

Handout 1.1: Introduction to Modern Regression Methods

Dr. Bean - Stat 5100

1 About me



- Graduated from BYU-Idaho in 2014 with a Bachelors of Science in Applied Mathematics.
- Graduated from Utah State University in 2019 with a PhD in Mathematical Sciences.
- Current interests include basketball, cross country skiing, hiking and spending time with my wife and daughter.

What is a creative, yet appropriate, question that you have about the life/career of the instructor?

2 Why Modern Regression Methods?

Statistics, in the words of Dr. Bin Yu, is the “science that solves data problems.” This science becomes more and more relevant in a world inundated with data. From the late Leo Breiman:

The uses of statistics pervade our society. They are used and terribly misused all through the social sciences and health fields. ... It is surprising how much the world around us depends on the use of statistics. ... It’s odd that even though the articles

involving statistics in the newspapers far outnumber those involving say, physics or chemistry, people in general know very little about what we do.

In this class, we will learn several of the foundational approaches for using data to make **predictions**. Perhaps more importantly, we will discuss the **cautions** we must consider when using and interpreting model output.

Why are YOU taking this course?

3 Functional vs Statistical Modeling

We learn about functions in Math 1050 (College Algebra), a functional model takes a set of inputs X and produces a (set of) outputs Y , i.e.

$$Y = f(X)$$

Example: You write a function to model the profits from your lemonade stand. You rent the stand for \$200 a month and sell each glass of lemonade for \$1.00. If it costs you \$0.25 to make the lemonade then your monthly profits Y could be modeled as a function of the number of lemonade glasses you sell x

$$Y = 0.75x - 200.$$

The key to a functional model is that each input x produces a **unique** output Y .

In a **statistical model** we assume that the values of Y can be modeled by a function *plus* some “random noise” ϵ . The presence of the ϵ term allows for many different values of Y for the same set in inputs x .

$$Y = f(X) + \epsilon$$

Example: The relationship between ground snow and elevation in Utah (see figure 1).

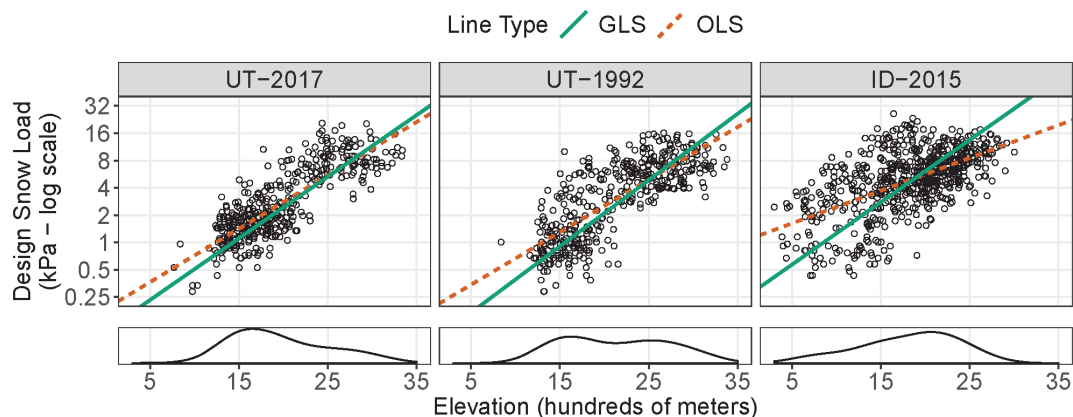


Figure 1: Plot of design ground snow loads (log-scale) vs elevation in and near Utah.

4 The Key Assumption (why this class exists)

The key assumption (and the foundation for this course) we make is that ϵ follows a probability distribution. Specifically we assume that

$$\epsilon \sim^{i.i.d} N(0, \sigma^2). \quad (1)$$

If this assumption is valid, then our **estimates** of the **model parameters** will come from well-defined probability distributions, which will allow us to determine if the linear relationships between our explanatory variables and our response variable are significant. This process is often called **statistical inference**.

What do each of the symbols mean in (1) and why might they be important?

- **independence:** Knowing the value of one of the residuals should tell us nothing about the rest.
- **identically distributed:** Each of the residuals come from the same probability distribution.
- **zero mean:** The residuals have an average value of zero (i.e. the model is not biased).
- **constant variance:** The spread of the residuals is constant across all predictions.

Why they are important is something we will talk about for the next several weeks.

5 Why “Linear Regression”?

5.1 Why Linear?

Model are composed of:

- **coefficients:** These are *constant* values that are *estimated* to optimize the model fit.
- **variables:** These are the *observed* values, calculated from the data that we use to estimate parameters or make predictions.

A model is considered “linear” if it can be written as a sum of coefficients β multiplied by a set of variables X_k , ($k = 1, 2, \dots, p - 1$), i.e.

$$Y = \sum_i \beta_i f_i(X_1, \dots, X_{p-1})$$

This means that you can have nonlinear variables as long as the coefficients are linear.

Which of the following models are linear and which are non-linear?

- $Y = \beta_0 + \beta_1 X_1 + \epsilon$
- $Y = \beta_0 + \beta_1 e^{X_1} + \epsilon$
- $Y = \beta_0 + \beta_1 X_1 X_2 + \epsilon$
- $Y = \beta_0 + X_1^{\beta_1} + \epsilon$

5.2 Why Regression?

Based on concept that things tend to “regress” to the mean:

Example: heights of fathers vs sons:

- Tall fathers tend to have tall sons, but those sons will tend to be shorter than their fathers.
- Short fathers tend to have short sons, but those sons will tend to be taller than their fathers.
- Thus, a line comparing “standard deviations” of fathers and sons heights will have a slope approximately equal to one, while the **regression** line will have a slope that is less than one.
- Because things regress to the mean, the regression line will always be flatter than the standard deviation line.

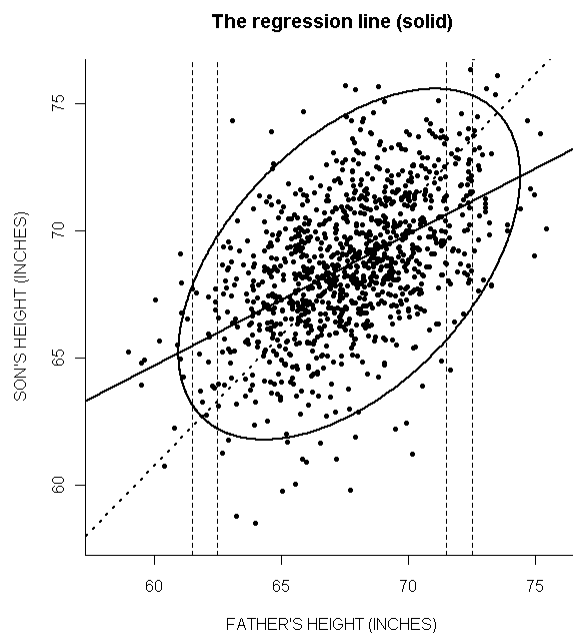


Figure 2: Plot of father vs sons' heights. The dotted line is the standard deviation line while the solid line is the regression line.

Handout 1.2: Introduction to Hypothesis Testing

Dr. Bean - Stat 5100

1 Why Hypothesis Testing?

In statistics, hypothesis tests are a way to determine if an **observed** difference is **significant** or simply due to chance. The key ingredients of a hypothesis test are:

- A null and alternative hypothesis.
- An observed statistic taken from a sample of the population.
 - Example: $t = \frac{\bar{X} - \mu_0}{SE\{\bar{X}\}}$
- An assumed probability distribution for the test statistic IF the null hypothesis is true.

The climax of a hypothesis test is the determination of the **p-value**. If the p-value is small (< 0.05), we reject the null hypothesis, and if it is not small, we fail to reject the null hypothesis.

Note that we *never* accept the alternative hypothesis, we simply *fail to reject* the null hypothesis.

What is a p-value?

The probability of obtaining our sample statistic, or one more extreme, if the null hypothesis was true.

Why is the assumed distribution for the test-statistic such a big deal?

We use the assumed probability distribution to calculate the p-value, and the p-value is how we determine the “significance” of our results. If the probability distribution is not appropriate then the p-value will be worthless.

