Linear Models for Data Science

Jeffrey Woo

2024-07-11

Contents

Preface **5**
- Who is this book for? 5
- Data sets used 6
- Chapters 6
- Other resources 6

1 Data Wrangling with R **9**
- 1.1 Introduction 9
- 1.2 Data Wrangling using Base R Functions 10
- 1.3 Data Wrangling Using dplyr Functions 18

2 Data Visualization with R Using ggplot2 **27**
- 2.1 Introduction 27
- 2.2 Visualizations with a Single Categorical Variable 28
- 2.3 Visualizations with a Single Quantitative Variable 37
- 2.4 Bivariate Visualizations 41
- 2.5 Multivariate Visualizations 49

3 Basics with Simple Linear Regression (SLR) **55**
- 3.1 Introduction 55
- 3.2 Simple Linear Regression (SLR) 58
- 3.3 Estimating Regression Coefficients in SLR 63
- 3.4 Estimating Variance of Errors in SLR 66
- 3.5 Assessing Linear Association 66
- 3.6 A Word of Caution 70
- 3.7 R Tutorial 70

4 Inference with Simple Linear Regression (SLR) **77**
- 4.1 Introduction 77
- 4.2 Hypothesis Testing in SLR 78
- 4.3 Confidence Intervals for Regression Coefficients 81
- 4.4 CI of the Mean Response 82
- 4.5 PI of a New Response 83

4.6 Supplemental Notes on Statistical Inference 84
4.7 R Tutorial 87

5 Model Diagnostics and Remedial Measures in SLR 93
5.1 Introduction 93
5.2 Assumptions in Linear Regression 93
5.3 Assessing Regression Assumptions 96
5.4 Remedial Measures: Variance Stabilizing Transformations 103
5.5 Remedial Measures: Linearization Transformations 105
5.6 R Tutorial 107

6 Introduction 121

7 Literature 123
7.1 New section 123

8 Methods 125
8.1 math example 125

9 Applications 127
9.1 Example one 127
9.2 Example two 127

10 Final Words 129

Preface

Who is this book for?

There are many books on linear models, with various expectations for different levels of familiarity with statistical, mathematical, and coding concepts. These books generally fall into one of two camps:

1. Little to no familiarity with statistical and mathematical concepts, but fairly familiar to coding. These books tend to be written for programmers who want to get into data science. These books tend to explain linear models while trying to avoid statistical and mathematical concepts as much, only covering these concepts if absolutely necessary. These books tend to present linear models in a recipe format giving readers directions on what to do to build their models.

The drawback of such books is that readers do not get much understanding of the underlying concepts of linear models. It is impossible to give directions covering every possible scenario in the real world as real data are messy. Practitioners of data science often have to think outside the box in order to make linear models work for their particular data, and it is difficult to do so without understanding the mathematical framework of linear models.

2. Familiarity with mathematical notation and introductory statistical concepts such as statistical inference, and little to no familiarity with coding. These books tend to be written for mathematicians (or anyone with a strong background in mathematics) who want to get into data science. These books cover the mathematical framework of linear models thoroughly.

The drawback of such books is that readers must be comfortable with mathematical notation. This limits the audience for such books to people with fairly thorough training in mathematics. People without such training will get lost trying to read such books, and do not understand why we need to know the mathematical foundations to use linear models in data science.

This book is meant to be readable by both groups of readers. Some foundational mathematical knowledge will be presented, but will be written so that is readable by anyone. This book will also explain what these knowledge mean in the context of data science. Practical advice, based on the foundational mathematical knowledge, will also be given.

This book accompanies the course STAT 6021: Linear Models for Data Science, for the Masters of Data Science (MSDS) program at the University of Virginia School of Data Science.

As introductory statistics and introductory programming are pre-requisites for entering the MSDS program, this book assumes basic knowledge of statistical inference and coding. Review materials covering these concepts are provided separately for enrolled students.

Data sets used

I have tried to use as many open source data sets as much as possible so that readers can work on the various examples I have provided on their own. However, some data sets may not be open source and have come from my experience teaching this class since 2019 (and variations of the class since 2013), and have used some data sets that were shared by other statistics and data science educators. It is my goal to eventually use only open source data sets.

Chapters

The chapters for the book is as follows:

Other resources

Some other resources that readers may want to check out:

- *OpenIntro Statistics*, 4th ed. Diez, Cetinkaya-Rundel, Barr, OpenIntro. Get free PDF version at https://leanpub.com/os, just set the price that you want to pay to $0. This is a good book for introductory statistics.

- *Linear Models with R*, 2nd ed. Faraway. This is probably one of the few books that balances between the two camps that I wrote about earlier. It does require familiarity with matrices and linear algebra though.

- *Introduction to Linear Regression Analysis*, 5th or 6th ed. Montgomery, Peck, Vining. You may be able to access an e-version of the book through your university library if you are affiliated with a university. This book is mathematically rigorous so is useful to those who are interested in mathematical proofs that is not covered.

- *Applied Linear Statistical Models* (ALSM), Kutner, Nachtsheim, Neter, Li, 5th ed. This book covers a wide range of topics in linear models and is also mathematically rigorous.

- *Applied Linear Regression Models* (ALRM), Kutner, Nachtsheim, Neter, 4th ed. ALRM is the same as the first 14 chapters of ALSM. The second part of ALSM covers topics in Design of Experiments, which I highly recommend if you are interested in those topics.

Chapter 1

Data Wrangling with R

1.1 Introduction

The data structure that we will be dealing with most often will be data frames. When we read data in to R, they are typically stored as a data frame. A data frame can be viewed like an EXCEL spreadsheet, where data are stored in rows and columns. Before performing any analysis, we want the data frame to have this basic structure:

- Each row of the data frame corresponds to an observation.
- Each column of the data frame corresponds to a variable.

Sometimes, data are not structured in this way, and we will have to transform the data to take on this basic data structure. This process is called data wrangling. The most common and basic operations to transform the data are:

- Selecting a subset of columns of the data frame.
- Selecting a subset of rows of the data frame based on some criteria.
- Change column names.
- Find missing data.
- Create new variables based on existing variables.
- Combine multiple data frames.

We will explore two approaches to data wrangling:

- Using functions that already come pre-loaded with R (sometimes called base R).
- Using functions from the `dplyr` package.

These two approaches are quite different but can achieve the same goals in data wrangling. Each user of R usually ends up with their own preferred way of performing data wrangling operations, but it is important to know both approaches so you are able to work with a broader audience.

1.2 Data Wrangling using Base R Functions

We will use the dataset `ClassDataPrevious.csv` as an example. The data were collected from an introductory statistics class at UVa from a previous semester. Download the dataset from Canvas and read it into R.

```
Data<-read.csv("ClassDataPrevious.csv", header=TRUE)
```

We check the number of rows and columns in this dataframe.

```
dim(Data)
```

```
## [1] 298   8
```

There are 298 rows and 8 columns: so we have 298 students and 8 variables. We can also check the names for each column.

```
colnames(Data)
```

```
## [1] "Year"     "Sleep"    "Sport"    "Courses"  "Major"    "Age"      "Computer"
## [8] "Lunch"
```

The variables are:

1. `Year`: the year the student is in
2. `Sleep`: how much sleep the student averages a night (in hours)
3. `Sport`: the student's favorite sport
4. `Courses`: how many courses the student is taking in the semester
5. `Major`: the student's major
6. `Age`: the student's age (in years)
7. `Computer`: the operating system the student uses (Mac or PC)
8. `Lunch`: how much the student usually spends on lunch (in dollars)

1.2.1 View specific row(s) and/or column(s) of a data frame

We can view specific rows and/or columns of any data frame using the square brackets `[]`, for example:

```
Data[1,2] ##row index first, then column index
```

```
## [1] 8
```

The row index is listed first, then the column index, in the square brackets. This means the first student sleeps 8 hours a night. We can also view multiple rows and columns, for example:

```
Data[c(1,3,4),c(1,5,8)]
```

```
##     Year                           Major Lunch
## 1 Second                        Commerce    11
## 3 Second Cognitive science and psychology    10
## 4  First                           Pre-Comm     4
```

to view the 1st, 5th, and 8th variables for observations 1, 3, and 4.

There are several ways to view a specific column. For example, to view the 1st column (which is the variable called `Year`):

```
Data$Year ##or
Data[,1] ##or
Data[,-c(2:8)]
```

Note the comma separates the indices for the row and column. An empty value before the comma means we want all the rows, and then the specific column. To view multiple columns, for example the first four columns:

```
Data[,1:4]
Data[,c(1,2,3,4)]
```

To view the values of certain rows, we can use

```
Data[c(1,3),]
```

to view the values for observations 1 and 3. An empty value after the comma means we want all the columns for those specific rows.

1.2.2 Select observations by condition(s)

We may want to only analyze certain subsets of our data, based on some conditions. For example, we may only want to analyze students whose favorite sport is soccer. The `which()` function in R helps us find the indices associated with a condition being met. For example:

```
which(Data$Sport=="Soccer")
```

```
##  [1]   3  20  25  26  31  32  33  38  44  46  48  50  51  64  67  71  87  92  98
## [20]  99 118 122 124 126 128 133 136 137 143 146 153 159 165 174 197 198 207 211
## [39] 214 226 234 241 255 259 260 266 274 278 281 283 294 295
```

informs us which rows belong to observations whose favorite sport is soccer, i.e. the 3rd, 20th, 25th (and so on) students. We can create a new data frame that contains only students whose favorite sport is soccer:

```
SoccerPeeps<-Data[which(Data$Sport=="Soccer"),]
dim(SoccerPeeps)
```

```
## [1] 52  8
```

We are extracting the rows which satisfy the condition, favorite sport being soccer, and storing these rows into a new data frame called `SoccerPeeps`. We can see that this new data frame has 52 observations.

Suppose we want to have a data frame that satisfies two conditions: that the favorite sport is soccer and they are 2nd years at UVa. We can type:

```
SoccerPeeps_2nd<-Data[which(Data$Sport=="Soccer" & Data$Year=="Second"),]
dim(SoccerPeeps_2nd)
```

```
## [1] 25  8
```

This new data frame `SoccerPeeps_2nd` has 25 observations.

We can also set conditions based on numeric variables, for example, we want students who sleep more than eight hours a night

```
Sleepy<-Data[which(Data$Sleep>8),]
```

We can also create a data frame that contains students who satisfy at least one out of two conditions, for example, the favorite sport is soccer or they sleep more than 8 hours a night:

```
Sleepy_or_Soccer<-Data[which(Data$Sport=="Soccer" | Data$Sleep>8),]
```

1.2.3 Change column name(s)

For some datasets, the names of the columns are complicated or do not make sense. We should always give descriptive names to columns that make sense. For this dataset, the names are self-explanatory so we do not really need to change them. As an example, suppose we want to change the name of the 7th column from `Computer` to `Comp`:

```
names(Data)[7]<-"Comp"
```

To change the names of multiples columns (for example, the 1st and 7th columns), type:

```
names(Data)[c(1,7)]<-c("Yr","Computer")
```

1.2.4 Find and remove missing data

There are a few ways to locate missing data. Using the `is.na()` function directly on a data frame produces a lot of output that can be messy to view:

```
is.na(Data)
```

On the other hand, using the `complete.cases()` function is more pleasing to view:

```
Data[!complete.cases(Data),]
```

```
##         Yr Sleep      Sport Courses                                   Major
## 103 Second    NA Basketball       7 psychology and youth and social innovation
## 206 Second     8       None       4                       Cognitive Science
##     Age Computer Lunch
## 103  19      Mac    10
## 206  19      Mac    NA
```

The code above will extract rows that are not complete cases, in other words, rows that have missing entries. The output informs us observation 103 has a missing value for `Sleep`, and observation 206 has a missing value for `Lunch`.

If you want to remove observations with a missing value, you can use one of the following two lines of code to create new data frames with the rows with missing values removed:

```
Data_nomiss<-na.omit(Data) ##or
Data_nomiss2<-Data[complete.cases(Data),]
```

A word of caution: these lines of code will remove the entire row as long as at least a column has missing entries. As noted earlier, observation 103 has a missing value for only the `Sleep` variable. But this observation still provides information on the other variables, which are now removed.

1.2.5 Summarizing variable(s)

Very often, we want to obtain some characteristics of our data. A common way to summarize a numerical variable is to find its mean. We have four numerical variables in our data frame, which are in columns 2, 4, 6, and 8. To find the mean of all four numerical variables, we can use the `apply()` function:

```
apply(Data[,c(2,4,6,8)],2,mean)
```

```
##     Sleep   Courses       Age     Lunch
##        NA  5.016779 19.573826        NA
```

```
apply(Data[,c(2,4,6,8)],2,mean,na.rm=T)
```

```
##      Sleep    Courses        Age      Lunch
## 155.559259   5.016779  19.573826 156.594175
```

Notice that due to the missing values, the first line has `NA` for some of the variables. The second line includes an optional argument, `na.rm=T`, which will remove the observations with an `NA` value for the variable from the calculation of the mean.

There are at least 3 arguments that are supplied to the `apply()` function:

1. The first argument is a data frame containing all the variables which we want to find the mean of. In this case, we want columns 2, 4, 6, and 8 of the data frame `Data`.

2. The second argument takes on the value `1` or `2`. Since we want to find the mean of columns, rather than rows, we type `2`. If want to mean of a row, we will type `1`.

3. The third argument specifies the name of the function you want to apply to the columns of the supplied data frame. In this case, we want the mean. We can change this to find the median, standard deviation, etc, of these numeric variables if we want to.

We notice the means for some of the variables are suspiciously high, so looking at the medians will be more informative.

```
apply(Data[,c(2,4,6,8)],2,median,na.rm=T)
```

```
##   Sleep Courses     Age   Lunch
##     7.5     5.0    19.0     9.0
```

1.2.6 Summarizing variable by groups

Sometimes we want to summarize a variable by groups. Suppose we want to find the median amount of sleep separately for 1st years, 2nd years, 3rd years, and 4th years get. We can use the `tapply()` function:

```
tapply(Data$Sleep,Data$Yr,median,na.rm=T)
```

```
##  First Fourth Second  Third
##    8.0    7.0    7.5    7.0
```

This informs us the median amount of sleep first years get is 8 hours a night; for fourth years the median amount is 7 hours a night.

There are at least 3 arguments that are supplied to the `tapply()` function:

1. The first argument contains the vector which we want to summarize.
2. The second argument contains the factor which we use to subset our data. In this example, we want to subset according to `Yr`.
3. The third argument is the function which we want to apply to each subset of our data.
4. The fourth argument is optional, in this case, we want to remove observations with missing values from the calculation of the mean.

Notice the output orders the factor levels in alphabetical order. For our context, it is better to rearrange the levels to First, Second, Third, Fourth using the `factor()` function:

```
Data$Yr<-factor(Data$Yr, levels=c("First","Second","Third","Fourth"))

levels(Data$Yr)
```

```
## [1] "First"  "Second" "Third"  "Fourth"
```

```
tapply(Data$Sleep,Data$Yr,median,na.rm=T) ##much nicer
```

```
##  First Second  Third Fourth
##    8.0    7.5    7.0    7.0
```

This output makes a lot more sense for this context.

If we want to summarize a variable on groups formed by more than one variable, we need to adjust the second argument in the `tapply()` function by creating a list. Suppose we want to find the median sleep hour based on the `Yr` and the preferred operating system of the observations,

```
tapply(Data$Sleep,list(Data$Yr,Data$Computer),median,na.rm=T)
```

```
##           Mac   PC
## First  NA 8.0 7.50
## Second  7 7.5 7.50
## Third  NA 7.5 7.00
## Fourth NA 7.0 7.25
```

Interestingly, it looks like there were observations who did not specify which operating system they use, hence the extra column in the output.

1.2.7 Create a new variable based on existing variable(s)

Depending on the context of our analysis, we may need to create new variables based on existing variables. There are a few variations of this task, based on the type of variable you want to create, and the type of variable it is based on.

1.2.7.1 Create a numeric variable based on another numeric variable

The variable `Sleep` is in number of hours. Suppose we need to convert the values of `Sleep` to number of minutes, we can simply perform the following mathematical operation:

```
Sleep_mins<-Data$Sleep * 60
```

and store the transformed variable into a vector called `Sleep_mins`.

1.2.7.2 Create a binary variable based on a numeric variable

Suppose we want to create a binary variable (categorical variable with two levels), called `deprived`. An observation will obtain a value of “yes” if they sleep for less than 7 hours a night, and “no” otherwise. The `ifelse()` function is useful in creating binary variables:

```
deprived<-ifelse(Data$Sleep<7, "yes", "no")
```

There are 3 arguments associated with the `ifelse()` function:

1. The first argument is the condition that we wish to use.

2. The second argument is the value of the observation if the condition is true.

3. The third argument is the value of the observation if the condition if false.

1.2.7.3 Create a categorical variable based on a numeric variable

Suppose we want to create a categorical variable based on the number of courses a student takes. We will call this new variable `CourseLoad`, which takes on the following values:

- `light` if 3 courses or less,
- `regular` if 4 or 5 courses,
- `heavy` if more than 5 courses .

The `cut()` function is used in this situation

```
CourseLoad<-cut(Data$Courses, breaks = c(-Inf, 3, 5, Inf),
                labels = c("light", "regular", "heavy"))
```

There are three arguments that are applied to the `cut()` function:

1. The first argument is the vector which you are basing the new variable on.

2. The argument breaks lists how you want to set the intervals associated with `Data$Courses`. In this case, we are creating three intervals: one from $(-\infty, 3]$, another from $(3, 5]$, and the last interval from $(5, \infty]$.

3. The argument labels gives the label for `CourseLoad` associated with each interval.

1.2.7.4 Collapse levels

Sometimes, a categorical variable has more levels than we need for our analysis, and we want to collapse some levels. For example, the variable `Yr` has four levels: First, Second, Third, and Fourth. Perhaps we are more interested in comparing upperclassmen and underclassmen, so we want to collapse First and Second years into underclassmen, and Third and Fourth years into upperclassmen:

```
levels(Data$Yr)
```

```
## [1] "First"  "Second" "Third"  "Fourth"
```

```
new.levels<-c("und", "und", "up","up")
Year2<-factor(new.levels[Data$Yr])
levels(Year2)
```

```
## [1] "und" "up"
```

The levels associated with the variable `Yr` are ordered as First, Second, Third, Fourth. The character vector new.levels has `under` as the first two characters, and upper as the last two characters to correspond to the original levels in the variable Yr. The new variable is called Year2.

1.2.8 Combine data frames

We have created four new variables, Sleep_mins, deprived, CourseLoad, and Year2, based on previously existing variables. Since these variables are all based on the same observations, we can combine them with an existing data frame using the data.frame() function:

```
Data<-data.frame(Data,Sleep_mins,deprived,CourseLoad,Year2)
head(Data)
```

```
##       Yr Sleep      Sport Courses                            Major Age Computer
## 1 Second     8 Basketball       6                         Commerce  19      Mac
## 2 Second     7     Tennis       5                       Psychology  19      Mac
## 3 Second     8     Soccer       5 Cognitive science and psychology  21      Mac
## 4  First     9 Basketball       5                         Pre-Comm  19      Mac
## 5 Second     4 Basketball       6                      Statistics   19       PC
## 6  Third     7       None       4                       Psychology  20       PC
##   Lunch Sleep_mins deprived CourseLoad Year2
## 1    11        480       no      heavy   und
## 2    10        420       no    regular   und
## 3    10        480       no    regular   und
## 4     4        540       no    regular   und
## 5     0        240      yes      heavy   und
## 6    11        420       no    regular    up
```

Notice that since we listed the four new variables after Data in the data.frame() function, they appear after the original columns in the data frame.

Alternatively, we can use the cbind() function which gives the same data frame:

```
Data2<-cbind(Data,Sleep_mins,deprived,CourseLoad,Year2)
```

If you are combining data frames which have different observations but the same columns, we can merge them using rbind():

```
dat1<-Data[1:3,1:3]
dat3<-Data[6:8,1:3]
res.dat2<-rbind(dat1,dat3)
head(res.dat2)
```

```
##       Yr Sleep      Sport
## 1 Second     8 Basketball
## 2 Second     7     Tennis
## 3 Second     8     Soccer
## 6  Third     7       None
## 7 Second     7 Basketball
## 8  First     7 Basketball
```

1.2.9 Export data frame in R to a .csv file

To export a data frame to a .csv file, type:

```
write.csv(Data, file="newdata.csv", row.names = FALSE)
```

A file newdata.csv will be created in the working directory. Note that by default, the argument `row.names` is set to be `TRUE`. This will add a column in the dataframe which is an index number. I do not find this step useful in most analyses so I almost always set `row.names` to be `FALSE`.

1.2.10 Sort data frame by column values

To sort your data frame in ascending order by `Age`:

```
Data_by_age<-Data[order(Data$Age),]
```

To sort in descending order by `Age`:

```
Data_by_age_des<-Data[order(-Data$Age),]
```

To sort in ascending order by `Age` first, then by `Sleep`:

```
Data_by_age_sleep<-Data[order(Data$Age, Data$Sleep),]
```

1.3 Data Wrangling Using dplyr Functions

In the previous section, we were performing data wrangling operations using functions that are built in with base R. In this module, we will be using functions mostly from a package called `dplyr`, which can perform the same operations as well.

Before performing data wrangling operations, let us clear our environment, so that previously declared objects are removed. This allows us to start with a clean slate, which is often desirable when starting on a new analysis. This is done via:

```
rm(list = ls())
```

The `dplyr` package is a subset of the `tidyverse` package, so we can access these functions after installing and loading either package. After installing the `tidyverse` package, load it by typing:

```
##library(dplyr) or
library(tidyverse)
```

The `dplyr` package was developed to make the syntax more intuitive to a broader range of R users, primarily through the use of pipes. However, the code involved with functions from `dlpyr` tends to be longer than code involved with base R functions, and there are more functions to learn with `dplyr`.

You will find a lot of articles on the internet by various R users about why each of them believes one approach to be superior to the other. I am not fussy about which approach you use as long as you can perform the necessary operations. It is to your benefit to be familiar with both approaches so you can work with a broader range of R users.

We will continue to use the dataset `ClassDataPrevious.csv` as an example. Download the dataset from Canvas and read it into R:

```
Data<-read.csv("ClassDataPrevious.csv", header=TRUE)
```

In the examples below, we are performing the same operations as in the previous section, but using `dplyr` functions instead of base R functions.

1.3.1 Select specific column(s) of a data frame

The `select()` function is used to select specific columns. There are a couple of ways to use this function. First:

```
select(Data,Year)
```

to select the column `Year` from the data frame called `Data`.

1.3.1.1 Pipes

Alternatively, we can use pipes:

```
Data%>%
  select(Year)
```

Pipes in R are typed using `%>%` or by pressing Ctrl + Shift + M on your keyboard. To think of the operations above, we can read the code as

1. take the data frame called Data
2. and then select the column named Year.

We can interpret a pipe as “and then”. Commands after a pipe should be placed on a new line (press enter). Pipes are especially useful if we want to execute several commands in sequence, which we will see in later examples.

1.3.2 Select observations by condition(s)

The `filter()` function allows us to subset our data based on some conditions, for example, to select students whose favorite sport is soccer:

```
filter(Data, Sport=="Soccer")
```

We can create a new data frame called SoccerPeeps that contains students whose favorite sport is soccer:

```
SoccerPeeps<-Data%>%
  filter(Sport=="Soccer")
```

Suppose we want to have a data frame, called SoccerPeeps_2nd, that satisfies two conditions: that the favorite sport is soccer and they are 2nd years at UVa:

```
SoccerPeeps_2nd<-Data%>%
  filter(Sport=="Soccer" & Year=="Second")
```

We can also set conditions based on numeric variables, for example, we want the students who sleep more than eight hours a night:

```
Sleepy<-Data%>%
  filter(Sleep>8)
```

We can also create a data frame that contains observations as long as they satisfy at least one out of two conditions: the favorite sport is soccer or they sleep more than 8 hours a night:

```
Sleepy_or_Soccer<-Data%>%
  filter(Sport=="Soccer" | Sleep>8)
```

1.3.3 Change column name(s)

It is straightforward to change the names of columns using the rename() function. For example:

```
Data<-Data%>%
  rename(Yr=Year, Comp=Computer)
```

allows us to change the name of two columns: from Year and Computer to Yr and Comp.

1.3.4 Summarizing variable(s)

The summarize() function allows us to summarize a column. Suppose we want to find the mean value of the numeric columns: Sleep, Courses, Age, and Lunch:

```
Data%>%
  summarize(mean(Sleep,na.rm = T),mean(Courses),mean(Age),mean(Lunch,na.rm = T))
```

```
##   mean(Sleep, na.rm = T) mean(Courses) mean(Age) mean(Lunch, na.rm = T)
## 1               155.5593      5.016779  19.57383               156.5942
```

The output looks a bit cumbersome. We can give names to each summary

```
Data%>%
  summarize(avgSleep=mean(Sleep,na.rm = T),avgCourse=mean(Courses),avgAge=mean(Age),
            avgLun=mean(Lunch,na.rm = T))
```

```
##   avgSleep avgCourse   avgAge   avgLun
## 1 155.5593  5.016779 19.57383 156.5942
```

As mentioned previously, the means look suspiciously high for a couple of variables, so looking at the medians may be more informative:

```
Data%>%
  summarize(medSleep=median(Sleep,na.rm = T),medCourse=median(Courses),
            medAge=median(Age),medLun=median(Lunch,na.rm = T))
```

```
##   medSleep medCourse medAge medLun
## 1      7.5         5     19      9
```

Note: For a lot of functions in the `dplyr` package, using American spelling or British spelling works. So we can use `summarise()` instead of `summarize()`.

1.3.5 Summarizing variable by groups

Suppose we want to find the median amount of sleep 1st years, 2nd years, 3rd years, and 4th years get. We can use the `group_by()` function:

```
Data%>%
  group_by(Yr)%>%
  summarize(medSleep=median(Sleep,na.rm=T))
```

```
## # A tibble: 4 x 2
##   Yr     medSleep
##   <chr>     <dbl>
## 1 First       8
## 2 Fourth      7
## 3 Second      7.5
## 4 Third       7
```

The way to read the code above is

1. Get the data frame called `Data`,
2. and then group the observations by `Yr`,
3. and then find the median amount of sleep by each `Yr` and store the median in a vector called `medSleep`.

As seen previously, the ordering of the factor levels is in alphabetical order. For our context, it is better to rearrange the levels to First, Second, Third, Fourth. We can use the `mutate()` function whenever we want to transform or create a new variable. In this case, we are transforming the variable `Yr` by reordering the factor levels with the `fct_relevel()` function:

```
Data<- Data%>%
  mutate(Yr = Yr%>%
           fct_relevel(c("First","Second","Third","Fourth")))
```

1. Get the data frame called `Data`,
2. and then transform the variable called `Yr`,
3. and then reorder the factor levels.

Then, we use pipes, the `group_by()`, and `summarize()` functions like before:

```
Data%>%
  group_by(Yr)%>%
  summarize(medSleep=median(Sleep,na.rm=T))
```

```
## # A tibble: 4 x 2
##   Yr     medSleep
##   <fct>     <dbl>
## 1 First       8
## 2 Second      7.5
## 3 Third       7
## 4 Fourth      7
```

This output makes a lot more sense for this context.

