbf{W}_i \mathbf{X})^{-1} \mathbf{X}^T \mathbf{W}_i \mathbf{Y}$$

where $\mathbf{W}_i$ is a diagonal matrix denoting the geographical weighting of each observed data for regression point i. The weighting is calculated with a chosen kernel function based on proximities between the location i and the location of the other data points. One kernel function that could be used is the Gaussian kernel, such that

$$w_{ij} = \exp\left[\frac{-1}{2}\left(\frac{d_{ij}}{b}\right)^2\right]$$

where d_{ij} is the distance between observation point j and regression point i and b is the kernel bandwidth. The optimal bandwidth can be chosen using cross-validation or a goodness-of-fit measure such as AIC or BIC.

```
library(spgwr)
```

```
NOTE: This package does not constitute approval of GWR
as a method of spatial analysis; see example(gwr)
```

```
GWRbandwidth <- gwr.sel(log(SID74+1) ~ BIR74, data = nc2, coords = as.matrix(nc2[,c('x','y')]
```

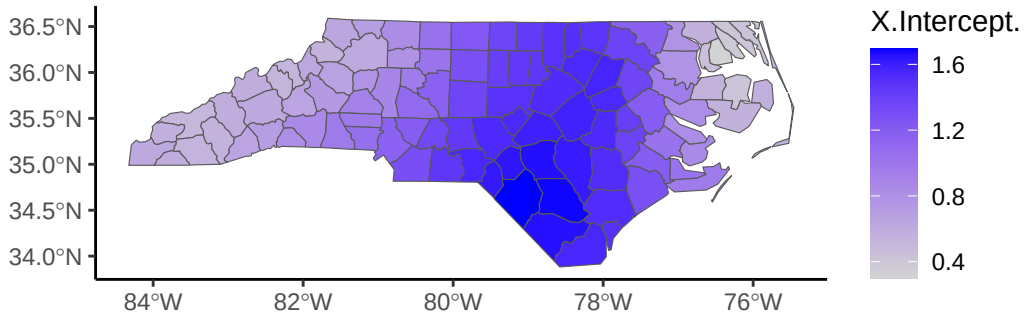
```
Adaptive q: 0.381966 CV score: 40.10318
Adaptive q: 0.618034 CV score: 43.03422
Adaptive q: 0.236068 CV score: 36.54116
Adaptive q: 0.145898 CV score: 32.79307
Adaptive q: 0.09016994 CV score: 30.15781
Adaptive q: 0.05572809 CV score: 30.70451
Adaptive q: 0.08427944 CV score: 30.29308
Adaptive q: 0.1114562 CV score: 30.87369
Adaptive q: 0.09830056 CV score: 30.27686
Adaptive q: 0.0915056 CV score: 30.16935
Adaptive q: 0.08791997 CV score: 30.16358
Adaptive q: 0.08945536 CV score: 30.1547
Adaptive q: 0.08932905 CV score: 30.15461
Adaptive q: 0.08928836 CV score: 30.15461
Adaptive q: 0.08936974 CV score: 30.15462
Adaptive q: 0.08932905 CV score: 30.15461
```

```
gwr.model <- gwr(log(SID74+1) ~ BIR74, data = nc2,
                 adapt=GWRbandwidth,
                 coords = as.matrix(nc2[,c('x','y')]),
                 hatmatrix=TRUE,
                 se.fit=TRUE)

gwr.model
```