How can we know the distribution of the test statistic?

- Through visualizations: Histograms, qqplots, boxplots.
- More often, the **Central Limit Theorem** assures us that the test statistic will follow a normal probability distribution.

2 Example:

Researchers have studied how the amount of sunlight bamboo is exposed to affects the speed of growth. One study compared the growth of 50 bamboo shoots grown under standard conditions to the growth of 49 bamboo shoots that had been exposed to 10% more sunlight. The growth was measured 40 days after planting. The observed mean and sd for bamboo under the standard growing conditions were 32.04 inches and 5.82 inches while the observed mean and sd for the more sunlight bamboo were 28.61 inches and 6.32 inches.

Hypothesis:

$$\begin{aligned}H_0 : \mu_1 &= \mu_2 \\H_A : \mu_1 &\neq \mu_2\end{aligned}$$

We will never know the values of μ_1 and μ_2 . They are **population parameters** that we could only know if we sampled every unit in the population (which, in this case, would be all bamboo shoots grown in the two lighting conditions).

Rather, we **estimate** the values of these population parameters with our samples, giving us values of $\bar{X}_1 = 32.04$ and $\bar{X}_2 = 28.61$ and an observed difference of $\bar{X}_1 - \bar{X}_2 = -3.43$.

If the null hypothesis was true then the observed difference would follow a t-distribution centered at 0 with a standard deviation (assuming pooled variances) of 1.221. This also means that our observed value of $t = \frac{-3.43}{1.221} = -2.81$. The p-value associated with our observation is 0.006.

The p-value in this setting is:

The probability of having an observed difference between the two bamboo groups as, or more, extreme than -3.43 IF the null hypothesis (no difference) was in fact true.

The p-value says that our observed difference would have been **very unlikely** (less than a 1/100 chance) if the null hypothesis was actually true. This gives us evidence to **reject the null hypothesis** and conclude that the growth rate of bamboo is different when sunlight conditions change.

To summarize:

1. We make a claim about the value of the population parameters.
2. We test the claim by obtaining statistics from a sample of the population.
3. We determine the probability of obtaining our sample statistic (or something more extreme) IF the null hypothesis was true.
4. If the probability of our observation is LOW, we reject the null hypothesis, if it is NOT LOW, then we fail to reject the null hypothesis.

Suppose the p-value for the above example had been 0.02 instead of 0.006. Would your conclusions change? What about if the p-value had been 0.13? How about 0.98?

Same conclusions for 0.02, but fail to reject the null hypothesis for p-values of 0.13 and 0.98.

3 Inference vs Prediction

In linear modeling, we assume that the population follows the model:

$$Y = \beta_0 + \beta_1 X_{i,1} + \beta_2 X_{i,2} + \cdots + \beta_{p-1} X_{i,p-1} + \epsilon_i$$

In these models, we can conduct **inference** to determine if the linear relationship between an explanatory variable X_k and the response variable Y is significant.

HOWEVER, we can also use these same models to try and make accurate **predictions** of Y .

Models that are accurate tend to have significant coefficients, but models with significant coefficients are not always accurate.

Our approach to linear modeling in this class changes slightly when our primary interest is establishing significance, vs being accurate.

Stat 5100 in R: Setup and Introduction

Contents

1 Introduction

2 Should I take this class using R?

3 R Setup

- 3.1 Download R
- 3.2 Download RStudio
- 3.3 Download the Stat 5100 R Package
- 3.4 Test your R installation

4 R Basics

- 4.1 Running commands and scripts
- 4.2 Getting help on functions
- 4.3 Reading in data
 - 4.3.1 Reading in Stat 5100 datasets
 - 4.3.2 Reading in data by CSV
 - 4.3.3 Creating data by hand
- 4.4 Altering datasets
- 4.5 Filtering datasets

5 Miscellaneous Notes and FAQ

1 Introduction

As of Spring 2021, the Stat 5100 course is being offered in R for the very first time. You have the option to complete this course in either SAS or R. Both languages are very popular and useful for the material you will learn in this class, but are very different in style.

Here are the main things you should know about each language:

SAS

- Costs a lot of money (You get to use SAS for free in this class because you are affiliated with an academic institution)
- Very useful “out of the box.” Many powerful features are available right away.
- Commonly used at large corporations with large datasets.
- The syntax (the rules of the language) are incredibly different from nearly every other programming language, so learning SAS will not necessarily help you learn general programming skills.
- The philosophy of SAS is “let’s give you a huge amount of output, we’ll let you figure out which parts of the output you need.” If you create a regression model in SAS, you will be thrown a huge amount of output and diagnostics right away by default. Although the large amount of output can feel overwhelming, it can be very nice to have when you’re not sure what you’re looking for.

R

- Completely free and open-source for everyone.
- A very lightweight and small language (compared to SAS), so many powerful features are not available natively.
- The most commonly used language for data science and statistics in both academia and industry. Behind Python, R is the second most commonly used language for data science in industry. (We would love to offer this course in Python eventually!)
- The syntax of R is similar to some other scripting languages such as Python, so learning R will help you learn general programming skills.
- Requires stronger programming skills to get started.
- The philosophy of R is “We assume you know what you’re looking for, so we’re going to give you the bare minimum.” In contrast with SAS, R does not give very much output and diagnostics. To obtain the same things that SAS gives you, you must know exactly what you are looking for and run appropriate functions to give you the output you want.
- To expand the power of R, we use what are called “packages.” Packages are code that is written by other people, and we can use many of these packages to do the same things that SAS can do.

2 Should I take this class using R?

Take this class using R if:

1. You have used R before and you are comfortable programming. If you have taken Stat 5050: Introduction to R here at USU, then you should be well-prepared to take this class using R. Ideally you should know the basics of manipulating dataframes and running simple functions.
2. You are okay being a guinea pig: this is the first semester this course is being offered in R so we can not guarantee a 100% smooth experience in all cases.

3 R Setup

To get ready to complete this class using R, we’ll need to set up your system to have both R and RStudio.

R is the language itself, and RStudio is simply a program that interacts with the language and provides a nice and clean environment for working with R. If you are familiar with programming, RStudio is just an IDE (integrated development environment) for R. RStudio is by far the most popular IDE for R. You can download and work with R without using RStudio, but you are restricted to doing so in your system’s terminal which can be very hard to work with.

3.1 Download R

Navigate yourself to [the official R download page](#). Click on any of the mirror links, and then click “Download R” for whatever operating system you use. Follow the instructions of the installer, and then you should be good to go.

3.2 Download RStudio

Navigate yourself to [the official RStudio download page](#). Click on “RStudio Desktop,” then click on “Download RStudio Desktop” underneath the open source edition. Click on the download button again under the free version. When you are set up, you should be able to open up RStudio and have it automatically identify your R installation.

3.3 Download the Stat 5100 R Package

Next, you will want to download the Stat 5100 R package. This package contains all of the data that we will use in this class, as well as many complex functions and plotting routines. The goal of this package is to take the focus off of the programming and move the focus to the topic of the class: regression.

To download and install the package, type the following into the console in RStudio:

```
install.packages("devtools")
devtools::install_github("ethanancell/stat5100package")
```

3.4 Test your R installation

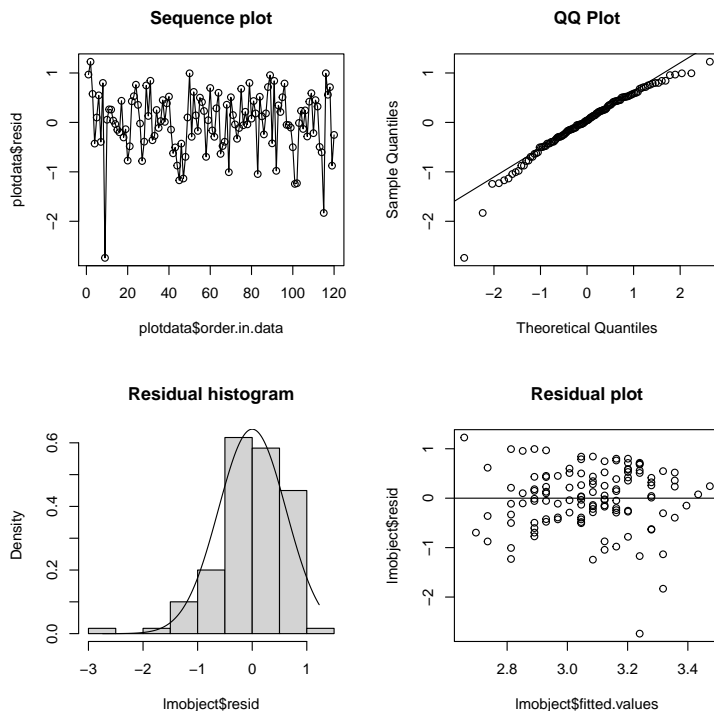
Try the following in your R console to make sure that you have your installation set up correctly. When you plot an image, you should see it pop up in the bottom-right corner of your screen under the “Plots” tab of RStudio.

```
# Load the Stat 5100 package into memory
library(stat5100)

# Load an included dataset from the library
data("college")

# Fit a regression model on the college dataset
college_lm <- lm(gpa ~ act, data = college)

# Show some plots
visual_assumptions(college_lm)
```



If you get the same results as the above, then your installation should be correctly setup and you will be good to go!

