R Companion to *Real Econometrics*

Tony Carilli

2019-09-23

Contents

Chapter 1

Prerequisites

The intended audience for this book is anyone make using of *Real Econometrics: The Right Tools to Answer Important Questions* 2nd ed. by Michael Bailey who would like to learn the R code necessary to complete the end of chapter exercises. We really heavily on the `tidyverse` a collection of packages that shares an underlying design philosophy, grammar, and data structures. We also make use of a variety of packages (bundles of code) where it will make coding more straightfoward in terms of writing, understanding, and editing.

Chapter 2

Introduction

In this chapter I will offer description of R, RStudio, and R Markdown. These are the software/progams you will need throughout the manual.

2.1 Install R and RStudio

R is an open source statistical package that is free and platform independent. R can be downloaded for any platform The Comprehensive R Archive Network (CRAN). After clicking the link, choose the appropriate (Linux, (Mac) OS X, or Windows) installation file for R. Download and install R on your computer. I strongly recommend the use of RStudio as opposed to the R GuI to do your work in R. RStudio is an integrated development environment (IDE) for R. It includes a console, syntax-highlighting editor that supports code execution, as well as tools for plotting, history, debugging, workspace management, and report writing. Like R, RStudio is open source and free to download and use. Following the link RStudio Desktop. Choose the appropriate version for your environment. In addition, since you will need to write reports, R Markdown gives you the ability to integrate documents with your code to produce outputs in a variety of formats.

Soren L. Kristiansen has written comprehensive guides to installing all of the above files for both the macOS and Windows. Please follow the appropriate link to ensure you have installed everything properly on your machine. Follow the steps exactly and you will have no issues. Don't follow them at your own peril.

In addition these guides will walk you through the creation of your first R Markdown file. You are expected to use R Markdown for your work. This guide was written using R Markdown.

RStudio also includes a code editor which allows you to maintain a file of your 'scripts' as you complete your code. This script file also allows for relatively easy editing and debugging of your code as you write it.

2.2 Using RStudio

RStudio contains 4 panes that make elements of R easier to work with than they would be working in R GUI.

2.2.1 Source Pane

The Source Pane in the upper left contains your code and can be accessed with the keyboard shortcut CTR+1. This pain includes any R Markdown files, R Notebook files, and R Script files that you may have opened. It will also contain any data that you `view` or information about attributes of an object when clicked in the environment pane.

This is the pane in which you would create or open a script file. Create a new script file by clicking the icon with the green "+" and clicking R Script, by clicking `File > New File > R Script`, or with `Ctrl+Shift+n` or `cmd+Shift+n`.

2.2.2 Environment/History Pane

You'll find the Environment/History pane in the top right. The Environment tab shows you the names of all the data objects you have defined in the current R session. You can directly access the environment with `ctrl+8`. This tab is `objects()` command mentioned in the text on steroids. By clicking on the triangle icon next the object name you receive the same information as calling `str()` on the object. Clicking on the grid icon the right of the name will call `view()` and the object will be displayed in the Source Pane. `view` will display all of the data in a data frame. `head` displays the first six observations. The History tab contains all the code you that you've run. You can directly access the history tab with `ctrl+4`. If you'd like to re-use a line from your history, click the To Console icon to send the command to the Console Pane, or the To Source icon to send the command to the Source Pane.

2.2.3 Console Pane

The Console Pane in the bottom left is essentially what you would see if you were using the R GUI without R Studio. It is where the code is evaluated. You can type code directly into the console and get an immediate response. You can access the Console Pane directly with `ctrl+2`. With your cursor in the Console you can access any previous code with ctrl+up and use the arrow keys to pick the line you'd like to use. Using the up arrow will show the lines of code one at a time from the last line ran to the first line available in the R session. If you type the first letter of the command followed by ctrl+up you will get all of the commands that you have used that begin with that letter. Highlight the command and press return to place the command at the prompt.