To summarize a variable on groups formed by more than one variable, we just add the other variables in the `group_by()` function:

```
Data%>%
  group_by(Yr,Comp)%>%
  summarize(medSleep=median(Sleep,na.rm=T))
```

```
## `summarise()` has grouped output by 'Yr'. You can override using the `.groups`
## argument.

## # A tibble: 9 x 3
## # Groups:   Yr [4]
##   Yr     Comp  medSleep
##   <fct>  <chr>    <dbl>
## 1 First  "Mac"     8
## 2 First  "PC"      7.5
## 3 Second ""        7
## 4 Second "Mac"     7.5
## 5 Second "PC"      7.5
## 6 Third  "Mac"     7.5
## 7 Third  "PC"      7
## 8 Fourth "Mac"     7
## 9 Fourth "PC"      7.25
```

1.3.6 Create a new variable based on existing variable(s)

As mentioned in the previously, the `mutate()` function is used to transform a variable or to create a new variable. There are a few variations of this task, based on the type of variable you want to create, and the type of variable it is based on.

1.3.6.1 Create a numeric variable based on another numeric variable

The variable `Sleep` is in number of hours. Suppose we need to convert the values of `Sleep` to number of minutes, we can simply perform the following mathematical operation:

```
Data<-Data%>%
  mutate(Sleep_mins = Sleep*60)
```

and store the transformed variable called `Sleep_mins` and add `Sleep_mins` to the data frame called `Data`.

1.3.6.2 Create a binary variable based on a numeric variable

Suppose we want to create a binary variable , called `deprived`. An observation will obtain a value of `yes` if they sleep for less than 7 hours a night, and `no` otherwise. We will then add this variable deprived to the data frame called `Data`:

```
Data<-Data%>%
  mutate(deprived=ifelse(Sleep<7, "yes", "no"))
```

1.3.6.3 Create a categorical variable based on a numeric variable

Suppose we want to create a categorical variable based on the number of courses a student takes. We will call this new variable `CourseLoad`, which takes on the following values:

- `light` if 3 courses or less,
- `regular` if 4 or 5 courses,
- `heavy` if more than 5 courses

and then add `CourseLoad` to the data frame `Data`. We can use the `case_when()` function from the `dplyr` package, instead of the `cut()` function:

```
Data<-Data%>%
  mutate(CourseLoad=case_when(Courses <= 3 ~ "light",
                              Courses >3 & Courses <=5 ~ "regular",
                              Courses > 5 ~ "heavy"))
```

Notice how the names of the categorical variable are supplied after a specific condition is specified.

1.3.6.4 Collapsing levels

Sometimes, a categorical variable has more levels than we need for our analysis, and we want to collapse some levels. For example, the variable Yr has four levels: First, Second, Third, and Fourth. Perhaps we are more interested in comparing between upperclassmen and underclassmen, so we want to collapse First and Second Yrs into underclassmen, and Third and Fourth Yrs into upperclassmen. We will use the `fct_collapse()` function:

```
Data<-Data%>%
  mutate(UpUnder=fct_collapse(Yr,under=c("First","Second"),up=c("Third","Fourth")))
```

We are creating a new variable called `UpUnder`, which is done by collapsing `First` and `Second` into a new factor called `under`, and collapsing `Third` and `Fourth` into a new factor called `up`. `UpUnder` is also added to the data frame `Data`.

1.3.7 Combine data frames

To combine data frames which have different observations but the same columns, we can combine them using `bind_rows()`:

```
dat1<-Data[1:3,1:3]
dat3<-Data[6:8,1:3]
res.dat2<-bind_rows(dat1,dat3)
head(res.dat2)
```

```
##       Yr Sleep      Sport
## 1 Second     8 Basketball
## 2 Second     7     Tennis
## 3 Second     8     Soccer
## 4  Third     7       None
## 5 Second     7 Basketball
## 6  First     7 Basketball
```

`bind_rows()` works the same way as `rbind()`. Likewise, we can use `bind_cols()` instead of `cbind()`.

1.3.8 Sort data frame by column values

To sort your data frame in ascending order by `Age`:

```
Data_by_age<-Data%>%
  arrange(Age)
```

To sort in descending order by `Age`:

```
Data_by_age_des<-Data%>%
  arrange(desc(Age))
```

To sort in ascending order by `Age` first, then by `Sleep`:

```
Data_by_age_sleep<-Data%>%
  arrange(Age,Sleep)
```

Chapter 2

Data Visualization with R Using ggplot2

2.1 Introduction

Data visualizations are tools that summarize data. Consider the visuals from the CDC covid tracker dashboard to an external site. Without actually having access to the actual data, we have a sense of trends associated with hospitalizations and deaths. Good visualizations are easy to interpret for a wide variety of audiences, and are easier to explain than statistical models.

In this module, you will learn how to create common data visualizations. The choice of data visualization is almost always determined by whether the variable(s) involved is categorical or quantitative. Discrete variables are interesting because depending on the circumstance, they can be viewed as either categorical or quantitative in the context of data visualizations.

We will be using functions from the `ggplot2` package to create visualizations. The `ggplot2` package enables users to create various kinds of data visualizations, beyond the visualizations that can be made in base R. The `ggplot2` package is automatically loaded when we load the `tidyverse` package, although we can load `ggplot2` on its own.

```
library(tidyverse)
```

We will use the dataset `ClassDataPrevious.csv` as an example. The data were collected from an introductory statistics class at UVa from a previous semester. Download the dataset from Canvas and read it into R.

```
Data<-read.csv("ClassDataPrevious.csv", header=TRUE)
```

The variables are:

1. `Year`: the year the student is in
2. `Sleep`: how much sleep the student averages a night (in hours)
3. `Sport`: the student's favorite sport
4. `Courses`: how many courses the student is taking in the semester
5. `Major`: the student's major
6. `Age`: the student's age (in years)
7. `Computer`: the operating system the student uses (Mac or PC)
8. `Lunch`: how much the student usually spends on lunch (in dollars)

2.2 Visualizations with a Single Categorical Variable

2.2.1 Frequency tables

Frequency tables are a common tool to summarize categorical variables. These tables give us the number of observations (sometimes called counts) that belong to each class of a categorical variable. These tables are created using the `table()` function. Suppose we want to see the number of students in each year in our data:

```
table(Data$Year)
```

```
##
##  First Fourth Second  Third
##     83     30    139     46
```

Notice the order of the years could be rearranged to make more sense:

```
Data$Year<-factor(Data$Year, levels=c("First","Second","Third","Fourth"))
levels(Data$Year)
```

```
## [1] "First"  "Second" "Third"  "Fourth"
```

```
mytab<-table(Data$Year)
mytab
```

```
##
##  First Second  Third Fourth
##     83    139     46     30
```

So we have 83 first years, 139 second years, 46 third years, and 30 fourth years in our dataset.

We can report these numbers using proportions instead of counts, using `prop.table()`:

```
prop.table(mytab)
```

```
##
```

```
##     First    Second     Third    Fourth
## 0.2785235 0.4664430 0.1543624 0.1006711
```

or percentages by multiplying by 100:

```
prop.table(mytab) * 100
```

```
##
##    First   Second    Third   Fourth
## 27.85235 46.64430 15.43624 10.06711
```

To round the percentages to two decimal places, use the `round()` function:

```
round(prop.table(mytab) * 100, 2)
```

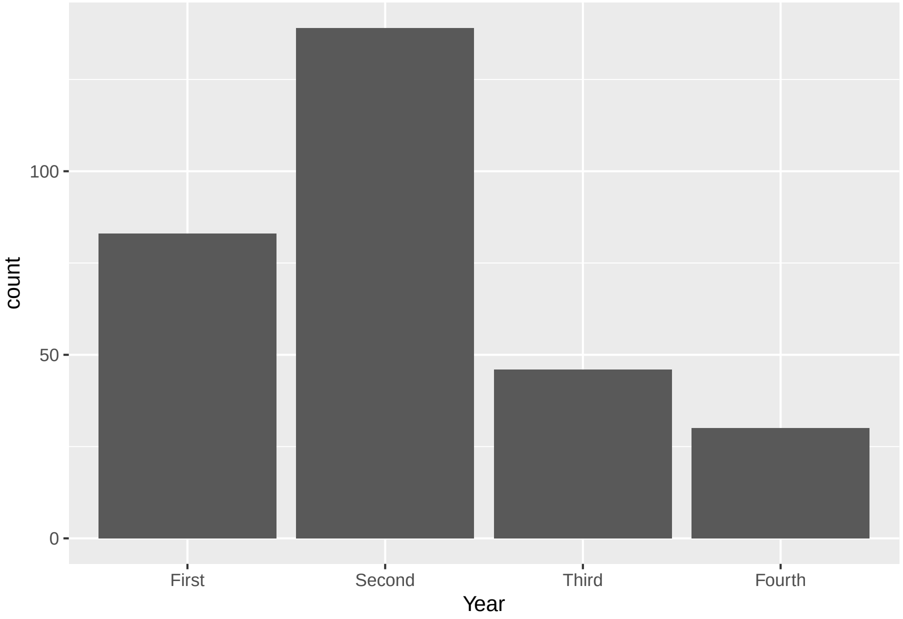
```
##
##  First Second  Third Fourth
##  27.85  46.64  15.44  10.07
```

So about 27.85% of these students are first years, 46.64% are second years, 15.44% are third years, and 10.07% are fourth years.

2.2.2 Bar charts

Bar charts are a simple way to visualize categorical data, and can be viewed as a visual representation of frequency tables. To create a bar chart for the years of these students, we use:

```
ggplot(Data, aes(x=Year))+
  geom_bar()
```



We can read the number of students who are first, second, third, and fourth years by reading off the corresponding value on the vertical axis.

From these two lines we code, we can see the basic structure of creating data visualizations with the `ggplot()` function:

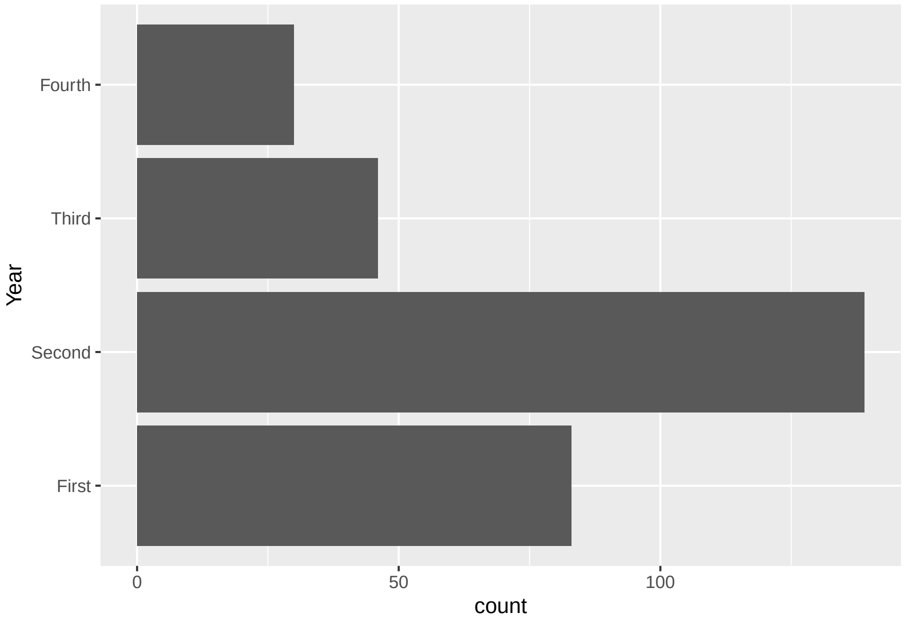
1. Use the `ggplot()` function, and supply the name of the data frame, and the x- and/or y- variables via the aes() function. End this line with a `+` operator, and then press enter.

2. In the next line, specify the type of graph we want to create (called `geoms`). For a bar chart, type `geom_bar()`.

Some describe these lines of code as two layers of code. These two layers must be supplied for all data visualizations with `ggplot()`.

Additional optional layers can be added (these usually deal with the details of the visuals). Suppose we want to change the orientation of this bar chart, we can add an optional line, or layer:

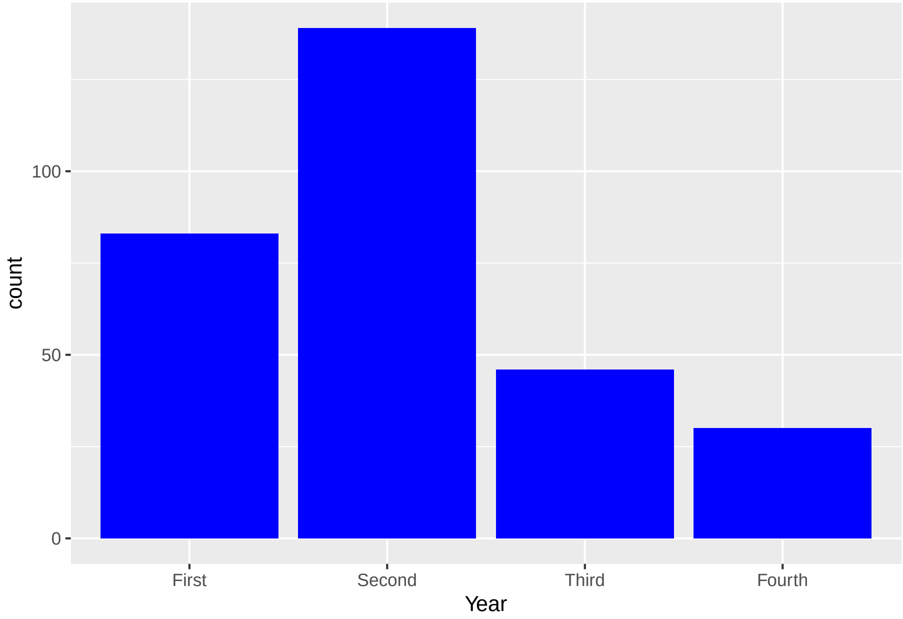
```
ggplot(Data, aes(x=Year))+
  geom_bar()+
  coord_flip()
```



It is recommended that each layer is typed on a line below the previous layer. A + sign is used at the end of each layer to add another layer below.

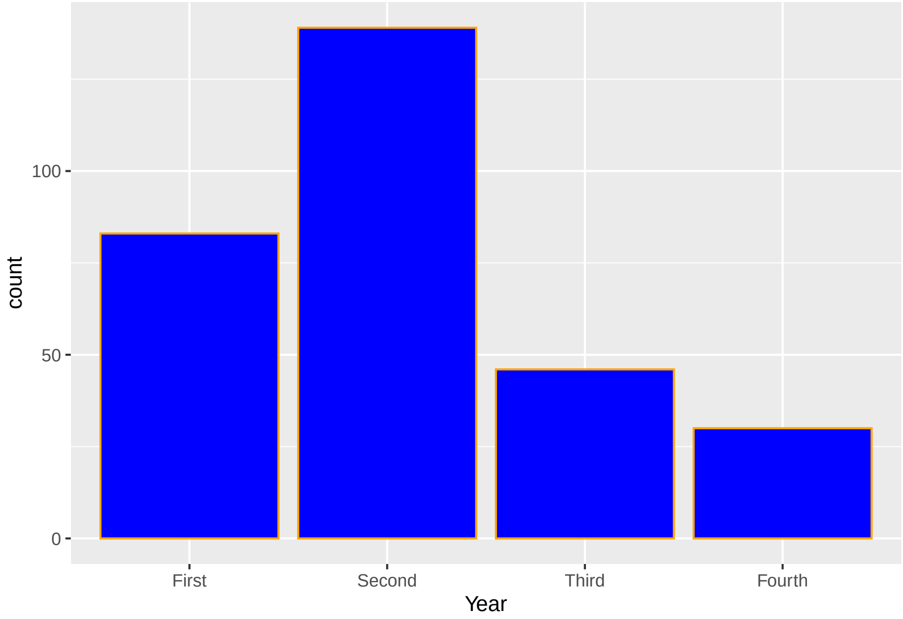
To change the color of the bars:

```
ggplot(Data, aes(x=Year))+
  geom_bar(fill="blue")
```



To have a different color to outline the bars:

```
ggplot(Data, aes(x=Year))+
  geom_bar(fill="blue",color="orange")
```

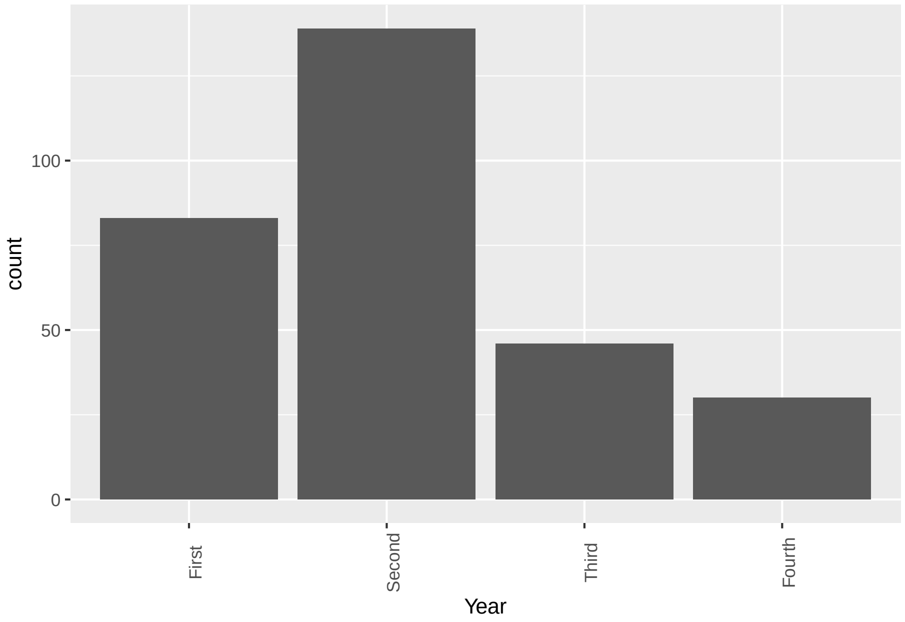


2.2.2.1 Customize title and labels of axes in bar charts

To change the orientation of the labels on the horizontal axis, we add an extra layer called `theme`. This will be useful when we have many classes and/or labels with long names.

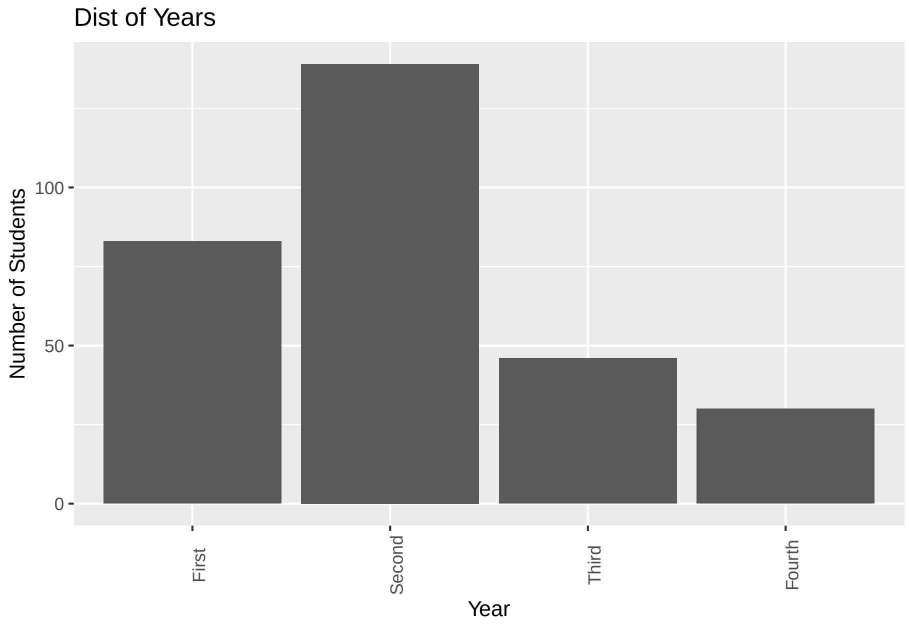
To rotate the labels on the horizontal by 90 degrees:

```r
ggplot(Data, aes(x=Year))+
  geom_bar()+
  theme(axis.text.x = element_text(angle = 90))
```



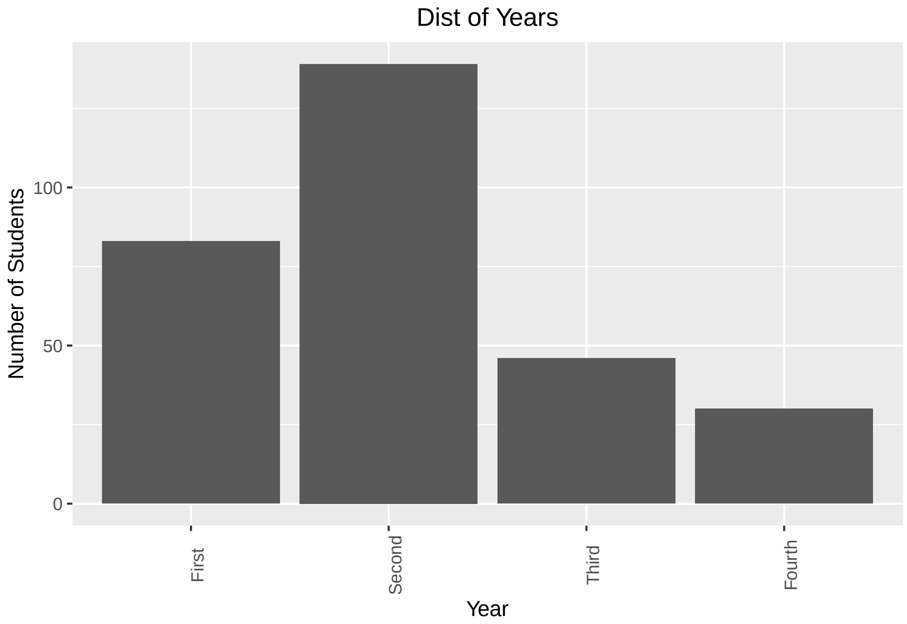
As we create more visualizations, it is good practice to give short but meaningful and descriptive names for each axis and provide a title. We can change the labels of the x- and y- axes, as well as add a title for the bar chart by adding another layer called `labs`:

```r
ggplot(Data, aes(x=Year))+
  geom_bar()+
  theme(axis.text.x = element_text(angle = 90))+
  labs(x="Year", y="Number of Students", title="Dist of Years")
```



We can also adjust the position of the title, for example, center-justify it via `theme`:

```
ggplot(Data, aes(x=Year))+
  geom_bar()+
  theme(axis.text.x = element_text(angle = 90),
        plot.title = element_text(hjust = 0.5))+
  labs(x="Year", y="Number of Students", title="Dist of Years")
```



2.2.2.2 Create a bar chart using proportions

Suppose we want to create a bar chart where the y-axis displays the proportions, rather than the counts of each level. There are a few steps to produce such a bar chart. First, we create a new dataframe, where each row represents a year, and we add the proportion of each year into a new column:

```
newData<-Data%>%
  group_by(Year)%>%
  summarize(Counts=n())%>%
  mutate(Percent=Counts/nrow(Data))
```

The code above does the following:

1. Creates a new data frame called `newData` by taking the data frame called `Data`,
2. and then groups the observations by `Year`,
3. and then counts the number of observations in each `Year` and stores these values in a vector called `Counts`,
4. and then creates a new vector called `Percent` by using the mathematical operations as specified in `mutate()`. `Percent` is added to `newData`.

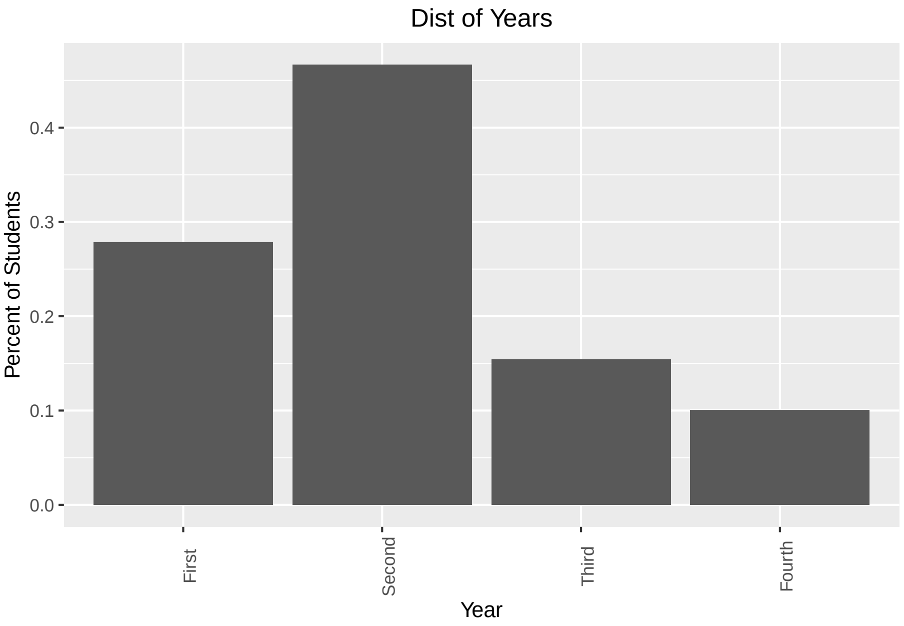
We can take a look at the contents of `newData`:

```
newData
```

```
## # A tibble: 4 x 3
##   Year   Counts Percent
##   <fct>    <int>   <dbl>
## 1 First       83   0.279
## 2 Second     139   0.466
## 3 Third       46   0.154
## 4 Fourth      30   0.101
```

To create a bar chart using proportions:

```
ggplot(newData, aes(x=Year, y=Percent))+
  geom_bar(stat="identity")+
  theme(axis.text.x = element_text(angle = 90),
        plot.title = element_text(hjust = 0.5))+
  labs(x="Year", y="Percent of Students", title="Dist of Years")
```



Note the following:

1. In the first layer, we use `newData` instead of the old data frame. In `aes()`, we specified a y-variable, which we want to be `Percent`.
2. In the second layer, we specified `stat="identity"` inside `geom_bar()`.

2.3 Visualizations with a Single Quantitative Variable

2.3.1 5 number summary

The `summary()` function, when applied to a quantitative variable, produces the 5 number summary: the minimum, the first quartile (25th percentile), the median (50th percentile), the third quartile (75th percentile), and the maximum, as well as the mean. For example, to obtain the 5 number summary of the ages of these students:

```
summary(Data$Age)
```

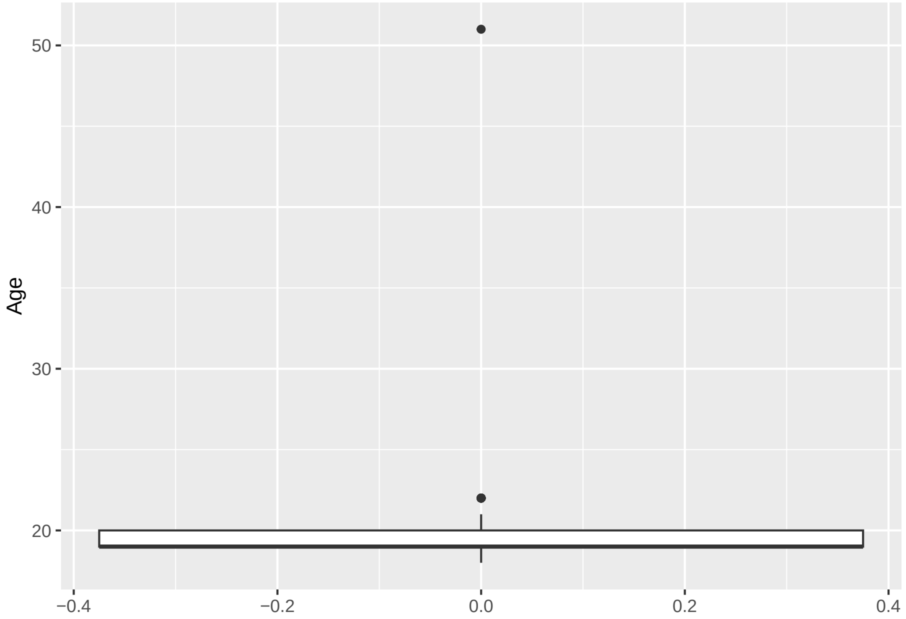
```
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max. 
##   18.00   19.00   19.00   19.57   20.00   51.00
```

The average age of the observations in this dataset is 19.57 years old. Notice the first quartile and the median are both 19 years old, that means at least a quarter of the observations are 19 years old. Also note the maximum of 51 years old, so we have a student who is quite a lot older than the rest.