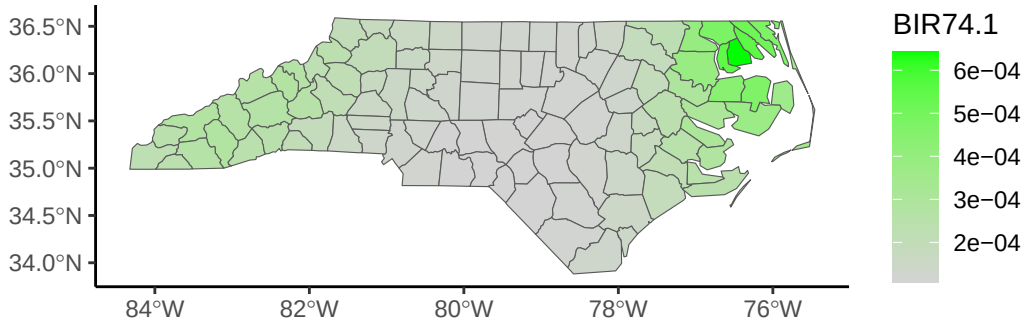
```
Call:
gwr(formula = log(SID74 + 1) ~ BIR74, data = nc2, coords = as.matrix(nc2[,
    c("x", "y")]), adapt = GWRbandwidth, hatmatrix = TRUE, se.fit = TRUE)
Kernel function: gwr.Gauss
Adaptive quantile: 0.08932905 (about 8 of 100 data points)
Summary of GWR coefficient estimates at data points:
                   Min.    1st Qu.     Median    3rd Qu.       Max. Global
X.Intercept. 0.29756084 0.64874576 1.04875034 1.46053450 1.69422536 1.0530
BIR74        0.00010846 0.00012462 0.00016401 0.00024836 0.00064206 0.0002
Number of data points: 100
Effective number of parameters (residual: 2traceS - traceS'S): 15.4748
Effective degrees of freedom (residual: 2traceS - traceS'S): 84.5252
Sigma (residual: 2traceS - traceS'S): 0.53242
Effective number of parameters (model: traceS): 11.33877
Effective degrees of freedom (model: traceS): 88.66123
Sigma (model: traceS): 0.519853
Sigma (ML): 0.4894941
```

```
AICc (GWR p. 61, eq 2.33; p. 96, eq. 4.21): 169.387
AIC (GWR p. 96, eq. 4.22): 152.2499
Residual sum of squares: 23.96045
Quasi-global R2: 0.7270107
```

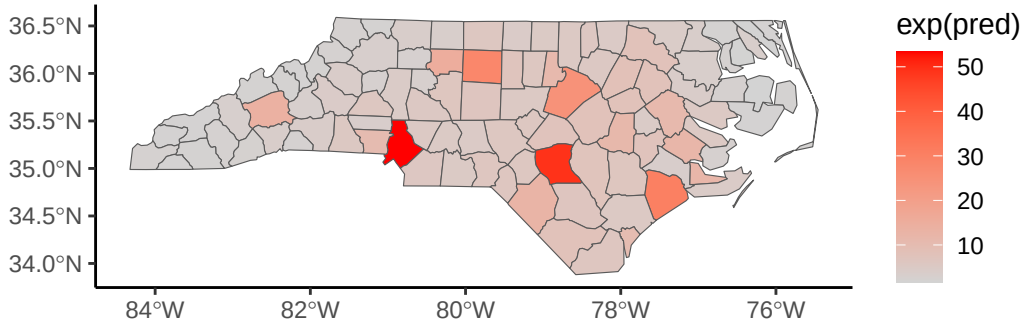
```r
gwr.map <- cbind(nc, as.matrix(as.data.frame(gwr.model$SDF))) %>%
  mutate(gwr_resid = log(SID74+1) - pred)

#map the coef, map the r2, map the residuals
gwr.map %>%
  ggplot(aes(fill = X.Intercept.)) +
  geom_sf() +
  scale_fill_gradient(low = 'lightgrey', high='blue')+
  theme_classic()
```