2.2.4 Files/Plots/Packages/Help Pane

The last pane is on the bottom right. The files tab (`ctrl+5`) will show you the files in the current working directory. The plots tab (`ctrl+6`) will contain any plots that you have generated with the base R plotting commands. The packages tab (`ctrl+7`) will show you all of the packages you have installed with checks next to the ones you have loaded. Packages are collections of commands that perform specific tasks and are one of the great benefits of being part of the R community. Finally, the help tab (`ctrl+3`) will allow you get help on any command in R, similarly to `?commandname`. Initially understanding R help files can be difficult; follow the link A little more about R from Kieran Healy's *Data Visualization* for a good introduction. In addition `args(commandname)` displays the argument names and corresponding default values (if any) for any command.

Double clicking a `csv` file in the Files tab will open the data in the Source Pane.

Find a nice overview of R Studio in YaRrr! The Pirate's Guide to R

2.3 R Markdown

You will complete your homework, reports, etc. using R Markdown gives you the ability to integrate documents with your code to produce outputs in a variety of formats.

R Markdown allows you to combine R code with your report for seamless integration. R code can be included in a markdown file as Code Chunks or directly within the text.

To create an new R Markdown file inside of RStudio by clicking `File > New File > R Markdown...` A dialog box will appear choose a title that is descriptive of the work you are doing, click `OK`. This will create a default R Markdown. The first thing it creates is yaml header. The header includes the title, author, date, and default output file type. You will want to retain this. It will also generate R code chunk with knitr[1] options. You will want to retain this R chunk. You need not retain any of the remaining parts of the file generated.

[1] knitr is an engine for dynamic report generating within R.

Markdown is a simple formatting syntax for authoring HTML, PDF, and MS Word documents. For more details on using R Markdown see http://rmarkdown.rstudio.com.

When you click the **Knit** button a document will be generated that includes both content as well as the output of any embedded R code chunks within the document.

Scroll down to the **Overview** at https://github.com/rstudio/rmarkdown for more on markdown. I strongly suggest you work through the Markdown formatting tutorial for an introduction to basic formatting in Markdown.

For more help getting started in R Markdown, please see the R Markdown website.

Chapter 3

The Quest for Causality

3.1 Introduction

In order to familiarize you with the R code necessary to complete the assignments in *Real Econometrics*, I will reproduce all examples from Chapter 1. As I present the examples, I will explain the syntax for each pieced of code. You will also be introduced to using R Markdown to produce a seamless integration of your code, your output, and your reports.

In subsequent chapters, I will take you through examples of the relevant code necessary to complete the exercises in R.

3.1.1 Table 1.1

Table 1.1 contains the necessary information to produce Figures 1.2 and 1.3. Creating Table 1.1 will give you an opportunity to create a data frame from four vectors. A data frame is used for storing data tables. A data frame is one of the many data structures in R. The others include vector, list, matrix, factors, and tables. A data frame is collection of vectors of the same length.

A vector is the most common and basic data structure in R. Vectors can be of two types: atomic vectors or lists. An atomic vector is a collection of observations of a single variable. The vectors in a data frame can be different types, however. Each vector must be of a single type. The atomic vector types or classes in R are logical, integer, numeric (real or decimal), complex, and character. A logical vector is one in which all of the values are TRUE, FALSE, and NA. An integer vector contains only integers, a real vector contains only reals, etc. If a vector contains more than one type of value, the vector and each element of it is coerced to the most general class in the vector.

Let's start by creating each vector in Table 1.1. To assign values to a vector, use the assignment operator `<-` and the concatenate or combine function `c()`.

```
observation_number <- c(1:13) # The colon tells R to create a sequence from 1 to 13 by
name <- c("Homer", "Marge", "Lisa", "Bart", "Comic Book Guy", "Mr. Burns",
          "Smithers", "Chief Wiggum", "Principle Skinner", "Rev. Lovejoy",
          "Ned Flanders", "Patty", "Selma")  # Each "string" is enclosed in quotes.  T
donuts_per_week <- c(14, 0, 0, 5, 20, 0.75, 0.25, 16, 3, 2, 0.8, 5, 4) # This is a num
weight <- c(275, 141, 70, 75, 310, 80, 160, 263, 205, 185, 170, 155, 145) # This is a
```

Note in the code chunk above that the symbol, #, is used to create comments within the code. Those things set off by the # will not be executed as code. These are useful for creating notes to yourself or collaborators about what you are doing with certain lines of code.