2.3.2 Boxplots

A boxplot is a graphical representation of the 5 number summary. To create a generic boxplot, we have the following two lines of code when using the `ggplot()` function:

```
ggplot(Data, aes(y=Age))+
  geom_boxplot()
```



Note we are still using the same structure when creating data visualizations with `ggplot()`:

1. Use the `ggplot()` function, and supply the name of the data frame, and the x- and/or y- variables via the aes() function. End this line with a `+` operator, and then press enter.

2. In the next line, specify the type of graph we want to create (called `geoms`). For a boxplot, type `geom_boxplot`.

Notice there are outliers (observations that are a lot older or younger) that are denoted by the dots. One is the 51 year old, and 22 year olds are deemed to be outliers. The rule being used is the $1.5 \times IQR$ rule.

Similar to bar charts, we can change the orientation of boxplots by adding an additional layer as before:

```
ggplot(Data, aes(y=Age))+
  geom_boxplot()+
  coord_flip()
```

We can change the color of the box and the outliers similarly:

```
ggplot(Data, aes(y=Age))+
  geom_boxplot(color="blue", outlier.color = "orange" )
```

2.3.3 Histograms

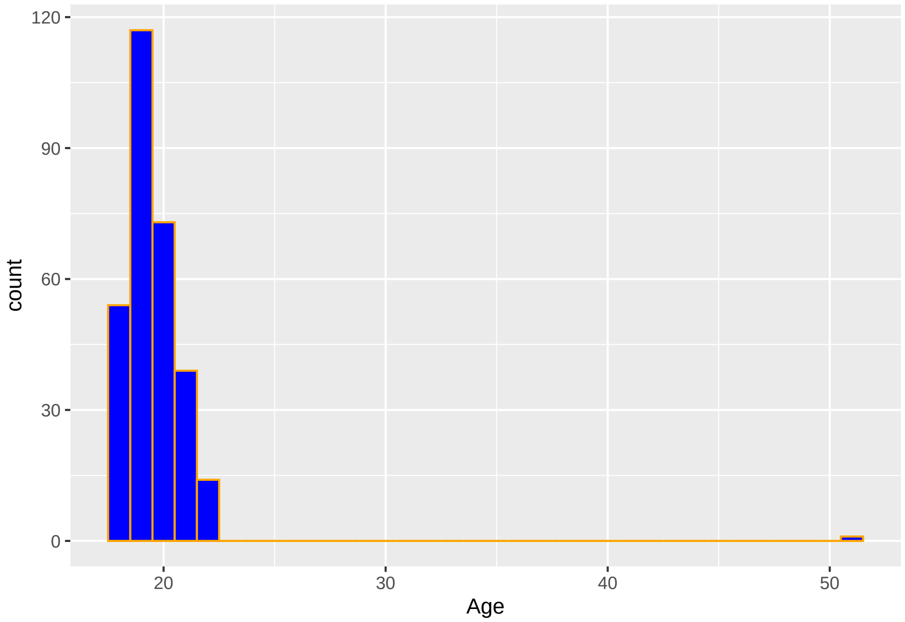
A histogram displays the number of observations within each bin on the x-axis:

```
ggplot(Data,aes(x=Age))+
  geom_histogram()
```

```
## `stat_bin()` using `bins = 30`. Pick better value with `binwidth`.
```

Notice a warning message is displayed when creating a basic histogram. To fix this, we use the binwidth argument within `geom_histogram`. We try `binwidth=1` for now, which means the width of the bin is 1 unit:

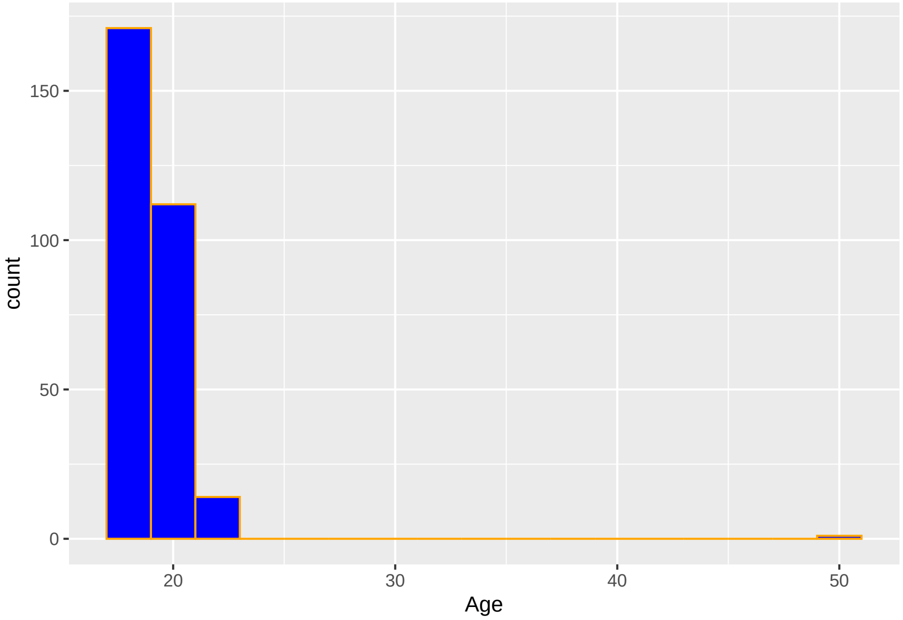
```
ggplot(Data,aes(x=Age))+
  geom_histogram(binwidth = 1,fill="blue",color="orange")
```



The ages of the students are mostly young, with 19 and 20 years olds being the most commonly occuring.

A well-known drawback of histograms is that the width of the bins can drastically affect the visual. For example, suppose we change the binwidth to 2:

```
ggplot(Data,aes(x=Age))+
  geom_histogram(binwidth = 2,fill="blue",color="orange")
```

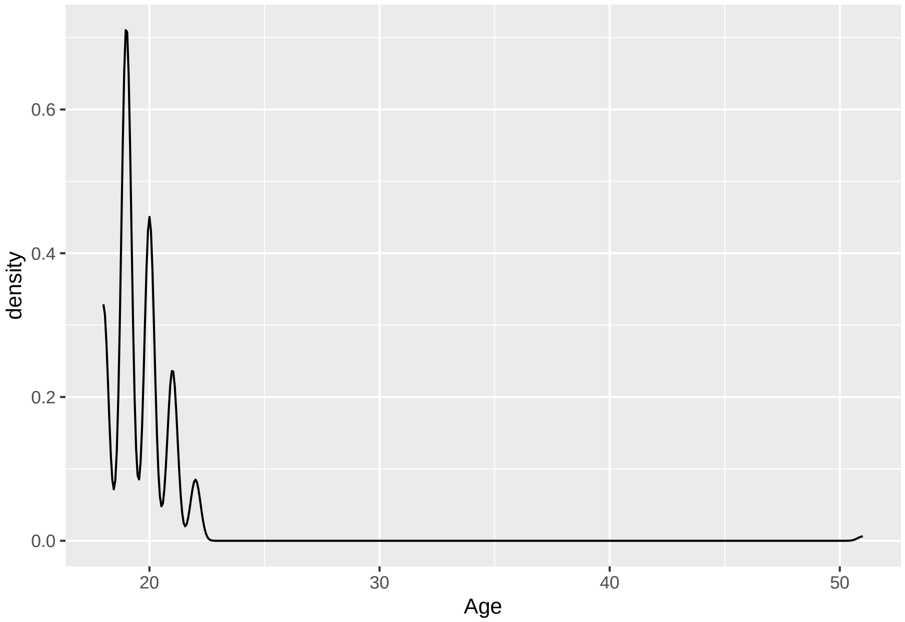


Each bar now contains two ages: the first bar contains the 18 and 19 year olds. Notice how the shape has been changed a little bit from the previous histogram with a different binwidth?

2.3.4 Density plots

Density plots are a variation of histograms, where the plot attempts to use a smooth mathematical function to approximate the shape of the histogram, and is unaffected by binwidth:

```
ggplot(Data,aes(x=Age))+
  geom_density()
```



We can see that 19 and 20 year olds are the most common ages in this data. Be careful in interpreting the values on the veritical axis: these do not represent proportions. A characteristic of density plots is that the area under the plot is always one.

2.4 Bivariate Visualizations

We will now look at visualizations we can create to explore the relationship between two variables. The term bivariate means that we are looking at two variables.

We will be using a new dataset as an example, so we clear the environment:

```
rm(list = ls())
```

We will be using the dataset, `gapminder`, from the `gapminder` package. Install and load the `gapminder` package. Also load the `tidyverse` package (which automatically loads the `ggplot2` package):

```
library(tidyverse)
library(gapminder)
```

We can take a look at the `gapminder` dataset:

```
gapminder[1:15,]
```

```
## # A tibble: 15 x 6
##    country     continent  year lifeExp      pop gdpPercap
##    <fct>       <fct>     <int>   <dbl>    <int>     <dbl>
##  1 Afghanistan Asia       1952    28.8  8425333      779.
##  2 Afghanistan Asia       1957    30.3  9240934      821.
##  3 Afghanistan Asia       1962    32.0 10267083      853.
##  4 Afghanistan Asia       1967    34.0 11537966      836.
##  5 Afghanistan Asia       1972    36.1 13079460      740.
##  6 Afghanistan Asia       1977    38.4 14880372      786.
##  7 Afghanistan Asia       1982    39.9 12881816      978.
##  8 Afghanistan Asia       1987    40.8 13867957      852.
##  9 Afghanistan Asia       1992    41.7 16317921      649.
## 10 Afghanistan Asia       1997    41.8 22227415      635.
## 11 Afghanistan Asia       2002    42.1 25268405      727.
## 12 Afghanistan Asia       2007    43.8 31889923      975.
## 13 Albania     Europe     1952    55.2  1282697     1601.
## 14 Albania     Europe     1957    59.3  1476505     1942.
## 15 Albania     Europe     1962    64.8  1728137     2313.
```

Per the documentation, the variables are

1. `country`
2. `continent`
3. `year`: from 1952 to 2007 in increments of 5 years
4. `lifeExp`: life expectancy at birth, in years
5. `pop`: population of country
6. `gdpPercap`: GDP per capita in US dollars, adjusted for inflation

We notice that data are collected from each country across a number of different years: 1952 to 2007 in increments of five years. For this example, we will mainly focus on the data for the most recent year, 2007:

```
Data<-gapminder%>%
  filter(year==2007)
```

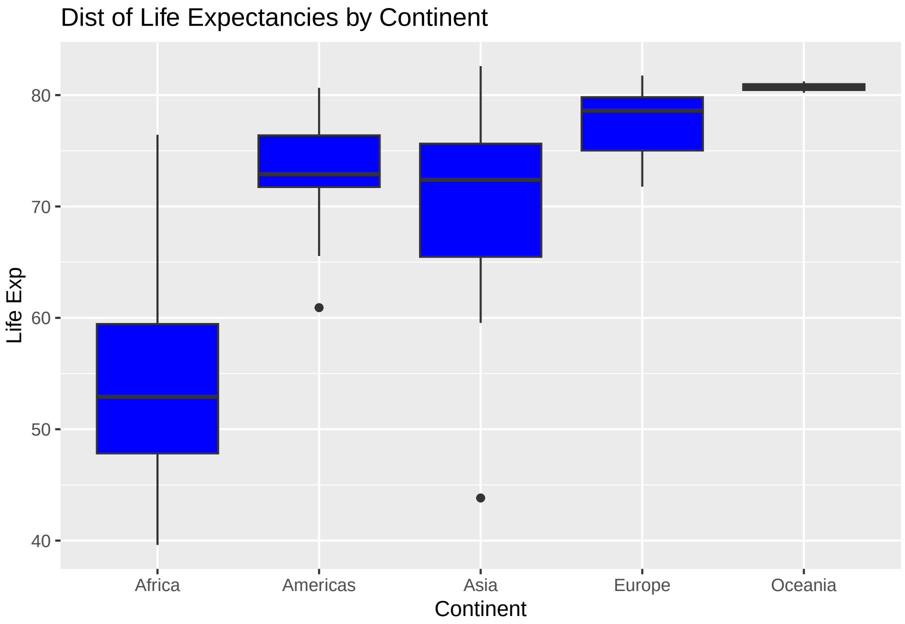
The specific visuals to use will again depend on the type of variables we are using, whether they are categorical or quantitative.

2.4.1 Compare quantitative variable across categories

2.4.1.1 Side by side boxplots

Side by side boxplots are useful to compare a quantitative variable across different classes of a categorical variable. For example, we want to compare life expectancies across the different continents in the year 2007:

```
ggplot(Data, aes(x=continent, y=lifeExp))+
  geom_boxplot(fill="Blue")+
  labs(x="Continent", y="Life Exp", title="Dist of Life Expectancies by Continent")
```

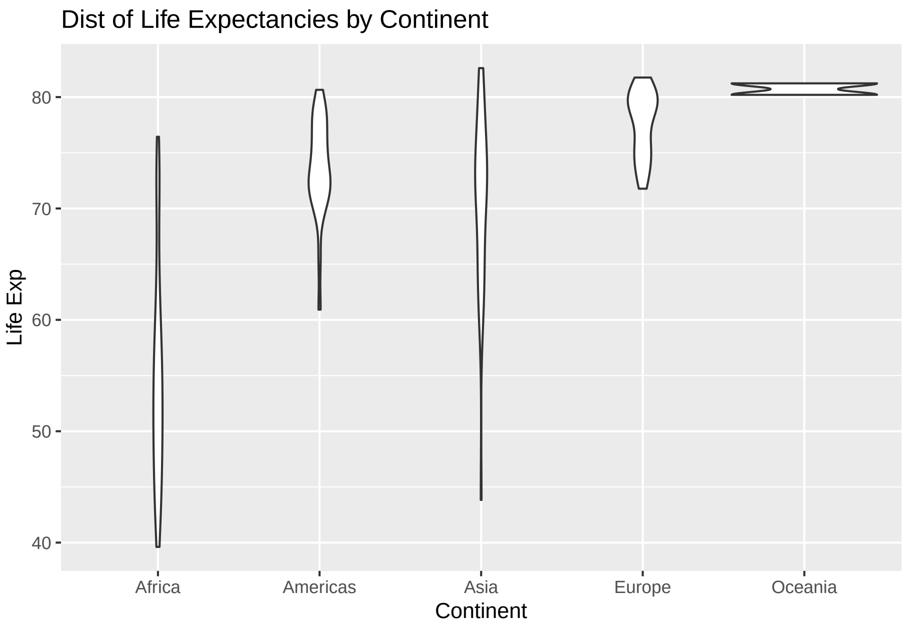


Countries in the Oceania region have long life expectancies with little variation. Comparing the Americas and Asia, the median life expectancies are similar, but the spread is larger for Asia.

2.4.1.2 Violin plots

Violin plots are an alternative to boxplots. To create these plots to compare life expectancies across the different continents in the year 2007:

```
ggplot(Data, aes(x=continent, y=lifeExp))+
  geom_violin()+
  labs(x="Continent", y="Life Exp", title="Dist of Life Expectancies by Continent")
```



The width of the violin informs us which values are more commonly occuring. For example, look at the violin for Europe. The violin is wider at higher life expectancies, so longer life expectancies are more common in European countries.

2.4.2 Summarizing two categorical variables

For this example, we create a new binary variable called `expectancy`, which will be denoted as `low` if the life expectancy in the country is less than 70 years, and `high` otherwise:

```
Data<-Data%>%
  mutate(expectancy=ifelse(lifeExp<70,"Low","High"))
```

2.4.2.1 Two-way tables

Suppose we want to see how `expectancy` varies across the continents. A two-way table can be created for produce counts when two categorical variables are involved:

```
mytab2<-table(Data$continent, Data$expectancy)
##continent in rows, expectancy in columns
mytab2
```

```
##
##              High Low
##   Africa        7  45
##   Americas   22   3
##   Asia       22  11
##   Europe     30   0
##   Oceania     2   0
```

The first variable in `table()` will be placed in the rows, the second variable will be placed in the columns.

From this table, we can see that 22 countries in the Americas have high life expectancies, while 3 countries in the Americas have low life expectancies.

We may be interested in looking at the proportions, instead of counts, of countries in each continent that have high or low life expectancies. To convert this table to proportions, we can use `prop.table()`:

```r
prop.table(mytab2, 1)
```

```
## 
##                 High       Low
##   Africa   0.1346154 0.8653846
##   Americas 0.8800000 0.1200000
##   Asia     0.6666667 0.3333333
##   Europe   1.0000000 0.0000000
##   Oceania  1.0000000 0.0000000
```

We want proportions for each continent, so we want proportions in each row to add up to 1. Therefore, the second argument in `prop.table()` is 1. Enter 2 for this argument if we want the proportions in each column to add up to 1.

As before, to convert to percentages and round to 2 decimal places:

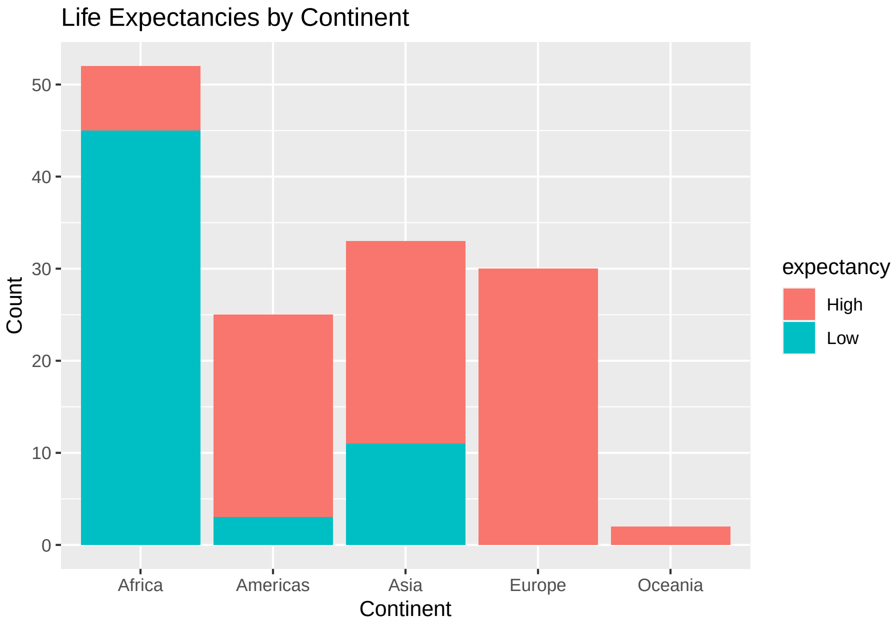
```r
round(prop.table(mytab2, 1) * 100, 2)
```

```
## 
##              High    Low
##   Africa    13.46  86.54
##   Americas  88.00  12.00
##   Asia      66.67  33.33
##   Europe   100.00   0.00
##   Oceania  100.00   0.00
```

2.4.2.2 Bar charts

A stacked bar chart can be used to display the relationship between the binary variable `expectancy` across continents:

```r
ggplot(Data, aes(x=continent, fill=expectancy))+
  geom_bar(position = "stack")+
  labs(x="Continent", y="Count", title="Life Expectancies by Continent")
```

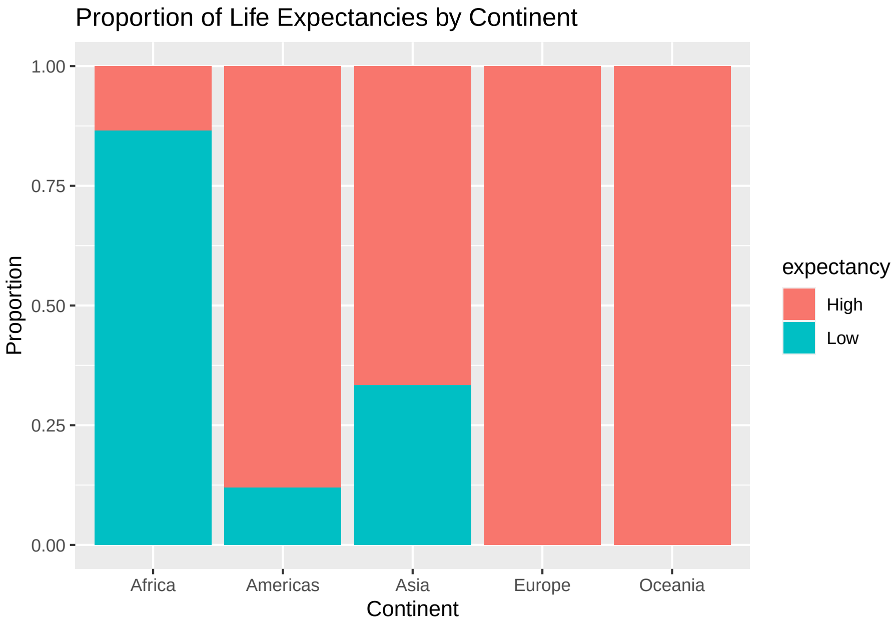


We can see how many countries exist in each continent, and how many of these countries in each continent have high or low life expectancies. For example, there are about 25 countries in the Americas with the majority having high life expectancies.

We can change the way the bar chart is displayed by changing `position` in `geom_bar()` to `position = "dodge"` or `position = "fill"`, the latter being more useful for proportions instead of counts:

```
ggplot(Data, aes(x=continent, fill=expectancy))+
  geom_bar(position = "dodge")
```

```
ggplot(Data, aes(x=continent, fill=expectancy))+
  geom_bar(position = "fill")+
  labs(x="Continent", y="Proportion",
       title="Proportion of Life Expectancies by Continent")
```

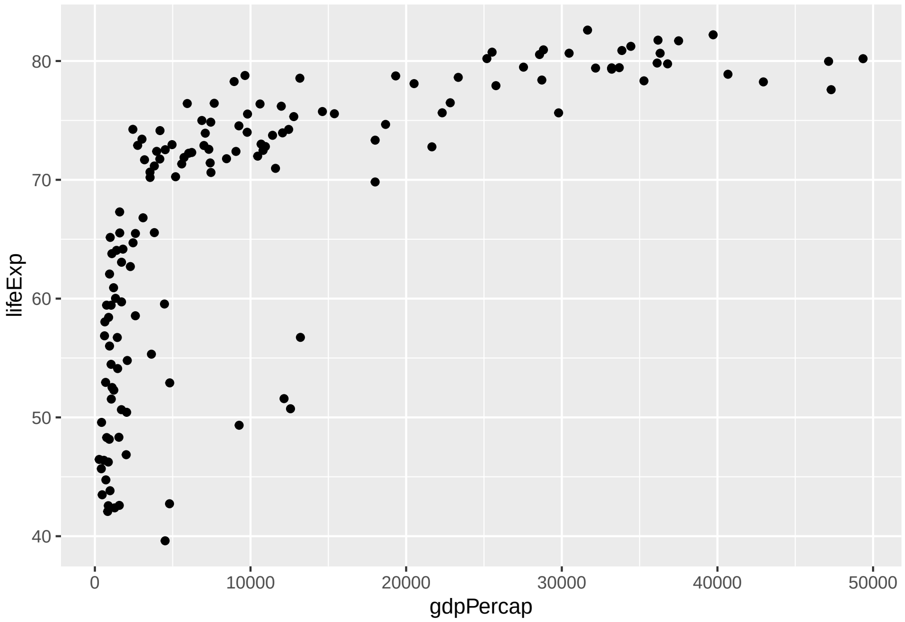


2.4.3 Summarizing two quantitative variables

2.4.3.1 Scatterplots

Scatterplots are the standard visualization when two quantitative variables are involved. To create a scatterplot for life expectancy against GDP per capita:

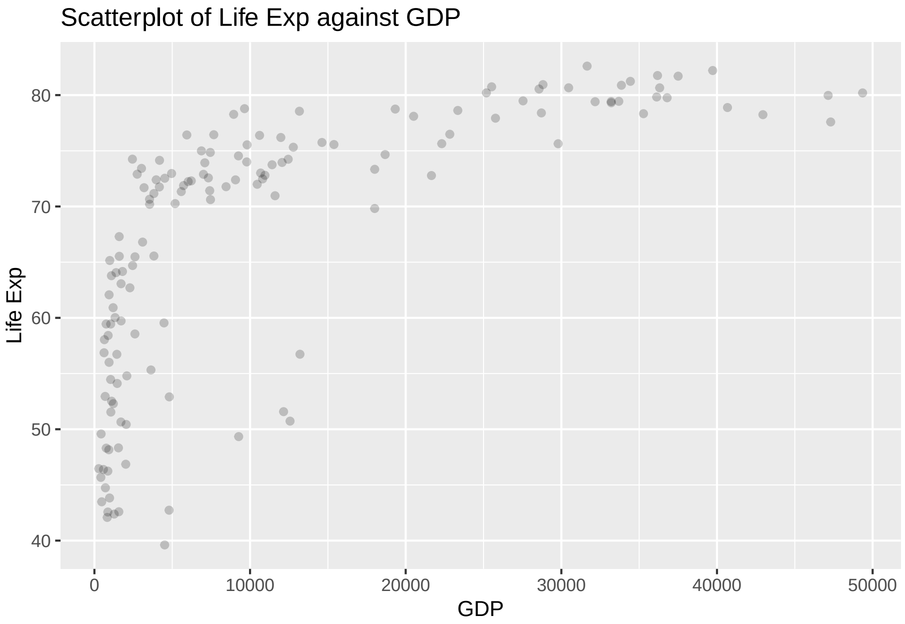
```
ggplot(Data, aes(x=gdpPercap,y=lifeExp))+
  geom_point()
```



We see a curved relationship between life expectancies and GDP per capita. Countries with higher GDP per capita tend to have longer life expectancies.

When there are many observations, plots on the scatterplot may actually overlap each other. To have a sense of how many of these exist, we can add a transparency scale called `alpha=0.2` inside `geom_point()`:

```
ggplot(Data, aes(x=gdpPercap,y=lifeExp))+
  geom_point(alpha=0.2)+
  labs(x="GDP", y="Life Exp",
       title="Scatterplot of Life Exp against GDP")
```



The default value for `alpha` is 1, which means the points are not at all transparent. The closer this value is to 0, the more transparent the points are. A darker point indicates more observations with those specific values on both variables.

2.5 Multivariate Visualizations

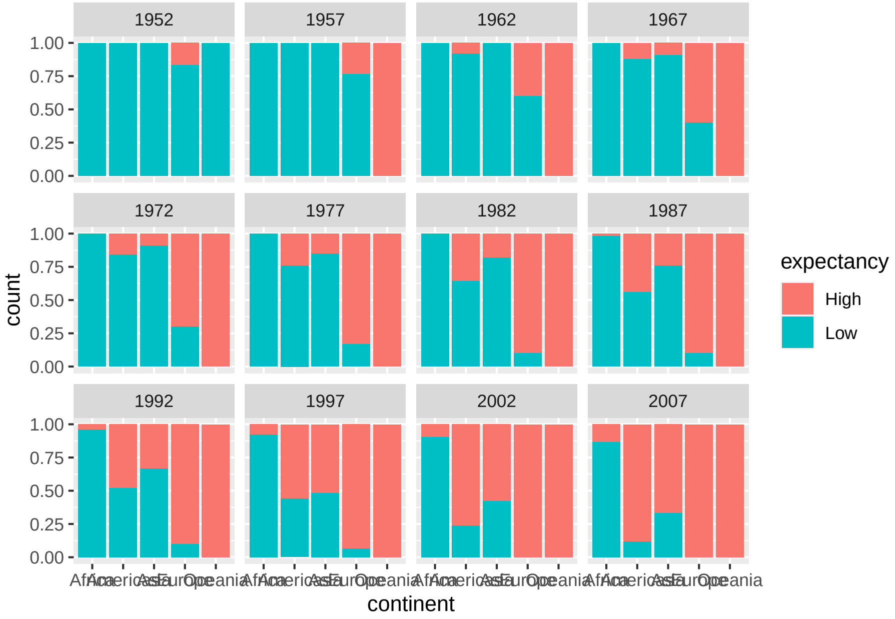
We will now look at visualizations we can create to explore the relationship between multiple (more than two) variables. The term multivariate means that we are looking at more than two variables.