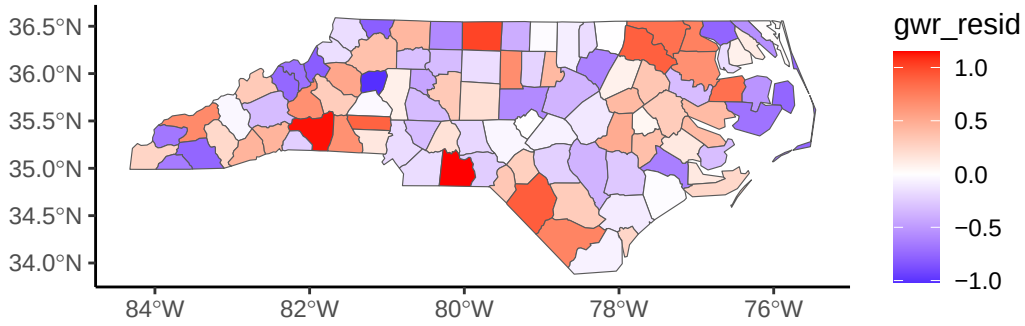

```r
gwr.map %>%
  ggplot(aes(fill = BIR74.1)) +
  geom_sf() +
  scale_fill_gradient(low = 'lightgrey', high='green')+
  theme_classic()
```



```
gwr.map %>%
  ggplot(aes(fill = exp(pred))) +
  scale_fill_gradient(low = 'lightgrey', high='red')+
  geom_sf() + theme_classic()
```



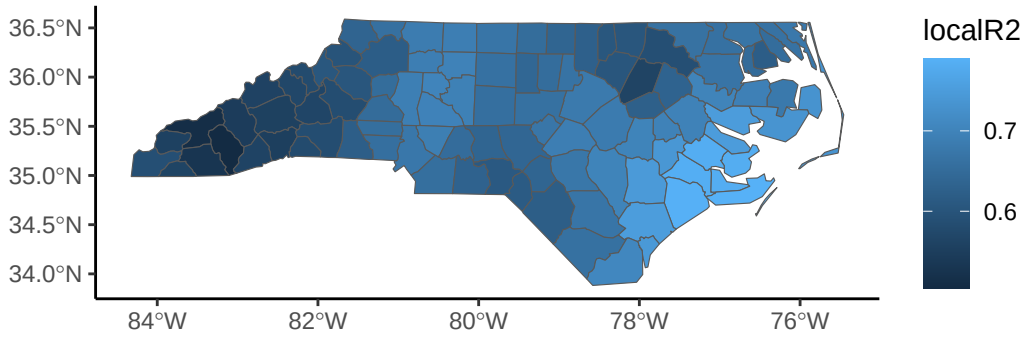

```
gwr.map %>%
  ggplot(aes(fill = gwr_resid)) +
  geom_sf() +
  scale_fill_gradient2(mid = 'white', high= 'red', low ='blue') +
  theme_classic()
```




```
## RMSE
sqrt(mean(gwr.map$gwr_resid^2))
```

```
[1] 0.4894941
```

```
gwr.map %>%
  ggplot(aes(fill = localR2)) +
  geom_sf() + theme_classic()
```




```
# Double check the residuals after the  model are independent
spdep::moran.test(gwr.map$gwr_resid, nb2listw(Queen, style='W'), alternative = 'two.sided') #
```

```
    Moran I test under randomisation

data:  gwr.map$gwr_resid
weights: nb2listw(Queen, style = "W")

Moran I statistic standard deviate = 0.53231, p-value = 0.5945
alternative hypothesis: two.sided
sample estimates:
Moran I statistic       Expectation          Variance
      0.024710153      -0.010101010       0.004276652
```

7.6.5 Meaningful Distances

While it may be the easiest to define spatial distance using the Euclidean distance between centroids of the areal boundaries, that may not be the most meaningful.

Two locations that are "close" geographically might be quite different due to other environmental factors such as physical factors (e.g., rivers, road infrastructure, and associated conditions of accessibility), socio-economic factors (e.g., preferences for hospitals, schools, and stores), and administrative geographies.

If you study outcomes related to transportation, river/stream networks or some complex terrain conditions, typical distance metrics may fail to reflect true spatial proximity. Instead, others should be considered, such as road network distance, travel time, water-based distance (along a river or stream or coastline), or landscape-based (i.e. complex terrain) distance.

References

Chatfield, Chris, and Haipeng Xing. 2019. *The Analysis of Time Series: An Introduction with r*. Chapman; Hall/CRC.

Cryer, Jonathan D, and Kung-Sik Chan. 2008. *Time Series Analysis: With Applications in r*. Springer.

Shumway, Robert H, and David S Stoffer. 2019. *Time Series: A Data Analysis Approach Using r*. Chapman; Hall/CRC.