4 R Basics

Here is quick review of some basic skills you should be familiar with in R. If you feel comfortable doing all of the things below, then you should be just fine in this class.

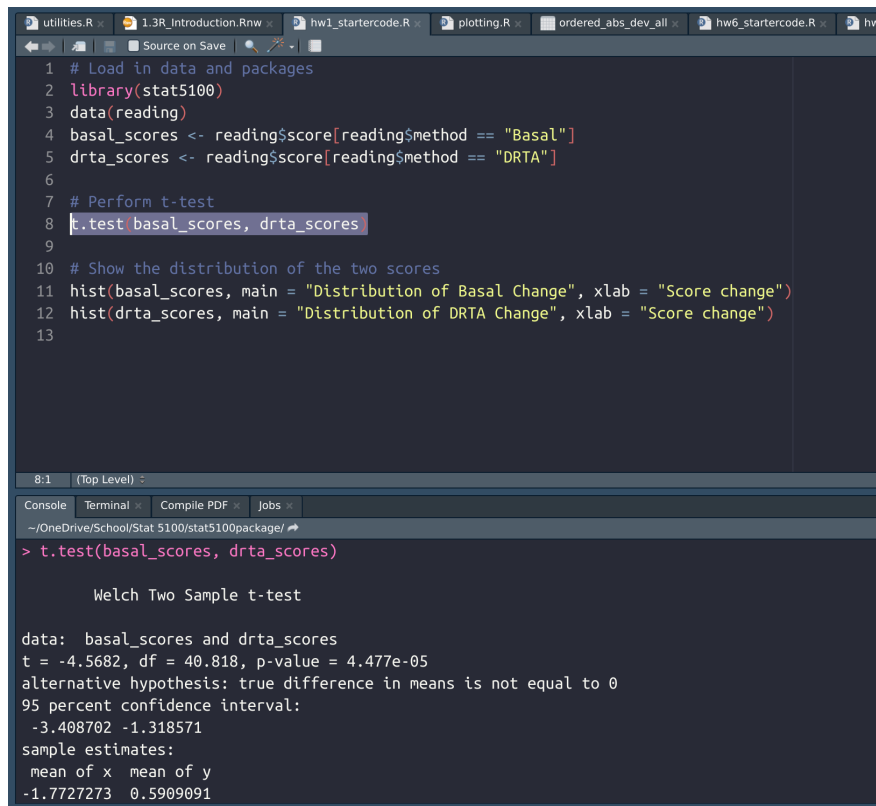
4.1 Running commands and scripts

In RStudio, there are two places where you see code. You have a console, and you should also have a script window. By default, just the console window will be open. In there, you can type R commands and there will be ran in an interactive manner.

```
> print("Running some basic commands in my console window")
[1] "Running some basic commands in my console window"
> 3+5
[1] 8
> rnorm(4)
[1] 1.0571192 0.2427443 2.1226940 -0.2715278
> |
```

You can also open R code inside a script window. If desired, you can run the entire R script. More often, you will be running bits and pieces of an R script at a time. Putting your code in a script has an advantage in that it's very easy to jump around your code and test little bits at a time without needing to type out your code every time.

If you go to File → Open File, you can open an R file inside RStudio. To run one chunk of code, select it with your mouse and then click either “Run” at the top right part of the script window, or Ctrl+Enter (my favorite method and much quicker than clicking run!)



The screenshot shows the RStudio interface. The top pane displays an R script with the following code:

```
1 # Load in data and packages
2 library(stat5100)
3 data(reading)
4 basal_scores <- reading$score[reading$method == "Basal"]
5 drta_scores <- reading$score[reading$method == "DRTA"]
6
7 # Perform t-test
8 t.test(basal_scores, drta_scores)
9
10 # Show the distribution of the two scores
11 hist(basal_scores, main = "Distribution of Basal Change", xlab = "Score change")
12 hist(drta_scores, main = "Distribution of DRTA Change", xlab = "Score change")
13
```

The bottom pane shows the console output for the command `t.test(basal_scores, drta_scores)`:

```
> t.test(basal_scores, drta_scores)

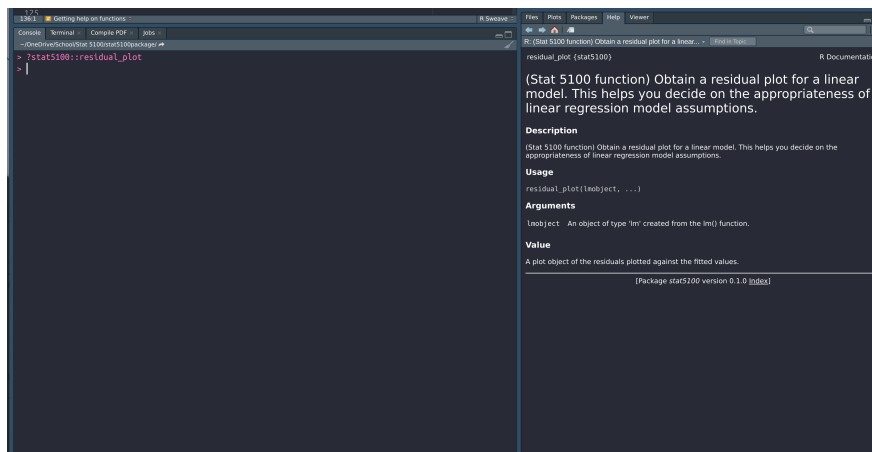
Welch Two Sample t-test

data: basal_scores and drta_scores
t = -4.5682, df = 40.818, p-value = 4.477e-05
alternative hypothesis: true difference in means is not equal to 0
95 percent confidence interval:
 -3.408702 -1.318571
sample estimates:
mean of x mean of y
-1.772723 0.5909091
```

4.2 Getting help on functions

Not even the best R programmers memorize how to use every single function. Very often, you will know that a function exists but you forget the exact parameters to enter and how to use the function. Fortunately, R

has integrated a very easy help page lookup feature. To use this lookup feature, prepend the command you want to learn about with a question mark “?” and enter it into the console. For example, to get help on the `stat5100::residual_plot()` function, type “`?stat5100residual_plot`” into the console.



Every single function that you will ever use in this class will come with a help page that can be accessed this way.

4.3 Reading in data

4.3.1 Reading in Stat 5100 datasets

Nearly all of the datasets that we use in this class (with a few exceptions) are made available right away in the `stat5100` package. To load them into memory, simply use the `data()` function with the name of the dataset in the parenthesis.

```
data(surgical)
```

Note that for this to work, you *must* have the Stat 5100 package loaded already:

```
library(stat5100)
```

4.3.2 Reading in data by CSV

To read in a CSV file, it is a little bit more tricky but still very easy. To load a CSV file, we use the `read.csv()` function in R with the location of the file as a string inside the parenthesis. Note that this function requires we assign the CSV data to an object name.

```
my_data_name <- read.csv(".././../././data_csv/grocery.csv")
```

Note that the file location is specified relevant to your current working directory. To access a file that is “up” a directory, use `..` to indicate this (you see this in the above command!)