2.5.1 Bar charts

Previously, we created a bar chart to look at how `expectancy` varies across the continents. Suppose we want to see how these bar graphs vary across the years, so we use the `year` variable as a third variable via a layer `facet_wrap`:

```
##another data frame across all years plus a binary variable
##for expectancy
Data.all<-gapminder%>%
  mutate(expectancy=ifelse(lifeExp<70,"Low","High"))

ggplot(Data.all,aes(x=continent, fill=expectancy))+
  geom_bar(position = "fill")+
  facet_wrap(~year)
```

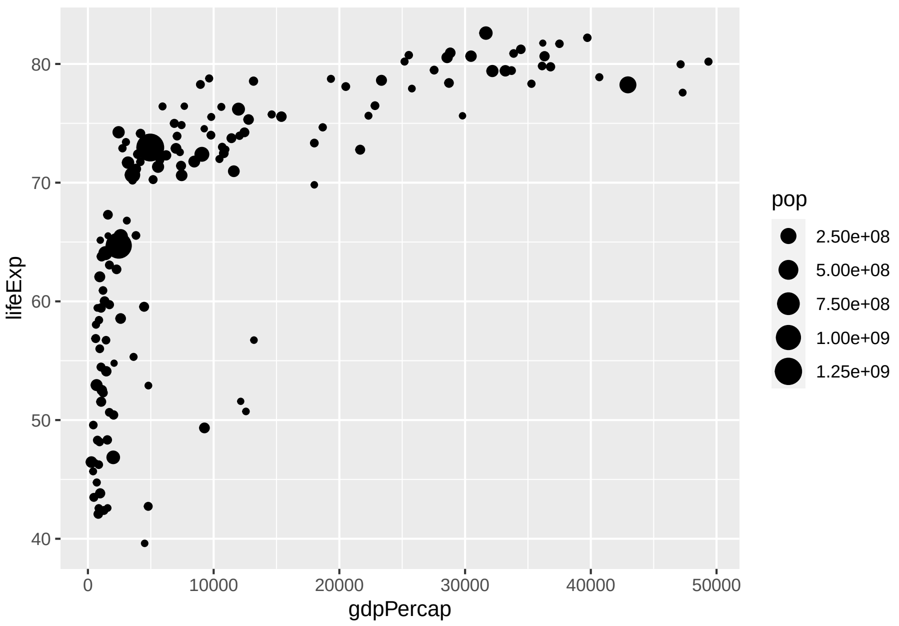


Notice that three categorical variables are summarized in this bar chart. Is there something that can be done to improve this bar chart? How would you make this improvement?

2.5.2 Scatterplots

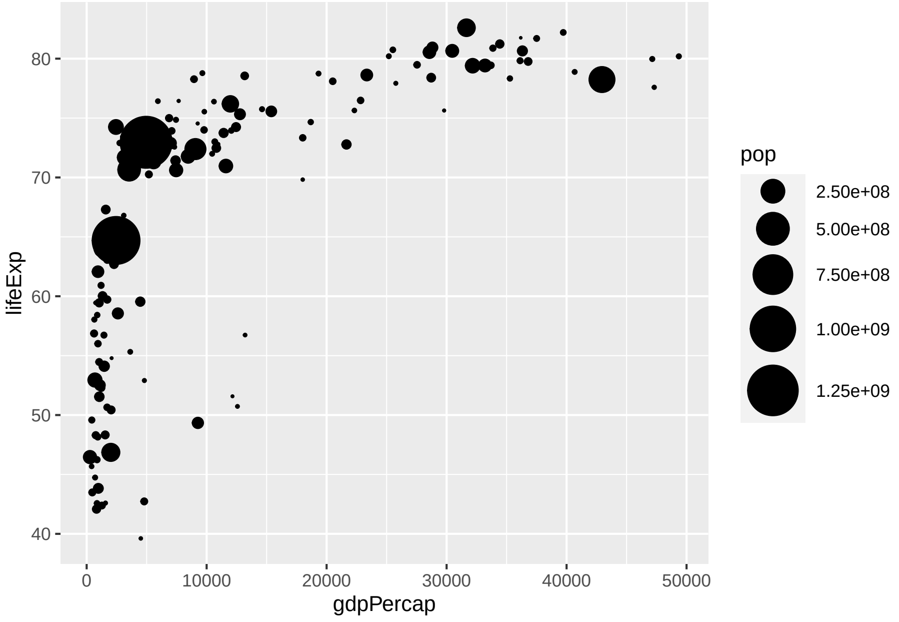
Previously, we created a scatterplot of life expectancy against GDP per capita. We can include another quantitative variable in the scatterplot, by using the size of the plots. We can use the size of the plots to denote the population of the countries. This is supplied via `size` in `aes()`:

```
ggplot(Data, aes(x=gdpPercap, y=lifeExp, size=pop))+
  geom_point()
```



We can adjust the size of the plots by adding a layer `scale_size()`:

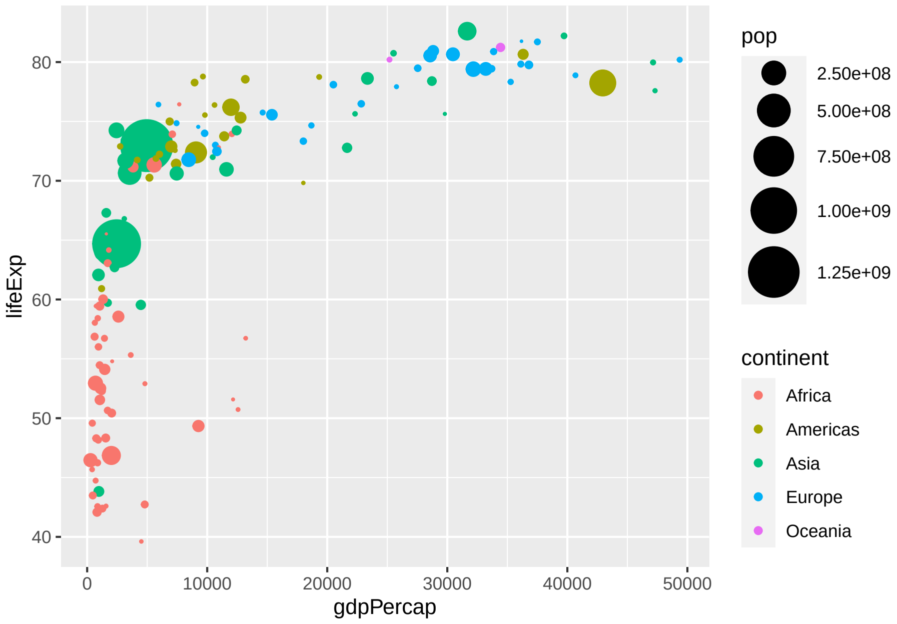
```
ggplot(Data, aes(x=gdpPercap, y=lifeExp, size=pop))+
  geom_point()+
  scale_size(range = c(0.1,12))
```



This scatterplot summarizes three quantitative variables.

We can use different-colored plots to denote which continent each point belongs to:

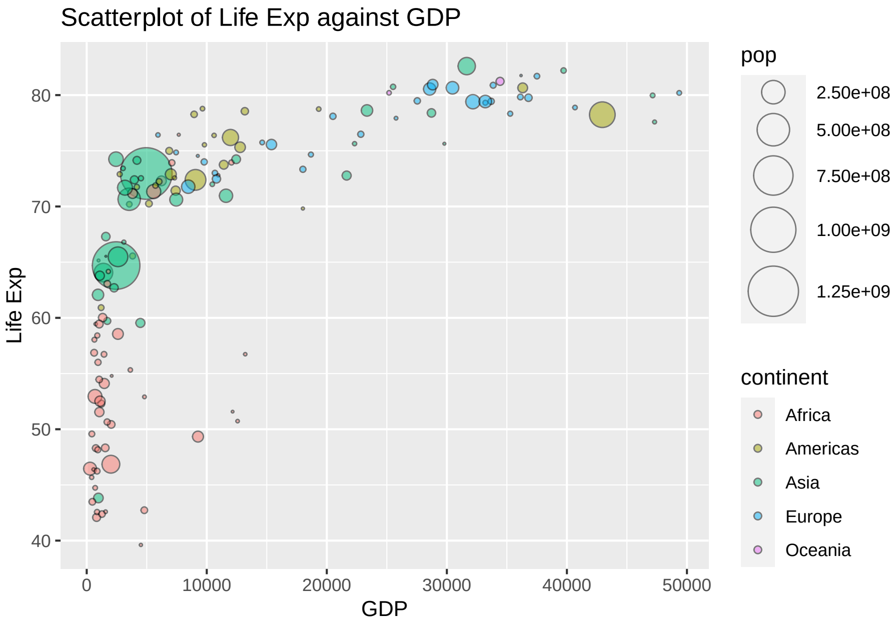
```
ggplot(Data, aes(x=gdpPercap, y=lifeExp, size=pop, color=continent))+
  geom_point()+
  scale_size(range = c(0.1,12))
```



This scatterplot summarizes three quantitative variables and one categorical variable.

We can adjust the plots by changing its shape and making it more translucent via `shape` and `alpha` in `aes()`:

```
ggplot(Data, aes(x=gdpPercap, y=lifeExp, size=pop, fill=continent))+
  geom_point(shape=21, alpha=0.5)+
  scale_size(range = c(0.1,12))+
  labs(x="GDP", y="Life Exp", title="Scatterplot of Life Exp against GDP")
```


Scatterplot of Life Exp against GDP
Life Exp
GDP
80
70
60
50
40
0
10000
20000
30000
40000
50000
pop
2.50e+08
5.00e+08
7.50e+08
1.00e+09
1.25e+09
continent
Africa
Americas
Asia
Europe
Oceania

Chapter 3

Basics with Simple Linear Regression (SLR)

3.1 Introduction

We will start this module by introducing the simple linear regression model. Simple linear regression uses the term "simple," because it concerns the study of only one predictor variable with one quantitative response variable. In contrast, multiple linear regression, which we will study in future modules, uses the term "multiple," because it concerns the study of two or more predictor variables with one quantitative response variable. We start with simple linear regression as it is much easier to visualize concepts in regression models when there is only one predictor variable.

For the time being, we will only consider predictor variables that are quantitative. We will consider predictor variables that are categorical in future modules.

The most common way of visualizing the relationship between one quantitative predictor variable and one quantitative response variable is with a scatter plot. In the simulated example below, we have data from 6000 UVa undergraduate students on the amount of time they spend studying in a week (in minutes), and how many courses they are taking in the semester (3 or 4 credit courses).

```r
##create dataframe
df<-data.frame(study,courses)

##fit regression
result<-lm(study~courses, data=df)
```

```r
##create scatterplot with regression line overlaid
plot(df$courses, df$study, xlab="# of Courses", ylab="Study Time (Mins)")
```

```
abline(result)
```

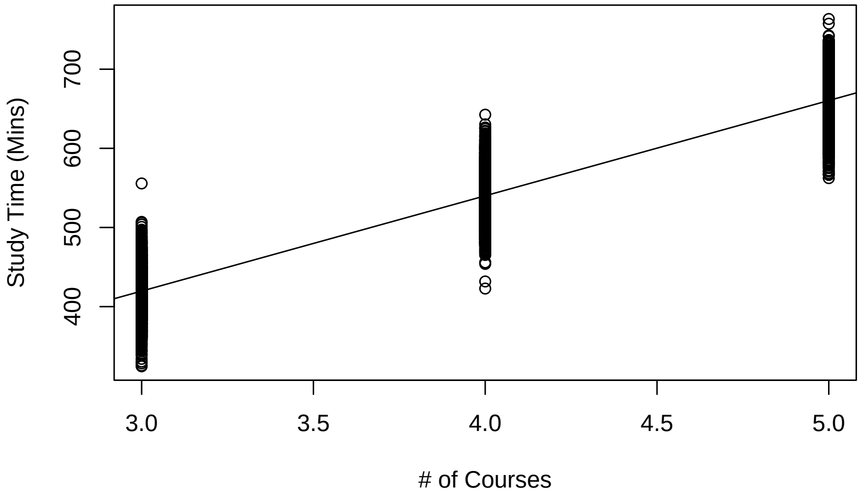


Figure 3.1: Scatterplot of Study Time against Number of Courses Taken

Questions that we may have include:

- Are study time and the number of courses taken related to one another?
- How strong is this relationship?
- Could we use the data to make a prediction for the study time of a student who is not in this scatterplot?
- How confident are we of the prediction?

These questions can be answered using simple linear regression.

Note that we will only be learning about models with just one response variable. We will not cover multivariate regression, which is used when there is more than one response variable. There may be some confusion between “multiple” linear regression and “multivariate” regression due to the closeness in terminology.

3.1.1 Basic Ideas with Statistics

3.1.1.1 Population vs Sample

Statistical methods are usually used to make inferences about the **population** based on information from a **sample**.

- A sample is the collection of units that is actually measured or surveyed in a study.
- The population includes all units of interest.

In the study time example above, the population is all UVa undergraduate students, while the sample is the 6000 students that we have data on and are displayed on the scatterplot.

3.1.1.2 Parameters vs Statistics

- **Parameters** are numerical quantities that describe a population.
- **Statistics** are numerical quantities that describe a sample.

In the study time example, an example of a parameter will be the average study time among all UVa undergraduate students (called the population mean), and an example of a statistic will be the average study time among the 6000 UVa students we have data on (called the sample mean).

Notice that in real life, we will rarely know the actual numerical value of a parameter. So we use the numerical value of the statistic to **estimate** the unknown numerical value of the corresponding parameter.

We also have different notation for parameters and statistics. For example,

- the population mean is denoted as μ.
- the sample mean is denoted as $\bar{x}$.

We say that $\bar{x}$ is an **estimator** of μ.

It is important to pay attention to whether we are describing a statistic (a known value that can be calculated) or a parameter (an unknown value).

3.1.2 Motivation

Linear regression models generally have two primary uses:

1. **Prediction**: Predict a future value of a response variable, using information from predictor variables.
2. **Association**: Quantify the relationship between variables. How does a change in the predictor variable change the value of the response variable?

We always distinguish between a **response variable, denoted by** y, and a **predictor variable, denoted by** x. In most statistical models, we say that the response variable can be approximated by some **mathematical function, denoted by** f, of the predictor variable, i.e.

$$y \approx f(x).$$

Oftentimes, we write this relationship as

$$y = f(x) + \epsilon,$$

where ϵ **denotes a random error term**, with a mean of 0. The error term cannot be predicted based on the data we have.

There are various statistical methods to estimate f. Once we estimate f, we can use our method for prediction and / or association.

Using the study time example above:

- a prediction example: a student intends to take 4 courses in the semester. What is this student's predicted study time, on average?
- an association example: we want to see how taking more courses increases study time.

3.1.2.1 Practice questions

In the examples below, are we using a regression model for prediction or for association?

1. It is early in the morning and I am heading out for the rest of the day. I want to know the weather forecast for the rest of the day so I know what to wear.
2. An executive for a sports league wants to assess how increasing the length of commercial breaks may impact the enjoyment of sports fans who watch games on TV.
3. The Education Secretary would like to evaluate how certain factors such as use of technology in classrooms and investment in teacher training and teacher pay are associated with reading skills of students.
4. When buying a home, the prospective buyer would like to know if the home is under- or over- priced, given its characteristics.

3.2 Simple Linear Regression (SLR)

In simple linear regression (SLR), the function f that relates the predictor variable with the response variable is typically $\beta_0+\beta_1 x$. Mathematically, we express this as

$$y \approx \beta_0 + \beta_1 x,$$

or in other words, that the response variable has an approximately linear relationship with the predictor variable.

In SLR, this relationship is more explicitly formulated as the **simple linear regression equation**:

$$E(y|x) = \beta_0 + \beta_1 x. \tag{3.1}$$

- β_0 and β_1 are parameters in the SLR equation, and we want to estimate them.
- These parameters are sometimes called **regression coefficients**.
- β_1 is also called the **slope. It denotes the change in y, on average, when x increases by one unit.**
- β_0 is also called the **intercept. It denotes the average of y when $x = 0$.**
- The notation on the left hand side of (3.1) denotes the **expected value** of the response variable, for a fixed value of the predictor variable. What (3.1) implies is that, for each value of the predictor variable x, the expected value of the response variable y is $\beta_0 + \beta_1 x$. The expected value is also the population mean. Applying (3.1) to our study time example, it implies that:
  - for students who take 3 courses, their expected study time is equal to $\beta_0 + 3\beta_1$,
  - for students who take 4 courses, their expected study time is equal to $\beta_0 + 4\beta_1$,
  - for students who take 5 courses, their expected study time is equal to $\beta_0 + 5\beta_1$.

So $f(x) = \beta_0 + \beta_1 x$ gives us the value of the expected value of the response variable for a specific value of the predictor variable. But, for each value of the predictor variable, the value of the response variable is not a constant. We say that for each value of x, the response variable y has some variance. The variance of the response variable for each value of x is the same as the variance of the error term, ϵ. Thus we have the **simple linear regression model**

$$y = \beta_0 + \beta_1 x + \epsilon. \tag{3.2}$$

We need to make some assumptions for the error term ϵ. Generally, the assumptions are:

1. The errors have mean 0.
2. The **errors have variance denoted by σ^2.** Notice this variance is constant.
3. The errors are independent.
4. The errors are normally distributed.

From (3.2), notice we have another parameter, σ^2.

We will go into more detail about what these assumptions mean, and how to assess whether they are met, in module 5.

What these assumptions mean is that for each value of the predictor variable x, the response variable:

1. follows a normal distribution,
2. with mean equal to $\beta_0 + \beta_1 x$,
3. and variance equal to σ^2.

Using our study time example, it means that:

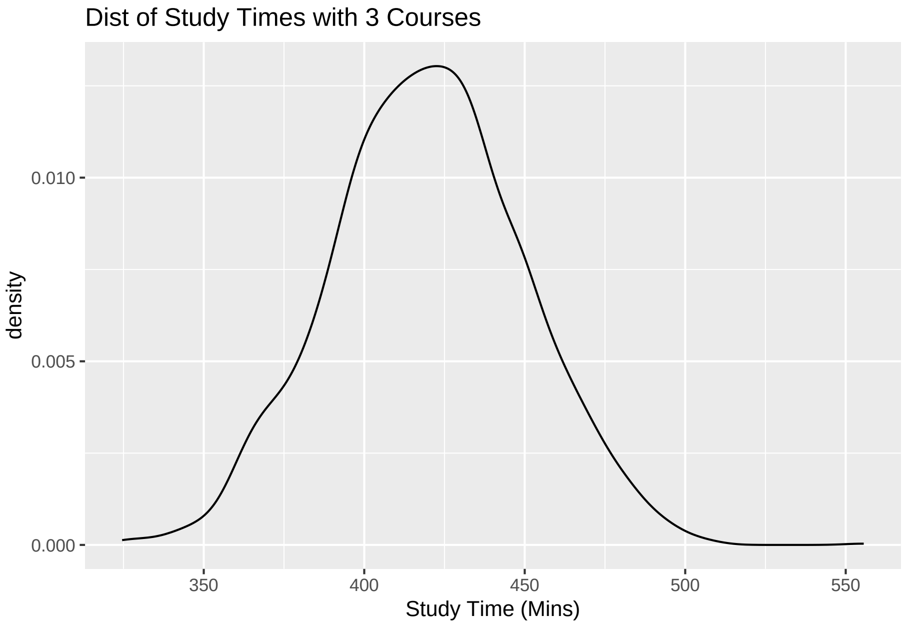
- for students who take 3 courses, the distribution of their study times is $N(\beta_0 + 3\beta_1, \sigma^2)$.
- for students who take 4 courses, the distribution of their study times is $N(\beta_0 + 4\beta_1, \sigma^2)$.
- for students who take 5 courses, the distribution of their study times is $N(\beta_0 + 5\beta_1, \sigma^2)$.

So if we were to subset our dataframe into three subsets, one with students who take 3 courses, another subset for students who take 4 courses, and another subset for students who take 5 courses, and then create a density plot of study times for each subset, each density plot should follow a normal distribution, with different means, and the same spread.

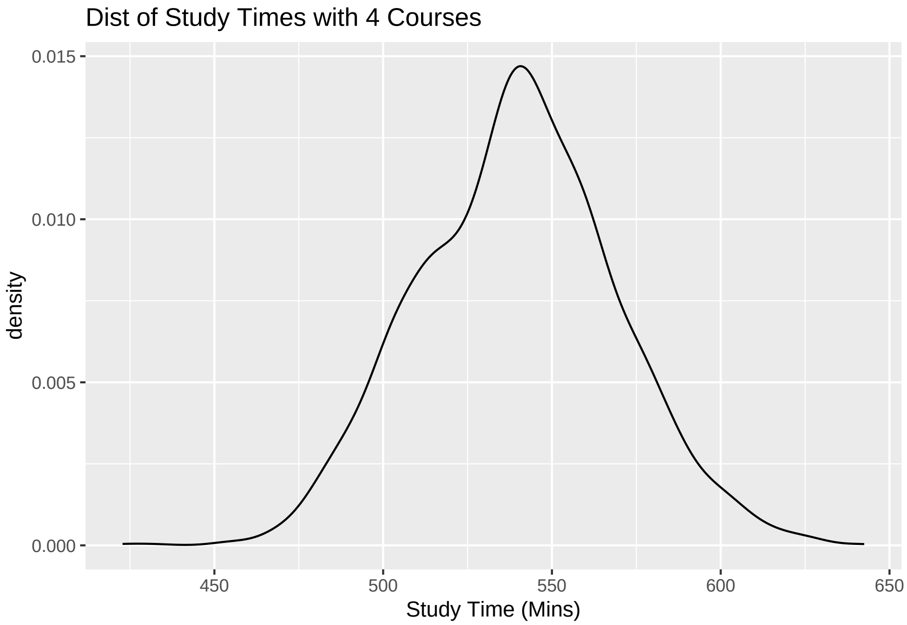
Let us take a look at these density plots next.

```r
library(tidyverse)
```

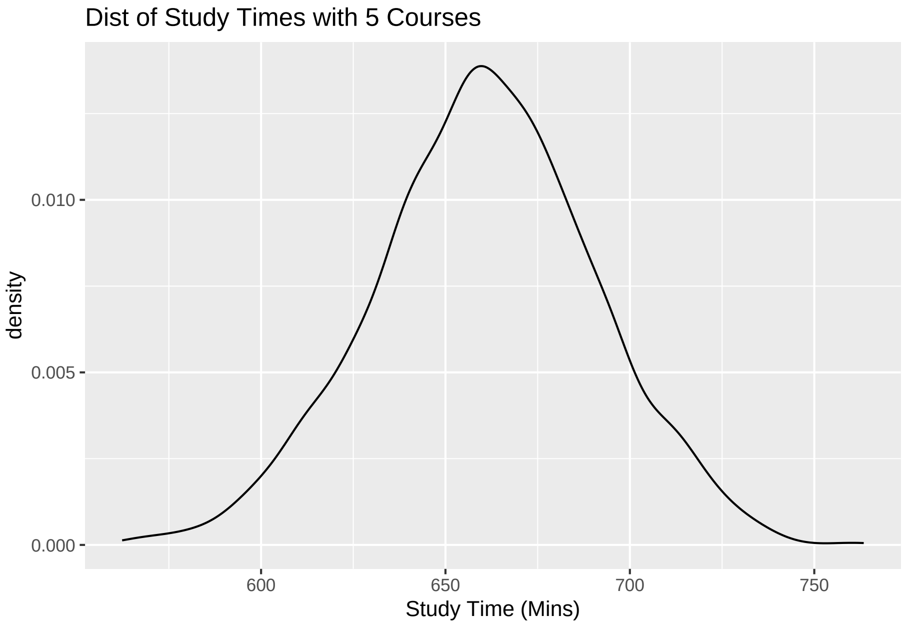
```r
##subset dataframe
x.3<-df[which(df$courses==3),]
##density plot of study time for students taking 3 courses
ggplot(x.3,aes(x=study))+
  geom_density()+
  labs(x="Study Time (Mins)", title="Dist of Study Times with 3 Courses")
```



```r
##subset dataframe
x.4<-df[which(df$courses==4),]
##density plot of study time for students taking 4 courses
ggplot(x.4,aes(x=study))+
  geom_density()+
  labs(x="Study Time (Mins)", title="Dist of Study Times with 4 Courses")
```



```
##subset dataframe
x.5<-df[which(df$courses==5),]
##density plot of study time for students taking 5 courses
ggplot(x.5,aes(x=study))+
  geom_density()+
  labs(x="Study Time (Mins)", title="Dist of Study Times with 5 Courses")
```



Notice all of these plots are normal, with different means (centers), and similar spreads.

Please see the associated video for more explanation about the distribution of the response variable, for each value of the predictor variable, in an SLR setting.

3.3 Estimating Regression Coefficients in SLR

From (3.1) and (3.2), we noted that we have to estimate the regression coefficients β_0, β_1 as well as the parameter σ^2 associated with the error term. As mentioned earlier, we are unable to obtain numerical values of these parameters as we do not have data from the entire population. So what we do is use the data from our sample to estimate these parameters.

We estimate β_0, β_1 using $\hat{\beta}_0, \hat{\beta}_1$ based on a sample of observations (x_i, y_i) of size n.

The subscripts associated with the response and predictor variables denote which data point that value belongs to. Let us take a look at the first few rows of the data frame for the study time example:

```
head(df)
```

```
##      study courses
## 1 429.8311       3
## 2 458.4588       3
## 3 391.9406       3
## 4 378.0196       3
## 5 397.9856       3
## 6 405.7145       3
```

For example, x_1 denotes the number of courses taken by student number 1 in the dataframe, which is 3. y_4 denotes the study time for student number 4 in the dataframe, which is 378.0196456.

Following (3.1) and (3.2), the sample versions are

$$\hat{y} = \hat{\beta}_0 + \hat{\beta}_1 x \tag{3.3}$$

and

$$y = \hat{\beta}_0 + \hat{\beta}_1 x + e \tag{3.4}$$

respectively. (3.3) is called the **estimated SLR equation**, or **fitted SLR equation**. (3.4) is called the **estimated SLR model**.

$\hat{\beta}_1, \hat{\beta}_0$ are the estimators for β_1, β_0 respectively. These estimators can be interpreted in the following manner:

- $\hat{\beta}_1$ **denotes the change in the predicted** y **when** x **increases by 1 unit. Alternatively, it denotes the change in** y**, on average, when** x **increases by 1 unit.**
- $\hat{\beta}_0$ **denotes the predicted** y **when** $x = 0$**. Alternatively, it denotes the average of** y **when** $x = 0$**.**

From (3.4), notice we use e **to denote the residual**, or in other words, the "error" in the sample.

From (3.3) and (3.4), we have the following quantities that we can compute:

$$\text{Predicted/Fitted values: } \hat{y}_i = \hat{\beta}_0 + \hat{\beta}_1 x_i. \tag{3.5}$$

$$\text{Residuals: } e_i = y_i - \hat{y}_i. \tag{3.6}$$

$$\text{Sum of Squared Residuals: } SS_{res} = \sum_{i=1}^{n} (y_i - \hat{y}_i)^2. \tag{3.7}$$

We compute the estimated coefficients $\hat{\beta}_1, \hat{\beta}_0$ using the **method of least squares**, i.e. choose the numerical values of $\hat{\beta}_1, \hat{\beta}_0$ that minimize SS_{res} as given in (3.7).