Stoffer, David. 2025. *Astsa: Applied Statistical Time Series Analysis*. https://doi.org/10.32614/CRAN.package.astsa.

A Matrix Algebra

In this class, we will be using matrices to help us organize data and information. Thus, it is useful to know some basic definitions and properties of matrices. Linear algebra is not a prerequisite for this course, so if this is new to you, don't fret!

Use the this mathematical section as a reference as we go throughout the course. I've purposefully put more here than we might need. If I refer to a property or term that you are unfamiliar with, come back to check this section first.

A.1 Matrix & Vector Addition

If $\mathbf{A}$ is a $(r \times c)$ matrix and $\mathbf{B}$ is a $(r \times c)$ matrix (Note: nrow(A) = nrow(B) and ncol(A) = ncol(B)!), then $\mathbf{A} + \mathbf{B}$ is a $(r \times c)$ matrix with (i, j)th element

$$(\mathbf{A} + \mathbf{B})_{ij} = a_{ij} + b_{ij}$$

```
A = matrix(c(1,2,3,4),nrow=2,ncol=2)
B = matrix(c(5,6,7,8),nrow=2,ncol=2)

A + B #element wise addition
```

```
     [,1] [,2]
[1,]    6   10
[2,]    8   12
```

A.1.1 Properties

Here are some useful properties of matrices:

1. Commutative Property

$$\mathbf{A} + \mathbf{B} = \mathbf{B} + \mathbf{A}$$

2. Associative Property

$$\mathbf{A} + (\mathbf{B} + \mathbf{C}) = (\mathbf{A} + \mathbf{B}) + \mathbf{C}$$

3. Additive Identity

$$\mathbf{A} + \mathbf{0} = \mathbf{A}$$

where $\mathbf{0}$ is a $r \times c$ matrix with 0 elements.

4. Additive Inverse

- For any $r \times c$ matrix $\mathbf{A}$, there is a $r \times c$ matrix $\mathbf{B}$ $(= -\mathbf{A})$ such that

$$\mathbf{A} + \mathbf{B} = \mathbf{0}$$

A.2 Matrix & Vector Multiplication

If $\mathbf{A}$ is a $(r \times q)$ matrix and $\mathbf{B}$ is a $(q \times c)$ matrix (Note: ncol(A) = nrow(B)!), then $\mathbf{AB}$ is a $(r \times c)$ matrix with (i, j)th element

$$(\mathbf{AB})_{ij} = \sum_{k=1}^{q} a_{ik} b_{kj}$$

where a_{ik} is the element in the ith row and kth column of $\mathbf{A}$ and b_{kj} is the element in the kth row and jth column of $\mathbf{B}$.

Here, we say that $\mathbf{A}$ is postmultiplied by $\mathbf{B}$ and, equivalently, that $\mathbf{B}$ is premultiplied by $\mathbf{A}$. Here, the order matters!

```
A = matrix(c(1,2,3,4),nrow=2,ncol=2)
B = matrix(c(5,6,7,8),nrow=2,ncol=2)

A * B #element wise multiplication (not what you want....)
```

```
     [,1] [,2]
[1,]    5   21
[2,]   12   32
```

```
A %*% B #matrix multiplication
```

```
     [,1] [,2]
[1,]   23   31
[2,]   34   46
```

A.2.1 Properties

Here are some useful properties of matrices:

1. Associative Property

$$\mathbf{A}(\mathbf{BC}) = (\mathbf{AB})\mathbf{C}$$

2. Distributive Property

$$\mathbf{A}(\mathbf{B}+\mathbf{C}) = \mathbf{AB}+\mathbf{AC}, (\mathbf{A}+\mathbf{B})\mathbf{C} = \mathbf{AC}+\mathbf{BC}$$

3. Multiplicative Identity

$$\mathbf{I_r A} = \mathbf{A I_c} = \mathbf{A}$$

where $\mathbf{I_r}$ is a $r \times r$ matrix with 1's along the diagonal and 0's otherwise. Generally, we'll drop the subscript and we'll assume the dimension by its context.