We now have four named vectors that we can put into a data frame. A note on naming conventions in R. While there are many name conventions in R, I recommend using snake case where each word is separated by an under score and no capital letters are used. See Hadley Wickhams Style Guide for style suggestions for all parts of R programming, not just variable names. Following these guidelines will make your code easier to read and edit.

```
library(tidyverse) # load the tidyverse package
donuts <- tibble(observation_number, name, donuts_per_week, weight) # create the donut
save(donuts, file = "donuts.RData")
```

A tibble is an update to the traditional data frame. For most of what we will do, it will act the same as a data frame. The two main differences in data frames and tibbles are printing and subsetting. For more on tibbles type `vignette("tibble")` in the console.

`tidyverse` is one of the many packages developed within the R community. In R, a package is shareable code that bundles together code, data, documentation, tests, etc. To use a package, it must first be installed and then be loaded. To install a package, call `install.packages("package_name")`[1]. To make use of a package, load it by calling `library(packagename)`[2]. Currently there are more than 14,000 packages available, to see the packages visit Contributed Packages. CRAN Task Views shows relevant packages by task. You may want to visit CRAN Task View: Econometrics to see the extensive array of packages for use in econometrics.

The `tidyverse` package is a collection of packages that share an underlying design philosophy, grammar, and data structures. For more on the tidyverse follow this link. The `dplyr` package loaded below is a grammar of data manipulation that can be used to solve most data manipulation problems.

[1] You need install a package only once

[2] You must load the package during each R session to make use of it.

```r
# Print the tibble to the console by typing its name.
donuts
```

```
# A tibble: 13 x 4
   observation_number name               donuts_per_week weight
                <int> <chr>                        <dbl>  <dbl>
 1                  1 Homer                        14       275
 2                  2 Marge                         0       141
 3                  3 Lisa                          0        70
 4                  4 Bart                          5        75
 5                  5 Comic Book Guy               20       310
 6                  6 Mr. Burns                     0.75     80
 7                  7 Smithers                      0.25    160
 8                  8 Chief Wiggum                 16       263
 9                  9 Principle Skinner             3       205
10                 10 Rev. Lovejoy                  2       185
11                 11 Ned Flanders                  0.8     170
12                 12 Patty                         5       155
13                 13 Selma                         4       145
```

```r
# glimpse will provide information about the data frame, its observations, variables, and their c
library(dplyr)
glimpse(donuts)
```

```
Observations: 13
Variables: 4
$ observation_number <int> 1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13
$ name               <chr> "Homer", "Marge", "Lisa", "Bart", "Comic Bo...
$ donuts_per_week    <dbl> 14.00, 0.00, 0.00, 5.00, 20.00, 0.75, 0.25,...
$ weight             <dbl> 275, 141, 70, 75, 310, 80, 160, 263, 205, 1...
```

Use the `kable` function in `knitr` to create Table 1.1.

```r
knitr::kable(donuts,
             caption = 'Table 1.1 Donut Consumption and Weight',
             col.names = c("Observation</br> number", # </br> is html code to insert a line break
                           "Name", "Donuts</br> per week",
                           "Weight</br> (pounds)"),
             escape = F,  # necessary to force the line breaks
             align = 'cccc') # request that the four columns be centered
```

Table 3.1: Table 1.1 Donut Consumption and Weight

Observation</br> number	Name	Donuts</br> per week	Weight</br> (pounds)
1	Homer	14.00	275
2	Marge	0.00	141
3	Lisa	0.00	70
4	Bart	5.00	75
5	Comic Book Guy	20.00	310
6	Mr. Burns	0.75	80
7	Smithers	0.25	160
8	Chief Wiggum	16.00	263
9	Principle Skinner	3.00	205
10	Rev. Lovejoy	2.00	185
11	Ned Flanders	0.80	170
12	Patty	5.00	155
13	Selma	4.00	145