By minimizing SS_{res} with respect to $\hat{\beta}_0$ and $\hat{\beta}_1$, the estimated coefficients in the simple linear regression equation are

$$\hat{\beta}_1 = \frac{\sum\limits_{i=1}^{n}(x_i - \bar{x})(y_i - \bar{y})}{\sum\limits_{i=1}^{n}(x_i - \bar{x})^2} \tag{3.8}$$

and

$$\hat{\beta}_0 = \bar{y} - \hat{\beta}_1\bar{x} \tag{3.9}$$

$\hat{\beta}_1, \hat{\beta}_0$ are called **least squares estimators**.

The minimization of SS_{res} with respect to $\hat{\beta}_0$ and $\hat{\beta}_1$ is done by taking the partial derivatives of (3.7) with respect to $\hat{\beta}_1$ and $\hat{\beta}_0$, setting these two partial derivatives equal to 0, and solving these two equations for $\hat{\beta}_1$ and $\hat{\beta}_0$.

Let's take a look at the estimated coefficients for our study time example:

```
##fit regression
result<-lm(study~courses, data=df)
##print out the estimated coefficients
result
```

```
##
## Call:
## lm(formula = study ~ courses, data = df)
##
## Coefficients:
## (Intercept)      courses
##       58.45       120.39
```

From our sample of 6000 students, we have

- $\hat{\beta}_1 = 120.3930985$. The predicted study time increases by 120.3930985 minutes for each additional course taken.
- $\hat{\beta}_0 = 58.4482853$. The predicted study time is 58.4482853 when no courses are taken. Notice this value does not make sense, as a student cannot be taking 0 courses. If you look at our data, the number of courses taken is 3, 4, or 5. So we should only use our regression when $3 \leq x \leq 5$. We cannot use it for values of x outside the range of our data. Making predictions of the response variable for predictors outside the range of the data is called **extrapolation** and should not be done.

3.4 Estimating Variance of Errors in SLR

The estimator of σ^2, the variance of the error terms (also the variance of the probability distribution of y given x) is

$$s^2 = MS_{res} = \frac{SS_{res}}{n-2} = \frac{\sum_{i=1}^{n} e_i^2}{n-2}, \tag{3.10}$$

where MS_{res} is the called the **mean squared residuals**.

σ^2, the variance of the error terms, measures the spread of the response variable, for each value of x. The smaller this is, the closer the data points are to the regression equation.

3.4.1 Practice questions

Take a look at the scatterplot of study time against number of courses taken, Figure 3.1. On this plot, label the following:

- estimated SLR equation
- the fitted value when $x = 3$, $x = 4$, and $x = 5$.
- the residual for any data point on the plot of your choosing.

Try these on your own first, then view the associated video to see if you labeled the plot correctly!

3.5 Assessing Linear Association

As noted earlier, the variance of the error terms inform us how close the data points are to the estimated SLR equation. The smaller the variance of the error terms, the closer the data points are to the estimated SLR equation. This in turn implies the linear relationship between the variables is stronger.

We will learn about some common measures that are used to quantify the strength of the linear relationship between the response and predictor variables. Before we do that, we need to define some other terms.

3.5.1 Sum of squares

$$\text{Total Sum of Squares: } SS_T = \sum_{i=1}^{n}(y_i - \bar{y})^2. \tag{3.11}$$

Total sum of squares is defined as the **total variance in the response variable**. The larger this value is, the larger the spread is of the response variable.

$$\text{Regression sum of squares: } SS_R = \sum_{i=1}^{n}(\hat{y_i} - \bar{y})^2. \tag{3.12}$$

Regression sum of squares is defined as the **variance in the response variable that can be explained by our regression**.

We also have residual sum of squares, SS_{res}. Its mathematical formulation is given in (3.7). It is defined as the **variance in the response variable that cannot be explained by our regression**.

It can be shown that

$$SS_T = SS_R + SS_{res}. \tag{3.13}$$

Each of the sums of squares has its associated **degrees of freedom (df)**:

- df for SS_R: $df_R = 1$
- df for SS_{res}: $df_{res} = n - 2$
- df for SS_T: $df_T = n - 1$

Please see the associated video for more explanation about the concept behind degrees of freedom.

3.5.2 ANOVA Table

Information regarding the sums of squares is usually presented in the form of an **ANOVA (analysis of variance) table**:

Source of Variation	SS	df	MS	F
Regression	$SS_R = \sum (\hat{y}_i - \bar{y})^2$	$df_R = 1$	$MS_R = \frac{SS_R}{df_R}$	$\frac{MS_R}{MS_{res}}$
Error	$SS_{res} = \sum (y_i - \hat{y}_i)^2$	$df_{res} = n - 2$	$MS_{res} = \frac{SS_{res}}{df_{res}}$	***
Total	$SS_T = \sum (y_i - \bar{y})^2$	$df_T = n - 1$	***	***

Note:

- Dividing each sum of square with its corresponding degrees of freedom gives the corresponding mean square.
- In the last column, we report an F statistic, which equal to $\frac{MS_R}{MS_{res}}$. This F statistic is associated with an **ANOVA F test**, which we will look at in more detail in the next subsection.

To obtain the ANOVA table for our study time example:

```r
anova(result)
```

```
## Analysis of Variance Table
##
## Response: study
##             Df   Sum Sq  Mean Sq F value    Pr(>F)
## courses      1 57977993 57977993   65404 < 2.2e-16 ***
## Residuals 5998  5317017      886
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

Notice that R does not print out the information for the line regarding SS_T.

3.5.3 ANOVA F Test

In SLR, the ANOVA F statistic from the ANOVA table can be used to test if the slope of the SLR equation is 0 or not. In words, this means that whether there is a linear association between the variables or not. If the slope is 0, it means that changes in the value of the predictor variable do not change the value of the response variable, on average; hence the variables are not linearly associated.

The null and alternative hypotheses are:

$$H_0 : \beta_1 = 0, H_a : \beta_1 \neq 0.$$

The test statistic is

$$F = \frac{MS_R}{MS_{res}} \tag{3.14}$$

and is compared with an $F_{1,n-2}$ distribution. Note that $F_{1,n-2}$ is read as an **F distribution with 1 and $n-2$ degrees of freedom**.

Going back to the study time example, the F statistic is 6.5403586×10^4. The critical value can be found using

```
qf(1-0.05, 1, 6000-2)
```

```
## [1] 3.84301
```

Since our test statistic is larger than the critical value, we reject the null hypothesis. Our data support the claim that the slope is different from 0, or in other words, that there is a linear association between study time and number of courses taken.

3.5.4 Coefficient of determination

The **coefficient of determination, R^2,** is

$$R^2 = \frac{SS_R}{SS_T} = 1 - \frac{SS_{res}}{SS_T}. \tag{3.15}$$

R^2 is an indication of how well the data fits our model. In the context of simple linear regression, it denotes **the proportion of variance in the response variable that is explained by the predictor**.

A few notes about R^2:

- $0 \leq R^2 \leq 1$.
- Values closer to 1 indicate a better fit; values closer to 0 indicate a poorer fit.
- Sometimes reported as a percentage.

To obtain R^2 for our study time example:

```r
anova.tab<-anova(result)
##SST not provided, so we add up SSR and SSres
SST<-sum(anova.tab$"Sum Sq")
##R2
anova.tab$"Sum Sq"[1]/SST
```

```
## [1] 0.9159963
```

This implies that the proportion of variance in study time that can be explained by the number of courses taken is 0.9159963.

3.5.5 Correlation

A measure used to quantify the strength of the linear association between two quantitative variables is the **sample correlation**. The sample correlation, $\text{Corr}(x, y)$ or r, is given by

$$r = \frac{\sum\limits_{i=1}^{n}(x_i - \bar{x})(y_i - \bar{y})}{\sqrt{\sum\limits_{i=1}^{n}(x_i - \bar{x})^2(y_i - \bar{y})^2}}. \tag{3.16}$$

A few notes about r:

- $-1 \leq r \leq 1$.
- Sign of correlation indicates direction of association. A positive value indicates a positive linear association: as the predictor variable increases, so does the response variable, on average. A negative value indicates a negative linear association: as the predictor variable increases, the response variable decreases, on average.

- Values closer to 1 or -1 indicate a stronger linear association; values closer to 0 indicate a weaker linear association.
- In SLR, it turns out that $r^2 = R^2$.

Using our study time example, the correlation between study time and number of courses taken is

```r
cor(df$study, df$courses)
```

```
## [1] 0.9570769
```

This value indicates a very strong and positive linear association between study time and number of courses taken (remember that this is simulated data and is not real).

3.5.5.1 How strong is strong?

A question that is often raised is how large should the magnitude of the sample correlation be for it to be considered strong? The answer is: it depends on the context. If you are conducting an experiment that is governed by scientific laws (e.g an experiment verifying Newton's 2nd law that $F = ma$), we should expect an extremely high correlation. A correlation of 0.9 in such an instance may be considered weak. The value of the correlation you have should be compared with correlations from similar studies in that domain to determine if it is strong or not.

3.6 A Word of Caution

To be able to use the measures we have learned (such as correlation, R^2) and to interpret the estimated regression coefficients, we must verify via a scatterplot that the association between the two variables is approximately linear. If we see a non linear pattern in the scatterplot, we should not use or interpret these values. We will learn how to remedy the situation if we see a non linear pattern in the scatterplot in module 5.

Please see the associated video for a demonstration on how not looking at the scatterplot can lead to misleading interpretations.

3.7 R Tutorial

For this tutorial, we will work with the dataset `elmhurst` from the `openintro` package in R.

```r
library(tidyverse)
library(openintro)
Data<-openintro::elmhurst
```

Type `?openintro::elmhurst` to read the documentation for datasets in R. Always seek to understand the background of your data! The key pieces of information are:

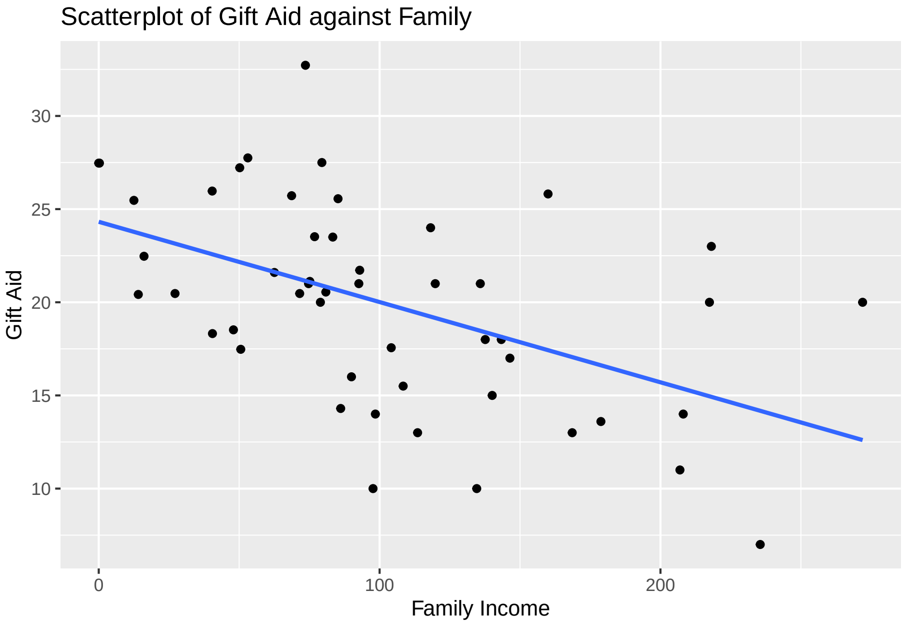
- A random sample of 50 students (all freshman from the 2011 class at Elmhurst College).
- Family income of the student (units are missing).
- Gift aid, in $1000s.

We want to explore how family income may be related to gift aid, in a simple linear regression framework.

Visualization

We should always verify with scatterplot that the relationship is (approximately) linear before proceeding with correlation and simple linear regression!

```
##scatterplot of gift aid against family income
ggplot2::ggplot(Data, aes(x=family_income,y=gift_aid))+
  geom_point()+
  geom_smooth(method = "lm", se=FALSE)+
  labs(x="Family Income", y="Gift Aid", title="Scatterplot of Gift Aid against Family")
```



We note that the observations are fairly evenly scattered on both sides of the regression line, so a linear association exists. We see a negative linear association. As family income increases, the gift aid, on average, decreases.

We also do not see any observation with weird values that may warrant further investigation.

Regression

We use the `lm()` function to fit a regression model:

```r
##regress gift aid against family income
result<-lm(gift_aid~family_income, data=Data)
```

Use the `summary()` function to display relevant information from this regression:

```r
##look at information regarding regression
summary(result)
```

```
##
## Call:
## lm(formula = gift_aid ~ family_income, data = Data)
##
## Residuals:
##      Min       1Q   Median       3Q      Max
## -10.1128  -3.6234  -0.2161   3.1587  11.5707
##
## Coefficients:
##               Estimate Std. Error t value Pr(>|t|)
## (Intercept)   24.31933    1.29145  18.831  < 2e-16 ***
## family_income -0.04307    0.01081  -3.985 0.000229 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 4.783 on 48 degrees of freedom
## Multiple R-squared:  0.2486, Adjusted R-squared:  0.2329
## F-statistic: 15.88 on 1 and 48 DF,  p-value: 0.0002289
```

We see the following values:

- $\widehat{\beta}_1$ = -0.0430717. The estimated slope informs us the the predicted gift aid decreases by 0.0430717 thousands of dollars (or $43.07) per unit increase in family income.
- $\widehat{\beta}_0 = 24.319329$. For students with no family income, their predicted gift aid is $24 319.33. Note: from the scatterplot, we have an observation with 0 family income. We must be careful in not extrapolating when making predictions with our regression. We should only make predictions for family incomes between the minimum and maximum values of family incomes in our data.
- $s = 4.7825989$, is the estimate of the standard deviation of the error terms. This is reported as residual standard error in R. Squaring this gives the estimated variance.

- $F = 15.8772043$. This is the value of the ANOVA F statistic. The corresponding p-value is reported. Since this p-value is very small, we reject the null hypothesis. The data support the claim that there is a linear association between gift aid and family income.
- $R^2 = 0.2485582$. The coefficient of determination informs us that about 24.86% of the variation in gift aid can be explained by family income.

Extract values from R objects

We can actually extract these values that are being reported from `summary(result)`. To see what can be extracted from an R object, use the `names()` function:

```
##see what can be extracted from summary(result)
names(summary(result))
```

```
##  [1] "call"          "terms"         "residuals"     "coefficients"
##  [5] "aliased"       "sigma"         "df"            "r.squared"
##  [9] "adj.r.squared" "fstatistic"    "cov.unscaled"
```

To extract the estimated coefficients:

```
##extract coefficients
summary(result)$coefficients
```

```
##                  Estimate Std. Error   t value     Pr(>|t|)
## (Intercept)   24.31932901 1.29145027 18.831022 8.281020e-24
## family_income -0.04307165 0.01080947 -3.984621 2.288734e-04
```

Notice the information is presented in a table. To extract a specific value, we can specify the row and column indices:

```
##extract slope
summary(result)$coefficients[2,1]
```

```
## [1] -0.04307165
```

```
##extract intercept
summary(result)$coefficients[1,1]
```

```
## [1] 24.31933
```

On your own, extract the values of the residual standard error, the ANOVA F statistic, and R^2.

Prediction

A use of regression models is prediction. Suppose I want to predict the gift aid of a student with family income of 50 thousand dollars (assuming the unit is in thousands of dollars). We use the `predict()` function:

```
##create data point for prediction
newdata<-data.frame(family_income=50)
##predicted gift aid when x=50
predict(result,newdata)
```

```
##        1
## 22.16575
```

This student's predicted gift aid is \$22 165.75. Alternatively, you could have calculated this by plugging $x = 50$ into the estimated SLR equation:

```
summary(result)$coefficients[1,1] + summary(result)$coefficients[2,1]*50
```

```
## [1] 22.16575
```

ANOVA table

We use the `anova()` function to display the ANOVA table

```
anova.tab<-anova(result)
anova.tab
```

```
## Analysis of Variance Table
##
## Response: gift_aid
##               Df  Sum Sq Mean Sq F value    Pr(>F)
## family_income  1  363.16  363.16  15.877 0.0002289 ***
## Residuals     48 1097.92   22.87
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

The report F statistic is the same as the value reported earlier from `summary(result)`.

The first line of the output gives SS_R, the second line gives SS_{res}. The function doesn't provide SS_T, but we know that $SS_T = SS_R + SS_{res}$.

Again, to see what can be extracted from `anova.tab`:

```
names(anova.tab)
```

```
## [1] "Df"      "Sum Sq"  "Mean Sq" "F value" "Pr(>F)"
```

So SS_T can be easily calculated:

```
SST<-sum(anova.tab$"Sum Sq")
SST
```

```
## [1] 1461.079
```

The R^2 was reported to be 0.2485582. To verify using the ANOVA table:

```
anova.tab$"Sum Sq"[1]/SST
```

```
## [1] 0.2485582
```

Correlation

We use the `cor()` function to find the correlation between two quantitative variables:

```
##correlation
cor(Data$family_income,Data$gift_aid)
```

```
## [1] -0.4985561
```

The correlation is -0.4985561. We have a moderate, negative linear association between family income and gift aid.

Chapter 4

Inference with Simple Linear Regression (SLR)

4.1 Introduction

Oftentimes, the data we collect come from a random sample that is representative of the population of interest. A common example is an election poll before a presidential election. Random sampling allows the sample to be representative of the population. However, if we obtain another random sample, the characteristics of the new sample are unlikely to be exactly the same as the first sample. For example, the sample proportion who will vote for a certain party is unlikely to be the same for both random samples. What this tells us is that even with representative samples, sample proportions are unlikely to be equal to the population proportion, and sample proportions vary from sample to sample.

Dr. W. Edwards Deming's Red Bead experiment illustrates this concept. A video of this experiment can be found here.

In this video, the number of red beads, which represent bad products, varies each time the worker obtains a random sample of 50 beads. The fact that the number of red beads increases in his second sample does not indicate that he performed his task any worse, as this increase is due to the random variation associated with samples.

Note: Deming's Red Bead experiment was developed to illustrate concepts associated with management. He is best known for his work in developing the Japanese economy after World War II. You will be able to find many blogs/articles discussing the experiment on the World Wide Web. Although many of the articles discuss how this experiment applies in management, it can be used to illustrate concepts of variation.

The same idea extends to the slope and intercept of a regression line. The estimated slope and intercept will vary from sample to sample and are unlikely to be equal to the population slope and intercept. In inferential statistics, we use hypothesis tests and confidence intervals to aid us in accounting for this random variation. In this module, you will learn how to account for and quantify the random variation associated with the estimated regression model, and how to interpret the estimated regression model while accounting for random variation.

4.1.1 Review from previous module

The **simple linear regression model** is written as

$$y = \beta_0 + \beta_1 x + \epsilon. \tag{4.1}$$

We make some assumptions for the error term ϵ. They are:

1. The errors have mean 0.
2. The **errors have variance denoted by σ^2**. Notice this variance is constant.
3. The errors are independent.
4. The errors are normally distributed.

These assumptions allow us to derive the distributional properties associated with our least squares estimators $\hat{\beta}_0, \hat{\beta}_1$, which then enables us to compute reliable confidence intervals and perform hypothesis tests on our SLR reliably.

$\hat{\beta}_1, \hat{\beta}_0$ are the estimators for β_1, β_0 respectively. These estimators can be interpreted in the following manner:

- **$\hat{\beta}_1$ denotes the change in the predicted y when x increases by 1 unit. Alternatively, it denotes the change in y, on average, when x increases by 1 unit.**
- **$\hat{\beta}_0$ denotes the predicted y when $x = 0$. Alternatively, it denotes the average of y when $x = 0$.**

How do the values of these estimators vary from sample to sample?

4.2 Hypothesis Testing in SLR

4.2.1 Distribution of least squares estimators

Gauss Markov Theorem: Under assumptions for a regression model, the least squares estimators $\hat{\beta}_1$ and $\hat{\beta}_0$ are unbiased and have minimum variance among all unbiased linear estimators.

Thus, the least squares estimators have the following properties:

1. $\text{E}(\hat{\beta}_1) = \beta_1$, $\text{E}(\hat{\beta}_0) = \beta_0$

Note: An estimator is **unbiased** if its expected value is exactly equal to the parameter it is estimating.

2. The variance of $\hat{\beta}_1$ is

$$\text{Var}(\hat{\beta}_1) = \frac{\sigma^2}{\sum (x_i - \bar{x})^2} \tag{4.2}$$

3. The variance of $\hat{\beta}_0$ is

$$\text{Var}(\hat{\beta}_0) = \sigma^2 \left[\frac{1}{n} + \frac{\bar{x}^2}{\sum (x_i - \bar{x})^2} \right] \tag{4.3}$$

4. $\hat{\beta}_1$ and $\hat{\beta}_0$ both follow a normal distribution.

Note that in (4.2) and (4.3), we use $s^2 = MS_{res}$ to estimate σ^2 since σ^2 is a unknown value.

What these imply is that if we standardize $\hat{\beta}_1$ and $\hat{\beta}_0$, these standardized quantities will follow a t_{n-2} distribution, i.e.

$$\frac{\hat{\beta}_1 - \beta_1}{se(\hat{\beta}_1)} \sim t_{n-2} \tag{4.4}$$

and

$$\frac{\hat{\beta}_0 - \beta_0}{se(\hat{\beta}_0)} \sim t_{n-2}, \tag{4.5}$$

where

$$se(\hat{\beta}_1) = \sqrt{\frac{MS_{res}}{\sum (x_i - \bar{x})^2}} = \frac{s}{\sqrt{\sum (x_i - \bar{x})^2}} \tag{4.6}$$

and

$$se(\hat{\beta}_0) = \sqrt{MS_{res} \left[\frac{1}{n} + \frac{\bar{x}^2}{\sum (x_i - \bar{x})^2} \right]} = s\sqrt{\frac{1}{n} + \frac{\bar{x}^2}{\sum (x_i - \bar{x})^2}} \tag{4.7}$$

Note:

- $se(\hat{\beta}_1)$ is read as the **standard error of** $\hat{\beta}_1$. The standard error of any estimator is essentially the sample standard deviation of that estimator, and measures the spread of that estimator.
- A t_{n-2} distribution is read as a t **distribution with** $n-2$ **degrees of freedom**.

4.2.2 Testing regression coefficients

Hypothesis testing is used to investigate if a population parameter is **different from a specific value**. In the context of SLR, we usually want to test if β_1 is 0 or not. If $\beta_1 = 0$, there is no linear relationship between the variables.

The general steps in hypothesis testing are:

- Step 1: State the null and alternative hypotheses.
- Step 2: A test statistic is calculated using the sample, assuming the null is true. The value of the test statistic measures how the **sample deviates from the null**.
- Step 3: Make conclusion, using either critical values or p-values.

In the previous module, we introduced the ANOVA F test. In SLR, this tests if the slope of the SLR equation is 0 or not. It turns out that we can also perform a t test for the slope. In the t test for the slope, the null and alternative hypotheses are:

$$H_0 : \beta_1 = 0, H_a : \beta_1 \neq 0.$$

The test statistic is

$$t = \frac{\hat{\beta}_1 - \text{ value in null}}{se(\hat{\beta}_1)} \tag{4.8}$$

which is compared with a t_{n-2} distribution. Notice that (4.8) comes from (4.4).

Let us go back to our simulated example that we saw in the last module. We have data from 6000 UVa undergraduate students on the amount of time they spend studying in a week (in minutes), and how many courses they are taking in the semester (3 or 4 credit courses).

```r
##create dataframe
df<-data.frame(study,courses)

##fit regression
result<-lm(study~courses, data=df)
##look at regression coefficients
summary(result)$coefficients
```

```
##              Estimate Std. Error   t value      Pr(>|t|)
## (Intercept)  58.44829  1.9218752  30.41211 4.652442e-189
## courses     120.39310  0.4707614 255.74125  0.000000e+00
```

The t statistic for testing $H_0 : \beta_1 = 0, H_a : \beta_1 \neq 0$ is reported to be 255.7412482, which can be calculated using (4.8): $t = \frac{120.39310-0}{0.4707614}$. The reported p-value is virtually 0, so we reject the null hypothesis. The data support the claim that there is a linear association between study time and the number of courses taken.

4.3 Confidence Intervals for Regression Coefficients

Confidence intervals (CIs) are similar to hypothesis testing in the sense that they are also based on the distributional properties of an estimator. CIs may differ in their use in the following ways:

1. We are not assessing if the parameter is different from a specific value.
2. We are more interested in exploring a plausible **range of values for an unknown parameter**.

Because CIs and hypothesis tests are based on the distributional properties of an estimator, their conclusions will be consistent (as long as the significance level is the same).

Recall the general form for CIs:

$$\text{estimator} \pm (\text{multiplier} \times \text{s.e of estimator}). \tag{4.9}$$

We have the following components of a CI

- **estimator (or statistic)**: numerical quantity that describes a sample
- **multiplier**: determined by confidence level and relevant probability distribution
- **standard error of estimator**: measure of variance of estimator (basically the square root of the variance of estimator)

Following (4.9) and (4.4), the $100(1-\alpha)\%$ CI for β_1 is

$$\hat{\beta}_1 \pm t_{1-\alpha/2;n-2} se(\hat{\beta}_1) = \hat{\beta}_1 \pm t_{1-\alpha/2;n-2} \frac{s}{\sqrt{\sum(x_i - \bar{x})^2}}. \tag{4.10}$$

Going back to our study time example, the 95% CI for β_1 is (119.470237, 121.3159601).

```
##CI for coefficients
confint(result,level = 0.95)[2,]
```

```
##    2.5 %   97.5 %
## 119.4702 121.3160
```

An interpretation of this CI is that we have 95% confidence that the true slope β_1 lies between (119.470237, 121.3159601). In other words, for each additional course taken, the predicted study time increases between 119.470237 and 121.3159601 minutes.

4.3.1 Thought questions

- Is the conclusion from this 95% CI consistent with the hypothesis test for $H_0 : \beta_1 = 0$ in the previous section at 0.05 significance level?
- I have presented hypothesis tests and CIs for the slope, β_1.
  - How would you calculate the t statistic if you wanted to test $H_0 : \beta_0 = 0, H_0 : \beta_0 \neq 0$?
  - How would you calculate the 95% CI for the intercept β_0?

Generally, we are usually more interested in the slope than the intercept.

4.4 CI of the Mean Response

We have established that the least squares estimators $\hat{\beta}_1, \hat{\beta}_0$ have their associated variances. Since the estimated SLR equation is

$$\hat{y} = \hat{\beta}_0 + \hat{\beta}_1 x, \tag{4.11}$$

it stands to reason that $\hat{y}$ has an associated variance as well, since it is a function of $\hat{\beta}_1, \hat{\beta}_0$.

There are two interpretations of $\hat{y}$:

1. it **estimates the mean of y when** $x = x_0$;
2. it **predicts the value of y for a new observation when** $x = x_0$.

Note: x_0 denotes a specific numerical value for the predictor variable.