A.3 Matrix Transpose

The **transpose** of any $(r \times c)$ $\mathbf{A}$ matrix is the $(c \times r)$ matrix denoted as $\mathbf{A}^T$ or $\mathbf{A}'$ such that a_{ij} is replaced by a_{ji} everywhere.

```
A
```

```
     [,1] [,2]
[1,]    1    3
[2,]    2    4
```

```
t(A)
```

```
     [,1] [,2]
[1,]    1    2
[2,]    3    4
```

A matrix is **square** if $c = r$. The diagonal of a square matrix are the elements of a_{ii} and the off-diagonal elements of a square matrix are a_{ij} where $i \neq j$.

If $\mathbf{A}$ is **symmetric**, then $\mathbf{A} = \mathbf{A}^T$.

For any matrix $\mathbf{A}$, $\mathbf{A}^T\mathbf{A}$ will be a square matrix.

A.3.1 Properties

The transpose of a matrix product is

$$(\mathbf{AB})^T = \mathbf{B}^T\mathbf{A}^T$$

The transpose of a matrix sum is

$$(\mathbf{A} + \mathbf{B})^T = \mathbf{A}^T + \mathbf{B}^T$$

A.4 Inner product

For two vectors $\mathbf{x}$ and $\mathbf{y}$, the standard **inner (dot) product** of the two vectors in $\mathbb{R}^k$ is

$$< \mathbf{x}, \mathbf{y} >= \mathbf{x}^T\mathbf{y} = \sum_{i=1}^{k} x_i y_i = x_1 y_1 + x_2 y_2 + \cdots + x_k y_k$$

To be an inner product, it has to satisfy 1) $< \mathbf{x}, \mathbf{x} >\geq 0$ with equality if $\mathbf{x} = \mathbf{0}$ 2) $< \mathbf{x}, \mathbf{y} >=< \mathbf{y}, \mathbf{x} >$ and 3) $< a\mathbf{x} + b\mathbf{y}, \mathbf{z} >= a < \mathbf{x}, \mathbf{z} > +b < \mathbf{y}, \mathbf{z} >$

```
x = 1:10
y = 21:30
x
```

```
 [1]  1  2  3  4  5  6  7  8  9 10
```

```
y
```

```
 [1] 21 22 23 24 25 26 27 28 29 30
```

```
class(x); class(y)
```

```
[1] "integer"
```

```
[1] "integer"
```

```
x %*% y #If x is a numeric vector (not a matrix), R doesn't differentiate between a column ve
```

```
     [,1]
[1,] 1485
```

```r
x = matrix(x,ncol=1) #If x is a column vector (matrix with 1 column), it matters.
y = matrix(y,ncol=1)

class(x); class(y)
```

```
[1] "matrix" "array"
```

```
[1] "matrix" "array"
```

```r
#x%*%y #Get an Error!

t(x)%*%y
```

```
     [,1]
[1,] 1485
```

A.5 Vector Difference

The **vector difference** between two p-dimensional vectors $\mathbf{x}$ and $\mathbf{y}$ is calculated element by element,

$$(\mathbf{x} - \mathbf{y})_i = x_i - y_i$$

A.6 Vector Length

For a vector $\mathbf{x} \in \mathbb{R}^k$, a general **vector norm** $||\mathbf{x}||_p$ for $p = 1, 2, ...$ is defined as

$$||\mathbf{x}||_p = \left(\sum_{i=1}^{m} |x_i|^p \right)^{1/p}$$

As $p \to \infty$, we have a special case,

$$||\mathbf{x}||_\infty = \max_i |x_i|$$

- L2-Norm: The most commonly used vector norm is when $p = 2$,

$$||\mathbf{x}||_2 = \sqrt{\sum_{i=1}^{m} (x_i)^2}$$

To be a norm, the following must be true, 1) $||\mathbf{x}|| \geq 0$, 2) $||a\mathbf{x}|| = |a|||\mathbf{x}||$, and 3) $||\mathbf{x}+\mathbf{y}|| \leq ||\mathbf{x}|| + ||\mathbf{y}||$ (triangle inequality).

```
sqrt(t(x)%*%x)
```

```
         [,1]
[1,] 19.62142
```

```
sqrt(sum(x^2))
```

```
[1] 19.62142
```