3.1.2 Figure 1.2

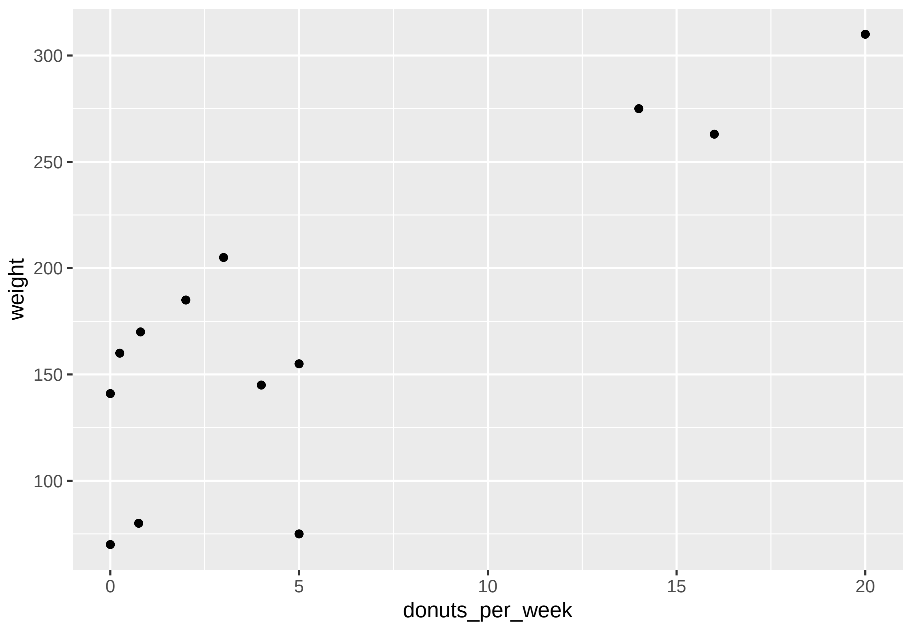
To create Figure 1.2 we will use the ggplot2 package. `ggplot2`, also part of the `tidyverse`, is a system for declarative creating graphics, based on The Grammar of Graphics. The Grammar of Graphics is built on two principles. First, graphics are built with distinct layers of grammatical elements. Second, meaningful plots are formed through aesthetic mappings.

Seven elements comprise the grammar of graphics: data, aesthetics, geometries, facets, statistics, coordinates, and themes. Every graphic must contain, at a minimum, data, aesthetics, and geometries. Data, typically a data frame or tibble, is the data set being plotted. Aesthetics are the scales onto which data are mapped. Aesthetics include x-axis, y-axis, color, fill, size, labels, alpha (transparency), shape, line width, and line type. Geometries are how we want the data plotted, *e.g.*, as points, lines, bars, histograms, boxplots, etc. Facets allow us to use more than one plot, statistics allow us to add elements like error bands, regression lines, etc. Coordinates allow us to control the space into which we plot the data. Finally, themes are all non-data ink in a graphic.

Follow this link for an overview of ggplot2. The Learning ggplot2 section points to three useful places to learn more about using ggplot2. While the use of data visualization is not emphasized in the econometrics, understanding the basic principles will help your data analysis.

```
# Load the ggplot2 library
library(ggplot2)
# Create an object p which includes the data and aesthetic mapping
p <- ggplot(data = donuts, mapping = aes(x = donuts_per_week, y = weight))
```

```r
# Add the geometry that creates the scatter plot
(p1 <- p + geom_point()) # putting parenthesis around the line of code force the output to the sc
```




```r
# the parentheses surrounding the function call cause the output to be printed.
```

This basic plot can be transformed into the figure in the text by adding layers to the graphic to change its appearance.

```r
# Change the axis labels and add a caption
(p2 <- p1 + labs(x = "Donuts", y = "Weight (in pounds)", caption = "Figure 1.2: Weight and Donuts
```