Depending on which interpretation we want, there are two different intervals based on $\hat{y}$. The first interpretation is associated with the **confidence interval for the mean response, $\hat{\mu}_{y|x_0}$, given the predictor**. This is used when we are interested in the average value of the response variable, when the predictor is equal to a specific value. This CI is

$$\hat{\mu}_{y|x_0} \pm t_{1-\alpha/2,n-2} s \sqrt{\frac{1}{n} + \frac{(x_0 - \bar{x})^2}{\sum(x_i - \bar{x})^2}}. \tag{4.12}$$

Going back to our study time example, suppose we want the average study time for students who take 5 courses, the 95% CI is

```r
##CI for mean y when x=5
newdata<-data.frame(courses=5)
predict(result, newdata, level=0.95, interval="confidence")
```

```
##        fit      lwr      upr
## 1 660.4138 659.2224 661.6052
```

We have 95% confidence that the average study time for students who take 5 courses is between 659.2223688 and 661.605187 minutes.

4.5 PI of a New Response

Previously, we found a CI for the mean of y given a specific value of x, (4.12). This CI gives us an idea about the location of the regression line at a specific of x.

Instead, we may have interest in finding an interval for a new value of $\hat{y}_0$, when we have a new observation $x = x_0$. This is called a **prediction interval (PI) for a future observation y_0 when the predictor is a specific value**. This interval follows from the second interpretation of $\hat{y}$.

The PI for $\hat{y}_0$ takes into account:

1. Variation in location for the distribution of y (i.e. where is the center of the distribution of y?).
2. Variation **within the probability distribution of** y.

By comparison, the confidence interval for the mean response (4.12) only takes into account the first element. The PI is

$$\hat{y}_0 \pm t_{1-\alpha/2,n-2} s \sqrt{1 + \frac{1}{n} + \frac{(x_0 - \bar{x})^2}{\sum(x_i - \bar{x})^2}}. \tag{4.13}$$

Going back to our study time example, suppose we have a newly enrolled student who wishes to take 5 courses, and the student wants to predict his study time

```
##PI for y when x=5
predict(result, newdata, level=0.95, interval="prediction")
```

```
##        fit      lwr      upr
## 1 660.4138 602.0347 718.7928
```

We have 95% confidence that the study time for this student is between 602.0347305 and 718.7928253 minutes.

4.5.1 Thought questions

- In the following two scenarios, decide if we are more interested in the CI for the mean response given the predictor (4.12), or the PI for a future response given the predictor (4.13).
  - We wish to estimate the waiting time, on average, of DMV customers if there are 10 people in line at the DMV.
  - I enter the DMV and notice 10 people in line. I want to estimate my waiting time.

- Look at the standard errors associated with the intervals given in (4.12) and (4.13). How are they related to each other?

4.6 Supplemental Notes on Statistical Inference

4.6.1 Hypothesis statements

Let's consider a t test for the regression parameter, β_1. Depending on context, the following could be null and alternative hypotheses

- $H_0 : \beta_1 = 0, H_a : \beta_1 \neq 0.$
- $H_0 : \beta_1 = 0, H_a : \beta_1 > 0.$
- $H_0 : \beta_1 = 0, H_a : \beta_1 < 0.$

The null hypothesis should be stated as a statement of **equality**. This concept holds true for hypothesis tests in general. Some other books / resources might state them as

- $H_0 : \beta_1 = 0, H_a : \beta_1 \neq 0.$
- $H_0 : \beta_1 \leq 0, H_a : \beta_1 > 0.$
- $H_0 : \beta_1 \geq 0, H_a : \beta_1 < 0.$

I prefer using the equality statement for the null hypothesis for the following reasons (theoretical, pedagogical, practical):

1. The null hypothesis being an equality aligns with the definition of the p-value.

- The p-value is the probability of observing our sample estimate (or a value more extreme), if the null hypothesis is true (i.e. β_1 is truly 0). This is what we are assuming in the calculation for the test statistic.

2. People tend to get confused between the null and alternative hypotheses if both involve inequalities (the alternative is the hypothesis you are trying to support).
3. Conclusions are made in terms of supporting (or not supporting) the alternative hypothesis.

4.6.2 Sample size and statistical inference

Generally speaking, there is a relationship between sample size and statistical inference (assuming other characteristics remain the same and our sample was randomly obtained or representative of the population of interest):

- Larger sample sizes (typically) lead to narrower confidence intervals (more precise intervals).
- Sample estimates based on larger samples are more likely to be closer to the true parameters.

- Larger sample (typically) lead to more evidence against the null hypothesis.
  - This means a larger sample size leads to a more powerful test. The power of a test is the probability a hypothesis test is able to correctly reject the null hypothesis.

4.6.2.1 Small sample sizes

Small sample sizes tend to result in:

- Confidence intervals that are wide.
- Sample estimates that are more likely to be further away from the true parameters.
- Hypothesis tests that are more likely to incorrectly fail to reject the null hypothesis when the alternative hypothesis is true.

While larger sample sizes have their advantages, there are also some disadvantages with sample sizes that are extremely large.

4.6.2.2 Large sample sizes

A "statistically significant" result does not necessarily mean that the result has practical consequences. Suppose a 95% confidence interval for β_1 is $(0.001, 0.002)$. The interval excludes 0, so it is "statistically significantly" different from 0 (because it is!), but does this result have practical consequences? A narrow CI that barely excludes the null value can happen when we have a large sample size.

If one was to conduct the corresponding hypothesis test, we would reject the null hypothesis that $\beta_1 = 0$. With large sample sizes, hypothesis tests are sensitive to small departures from the null hypothesis.

In such instances, it may be worth considering hypothesis tests involving a different value in the null hypothesis, one that makes sense for your question. For example, a practically significant slope may need to be greater than a specific numerical value for a certain context.

- Statistical inference to assess statistical significance.
- Subject area knowledge to assess practical significance.

4.6.2.3 Questions

Are the following results statistically significant? If so, are the results also practically significant? Assume a two-sided test with a null value of 0 (These are made up examples):

1. In assessing if studying more is associated with better test scores, a SLR is carried out with test scores (out of 100 points) against study time (in hours). The 95% confidence interval for the slope β_1 is (5.632, 7.829).

2. A SLR is carried out to explore the linear relationship between number of years in school with income (in thousands of dollars). The 95% confidence interval for the slope β_1 is (0.051, 0.243).

4.6.3 Cautions using SLR and Correlation

Simple linear regression and correlation are meant for assessing **linear** relationships. If the relationship is not linear, we will need to transform the variable(s) (so the transformed variables have a linear relationship. Will explore this in Module 5).

- Always verify via a scatterplot that the relationship is at least approximately linear.
- A high correlation or a significant estimated slope by themselves do not prove that we have a strong linear relationship between the variables. Conversely, a correlation close to 0 or an insignificant estimated slope is also not proof that we do not have a relationship between the variables.

4.6.3.1 Outliers

SLR and correlation are sensitive to outliers / influential observations. Generally speaking, these are data that are "far away" or very different from the rest of the observations. These data points can be visually inspected from a scatterplot. Some potential considerations when dealing with such data points:

- Investigate these observations. There is usually something that is making them "stand out" from the rest of the data.
- Data entry errors that can be corrected. Be sure to mention in the report.
- Revisit how the data were sampled. Perhaps the data point is is not part of the population of interest. If so, data point can be removed (this is legitimate), but be sure to mention in the report.

With regards to regression analysis:

- Exclusion of data points must be clearly documented.
- Fit the regression with and without the data points in question, and see how similar or different the conclusions become.
- If the data points have large value(s) on the predictor and/or response, a log transformation on the variable can pull in the large values.
- Consider subsetting your data and create separate models for each subset; or focus on a subset and make it clear your analysis is for a subset.
- Knowing your data and context can help a lot in these decisions.

4.6.3.2 Association and causation

Two correlated variables do not mean that one variable causes the other variable to change. For example, consider a plot of ice cream consumption and deaths by drowning during various months. There may be some positive correlation, and clearly, eating more ice cream does not cause more drownings. The correlation can be explained by a third (lurking) variable: the weather.

A **lurking variable** is a variable that has an impact on the relationship between the variables being studied, but is itself not studied.

A carefully designed **randomized experiment** can control for lurking variables, and causal relationships can be established. Typically, such experiments include:

- A control group and a treatment group.
- Random assignment of large number of observations into the treatment and control groups. Due to the random assignment, the general characteristics of of subjects in each group are similar.

Lurking variables are always an issue with **observational studies**. Researchers in observational studies do not intervene with the observations and simply observe the data that the observations generate. Causal relationships are much more difficult to establish with observational studies.

4.6.3.3 Questions

1. Consider the `palmerpenguins` dataset that we have been working on. The data contain size measurements for three different species of penguins on three islands in the Palmer Archipelago, Antarctica over three years. Is this an observational study or randomized experiment?

2. A fertilizer company wishes to evaluate how effective a new fertilizer is in terms of improving the yield of crops. A large field is divided into many smaller plots, and each smaller plot is randomly assigned to receive either the new fertilizer or the standard fertilizer. Is this an observational study or randomized experiment?

3. A professor wishes to evaluate the effectiveness of various teaching methods (traditional vs flipped classroom). The professor uses the traditional approach for a section that meets on Mondays, Wednesdays, and Fridays from 9 to 10am and uses the flipped classroom approach for a section that meets on Mondays, Wednesdays, and Fridays from 2 to 3pm. Students were free to choose whichever section that wanted to register for, with no knowledge of the teaching method being used. What are some potential lurking variables in this study?

4.7 R Tutorial

For this tutorial, we will continue to work with the dataset `elmhurst` from the `openintro` package in R.

```r
library(tidyverse)
library(openintro)
Data<-openintro::elmhurst
```

The key pieces of information are:

- A random sample of 50 students (all freshman from the 2011 class at Elmhurst College).
- Family income of the student (units are missing).
- Gift aid, in $1000s.

We want to explore how family income may be related to gift aid, in a simple linear regression framework.

Hypothesis test for β_1 (and β_0)

Applying the `summary()` function to `lm()` gives the results of hypothesis tests for β_1 and β_0:

```
##Fit a regression model
result<-lm(gift_aid~family_income, data=Data)

##look at t stats and F stat
summary(result)
```

```
##
## Call:
## lm(formula = gift_aid ~ family_income, data = Data)
##
## Residuals:
##      Min       1Q   Median       3Q      Max
## -10.1128  -3.6234  -0.2161   3.1587  11.5707
##
## Coefficients:
##               Estimate Std. Error t value Pr(>|t|)
## (Intercept)   24.31933    1.29145  18.831  < 2e-16 ***
## family_income -0.04307    0.01081  -3.985 0.000229 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 4.783 on 48 degrees of freedom
## Multiple R-squared:  0.2486, Adjusted R-squared:  0.2329
## F-statistic: 15.88 on 1 and 48 DF,  p-value: 0.0002289
```

Under coefficients, we can see the results of the hypothesis tests for β_1 and β_0. Specifically, for β_1:

- $\hat{\beta}_1 =$ -0.0430717
- $se(\hat{\beta}_1) =$ 0.0108095
- the test statistic is $t =$ -3.984621
- the corresponding p-value is 2.2887345×10^{-4}

You can work out the p-value using R (slight difference due to rounding):

```
##pvalue
2*pt(-abs(-3.985), df = 50-2)
```

```
## [1] 0.0002285996
```

Or find the critical value using R:

```
##critical value
qt(1-0.05/2, df = 50-2)
```

```
## [1] 2.010635
```

Either way, we end up rejecting the null hypothesis. The data support the claim that there is a linear association between gift aid and family income.

Note:

- the t tests for regression coefficients are based on $H_0 : \beta_j = 0, H_a : \beta_j \neq 0$. The reported p-value is based on this set of null and alternative hypotheses. If your null and alternative hypotheses are different, you will need to compute your own test statistic and p-value.
- For SLR, the two-sided t test for β_1 gives the exact same result as the ANOVA F test. Notice the p-values are the same. The F statistic of 15.88 is the squared of the t statistic, $(-3.985)^2$.

Confidence interval for β_1 (and β_0)

To find the 95% confidence intervals for the coefficients, we use the `confint()` function:

```
##to produce 95% CIs for all regression coefficients
confint(result,level = 0.95)
```

```
##                     2.5 %      97.5 %
## (Intercept)   21.72269421 26.91596380
## family_income -0.06480555 -0.02133775
```

The 95% CI for β_1 is (-0.0648056, -0.0213378). We have 95% confidence that for each additional thousand dollars in family income, the predicted gift aid decreases between $21.3378 and $64.8056.

Confidence interval for mean response for given x

Suppose we want a confidence interval for the average gift aid for Elmhurst College students with family income of 80 thousand dollars. We can use the `predict()` function:

```
##to produce 95% CI for the mean response when x=80,
newdata<-data.frame(family_income=80)
predict(result,newdata,level=0.95, interval="confidence")
```

```
##       fit      lwr      upr
## 1 20.8736 19.43366 22.31353
```

The 95% CI for the mean gift aid for students with family income of 80 thousand dollars is (19.4336609, 22.3135327). We have 95% confidence the mean gift aid for students with family income of 80 thousand dollars is between $19 433.66 and $22 313.53.

Prediction interval for a response for a given x

For a prediction interval for the gift aid of an Elmhurst College student with family income of 80 thousand dollars:

```
##and the 95% PI for the response of an observation when x=80
predict(result,newdata,level=0.95, interval="prediction")
```

```
##       fit      lwr      upr
## 1 20.8736 11.15032 30.59687
```

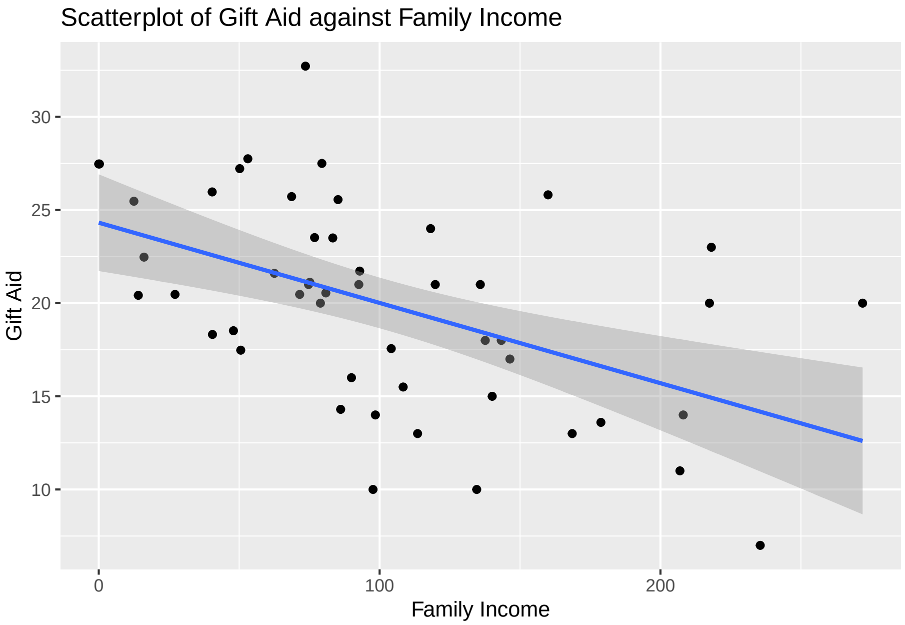
We have 95% confidence that for an Elmhurst College student with family income of 80, this student's gift aid is between $11 150.32 and $30 596.87.

Visualization of CI for mean response given x and PI of response given x

When using the `ggplot()` function to create a scatterplot, we can overlay the SLR equation by adding a layer via `geom_smooth(method = lm)`. By default, the CI for the mean response for each value of the predictor gets overlaid as well. In the previous tutorial, we removed this by adding `se=FALSE` inside `geom_smooth()`:

```
##regular scatterplot
##with regression line overlaid, and bounds of CI for mean y
ggplot2::ggplot(Data, aes(x=family_income, y=gift_aid))+
  geom_point() +
  geom_smooth(method=lm)+
  labs(x="Family Income",
       y="Gift Aid",
       title="Scatterplot of Gift Aid against Family Income")
```

```
## `geom_smooth()` using formula = 'y ~ x'
```



Overlaying prediction intervals require a bit more work. We need to compute the lower and upper bounds of the PI for each value of the predictor:

```
##find PIs for each observation
preds <- predict(result, interval="prediction")
```

```
## Warning in predict.lm(result, interval = "prediction"): predictions on current data refer to _
```

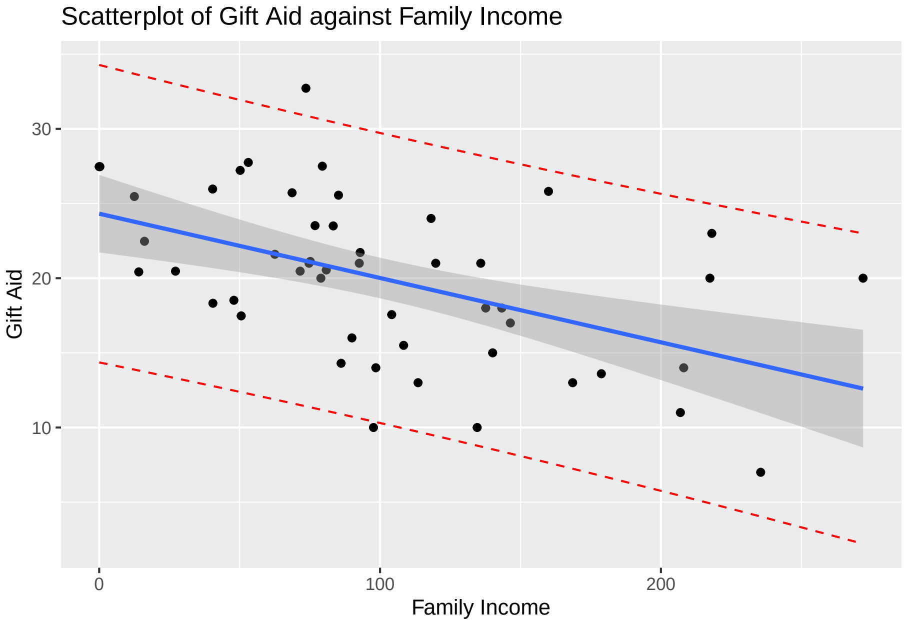
Previously, when we used the `predict()` function, we provided the numerical value of x to make a prediction on. If this is not supplied, the function will use all the current values of x to make predictions, and will actually print out a warning message. For our purpose, this is not an issue since this is exactly what we want.

We then add `preds` to the data frame in order to overlay the lower and upper bounds on the scatterplot, by adding extra layers via `geom_line()` in the `ggplot()` function:

```
##add preds to data frame
Data<-data.frame(Data,preds)

##overlay PIs via geom_line()
ggplot2::ggplot(Data, aes(x=family_income, y=gift_aid))+
  geom_point() +
  geom_line(aes(y=lwr), color = "red", linetype = "dashed")+
  geom_line(aes(y=upr), color = "red", linetype = "dashed")+
  geom_smooth(method=lm)+
  labs(x="Family Income",
       y="Gift Aid",
       title="Scatterplot of Gift Aid against Family Income")
```

```
## `geom_smooth()` using formula = 'y ~ x'
```



As mentioned in the notes, the CI captures the location of the regression line, whereas the PI captures the data points.

Chapter 5

Model Diagnostics and Remedial Measures in SLR

5.1 Introduction

The regression model is based on a number of assumptions. Those assumptions are made so that we can apply commonly used probability distributions to we quantify the variability associated with our estimated regression model. This means that if the assumptions are not met for our regression model, then how we quantify the variability associated with our model is no longer reliable. All our analysis with statistical inference becomes questionable.

In this module, you will learn how to assess whether the regression assumptions are met. We will explore ways in which we can transform our variables after diagnosing which assumptions are not met so that we can still proceed to build our regression model.

5.2 Assumptions in Linear Regression

In module 3, we stated the SLR model as

$$y = \beta_0 + \beta_1 x + \epsilon. \tag{5.1}$$

where $f(x) = \beta_0 + \beta_1 x$. We need to make some assumptions for the error term ϵ. Mathematically, the assumptions are expressed as

$$\epsilon_1, \dots, \epsilon_n \ i.i.d. \sim N(0, \sigma^2) \tag{5.2}$$

Breaking down (5.2) the assumptions can be expressed as the following:

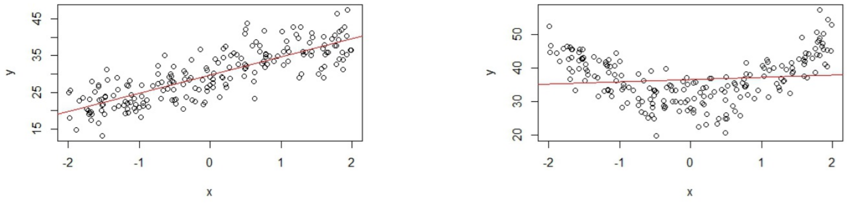
1. The errors have **mean 0**.
2. The errors have **constant variance denoted by** σ^2.
3. The errors are **independent**.
4. The errors are **normally distributed**.

Let's dig a little deeper into the meaning and implications of these 4 assumptions.

5.2.1 Assumption 1: Errors have mean 0.

For each value of the predictor, the errors have **mean 0**. A by-product of this statement is that the relationship between y and x, as expressed via $y \approx f(x)$, is correct. So, if $f(x) = \beta_0 + \beta_1 x$, then the relationship is approximately linear.

The plots in Figure 5.1 are based on simulated data. The scatterplot shown in Figure 5.1(a) is an example of when this assumption is met. As we move from left to right on the plot, the data points are generally evenly scattered on both sides of the regression line that is overlaid.



(a) Plot with Linear Relationship

(b) Plot with Non Linear Relationship

Figure 5.1: Assumption 1

The scatterplot shown in Figure 5.1(b) is an example of when this assumption is **not** met. As we move from left to right on the plot in Figure 5.1(b), the data points are generally not evenly scattered on both sides of the regression line that is overlaid.

- When $-2 \leq x \leq -1.2$, the data points are generally above the regression line;
- then when $-1.2 < x < 1$, the data points are generally below the regression line;
- and then when $x \geq 1$, the data points are generally above the regression line.

Please see the associated video for more explanation on how to use Figure 5.1 to assess assumption 1.

5.2.1.1 Consequences of violating this assumption

Predictions will be biased. This means that predicted values will systematically over- or under- estimate the true values of the response variable. Of the 4 assumptions listed, this is **most crucial assumption**.

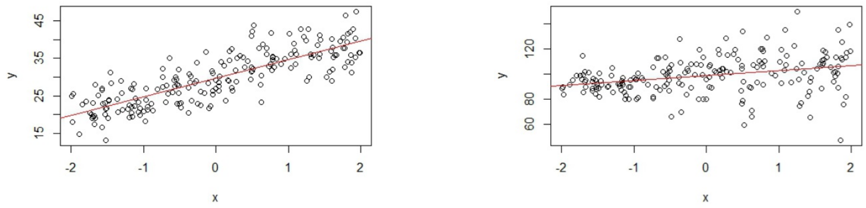
Using Figure 5.1(b) as an example, this implies that

- when $-2 \leq x \leq -1.2$, the regression line will systematically under-predict the response variable;
- then when $-1.2 < x < 1$, the regression line will systematically over-predict the response variable;
- and then when $x \geq 1$, the regression line will systematically under-predict the response variable.

5.2.2 Assumption 2: Errors have constant variance

For each value of the predictor, the error terms have **constant variance**, denoted by σ^2. This implies that when looking at a scatterplot, the vertical variation of data points around the regression equation has the same magnitude everywhere.

The plots in Figure 5.2 are based on simulated data. The scatterplot shown in Figure 5.2(a) is an example of when this assumption is met (this figure is actually the same as Figure 5.1(a), so the data that produced these plots satisfy both assumptions). As we move from left to right on the plot, the vertical variation of the data points about the regression line is approximately constant.



(a) Plot with Constant Variance

(b) Plot with Increasing Variance

Figure 5.2: Assumption 2

The scatterplot shown in Figure 5.2(b) is an example of when this assumption is **not** met. As we move from left to right on the plot in Figure 5.2(b), the vertical variation of the data points about the regression line becomes larger as the value of the response variable gets larger, so the variance is not constant.

Please see the associated video for more explanation on how to use Figure 5.2 to assess assumption 2.

5.2.2.1 Consequences of violating this assumption

Statistical inference will no longer be reliable. This means that the results from any hypothesis test, confidence interval, or prediction interval are no longer reliable.

Interestingly, for the scatterplot in Figure 5.2(b), we can say that assumption 1 is met, since the the data points are generally evenly scattered on both sides of the regression line. Predictions will still be unbiased; the predicted response, $\hat{y}$, do not systematically over- or under-predict the response variable. So if our goal is to assess if the relationship is approximately linear, this scatterplot is fine. We do lose the utility from hypothesis tests, CIs, and PIs.

5.2.3 Assumption 3: Errors are independent

A by-product of this assumption is that the values of the response variable, y_i, are independent from each other. Any y_i does not depend on other values of the response variable.

5.2.3.1 Consequences of violating this assumption

Statistical inference will no longer be reliable. This means that the results from any hypothesis test, confidence interval, or prediction interval are no longer reliable.

5.2.4 Assumption 4: Errors are normally distributed

If we were to create a density plot of the errors, the errors should follow a normal distribution.

5.2.4.1 Consequences of violating this assumption

The regression model is fairly robust to the assumption that the errors are normally distributed. In other words, violation of this particular assumption is not very consequential. **Of the 4 assumptions, this is the least crucial to satisfy.**

5.3 Assessing Regression Assumptions

There are a few visualizations that help in detecting violations of the regression assumptions. These visualizations are:

- Scatterplot of y against x (assumptions 1 and 2).
- Residual plot (assumptions 1 and 2).
- Autocorrelation function (ACF) plot of residuals (assumption 3).
- Normal probability plot of residuals (often called QQ plot) (assumption 4).

5.3.1 Scatterplot

We can examine the scatterplot of y against x to check for assumptions 1 and 2. We want to see the following in the scatterplot:

- **No nonlinear pattern** (assumption 1).
- Data points **evenly scattered** (for each value on the x-axis) around fitted line (assumption 1).
- Vertical variation of data points constant (assumption 2).

We have used Figure 5.2(a) as an example of a scatterplot that meets these assumptions. Let us take a look at another example that we have worked with. This scatterplot is from the `elmhurst` dataset from the `openintro` package that we have been seeing in tutorials. We are regressing the amount of gift aid a student receives based on the student's family income. The corresponding scatterplot is shown in in Figure 5.3.

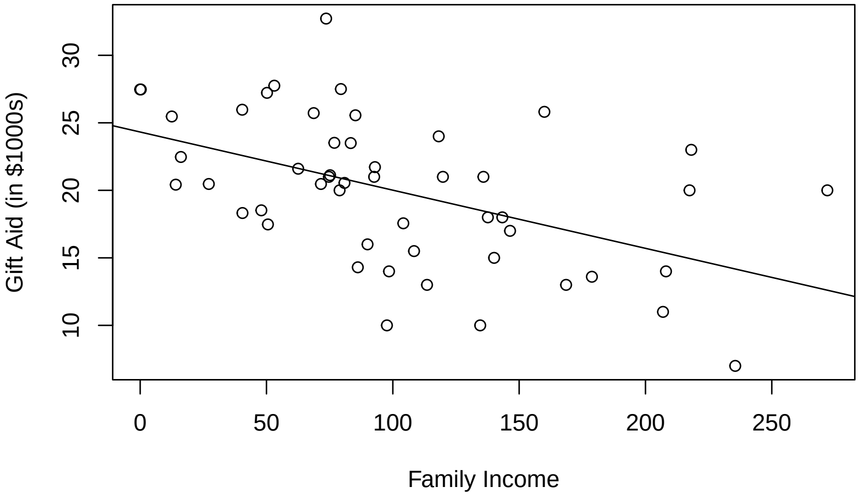


Figure 5.3: Scatterplot of Gift Aid Against Family Income

In Figure 5.3, we see that the data points are evenly scattered around the fitted line. We also see the vertical variation of the data points is fairly constant. So assumptions that the errors have 0 mean and constant variance appear to be met.