A.7 Vector Distance

- Euclidean distance: In Euclidean space, we often call the L2-norm a **Euclidean Norm**. This gives the length of the vector from the origin. More generally, the **Euclidean distance** between two vectors $\mathbf{x}$ and $\mathbf{y}$ is the L2-norm of the difference,

$$d(\mathbf{x},\mathbf{y}) = ||\mathbf{x}-\mathbf{y}||_2 = \sqrt{\sum_i (x_i - y_i)^2} = \sqrt{(\mathbf{x}-\mathbf{y})^T(\mathbf{x}-\mathbf{y})}$$

This is the distance "as the crow flies.''

```
sqrt(sum((x-y)^2))
```

```
[1] 63.24555
```

```
sqrt(t(x-y)%*%(x-y))
```

```
         [,1]
[1,] 63.24555
```

- Minkowski distance: A more general distance measurement between two vectors is based on the general definition of the vector norm called the **Minkowski distance**,

$$d(\mathbf{x},\mathbf{y}) = ||\mathbf{x}-\mathbf{y}||_p = \left(\sum_i |x_i - y_i|^p\right)^{1/p}$$

If $p = 2$, then you get the **Euclidean distance** (as the crow flies),

$$d(\mathbf{x}, \mathbf{y}) = \sqrt{\sum_i (x_i - y_i)^2}$$

If $p = 1$, then you get the **Manhattan distance** (city-block distance),

$$d(\mathbf{x}, \mathbf{y}) = \sum_i |x_i - y_i|$$

If $p \to \infty$, then you get **Chebyshev distance**,

$$d(\mathbf{x}, \mathbf{y}) = ||\mathbf{x} - \mathbf{y}||_\infty = \max_i |x_i - y_i|$$

- **Mahalanobis distance:** In the context of random variables and data analysis, imagine trying to find the distance between two individuals based on values of k different variables. We could find the differences in values amongst all of the variables between the two individuals. However, the scale and units of each might be different. We could standardized each difference by dividing by the standard deviation of the variable amongst the entire sample so that each difference is comparable. In order to incorporate the dependence (covariance) between the variables as well, the **Mahalanobis distance** is a modified Euclidean distance and is calculated as

  $$d(\mathbf{x}, \mathbf{y}) = \sqrt{(\mathbf{x} - \mathbf{y})^T \mathbf{S}^{-1} (\mathbf{x} - \mathbf{y})}$$

  where $\mathbf{S}$ is the $k \times k$ sample covariance matrix of the variables based on the sample.

A.8 Vector Space

A **vector space** is a set of vectors which is closed under vector addition and scalar multiplication. They must also adhere to typical axioms such as associativity, commutativity, etc (8 in all). We will focus on real vector spaces on $\mathbb{R}^k$.

The vector $\mathbf{y} = a_1\mathbf{x_1} + a_2\mathbf{x_2} + \cdots + a_k\mathbf{x_k}$ is a **linear combination** of the vectors $\mathbf{x_1}, \mathbf{x_2}, ..., \mathbf{x_k}$. The set of all linear combinations of $\mathbf{x_1}, \mathbf{x_2}, ..., \mathbf{x_k}$ is their **linear span**.

A set of vectors is said to be **linearly dependent** if one of the vectors in the set can be defined as a linear combination of the other vectors. In other words, if there exists k numbers $(a_1, a_2, ..., a_k)$, not all zero, such that

$$a_1\mathbf{x_1} + a_2\mathbf{x_2} + \cdots + a_k\mathbf{x_k} = \mathbf{0}$$

If no vector in the set can be written in this way such that $a_1, ..., a_k$ have to be 0, then the vectors are said to be **linearly independent**.

- **Theorem:** For a vector space V, a set of vectors are linearly dependent if and only if the matrix of the vectors is singular (see below for the definition of singularity).
- **Basis:** A basis is a set of vectors that spans the whole vector space (i.e. any vector in the vector space can be written as a linear combination of basis elements) and are linearly independent.