To find out where your current working directory is, run the following:

```
getwd()
```

```
## [1] "/home/ethan/OneDrive/School/Stat 5100/teaching/Stat5100/notes/Module 1"
```

To set your working directory, you can use the `setwd()` function, but it is easier to go to Session → Set Working Directory → Choose Directory inside of RStudio.

4.3.3 Creating data by hand

Sometimes, we might need to create a dataframe by hand. Here is an example where we create a very dataframe and display its contents in the console:

```
my_amazing_dataframe <- data.frame(column_name1 = c("Yellow", "Blue", "Red"),
                                   some_numbers = c(5, 7, 89),
                                   more_numbers = c(4, 8, 1),
                                   some_logicals = c(TRUE, TRUE, FALSE))

my_amazing_dataframe
```

	column_name1	some_numbers	more_numbers	some_logicals
## 1	Yellow	5	4	TRUE
## 2	Blue	7	8	TRUE
## 3	Red	89	1	FALSE

4.4 Altering datasets

There are a variety of ways we might want to alter a dataset:

```
# Create a new column
my_amazing_dataframe <- cbind(my_amazing_dataframe,
                              muchos_numeros = my_amazing_dataframe$some_numbers *
                                                my_amazing_dataframe$more_numbers,
                              numbers_exponent = my_amazing_dataframe$some_numbers^5.5)

my_amazing_dataframe
```

	column_name1	some_numbers	more_numbers	some_logicals	muchos_numeros
## 1	Yellow	5	4	TRUE	20
## 2	Blue	7	8	TRUE	56
## 3	Red	89	1	FALSE	89

```
## numbers_exponent
## 1 6.987712e+03
## 2 4.446714e+04
## 3 5.267991e+10

# Set all instances of the number 1 inside the "more_numbers" column to be the
# number e instead.
my_amazing_dataframe$more_numbers[my_amazing_dataframe$more_numbers == 1] <- exp(1)

my_amazing_dataframe
```

	column_name1	some_numbers	more_numbers	some_logicals	muchos_numeros
## 1	Yellow	5	4.000000	TRUE	20
## 2	Blue	7	8.000000	TRUE	56
## 3	Red	89	2.718282	FALSE	89

```
## numbers_exponent
## 1 6.987712e+03
## 2 4.446714e+04
## 3 5.267991e+10
```

4.5 Filtering datasets

To conditionally access parts of a dataframe, you can run any of the following:

```

# print all rows where "some_numbers" was 5
my_amazing_dataframe[my_amazing_dataframe$some_numbers == 5, ]

##   column_name1 some_numbers more_numbers some_logicals muchos_numeros
## 1      Yellow           5           4           TRUE           20
##   numbers_exponent
## 1      6987.712

# Print columns 1 and 3
my_amazing_dataframe[, c(1,3)]

##   column_name1 more_numbers
## 1      Yellow      4.000000
## 2      Blue      8.000000
## 3      Red      2.718282

# Print all elements of some_numbers where some_numbers is either 5 or 7.
my_amazing_dataframe$some_numbers[(my_amazing_dataframe$some_numbers == 5 |
                                     my_amazing_dataframe$some_numbers == 7)]

## [1] 5 7

```

5 Miscellaneous Notes and FAQ

Here are some notes you should be aware of:

- To create a comment in your code (lines in a script that are notes and not actual code), place a `#` right before what you want to say. You probably have seen this in some of the code examples above.

Other random questions:

Q: Where is the best place to go for R help?

A: To be completely honest, the best place to ask questions about R is on Google. Most likely, any question you have has been asked by somebody else. Finding help for your code online is an incredibly important skill: anyone who programs in any capacity should become familiar with this skill eventually.

However, if you aren't finding what you need online or you want some human help, either the TA or the professor is very happy to answer your questions via email or office hours.

Q: Can I view all the R code that is in the Stat 5100 package?

A: Yes! If you are a more complex R user and you wish to see what is happening behind the scenes in some of these functions, then you are welcome to look in the source code of the R package to see what code lies behind these functions. For example, the `visual_assumptions()` function in the code snippet above does not exist natively in R, it is a function that comes from the Stat 5100 R package.

To see the code on Github, navigate to <http://www.github.com/ethanancell/stat5100package>. You are welcome to clone this repository or peek around the source code. All the R code is contained in the "R" folder, and is in either "utilities.R" or "plotting.R." The plotting file should be very self-explanatory: it is where all the plotting functions we'll use in this class live. In utilities.R, we have more numerical methods for different complex calculations we will need for various things we do in this class.

Feel free to clone the repository and peek around. If you have any questions about the package for the course, feel free to direct them to me at ethanancell@aggiemail.usu.edu.

1.3: SAS Crash Course

Dr. Bean - Stat 5100

1 Why SAS?

SAS is a popular statistical software package used by many companies and researchers.

Advantages:

- Lots of output without having to write much code.
- Provides lots of graphical diagnostics automatically.

Disadvantages:

- Expensive (if you want a desktop version).
- Hard to customize output (particularly visualizations).

Teaching SAS in this class:

- (For non-math/stat majors): Has the smallest learning curve to create basic linear models.
- (For math/stat majors): Provides you exposure to multiple programming languages as several of our upper division courses use R.

In this class, we will focus on using SAS Studio, which is a free online version of SAS.

2 SAS Studio Online

- Navigate to https://odamid.oda.sas.com/SASLogon/login?service=https%3A%2F%2Fodamid.oda.sas.com%2FSASODAControlCenter%2Fj_spring_cas_security_check.
- Select the “Not registered or cannot sign in?” option.
- Follow directions for creating a SAS profile.
- Note that the email confirming your profile may take some time (5-10 minutes) to receive.
- Note also that the only tool we will use in this class is “SAS Studio”.

3 Getting Started

Once you have an account, your SAS studio window will look something like this.

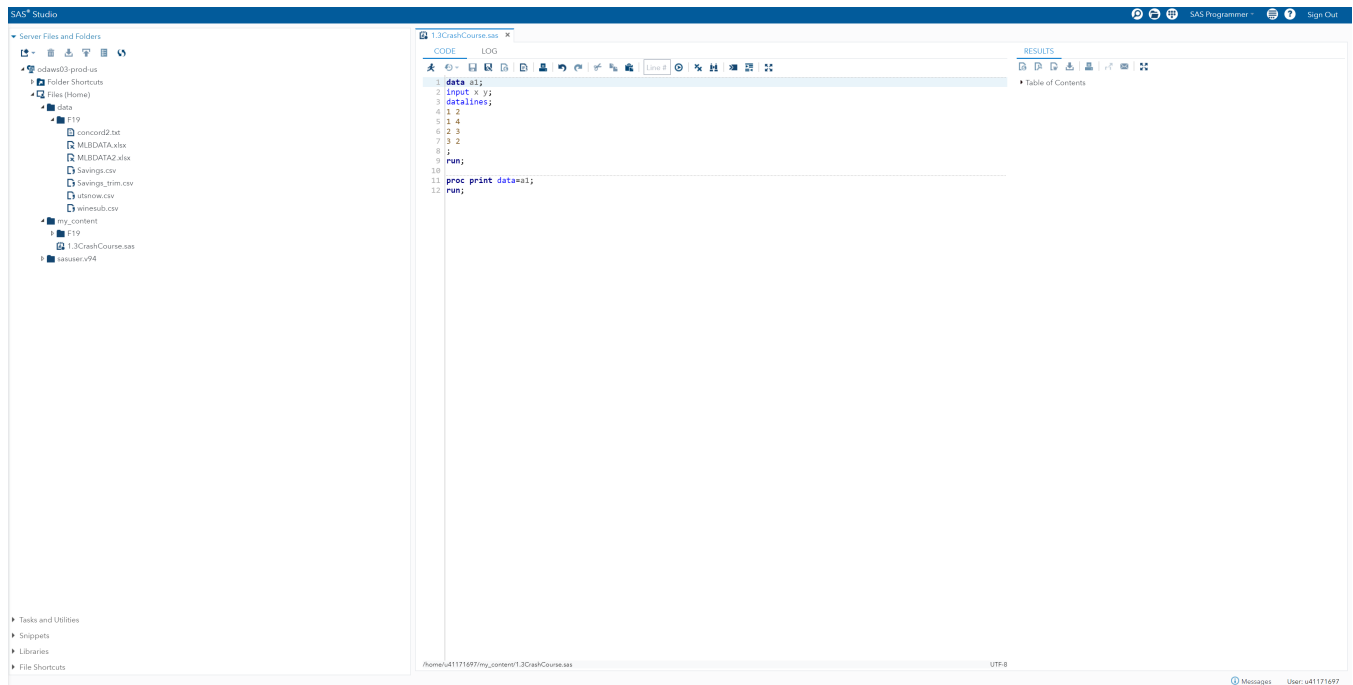


Figure 1: Sample SAS studio interface.