5.3.1.1 Practice question

The data are about the prices of used cars. We are regressing the sale price of the car against the age of the car. The corresponding scatterplot is shown in Figure 5.4. Based on Figure 5.4, which of assumptions 1 or 2 (or both, or neither), is met? We will go over this in the tutorial.

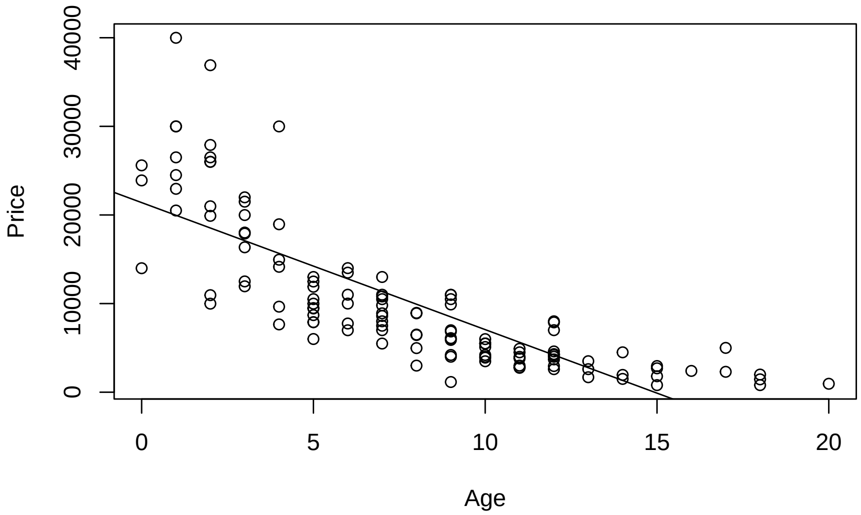


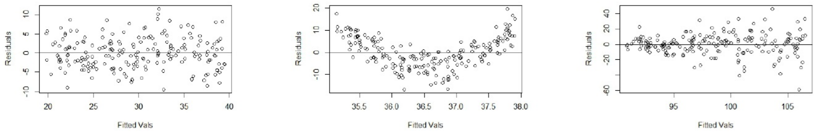
Figure 5.4: Scatterplot of Sale Price Against Age

5.3.2 Residual plot

While using the scatterplot is an intuitive way of assessing regression assumptions, it has a limitation. It cannot be used if we have multiple predictors in our regression, which we will encounter (and happens more often than just having one predictor). Another visualization that we can use to assess assumptions 1 and 2 is a **residual plot**. This is a scatterplot of residuals, e, against fitted values, $\hat{y}$. We want to observe the following in a residual plot.

- Residuals should be **evenly scattered** across the horizontal axis (assumption 1).
- The residuals should have **similar vertical variation** across the plot (assumption 2).
- Some writers combine these two points into the following statement: the residuals should fall in a **horizontal band around 0** with no apparent pattern (assumption 1, 2).

The residual plots in Figure 5.5 are based on simulated data from Figures 5.1(a), 5.1(b), and 5.2(b).



(a) Assumptions 1 & 2 Met (b) Assumption 1 Not Met (c) Assumption 2 Not Met

Figure 5.5: Residual Plots from Fig 1(a), 1(b), 2(b) Respectively

We make the following observations:

- From Figure 5.5(a), we see that the residuals are evenly scattered across the horizontal axis, and their vertical variation is fairly constant across the plot. So both assumptions are met.
- From Figure 5.5(b), we see that the residuals are **not** evenly scattered across the horizontal axis, although their vertical variation is fairly constant across the plot. So only assumption 1 is not met.
- From Figure 5.5(c), we see that the residuals are evenly scattered across the horizontal axis, but their vertical variation is **not constant** across the plot. In fact, the vertical variation is increasing as we move from left to right. So only assumption 2 is not met.

If you compare the conclusions from the residuals plots and scatterplots, they are the same. In SLR, the takeaways should be consistent.

Please see the associated video for more explanation on how to use Figure 5.5 to assess assumptions 1 and 2.

5.3.2.1 Practice questions

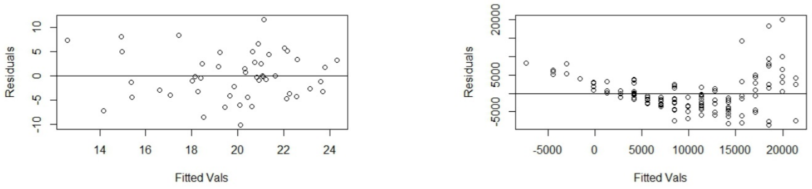
1. The residual plot in Figure 5.6(a) comes from regressing gift aid against family income for the `elmhurst` dataset. Based on this residual plot, which assumptions are met?

2. The residual plot in Figure 5.6(b) comes from regressing price of cars against age for the used cars dataset. Based on this residual plot, which assumptions are met?

Please see the associated video as I go over these practice questions.

5.3.3 ACF plot

Assumption 3 states that the errors are **independent**. This assumption implies that the values of the response variable are independent from each other. This assumption is typically assessed via knowing the nature of the data.



(a) Residual Plot Regressing Gift Aid Against Family Income

(b) Residual Plot Regressing Price Against Age for Used Cars

Figure 5.6: Residual Plots for Practice Questions

- If the observations were obtained from a random sample, it is likely that the observations will be independent from each other. This is the very nature of a random sample and why random samples are preferred over convenience samples.
- If the data has some inherent sequence, it is likely the observations will not be independent, and are dependent. For example, if I record the value of a stock at the end of each day, the value at day 2 is likely to be related to its value at day 1. So the values of stock prices at the end of each day are not independent.

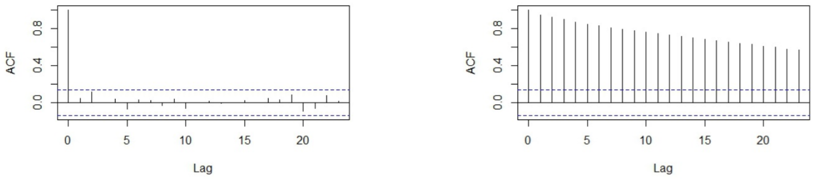
An autocorrelation function (ACF) plot of the residuals may be a used to help assess if the assumption that the errors are independent is met. However, the plot is not a substitute for using your understanding about the nature of the data and should only be used as a confirmation.

The ACF plot measures the correlation between a vector of observations and the lagged versions of the observations. If the observations are uncorrelated, the correlations between the vector of observations and lagged versions of these observations are theoretically 0. We may create an ACF plot for the residuals from our regression.

The ACF plot in Figure 5.7(a) and is based on simulated data that were independently generated.

A few notes about the ACF plot:

- The ACF at lag 0 is always 1. The correlation of any vector with itself is always 1.
- The dashed horizontal lines represent critical values. An ACF at any lag beyond the critical value indicates an ACF that is significant. We have evidence of correlation (and hence dependence) in our residuals.
- If the observed values for the response variable are independent, then we would expect the ACFs at lags greater than 0 to be insignificant. Do note that because we are conducting multiple hypothesis tests, do not be too



(a) ACF of Residuals that are Uncorrelated

(b) ACF of Residuals when Response was Ordered

Figure 5.7: Assumption 3

alarmed if the ACFs are slightly beyond the critical values at an isolated lag or 2.

Based on Figure 5.7(a), we see that the ACFs at all lags greater than 0 are insignificant. We do not have evidence the residuals are correlated with each other, so we do not have evidence that assumption 3 is not met.

Sometimes, the dataframe can be sorted in some manner (e.g. increasing order for response variable), and if so, we would actually expect to see significant correlations in the ACF plot. The ACF plot in Figure 5.7(b) is such an example. The residuals are from the same simulated dataset, only with the data sorted by the response variable. If we had just looked at the ACF plot in Figure 5.7(b) without understanding the data were simulated independently and then sorted, we would have erroneously concluded that the residuals are not independent and the regression assumption is not met.

5.3.4 QQ plot

A normal probability plot (also called a QQ plot) is used to assess if the distribution of a variable is normal. It typically plots the residuals against their theoretical residual if they followed a normal distribution. A QQ line is typically overlaid. If the plots fall closely to the QQ line, we have evidence that the observations follow a normal distribution. Figure 5.8 shows a QQ plot that comes from a normally distributed variable.

5.3.5 Remedial measures

We now know how to assess if specific regression assumptions are not met. The remedial measures involve transforming either the predictor variable and / or the response variable. These transformations are chosen to handle violations to assumptions 1 and / or 2 respectively. The general strategy on selecting which variable to transform:

- Transforming the response variable, y, affects both assumptions 1 and 2.

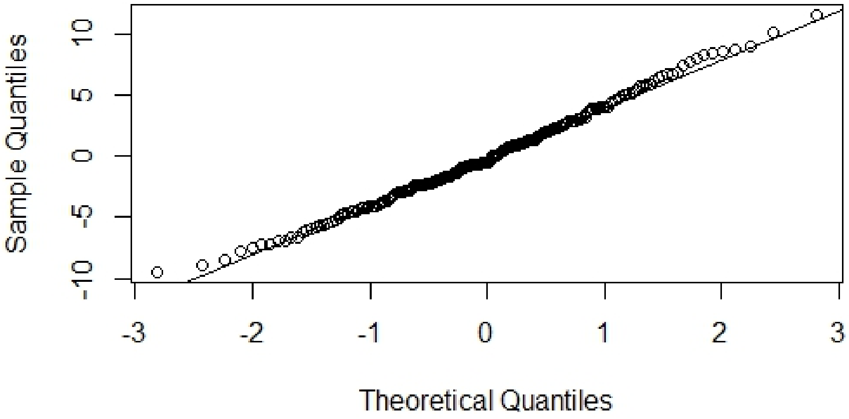


Figure 5.8: QQ Plot

  - Visually, we can think of transforming y in terms of stretching or squeezing the scatterplot of y against x vertically. Thus, transforming y affects the shape of the relationship and the vertical spread of the data points.
  - However, the **choice on how we transform y is based on handling assumption 2.**
- Transforming the predictor variable, x affects assumption 1 and does not theoretically affect assumption 2.
  - Visually, we can think of transforming x in terms of stretching or squeezing the scatterplot of y against x horizontally. Thus, transforming x affects the shape of the relationship but not the vertical spread of the data points.
  - Therefore, **transforming x is based on handling assumption 1.**
- If assumption 2 is not met, we transform y to stabilize the variance and make it constant.
- If assumption 1 is not met, we transform x to find the appropriate shape to relate the variables.
- If both assumptions are not met, we transform y first to stabilize the variance. Once assumption 2 is solved, check if assumption 1 is not met. If not met, transform x.

Assumption 1 deals with whether the way we have expressed how y and x are related, through $f(x)$, is appropriate. Assumption 2 deals with the vertical variation of the data points in the scatterplot.

5.4 Remedial Measures: Variance Stabilizing Transformations

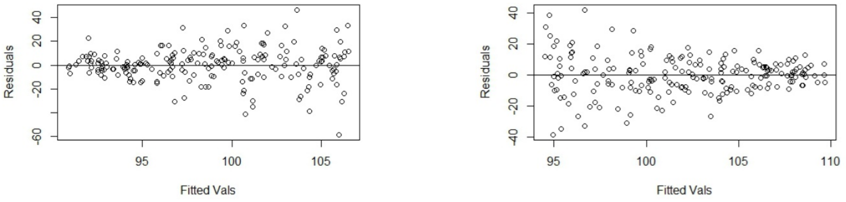
We transform the response variable to stabilize the variance (assumption 2). There are a couple of ways to decide the appropriate transformation:

1. Pattern seen in residual plot can guide choice in how to transform the response variable.
2. Box-Cox plot.

5.4.1 Use Pattern in Residual Plot

We can stabilize the variance of the errors based on the residual plot, if we see either of the following scenarios:

- vertical variation of residuals **increasing** as fitted response increases, or as we move from left to right, as in Figure 5.9(a), or
- vertical variation of residuals **decreasing** as fitted response increases, or as we move from left to right, as in Figure 5.9(b).



(a) Variance Increasing with Fitted Y

(b) Variance Decreasing with Fitted Y

Figure 5.9: Non Constant Variance in Residual Plot

Note that increasing variance as fitted response increases is much more common with real data. Generally, larger values of a variable are associated with larger spread.

We transform y using $y^* = y^\lambda$, with λ chosen based on whether the variance of the residuals is increasing or decreasing with fitted response:

- For Figure 5.9(a), choose $\lambda < 1$.
  - If $\lambda = 0$, it means we use a logarithmic transformation with base e, i.e. $y^* = \log(y)$.
  - Note that a logarithm with no base means a natural log, or ln.
- For Figure 5.9(b), choose $\lambda > 1$.

So based on the residual plot, we have a range of values for λ.

5.4.2 Box-Cox Plot

We can use a Box-Cox plot to help us narrow the range of λ to use. It is a plot of the log-likelihood function against λ, and we choose λ that maximizes this log-likelihood function. For example, Figure 5.10 shows the Box Cox plot generated for the regression associated with the residual plot in Figure 5.9(b).

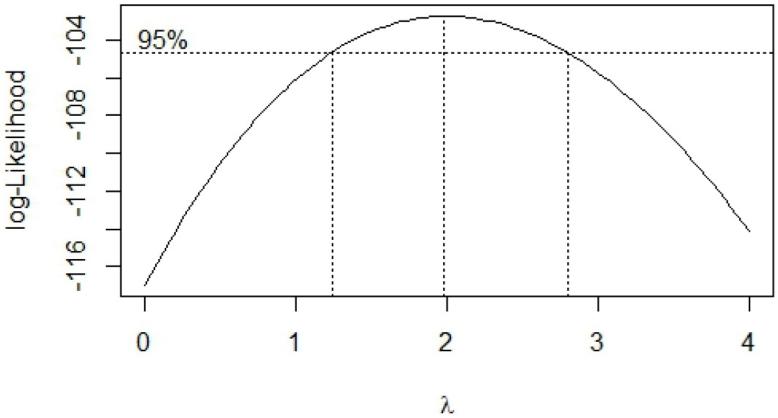


Figure 5.10: Box-Cox Plot based on Figure 9(b)

Notice an approximate 95% CI is provided for λ. A few comments on how to use the Box-Cox plot:

- Three vertical dashed lines are displayed: the middle line corresponds to the optimal value of λ; the other two lines are the lower and upper bounds of a 95% CI for λ.
- We choose λ within the CI (or even close to it) that is easy to understand. We do not have to choose the optimal value, especially if its value is difficult to interpret. In this example, I will choose $\lambda = 2$, so a square transformation for y. Transform response with $y^* = y^2$. Regress y^* against x.
- If 1 lies in the CI, **no transformation** on y may be needed.
- If a transformation is needed, a **log transformation** is preferred, since we can still interpret the estimated coefficients. It is difficult to interpret with any other type of transformation.
- View the Box-Cox procedure as a guide for selecting a transformation, rather than being definitive.
- Need to recheck the residuals after every transformation to assess if the transformation worked.

5.4.3 Interpretation with Log Transformed Response

A log transformation on the response is preferred over any other transformation, as we can still interpret regression coefficients. A couple of ways to interpret the estimated slope $\hat{\beta}_1$:

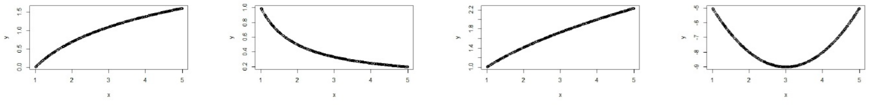
- The predicted response variable is **multiplied by a factor** of $\exp(\hat{\beta}_1)$ for a one-unit increase in the predictor.
- We can also subtract 1 from $\exp(\hat{\beta}_1)$ to express the change as a percentage.
  - If $\hat{\beta}_1$ is positive, we have a percent **increase**. The predicted response variable increases by $(\exp(\hat{\beta}_1) - 1) \times 100$ percent for a one-unit increase in the predictor.
  - If $\hat{\beta}_1$ is negative, we have a percent **decrease**. The predicted response variable decreases by $(1 - \exp(\hat{\beta}_1)) \times 100$ percent for a one-unit increase in the predictor.

Please see the associated video as I go over the math explaining how we interpret regression coefficients when the response variable is log transformed.

5.5 Remedial Measures: Linearization Transformations

We first ensure the variance has been stabilized and assumption 2 is met. If $f(x)$ does not accurately capture the relationship between the variables, we transform the predictor variable to meet assumption 1. Some writers call this a linearization transformation, as we seek to make the transformed version of the predictor variable, x^*, to be approximately linear with the response variable (or transformed y), i.e. $y = \beta_0 + \beta_1 x^* + \epsilon$. We do not consider transforming the response variable to deal with assumption 1, as transforming the response variable is likely to reintroduce violation of assumption 2.

The general strategy on how to transform the predictor is via a scatterplot of y (or y^*) against x. We use the pattern seen in the plot to decide how to transform the predictor. Some examples are shown in Figure 5.11 below.



(a) $x^* = \log(x)$ (b) $x^* = \frac{1}{x}$ (c) $x^* = \sqrt{x}$ (d) $x^* = x^2$

Figure 5.11: Transformations for x

5.5.1 Hierarchical Principle

One thing to be aware of is the **hierarchical principle**: if the relationship between the response and predictor is of a higher order polynomial (e.g. quadratic, cubic), the hierarchical principle states that the lower order terms should remain in the model. For example, if the relationship is of order h, fit $y = \beta_0 + \beta_1 x + \beta_2 x^2 + \cdots + \beta_h x^h + \epsilon$ via a multiple linear regression framework. We will see how to do this in the next module.

5.5.2 Interpretation with Log Transformed Predictor

A log transformation on the predictor is preferred over any other transformation, as we can still interpret the regression coefficient, $\hat{\beta}_1$, in a couple of ways:

- For an $a\%$ increase in the predictor, the predicted response **increases by $\hat{\beta}_1 \log(1 + \frac{a}{100})$.**
- $\log(1 + \frac{1}{100}) \approx \frac{1}{100}$ (Taylor series). So an alternative interpretation is: for a 1% increase in the predictor, the predicted response increases by approximately $\frac{\hat{\beta}_1}{100}$.

Please see the associated video as I go over the math explaining how we interpret regression coefficients when the predictor variable is log transformed.

5.5.3 Interpretation with Log Transformed Response and Predictor

If both response and predictor variables are log transformed, the regression coefficient, $\hat{\beta}_1$, can be interpreted in a couple of ways:

- For an $a\%$ increase in the predictor, the predicted response is **multiplied by $(1 + \frac{a}{100})^{\hat{\beta}_1}$.**
- $(1 + \frac{1}{100})^{\hat{\beta}_1} \approx 1 + \frac{\hat{\beta}_1}{100}$ (Taylor series). So an alternative interpretation is: for a 1% increase in the predictor, the predicted response **increases by approximately $\hat{\beta}_1$ percent.** Note that this approximation works better when $\hat{\beta}_1$ is small in magnitude.

Please see the associated video as I go over the math explaining how we interpret regression coefficients when the both response and predictor variables are log transformed.

5.5.4 Some General Comments about Assessing Assumptions and Transformations

- When assessing the assumptions with a residual plot, we are assessing if the assumptions are reasonably / approximately met.
- With real data, assumptions are rarely met 100%.

- If unsure, proceed with model building, and test how model performs on new data. If poor performance, go back to residual plot to assess what transformation will be appropriate.
- Assess the plots to decide which variables need to be transformed, and how. The choice of transformation should be guided by what you see in the plots, and not by trial and error.
- A residual plot should always be produced after each transformation. A Box Cox plot could also be produced. The plots should be assessed if the transformation helped in the way you intended.
- Solve assumption 2 first, then assumption 1.

5.6 R Tutorial

5.6.1 Example 1

The linear regression model involves several assumptions. Among them are:

1. The errors, for each fixed value of x, have mean 0. This implies that the relationship as specified in the regression equation is appropriate.
2. The errors, for each fixed value of x, have constant variance. That is, the variation in the errors is theoretically the same regardless of the value of x (or $\hat{y}$).
3. The errors are independent.
4. The errors, for each fixed value of x, follow a normal distribution.

To assess assumptions 1 and 2, we can examine scatterplots of:

- y versus x.
- residuals versus fitted values, $\hat{y}$.

Assumption 3 is assessed based on knowledge of the data. An autocorrelation (ACF) plot of the residuals may also be used.

Assumption 4 is assessed with a normal probability plot, and is considered the least crucial of the assumptions.

We will see how to generate the relevant graphical displays to help us assess whether the assumptions are met, and if needed, carry out transformations on the variable(s) so the assumptions are met.

For this tutorial, we will go over a dataset involving prices of used cars (Mazdas). The two variables are the sales price of the used car, and the age of the car in years. Download the data file, `mazda.txt` and read the data:

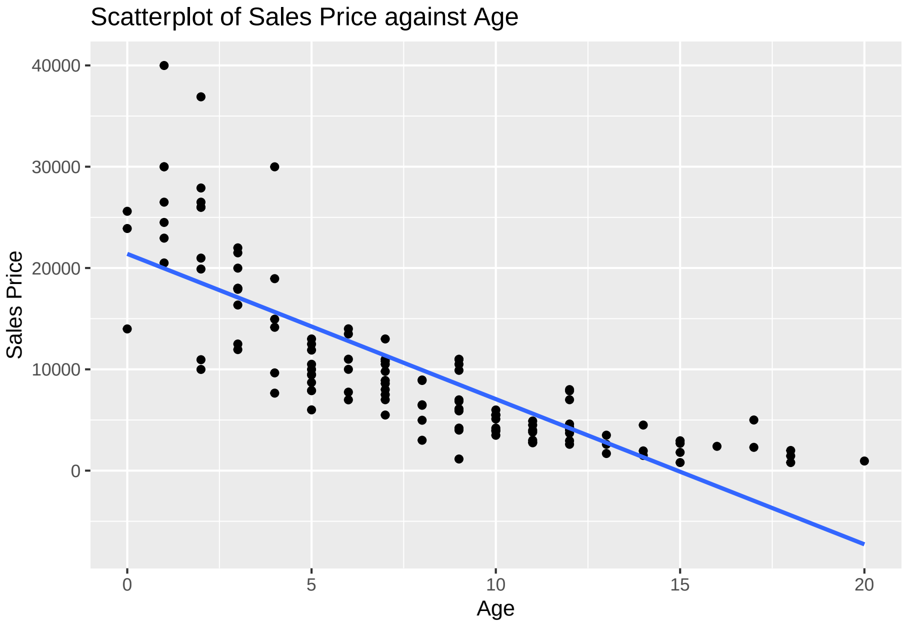
```r
Data<-read.table("mazda.txt", header=TRUE, sep="")
```

5.6.1.1 Model Diagnostics with Scatterplots

We can use a scatterplot of the response variable against the predictor to assess assumptions 1 and 2:

```
library(tidyverse)

##scatterplot, and overlay regression line
ggplot2::ggplot(Data, aes(x=Age,y=Price))+
  geom_point()+
  geom_smooth(method = "lm", se=FALSE)+
  labs(x="Age", y="Sales Price", title="Scatterplot of Sales Price against Age")
```



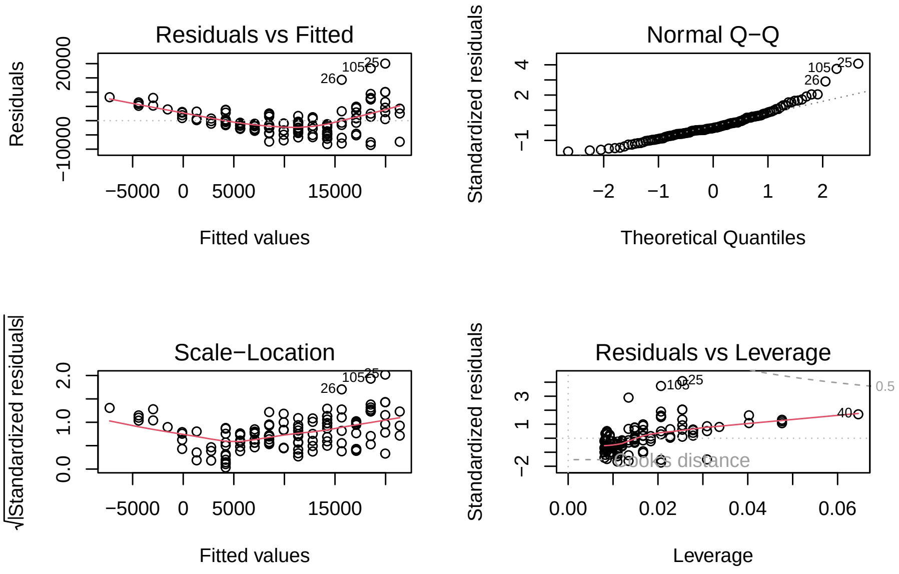
To assess assumption 1, the data points should be evenly scattered on both sides of the regression line, as we move from left to right. We do not see this in the scatterplot, so assumption 1 is not met. When age is between 0 and 2, the data points are mostly above the line. When age is between 5 and 11, the data points are mostly below the line, and when age is greater than 13, the data points are above the line.

To assess assumption 2, the vertical spread of the data points should be constant as we move from left to right. The spread seems to be decreasing as we move from left to right (or in other words, the spread is increasing as the response increases), so assumption 2 is not met.

5.6.1.2 Model Diagnostics with Residual Plots

Sometimes, a residual plot is easier to visualize than a scatterplot. We fit our SLR model using `lm()` as usual. Applying the `plot()` function to an object created with `lm()` actually produces a four diagnostic plots. To display the four diagnostic plots in a 2 by 2 array, we specify `par(mfrow = c(2, 2))` so the plotting window is split into a 2 by 2 array:

```r
result<-lm(Price~Age, data=Data)
par(mfrow = c(2, 2))
plot(result)
```



- The first plot (top left) is the residual plot, with residuals on the y-axis and fitted values on the x-axis. The residual plot can be used to address assumptions 1 and 2. A red line is overlayed to represent the average value of the residuals for differing values along the x-axis. This line should be along the x-axis without any apparent curvature to indicate the form of our model is reasonable. This is not what we see, as we see a clear curved pattern. So assumption 1 is not met. For assumption 2, we want to see the vertical spread of the residuals to be fairly constant as we move from left to right. We do not see this in the residual plot; the vertical spread increases as we move from left to right, so assumption 2 is not met.

- The second plot (top right) is the normal probability plot (also called a QQ plot), and addresses assumption 4. If the residuals are normal, the residuals should fall along the 45 degree line. The regression model is fairly robust to this assumption though; the normality assumption is the least crucial of the four.

- The third plot (bottom left) is a plot of the square root of the absolute value of the standardized residuals against the fitted values (scale-location). This plot should be used to assess assumption 2, the constant variance assumption. A red line is overlayed to represent the average value on the vertical axis for differing values along the x-axis. If the variance is constant, the red line should be horizontal and the vertical spread of the plot should be constant. This plot should be used to assess assumption 2, if we have a small sample size. Otherwise, this plot should tell a similar story to the first plot (top left) when assessing assumption 2.

- The last plot (bottom right) is a plot to identify influential outliers. Data points that lie in the contour lines with large Cook's distance are influential. None of our data points have Cook's distance greater than 0.5. As a general rule of thumb, observations with Cook's distance greater than 1 are flagged as influential. We will talk more about influential observations in a future module.

Now that we know that both assumptions 1 and 2 are not met. We need to transform the response variable first, to stabilize the variance.