A.9 Rank of matrix

The **rank** of $\mathbf{A}$ is dimension of row space of $\mathbf{A}$ (space spanned by rows of $\mathbf{A}$) which equals the dimension of the column space of $\mathbf{A}$ (space spanned by columns of $\mathbf{A}$).

- The rank is the maximum number of linearly independent columns of a matrix.
- A matrix is **full rank** if the rank of the matrix is equal to the number of columns.

A.10 Singularity

A square matrix $\mathbf{A}$ is **nonsingular** if $\mathbf{Ax} = \mathbf{0}$ implies that $\mathbf{x} = \mathbf{0}$. If a matrix fails to be nonsingular, it is called **singular**. - A square matrix is nonsingular if its rank is equal to the number of rows or columns.

A.11 Determinant

The **determinant** of a square $k \times k$ matrix $\mathbf{A}$ is the scalar

$$|\mathbf{A}| = \sum_{i=1}^{k} a_{1j} |\mathbf{A}_{1j}| (-1)^{1+j}$$

where $\mathbf{A}_{1j}$ is the $(k-1) \times (k-1)$ matrix obtained by deleting the first row and the jth column of $\mathbf{A}$.

```
A
```

```
     [,1] [,2]
[1,]    1    3
[2,]    2    4
```

```
det(A)
```

```
[1] -2
```

- **Theorem:** The determinant of a matrix $\mathbf{A}$ is 0 if and only if $\mathbf{A}$ is singular.

A.12 Matrix Inverse

The matrix $\mathbf{B}$ such that $\mathbf{AB} = \mathbf{BA} = \mathbf{I}$ (where $\mathbf{I}$ is the identity matrix) is called the **inverse** of $\mathbf{A}$ and is denoted by $\mathbf{A}^{-1}$.

- If $\mathbf{A}$ is a nonsingular square matrix, then there is a unique matrix $\mathbf{B}$ such that

$$\mathbf{AB} = \mathbf{BA} = \mathbf{I}$$

  To show the existence of an inverse, it is also equivalent to show that the determinant of $\mathbf{A}$ is not zero.

- For square matrices $\mathbf{A}$ and $\mathbf{B}$ with the same dimension,

$$(\mathbf{A}^{-1})^T = (\mathbf{A}^T)^{-1}$$

$$(\mathbf{AB})^{-1} = \mathbf{B}^{-1}\mathbf{A}^{-1}$$

A.13 Trace of matrix for square matrix

The **trace** of a square matrix is the sum of the diagonal elements, $tr(\mathbf{A}) = \sum_j a_{jj}$.

A.13.1 Properties

$$tr(c\mathbf{A}) = c\, tr(\mathbf{A})$$

$$tr(\mathbf{A} \pm \mathbf{B}) = tr(\mathbf{A}) \pm tr(\mathbf{B})$$

$$tr(\mathbf{AB}) = tr(\mathbf{BA})$$

A.14 Vector Projection

The **projection of a vector y on a vector x** is

$$\frac{\mathbf{y}^T\mathbf{x}}{|\mathbf{x}|_2^2}\mathbf{x}$$

The **orthogonal projection of a vector y on a the column space of matrix X** gives you the $\hat{\mathbf{y}} = \mathbf{X}\beta$ that minimizes $||\mathbf{y} - \hat{\mathbf{y}}||$.

$$(\mathbf{X}^T\mathbf{X})^{-1}\mathbf{X}^T\mathbf{y}$$

A.15 Orthogonality

A square matrix is **orthogonal** if its rows, considered as vectors, are mutually perpendicular and have unit lengths; that is, $\mathbf{A}\mathbf{A}^T = \mathbf{I}$.

A.16 Eigenvalues and Eigenvectors

The scalars, λ, that satisfy the polynomial characteristic equation $|\mathbf{A} - \lambda\mathbf{I}| = 0$ are called **eigenvalues** of matrix **A**.

If **e** is a non-zero vector such that

$$\mathbf{A}\mathbf{e} = \lambda\mathbf{e}$$

then **e** is said to be an **eigenvector** of the matrix **A** associated with the eigenvalue λ.