Main windows:

- CODE (or Editor): The window where you type in SAS commands. Font color matters in this window.
- LOG: tells what was “done”. Red notes indicate errors. This is the first thing you should check if your results are unexpected.
- RESULTS (or Results Viewer): displays results in HTML format. You can export the HTML output as a pdf or a rft (rich text file - Microsoft word) to obtain nice looking output that you can copy into your reports. Or, you can use the snipping tool to save relevant output and import them as figures in LaTeX (more on LaTeX in 1.4).

You run SAS code by clicking on the “Running Man”.



Necessary Components of a SAS Program:

- a semi-colon at the end of every statement
- a data statement that either creates or imports a dataset
- at least one space between each word or statement
- (almost always) a **procedure** that performs some type of analysis with your data
- a run statement

Data

There are three ways to read data into SAS. The first way is the easiest, but also not practical for large datasets.

Read in data “by hand”

```
/* Text between the asterisks are considered comments in SAS */

data a1; /* Create a new dataset named a1 */
input x y; /* List the variable names of the data that will be included in a1 */
datalines; /* (or 'cards') tells SAS to start reading in data */
1 2
1 4
2 3
3 2
;
run; /* Tells SAS to execute the above code */

proc print data = a1; /* Print the dataset to the output screen */
run;
```

Obs	x	y
1	1	2
2	1	4
3	2	3
4	3	2

Read in data from a file.

Assume that the same data as before our located in the SAS studio folder:

`/home/u41171697/data/mydata.txt`

Note that SAS Studio will not recognize file paths on your actual computer. All data that you wish to read into SAS must be uploaded to SAS studio first.

We could read the data directly from the file using:

```
data a1;
infile '/home/u41171697/data/mydata.txt'; /* Specify path to file */
input x y;
run;
```

Read in data from a file using a procedure.

Now suppose that the data are in the excel file mydata.xlsx with the variable names included on the first row.

```
proc import
datafile =  '/home/u41171697/data/mydata.txt'
dbms=xlsx /* Specify the file type (what separates the variables) */
out=work.a1 /* Specify the name of the dataset */
replace; /* Overwrite any datasets in the directory with the same name */
run;
```

Altering Datasets

We can add or change variables in a dataset using the commands:

```
data a2;
set a1; /* Create a copy of the dataset a1 */
  xy = x*y; /* Multiplication */
  xsq = x**2; /* Exponentiation */
  xeq1 = 0;
  if x = 1 then xeq1=1; /* Conditional Assignment */
run;

proc print data = a2;
var x y xy xeq1; /* Print all variables to the screen except xsq */
run;
```

Obs	x	y	xy	xeq1
1	1	2	2	1
2	1	4	4	1
3	2	3	6	0
4	3	2	6	0

Filtering Datasets

We can subset datasets using conditional statements.

```
data a3; set a2;
  if y < 3.5;
  /* Default 'then' is keep */
run;
/* Same as: */
data a3; set a2;
  if y >= 3.5 then delete;
run;
proc print data=a3;
var x y xeq1;
```

```

title1 'Smaller Set';
run;

```

Smaller Set			
Obs	x	y	xeq1
1	1	2	1
2	2	3	0
3	3	2	0

Procedures

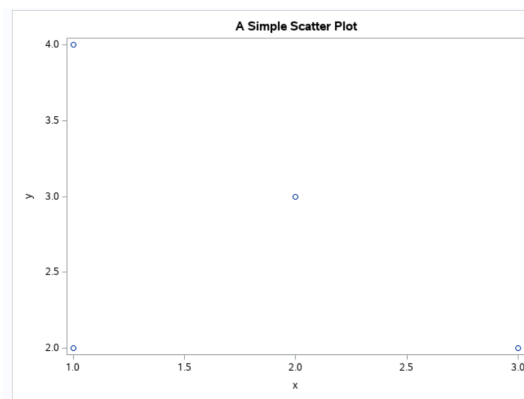
We prepare data in a SAS-friendly format using DATA steps. We analyze data using SAS procedures (PROC). Some PROCs that we will commonly use in this class include:

- Fitting models: PROC REG, PROC LOGISTIC, PROC ARIMA, and more
- Graphical Checks: PROC SGPLOT, PROC SGSCATTER, PROC BOXPLOT, PROC UNIVARIATE

```

proc sgplot data=a2;
  scatter x=x y=y ;
  title1 'A Simple Scatter Plot';
run;

```



```

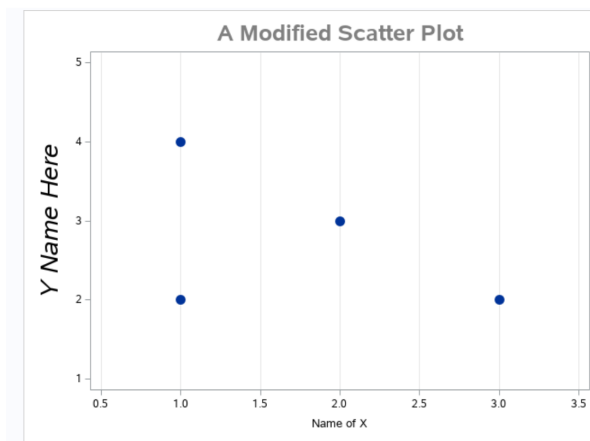
proc sgplot data=a2;
  scatter x=x y=y /
    markerattrs=(symbol='CIRCLEFILLED'
                  size=12);
  xaxis min=.5 max=3.5 grid
    label='Name of X';
  yaxis values=(1 to 5 by 1)
    label='Y Name Here'
    labelattrs=(size=20 style=ITALIC);

```

```

title1 height=2 color=grey
'A Modified Scatter Plot';
run;

```



```

/* @@ reads symbols in variable order and ignores new lines
$ indicates that variable z is a character and not numeric
. - indicates missing values */
data a1; input x y z $ @@; cards;
  1 2 alpha  1 4 .
  2 3 gamma  3 . delta
;
run;
proc means data=a1;
  var y;
  title1 "Means Output";
run;
proc print data=a1;
  var y x z;
  title2 "Subtitle";
run;

```

Means Output

The MEANS Procedure

Analysis Variable : y				
N	Mean	Std Dev	Minimum	Maximum
3	3.0000000	1.0000000	2.0000000	4.0000000

Means Output Subtitle

Obs	y	x	z
1	2	1	alpha
2	4	1	
3	3	2	gamma
4	.	3	delta

4 Miscellaneous Notes

- Missing semicolons are perhaps the most common bug in SAS code.
- The best way to get started with SAS programming is to look at the course example code, then figure out how to modify that code for your particular problem.
- Export output by copying from Results Viewer (sometimes helps to use the “Download results as RTF file”) and pasting into a word document, or saving the images and importing them into a LaTeX document.
- Help in SAS: The question mark symbol in the upper right hand corner of the SAS studio editor is a good place to look for function documentation. However, reading SAS documentation can be difficult if you aren’t already familiar with the procedure.
- SAS code can be written across lines. Line breaks are managed with semicolons. Data can be read in continuously with “@@”.
- Missing Values: SAS procedures will completely ignore an observation if one of the variables is missing; to code a value as “missing”, use the period (.) character.
- “Strings”: Read in character variables with \$ after the name in the input line.
- Comment Lines: To comment out a line up to the next semi-colon, put an asterisk (*) before it. To comment out an entire section, start with /* and end with */
- Selective Output: SAS will usually give you more output than you want or need, so you will need to know what you want in order to do anything useful with the output. **It is not appropriate to include ALL SAS output on homework assignments and project papers.**
- **Save Code:** SAS studio will not warn you about unsaved code when you try and close your browser. **Save your code** often to avoid frustrating loss of code.
- This class requires SAS version 9.3 or later. This is automatic for anyone using the online version of SAS studio.