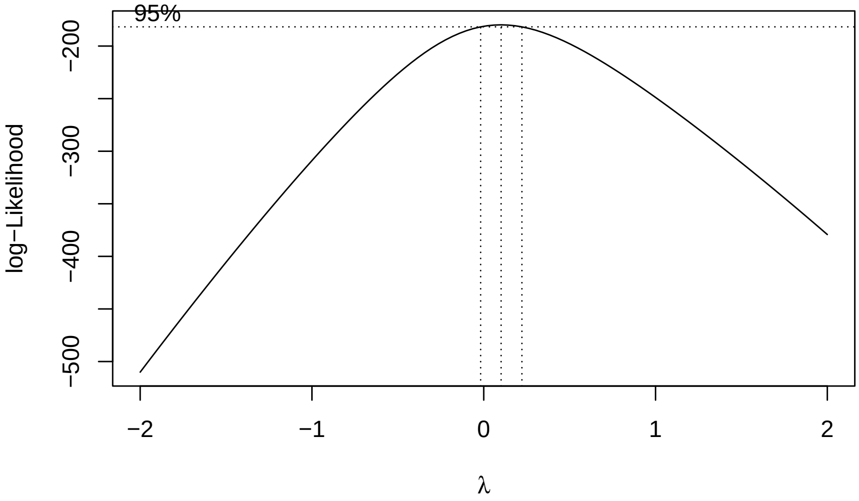
Based on the residual plot, we see that the variance of the residuals increases as we move from left to right. So we know we need to transform the response variable using $y^* = y^\lambda$ with $\lambda < 1$. A log transform should be considered since we can still interpret regression coefficients.

5.6.1.3 Box Cox Transformation on y

The Box Cox plot can be used to decide how to transform the response variable. The transformation takes the form $y^* = y^\lambda$, with the value of λ to be chosen. If $\lambda = 0$, we perform a log transformation.

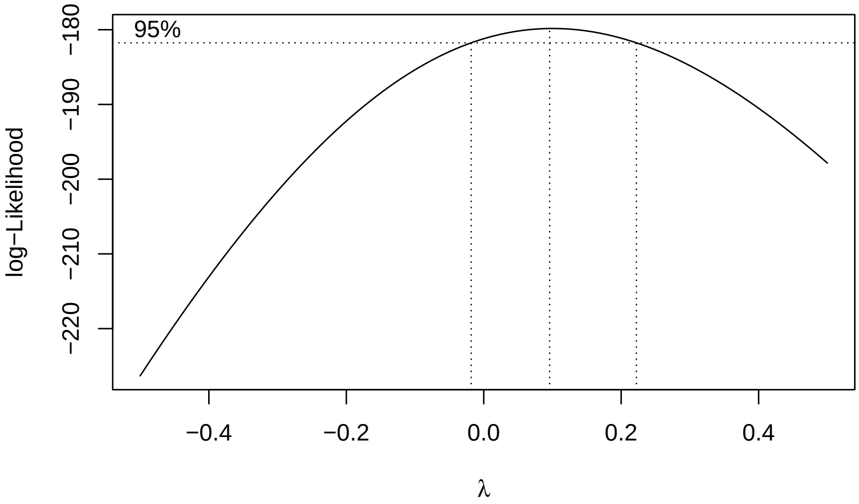
We will use the `boxcox()` function from the `MASS` package:

```
library(MASS) ##to use boxcox function
MASS::boxcox(result)
```



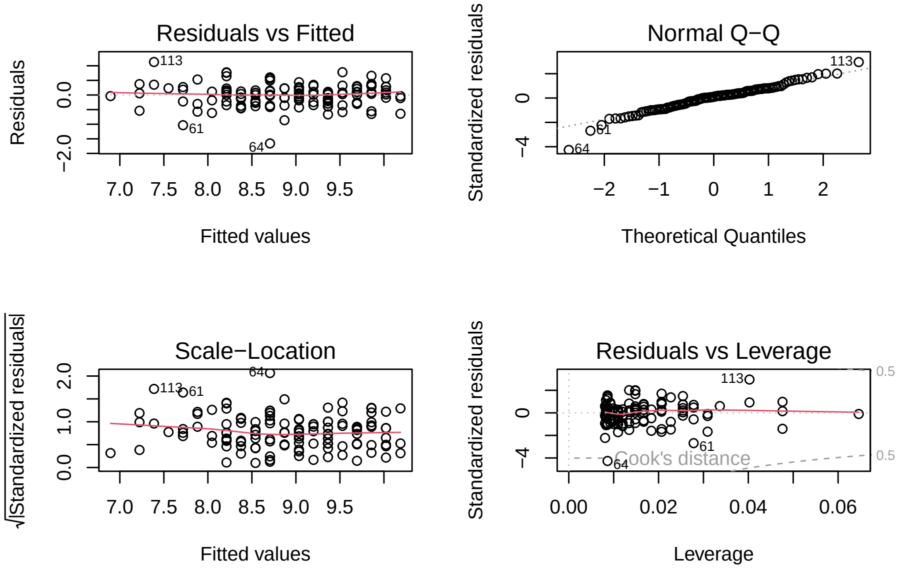
We can “zoom in” on the plot to have a better idea about the value of λ we can use, by specifying the range of `lambda` inside the function:

```
##adjust lambda for better visualization. Choose lambda between -0.5 and 0.5
MASS::boxcox(result, lambda = seq(-0.5, 0.5, 1/10))
```



We can choose any value of λ within the CI. A log transformation is preferred if possible, since we can still interpret coefficients. Since 0 lies in the CI, we choose $\lambda = 0$, to log transform the response variable to get $y^* = \log(y)$. We regress y^* against x, and check the resulting residual plot:

```r
##transform y and then regress ystar on x
ystar<-log(Data$Price)
Data<-data.frame(Data,ystar)
result.ystar<-lm(ystar~Age, data=Data)
par(mfrow = c(2, 2))
plot(result.ystar)
```



We need to reassess assumptions 1 and 2 after the transformation.

- For assumption 2, we see that the vertical spread of the residuals in the residual plot (top left) is fairly constant, as we move from left to right. So assumption 2 is met. The log transformation worked.
- We also notice that the residuals are now evenly scattered across the horizontal axis in the residual plot (top left). So assumption 1 is now met.

We do not need to perform any other transformations.

5.6.1.4 Interpreting Coefficients with Log Transformed Response

So our regression equation is

```
result.ystar
```

```
##
## Call:
## lm(formula = ystar ~ Age, data = Data)
##
## Coefficients:
## (Intercept)          Age
##     10.1878      -0.1647
```

$\hat{y^*} = 10.1878 - 0.1647x$, where $y^* = \log(y)$. To interpret the slope:

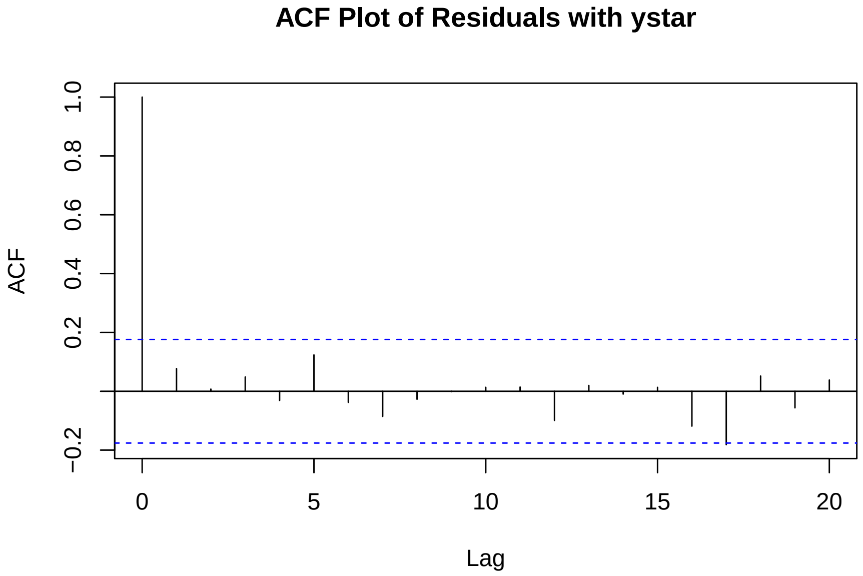
- The price of used Mazdas is multiplied by $\exp(-0.1647) = 0.8481481$ for each year older the car is.
- The price of used Mazdas decreases by $(1 - 0.8481481) \times 100$ percent, or 15.18519 percent, for each year older the car is.

5.6.1.5 ACF Plot of Residuals

We have yet to assess the assumption that the observed prices are independent from each other. Assuming that these prices are from different cars and not the same car measured repeatedly over time, there is no reason to think the prices are dependent on each other.

We an also produce an ACF plot to confirm our thought:

```
acf(result.ystar$residuals, main="ACF Plot of Residuals with ystar")
```



None of the ACFs beyond lag 0 are significant, so we don't have evidence that the observations are dependent on each other.

5.6.2 Example 2

For this second example, we will go over an example for the `faraway` package. The dataframe is called `gala`. The data are about species diversity on the Galapagos Islands. There are 30 islands, and for each island, we have data on 7 variables. We will focus on the variable `Species`, which denotes the number of plant species found on the island, and `Area`, the area of the island in squared kilometers. We wish to see how the number of plant species of an island is related to the area of the island.

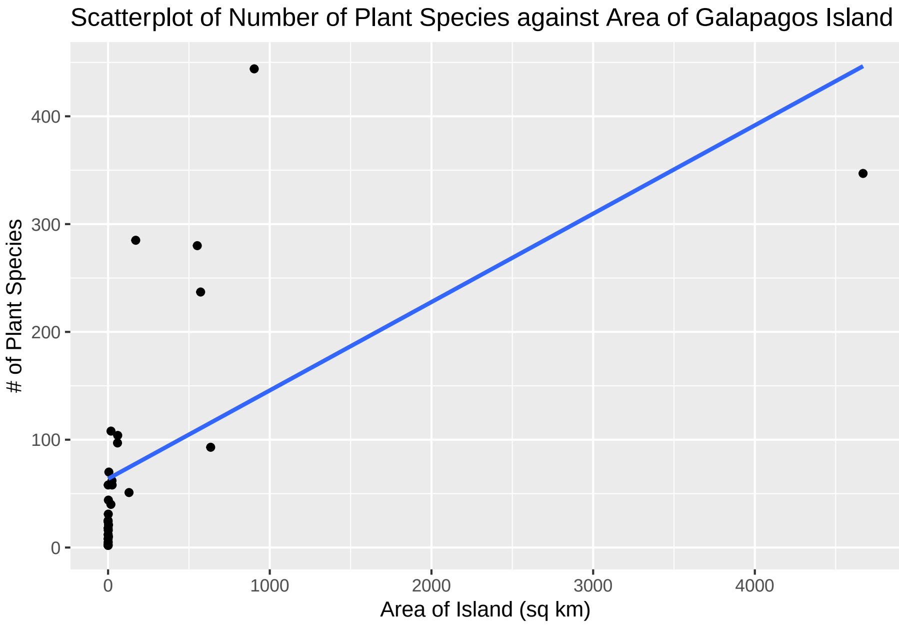
```
library(faraway)
Data<-faraway::gala
```

5.6.2.1 Model Diagnostics with Scatterplots

We can use a scatterplot of the response variable against the predictor to assess assumptions 1 and 2.

```
library(tidyverse)

##scatterplot, and overlay regression line
ggplot2::ggplot(Data, aes(x=Area,y=Species))+
  geom_point()+
  geom_smooth(method = "lm", se=FALSE)+
  labs(x="Area of Island (sq km)", y="# of Plant Species",
       title="Scatterplot of Number of Plant Species against Area of Galapagos Island")
```



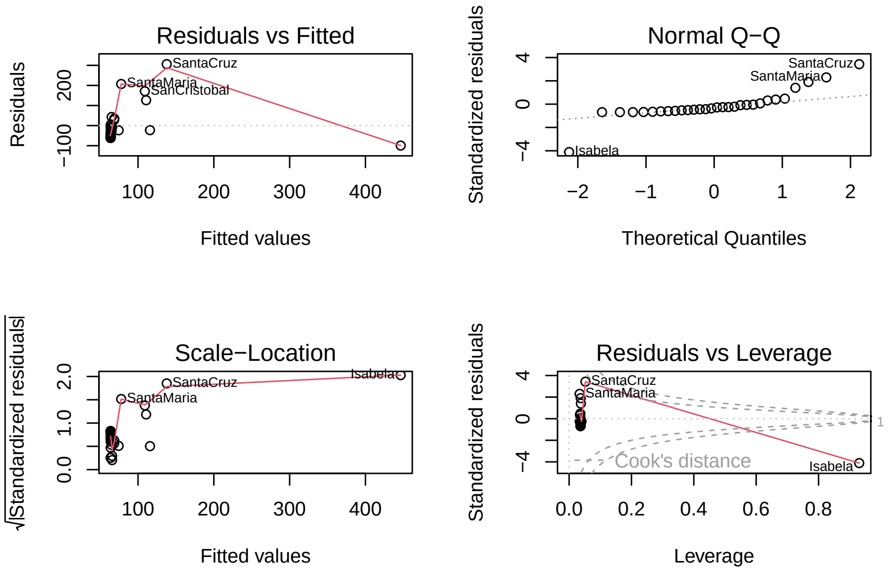
To assess assumption 1, the data points should be evenly scattered on both sides of the regression line, as we move from left to right. This plot looks nonlinear.

To assess assumption 2, the vertical spread of the data points should be constant as we move from left to right. This can be a bit difficult to assess with this scatterplot, although observations with small areas are closer to the line, which suggests the assumption is not met.

5.6.2.2 Model Diagnostics with Residual Plots

Fairly often, when assessing regression assumptions, a residual plot is easier to visualize than a scatterplot.

```
result<-lm(Species~Area, data=Data)
par(mfrow = c(2, 2))
plot(result)
```



- From the residual plot (top left), we see a curved pattern, so we have a nonlinear relationship. Assumption 1 is not met.

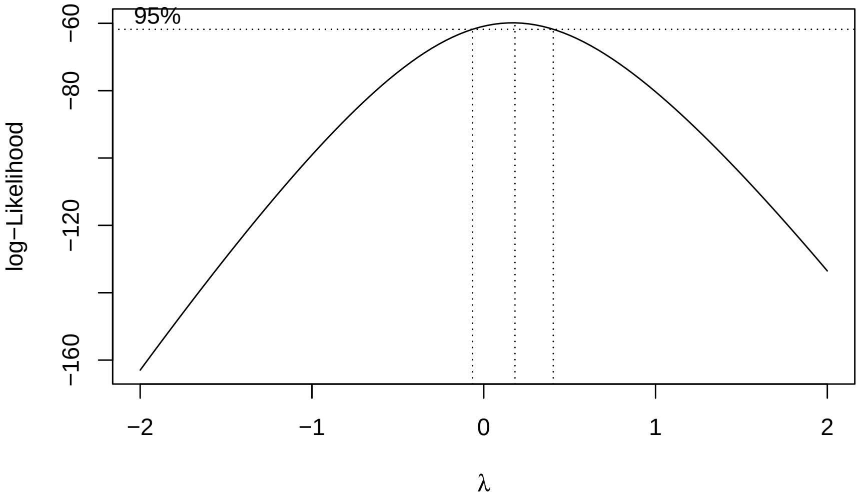
- From the scale-location plot (bottom left), the vertical variance of the plots appear to be higher for islands with larger fitted valies, so assumption 2 is not met.

Now that we know that both assumptions 1 and 2 are not met. We need to transform the response variable first, to stabilize the variance.

5.6.2.3 Box Cox Transformation on y

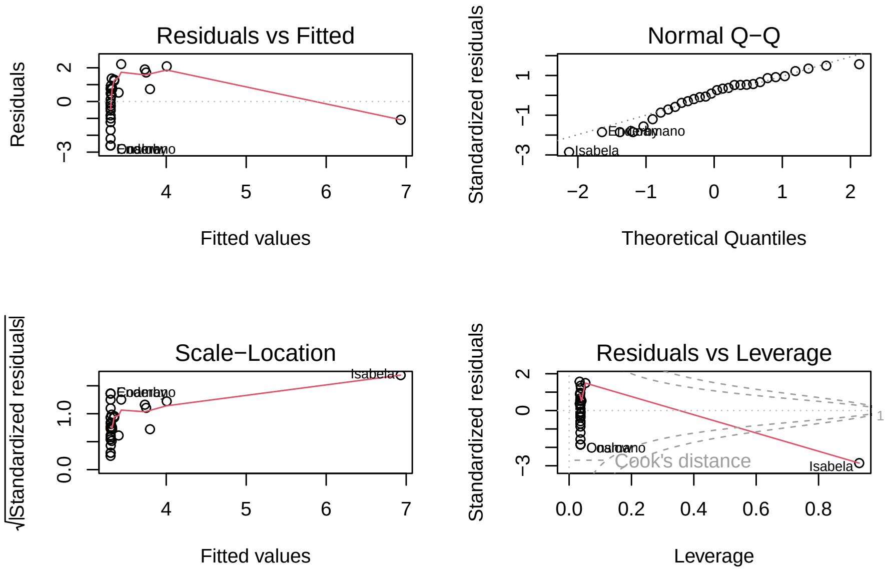
From the scale-location plot, we see the variance of the residuals is increasing, so we expect to transform the response variable with $y^* = y^\lambda$ with $\lambda < 1$. So see which specific value of λ to use, we can use the Box Cox plot:

```
library(MASS)
MASS::boxcox(result)
```



A log transformation is preferred if possible, since we can still interpret coefficients. Since 0 lies in the CI, we choose $\lambda = 0$, to log transform the response variable to get $y^* = \log(y)$. We regress y^* against x, and check the resulting residual plot:

```
##log transform response and add to dataframe
Data$y.star<-log(Data$Species)
##perform new regression
result.ystar<-lm(y.star~Area, data=Data)
par(mfrow = c(2, 2))
plot(result.ystar)
```



We need to reassess assumptions 1 and 2 after the transformation.

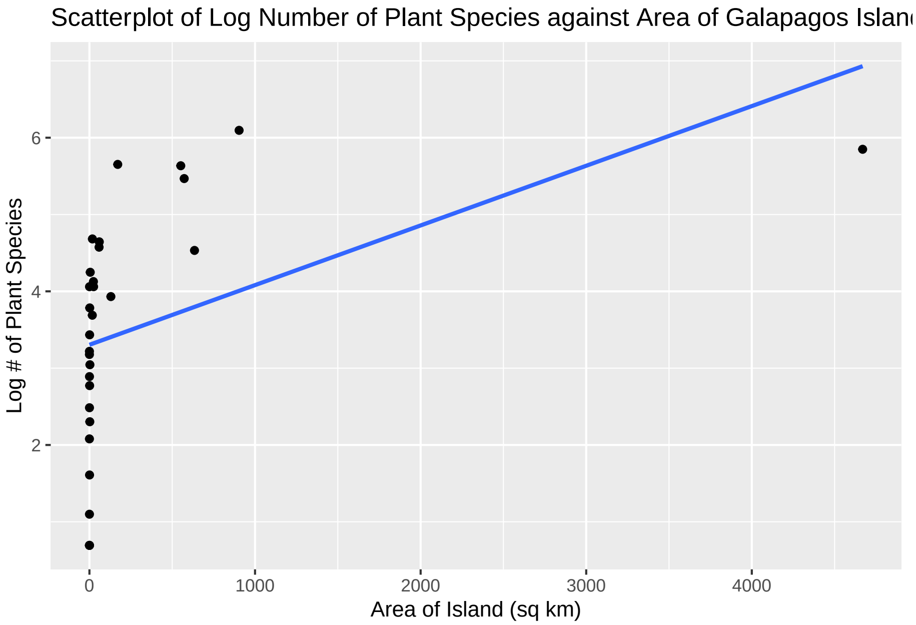
- For assumption 2, we see that the vertical spread of the residuals in the residual plot (top left) is fairly constant, as we move from left to right. So assumption 2 is met. The log transformation worked in stabilizing the variance.

- However, the residual plot still appears to be nonlinear. So assumption 1 is still not met.

5.6.2.4 Transformation on x

To see which specific transformation on the predictor to use, we create a scatterplot of the transformed response and the predictor.

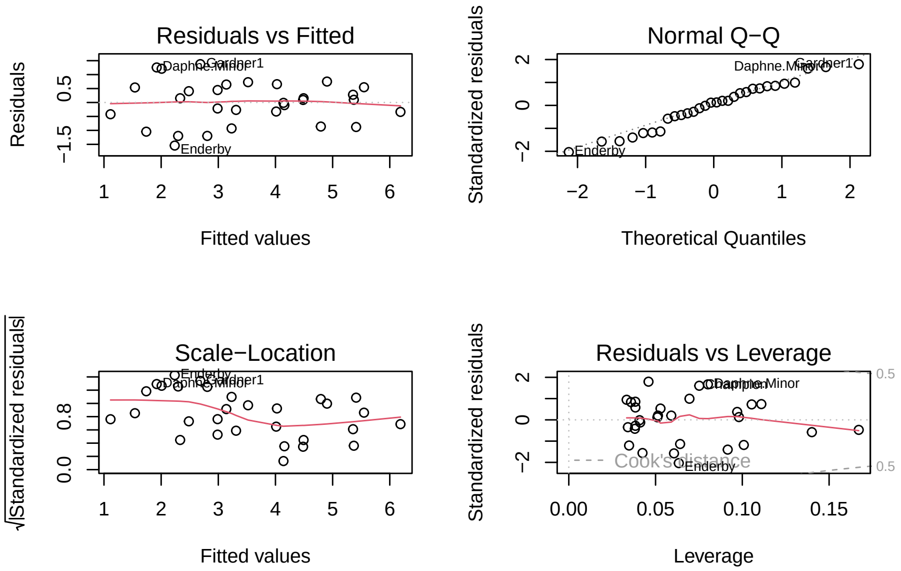
```
ggplot2::ggplot(Data, aes(x=Area,y=y.star))+
  geom_point()+
  geom_smooth(method = "lm", se=FALSE)+
  labs(x="Area of Island (sq km)", y="Log # of Plant Species",
       title="Scatterplot of Log Number of Plant Species against Area of Galapagos Isla
```



This plot resembles a logarithmic curve, so we use a log transformation on the predictor, so let $x^* = \log(x)$. As usual, a log transformation is preferred since we can still interpret the regression coefficients. We then perform a regression using both the log transformed response variable and predictor, and assess the diagnostic plots.

```
##log transform predictor and add to dataframe
Data$x.star<-log(Data$Area)

##perform new regression
result.xystar<-lm(y.star~x.star, data=Data)
par(mfrow = c(2, 2))
plot(result.xystar)
```



Based on the residual plot (top left), both assumptions are met. The residuals are evenly scattered across the horizontal axis with no pattern. The vertical spread of the residuals is also constant. So the transformations worked.

5.6.2.5 Interpreting Coefficients with Log Transformed Response and Predictor

So our regression equation is

```
result.xystar
```

```
##
## Call:
## lm(formula = y.star ~ x.star, data = Data)
##
## Coefficients:
## (Intercept)       x.star
##      2.9037       0.3886
```

$\hat{y^*} = 2.9037 + 0.3886x^*$, where $y^* = \log(y)$ and $x^* = \log(x)$. There are a couple of ways to interpret the slope:

- For a 1% increase in the area of a Galapagos island, the number of plant species found on the island is multiplied by $(1.01)^{0.3886} = 1.003874$, OR
- For a 1% increase in the area of a Galapagos island, the number of plant species increases by about 0.3886%.

Chapter 6

Introduction

You can label chapter and section titles using `{#label}` after them, e.g., we can reference Chapter 6. If you do not manually label them, there will be automatic labels anyway, e.g., Chapter 8.

Figures and tables with captions will be placed in `figure` and `table` environments, respectively.

```
par(mar = c(4, 4, .1, .1))
plot(pressure, type = 'b', pch = 19)
```

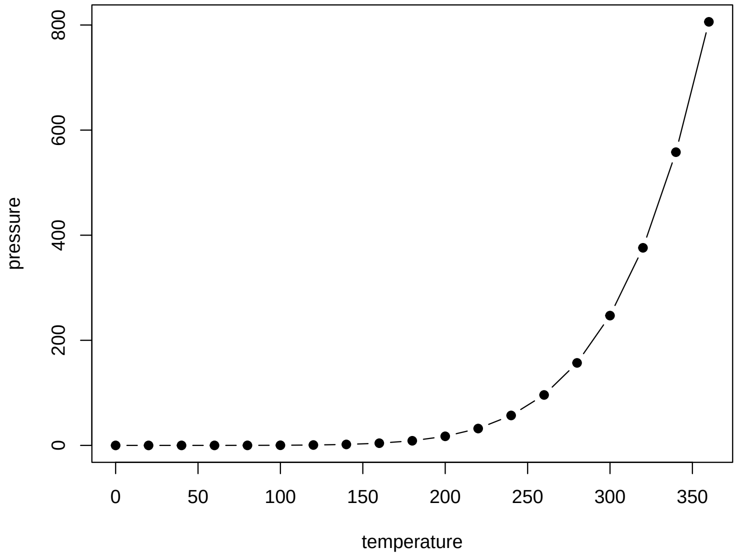


Figure 6.1: Here is a nice figure!

Table 6.1: Here is a nice table!

Sepal.Length	Sepal.Width	Petal.Length	Petal.Width	Species
5.1	3.5	1.4	0.2	setosa
4.9	3.0	1.4	0.2	setosa
4.7	3.2	1.3	0.2	setosa
4.6	3.1	1.5	0.2	setosa
5.0	3.6	1.4	0.2	setosa
5.4	3.9	1.7	0.4	setosa
4.6	3.4	1.4	0.3	setosa
5.0	3.4	1.5	0.2	setosa
4.4	2.9	1.4	0.2	setosa
4.9	3.1	1.5	0.1	setosa
5.4	3.7	1.5	0.2	setosa
4.8	3.4	1.6	0.2	setosa
4.8	3.0	1.4	0.1	setosa
4.3	3.0	1.1	0.1	setosa
5.8	4.0	1.2	0.2	setosa
5.7	4.4	1.5	0.4	setosa
5.4	3.9	1.3	0.4	setosa
5.1	3.5	1.4	0.3	setosa
5.7	3.8	1.7	0.3	setosa
5.1	3.8	1.5	0.3	setosa

Reference a figure by its code chunk label with the `fig:` prefix, e.g., see Figure 6.1. Similarly, you can reference tables generated from `knitr::kable()`, e.g., see Table 6.1.

```r
knitr::kable(
  head(iris, 20), caption = 'Here is a nice table!',
  booktabs = TRUE
)
```

You can write citations, too. For example, we are using the **bookdown** package (Xie, 2023) in this sample book, which was built on top of R Markdown and **knitr** (Xie, 2015).

Chapter 7

Literature

Here is a review of existing methods.

7.1 New section

Add some text here. BLAH BLAH

Chapter 8

Methods

We describe our methods in this chapter.

Math can be added in body using usual syntax like this

8.1 math example

p is unknown but expected to be around 1/3. Standard error will be approximated

$$SE = \sqrt(\frac{p(1-p)}{n}) \approx \sqrt{\frac{1/3(1-1/3)}{300}} = 0.027$$

You can also use math in footnotes like this[1].

We will approximate standard error to 0.027[2]

[1] where we mention $p = \frac{a}{b}$

[2] p is unknown but expected to be around 1/3. Standard error will be approximated

$$SE = \sqrt(\frac{p(1-p)}{n}) \approx \sqrt{\frac{1/3(1-1/3)}{300}} = 0.027$$

Chapter 9

Applications

Some *significant* applications are demonstrated in this chapter.

9.1 Example one

9.2 Example two

Chapter 10

Final Words

We have finished a nice book.

Bibliography

Xie, Y. (2015). *Dynamic Documents with R and knitr*. Chapman and Hall/CRC, Boca Raton, Florida, 2nd edition. ISBN 978-1498716963.

Xie, Y. (2023). *bookdown: Authoring Books and Technical Documents with R Markdown*. R package version 0.34.