A.17 Positive Definiteness

A **quadratic form** in k variables $x_1, ..., x_k$ is $\mathbf{x}^T\mathbf{A}\mathbf{x}$, where $\mathbf{x}^T = [x_1, ..., x_k]$ and **A** is a symmetric $k \times k$ matrix. It can be written as $\sum_{i=1}^{k}\sum_{j=1}^{k} a_{ij}x_ix_j$.

A symmetric matrix is **positive definite** if

$$\mathbf{x}^T\mathbf{A}\mathbf{x} > 0$$

for all vectors $\mathbf{x} \neq \mathbf{0}$. If a matrix is positive definite, then all of the eigenvalues will be positive.

A symmetric matrix is **nonnegative definite** if

$$\mathbf{x}^T\mathbf{A}\mathbf{x} \geq 0$$

for all vectors $\mathbf{x} \neq \mathbf{0}$.

Cholesky Decomposition: For a real-valued symmetric positive definite matrix $\mathbf{A}$, it can be decomposed as

$$\mathbf{A} = \mathbf{L}\mathbf{L}^T$$

where $\mathbf{L}$ is a lower triangular matrix with real and positive diagonal elements.

A.18 (Ordinary) Least Squares

A.18.1 Calculus Approach

Let y_i be the outcome for individual $i = 1, ..., n$ and $\mathbf{x}_i$ be a vector of explanatory variables for individual i, then when we fit a linear regression model, we want to find the slopes, $\beta = (\beta_0, ..., \beta_p)$, that minimize the sum of squared errors,

$$\sum_i (y_i - \mathbf{x}_i^T\beta)^2$$

If we stack the y_i on top of each other into a $n \times 1$ vector $\mathbf{y}$ and stack the $\mathbf{x}_i$ on top of each other into a $n \times (p+1)$ matrix $\mathbf{X}$, we can write the sum of squared errors as an inner product,

$$(\mathbf{y} - \mathbf{X}\beta)^T(\mathbf{y} - \mathbf{X}\beta)$$

By expanding this using matrix multiplication and properties, we get

$$\mathbf{y}^T\mathbf{y} - \beta^T\mathbf{X}^T\mathbf{y} - \mathbf{y}^T\mathbf{X}\beta + \beta^T\mathbf{X}^T\mathbf{X}\beta$$

This can be simplified to

$$\mathbf{y}^T\mathbf{y} - 2\beta^T\mathbf{X}^T\mathbf{y} + \beta^T\mathbf{X}^T\mathbf{X}\beta$$

To find the value of β that minimizes this quantity, we can find the derivative and set the equations equal to zero.

$$\frac{\partial}{\partial\beta}[\mathbf{y}^T\mathbf{y} - 2\beta^T\mathbf{X}^T\mathbf{y} + \beta^T\mathbf{X}^T\mathbf{X}\beta] = -2\mathbf{X}^T\mathbf{y} + 2\mathbf{X}^T\mathbf{X}$$

If we set this equal to 0 and solve for β, we get

$$\hat{\beta}_{OLS} = (\mathbf{X}^T\mathbf{X})^{-1}\mathbf{X}^T\mathbf{y}$$

A.18.2 Projection Approach

Another way to approach Least Squares is to say that we want to find the unique $\hat{\mathbf{y}}$ in $Col(X)$, the column space of X, such that we minimize $||\mathbf{y} - \hat{\mathbf{y}}||\$$. In order for $\hat{\mathbf{y}}$ to be in the column space of $\mathbf{X}$, $\hat{\mathbf{y}} = \mathbf{X}\hat{\beta}$.

The **orthogonal projection of a vector y on a the column space of matrix X** is

$$\hat{\mathbf{y}} = \mathbf{X}(\mathbf{X}^T\mathbf{X})^{-1}\mathbf{X}^T\mathbf{y}$$

and therefore,

$$\hat{\beta} = (\mathbf{X}^T\mathbf{X})^{-1}\mathbf{X}^T\mathbf{y}$$