5 How to “win” at SAS

- The best way to learn SAS is to **use it**. Start early on Homework 1 and 2 to start getting experience (and avoid trying to finish Homework 2 last minute, which never works for students).
- Take time to *understand* SAS code before you use it. This will make it much easier to modify the code on homework and projects.

1.4: Data Exploration

Dr. Bean - Stat 5100

1 Why Data Exploration

Data Modeling is a lot like:



In order to avoid disaster, you need to **look** before you **jump**.

Example: Consider four scenarios where we use to create a model that uses values of x to predict values of y . We make the assumption in each case that the data can be modeled as

$$Y_i = \beta_0 + \beta_1 X_{i,1} + \epsilon_i \quad (1)$$

This assumption means that we assume that X and Y share a linear relationship. That is, as X increases, Y will increase proportionally. We will explore this further in Handout 2.1.

Data Explorations BEFORE modeling will help us to detect:

- Skewed distributions
- Outlier points
- Non-linear trends

Often, we can use **variable transformations** to get data that are normal, or at least symmetric, in distribution.

Why symmetric data? Consider the “door hinge” problem.

2 Common Exploratory Plots

- **Boxplots::** Show the five quartiles of the data (min, 25th percentile, median, 75th percentile, and maximum).
 - Values that are farther than $1.5 \times \text{IQR}$ (Interquartile Range, which is the 75th percentile minus the 25th percentile) above the 75th percentile or below the 25th percentile are typically plotted as “outlier” points.
 - Great way to quickly summarize the range of values.

```
proc sgplot data=concord1;  
vbox Water81;  
run;
```

Add option “/ datalabel =” to identify potential outlier points.

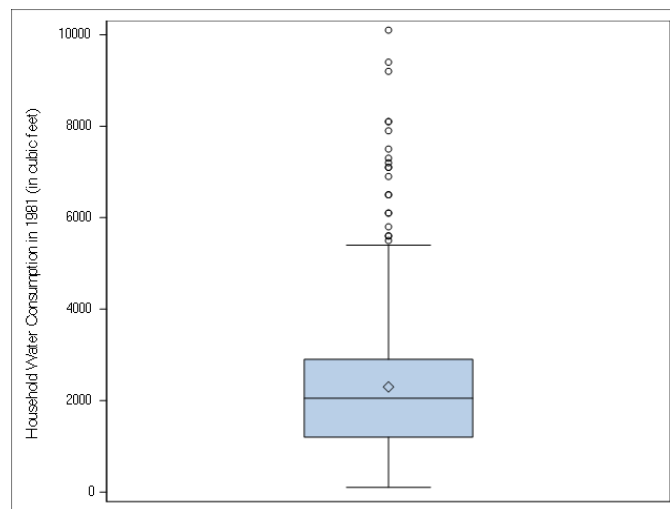


Figure 1: Sample boxplot.

- **Histograms:** Use bins to show the number of observations in a range.
 - Help us to visualize the distribution of the data by imagining a smooth curve running along the top of the bins.
 - Word of caution: the choice of bin width can drastically change the shape of a histogram.

```
PROC UNIVARIATE DATA = concord1 noprint;  
HISTOGRAM Water81;  
run;
```

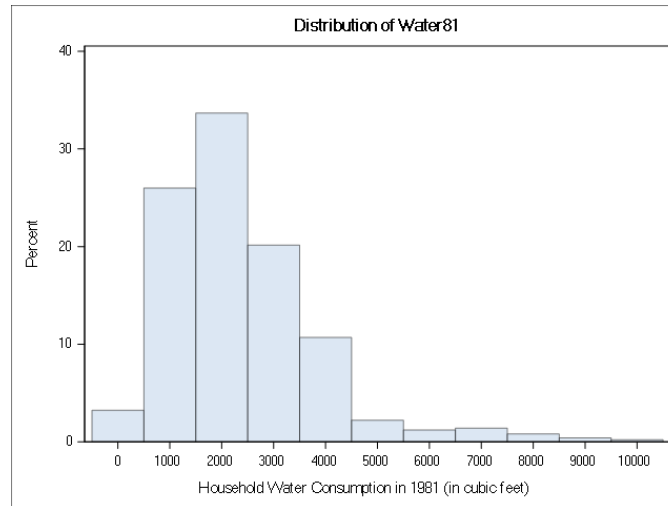


Figure 2: Sample histogram.

- **QQ Plot:** “Quantile Comparison” plots help to easily compare the observed distribution of points to a theoretical (typically normal) distribution.
 - Plots the data quantiles against the theoretical quantiles of similar observations that are normal in distribution.
 - Points that closely follow the diagonal line indicate that the observed data follow the theoretical distribution.
 - While they don’t help to visualize shape, qqplots are superior to histograms as a visual check for normality.

```
PROC UNIVARIATE DATA = concord1 noprint;
qqplot Water81 / NORMAL(mu=est sigma=est);
run;
```

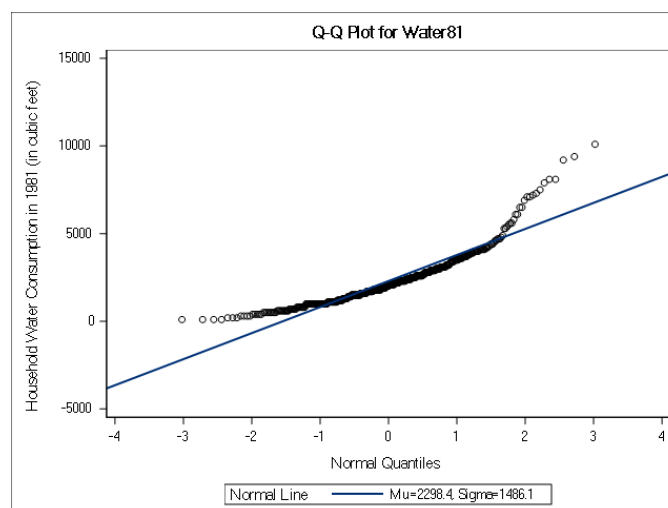


Figure 3: Sample quantile comparison plot for a normal probability distribution.

- **Scatterplots:** Plots paired observations from two variables as points on a two-dimensional plot.

- Excellent way to determine if two variables share a relationship.
- Can combine in a **scatterplot matrix** when looking at relationships between more than two variables.
- Subject to **overplotting** when you have thousands of observations that you are trying to plot at the same time.

```
proc sgscatter data=concord1;
matrix Water81 Water80 Water79;
run;
```

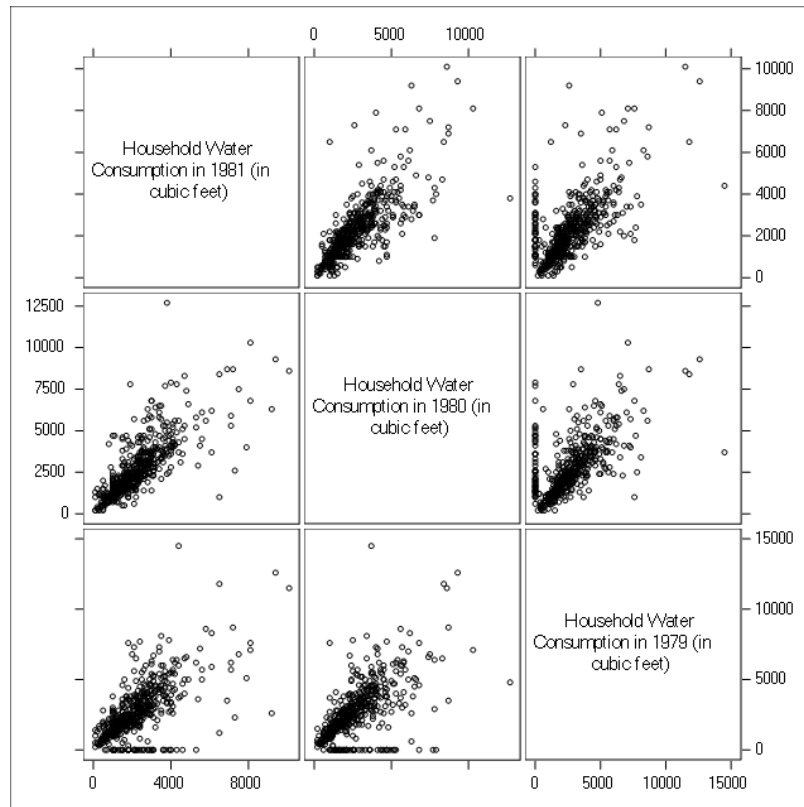


Figure 4: Sample scatterplot matrix.

See **Handout 1.4.2** for an extended example in SAS of data explorations.

1.4: Data Exploration

Dr. Bean - Stat 5100

1 Why Data Exploration

Data Modeling is a lot like:



In order to avoid disaster, you need to **look** before you **jump**.

Example: Consider four scenarios where we use to create a model that uses values of x to predict values of y . We make the assumption in each case that the data can be modeled as

$$Y_i = \beta_0 + \beta_1 X_{i,1} + \epsilon_i \quad (1)$$

This assumption means that we assume that X and Y share a linear relationship. That is, as X increases, Y will increase proportionally. We will explore this further in Handout 2.1.

Data Explorations BEFORE modeling will help us to detect:

- Skewed distributions
- Outlier points
- Non-linear trends

Often, we can use **variable transformations** to get data that are normal, or at least symmetric, in distribution.

Why symmetric data? Consider the “door hinge” problem.

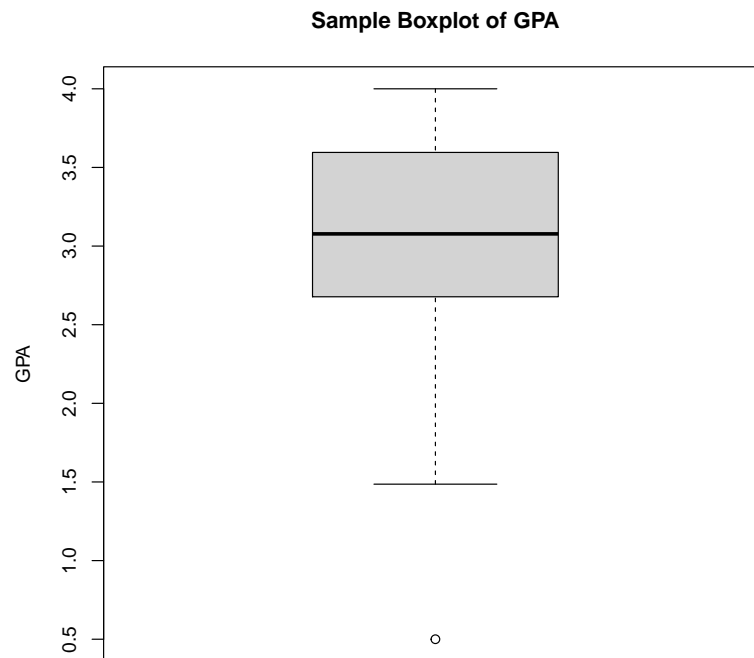
Common Exploratory Plots

- **Boxplots::** Show the five quartiles of the data (min, 25th percentile, median, 75th percentile, and maximum).

- Values that are farther than $1.5 \times \text{IQR}$ (Interquartile Range, which is the 75th percentile minus the 25th percentile) above the 75th percentile or below the 25th percentile are typically plotted as “outlier” points.
- Great way to quickly summarize the range of values.

```
library(stat5100)
data(college)

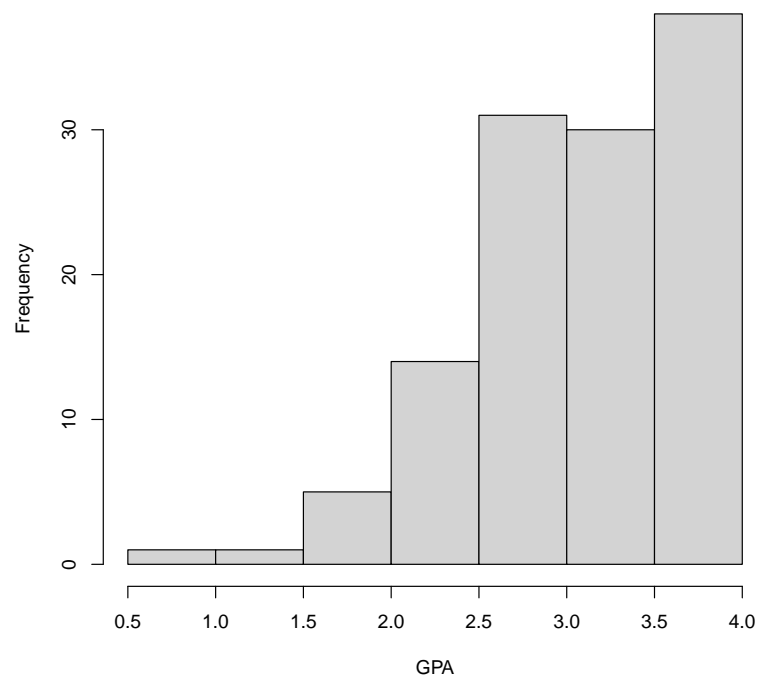
boxplot(college$gpa, main = "Sample Boxplot of GPA", ylab = "GPA")
```



- **Histograms:** Use bins to show the number of observations in a range.
 - Help us to visualize the distribution of the data by imagining a smooth curve running along the top of the bins.
 - Word of caution: the choice of bin width can drastically change the shape of a histogram.

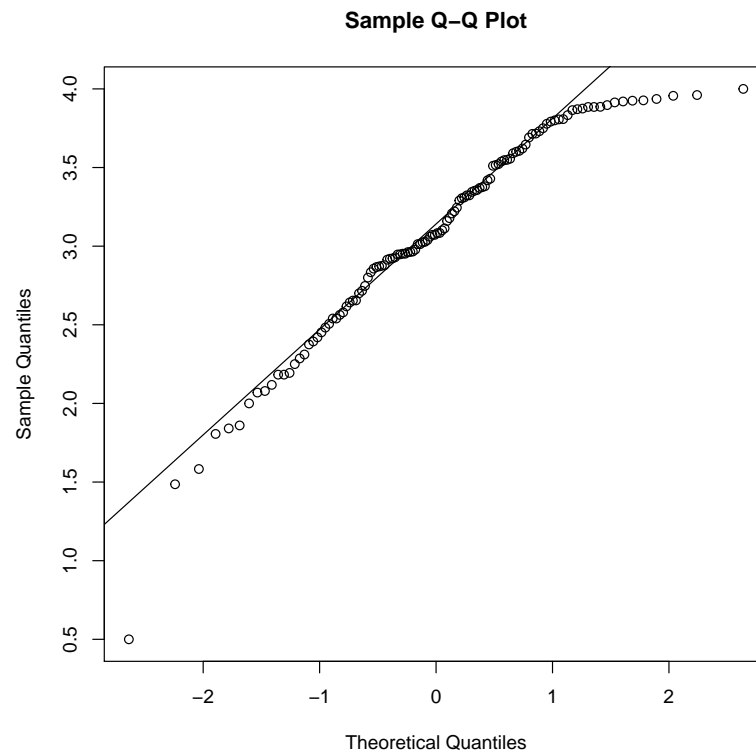
```
hist(college$gpa, main = "Sample Histogram of GPA", xlab = "GPA")
```

Sample Histogram of GPA



- **QQ Plot:** “Quantile Comparison” plots help to easily compare the observed distribution of points to a theoretical (typically normal) distribution.
 - Plots the data quantiles against the theoretical quantiles of similar observations that are normal in distribution.
 - Points that closely follow the diagonal line indicate that the observed data follow the theoretical distribution.
 - While they don’t help to visualize shape, qqplots are superior to histograms as a visual check for normality.

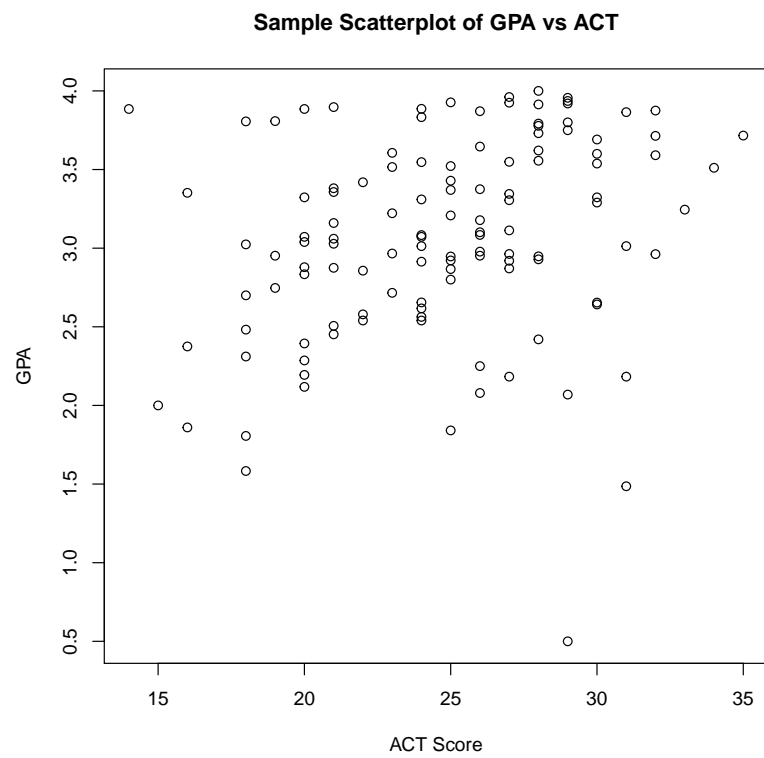
```
qqnorm(college$gpa, main = "Sample Q-Q Plot")  
qqline(college$gpa)
```



(The above shows that GPA is clearly not normally distributed. If they were, then the data points would lie on the line very nicely.)

- **Scatterplots:** Plots paired observations from two variables as points on a two-dimensional plot.
 - Excellent way to determine if two variables share a relationship.
 - Can combine in a **scatterplot matrix** when looking at relationships between more than two variables.
 - Subject to **overplotting** when you have thousands of observations that you are trying to plot at the same time.

```
plot(college$act, college$gpa, main = "Sample Scatterplot of GPA vs ACT",  
     xlab = "ACT Score", ylab = "GPA")
```



See **Handout 1.4.2** for an extended example in R of data explorations.

1.5: Introduction to Statistical Writing

Dr. Bean - Stat 5100

1 Writing Fellows

There will be three writing-intensive assignments this semester. Two will be completed individually and one will be completed in groups. For each one of these assignments, you will be required to make two submissions of your paper. The first will be a rough draft submission that must be complete in the content, but might need some refinements in the presentation of the content. The second will be a final, polished version of the papers. Between the two submissions, each of you will be **required** to meet with one of our writing fellows. The contact information for the writing fellows for this class is found on the “Syllabus” tab of the course canvas page.

Important points on writing fellows:

- You are **required** to meet with a writing fellow as part of your grade for each paper you write. However, the writing fellows do not grade any aspect of your paper.
 - Note that points will be deducted if you are late to your appointment with your writing fellow.
- Writing fellows are equipped to help you improve the clarity and the formatting of your writing. They will not comment on the technical content of your paper.
- You do not have to take every suggestion the writing fellow gives you, but it is in your best interest to incorporate most of their suggestions.

2 Why so much writing?

For the undergraduate students:

- The university requires that you take a communications intensive course beyond ENGL 2010.
- The university has determined that this course contains enough opportunities to communicate (through writing and presentations) to qualify as such a course.
 - If this course was not a CI course, some of our undergraduate statistics majors would fall short of the requirements of graduation.

For the graduates:

- Most, if not all, of you will be required to communicate your research, often in the form of a thesis, dissertation, or journal article.
- The ability to effectively communicate your efforts in these venues is absolutely vital to the success of your research.
- Despite this importance, we offer very little training in how to write effectively.

For all:

- **The ability to effectively communicate quantitative information will give you a competitive advantage in graduate school or in the workforce.**

3 General Writing Tips

Use concise, straightforward language.

- Intelligent writing doesn't require using fancy words.
- Concise writing will help retain interest in what you are saying.

(start)

We first employed forward variable selection on our model. This told us that we should remove the variables X_2 and X_7 . We acknowledge that the aforementioned method is suboptimal for use in variable selection. We then tried backwards variable selection, which told us to remove X_1 , X_2 , X_4 , and X_{11} . Finally, we tried the stepwise variable selection approach which told us to remove X_2 and X_{11} . Because the backwards and stepwise regression both suggested the removal of X_2 and X_{11} , we decided that we would eliminate these variables from our model.

(improved)

We tried several variable selection techniques, including forward, backwards, and stepwise selection. Each method suggested we remove different variables, but backwards and stepwise selection both recommended the removal of X_2 and X_{11} . This agreement across selection methods prompted us to remove these variables in our final model.

Use active voice whenever possible

- Active voice is more concise and gives you ownership over your results.
- Personal pronouns are OK, but use “we” instead of “I”, even if you are the only author.

(start)

It was determined that the variable X_2 should be removed from the model.

(better)

We removed the variable X_2 from our model.

Make sure you provide meaning to the numerical results in the introduction and conclusion of your paper.

- Your ultimate goal is to persuade people that there is valuable information contained in the data.
- Simply presenting a table of results fails to persuade people as to why the results are important.
- Providing a “why” in your writing makes readers more likely to pay attention to your analysis.

4 Using LaTeX

- LaTeX is a markup language intended for scientific writing.
- It is particularly good for including **references** and **mathematical equations**.

Writing equations:

% Add a comment to your document (ignored when compiling the document).
\$... \$ Add an equation to the current line of text.
\$\$... \$\$ Put an equation on its own line.
\[... \] Put an equation on its own line.

Referencing Equations

You can also number and label equations to include them in the text.

```
\begin{equation}
E = mc^2
\label{eq1}
\end{equation}
\begin{equation}
Y = \beta_0 X_{i,1} + \epsilon
\label{regression}
\end{equation}
Reference Equation \ref{eq1} and \ref{regression} in the text.
```

$$E = mc^2 \tag{1}$$

$$Y = \beta_0 X_{i,1} + \epsilon \tag{2}$$

Reference Equation 1 and 2 in the text.

The same goes for referencing figures and tables.

```
\begin{figure}[H] % H command requires 'float' package
\centering
\includegraphics[width = 0.25\textwidth]{../figures/module1/usu.png}
\caption{This is the Utah State Logo}
\label{fig1}
\end{figure}
```

The USU logo is included in Figure \ref{fig1}.
\end{figure}



Figure 1: This is the Utah State Logo

The USU logo is included in Figure 1.

The equation environment includes commands for all the greek letters as well. Check out:

https://www.overleaf.com/learn/latex/List_of_Greek_letters_and_math_symbols

Document Headers

LaTeX Documents include document headers that allow you to customize the overall format of the final document. These headers can seem like a pain at first but they are extremely valuable in helping you quickly change the format of your paper depending on the context. A typical document header looks something like this:

```
\documentclass[11pt]{article}

% Add necessary packages
\usepackage{amsmath}

% Begin Document
\begin{document}

%% ADD ALL PAPER CONTENT HERE

\end{document}
```

Getting Started with LaTeX

There are several free document editors that allow you to start using LaTeX, most notably:

- Windows: MiKTeX (<https://miktex.org/download>)
- Mac: MacTeX (<http://www.tug.org/mactex/>)
- Linux: TeXLive (<http://www.tug.org/texlive/>)

You can also get started with an online LaTeX editor called Overleaf. (<https://www.overleaf.com/>)

5 Using Microsoft Word

Microsoft Word also has an equation editor that can be quickly accessed using the “alt + equal” keystrokes. Many latex commands, particularly those for greek variables, subscripts, and superscripts, are recognized in Microsoft’s equation editor.

Remember: All equations included in your homework or papers must use a professional equation editor environment. Formatting points will be taken away for equations written outside of such an environment such as $Y = b_0 + b_1x$. vs $Y = b_0 + b_1x$.