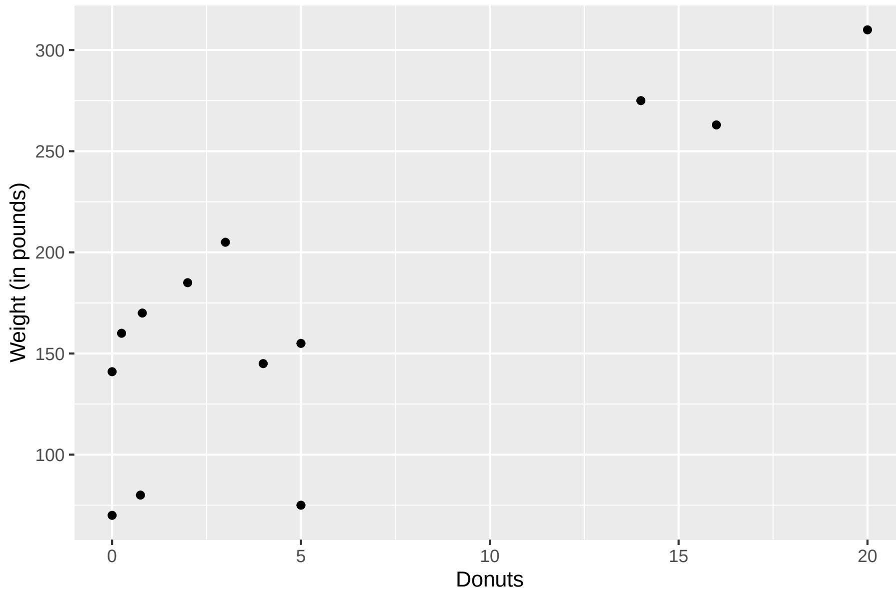


Figure 1.2: Weight and Donuts in Springfield

```
# Add the verticle line at 0
(p3 <- p2 + geom_vline(xintercept = 0, color = "gray80", size = 1.25))
```

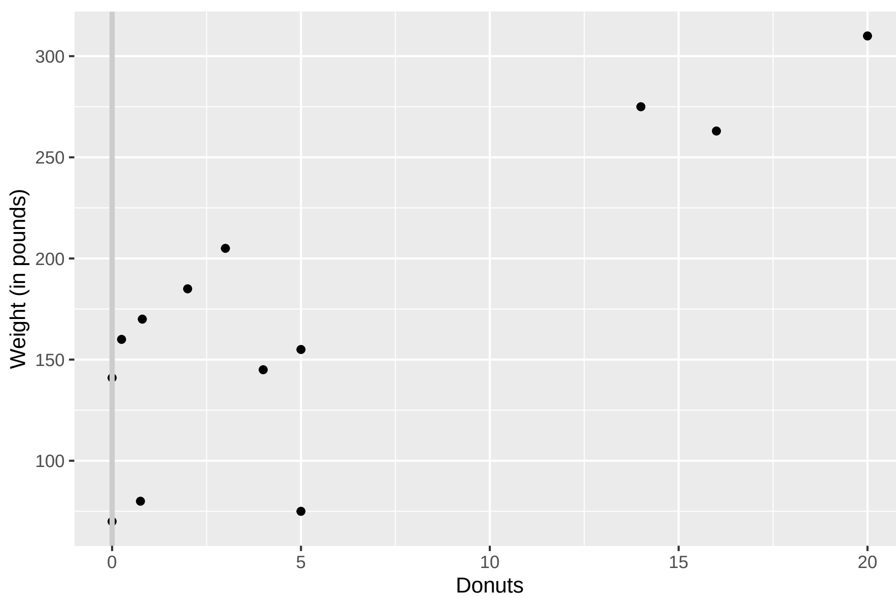


Figure 1.2: Weight and Donuts in Springfield

```
# the layering effect puts the line in front of the points, so we have to add it before geom_poir
(p4 <- p2 + #indentation makes to code easier to audit
    geom_vline(xintercept = 0, color = "gray80", size = 1) +
    geom_point())
```

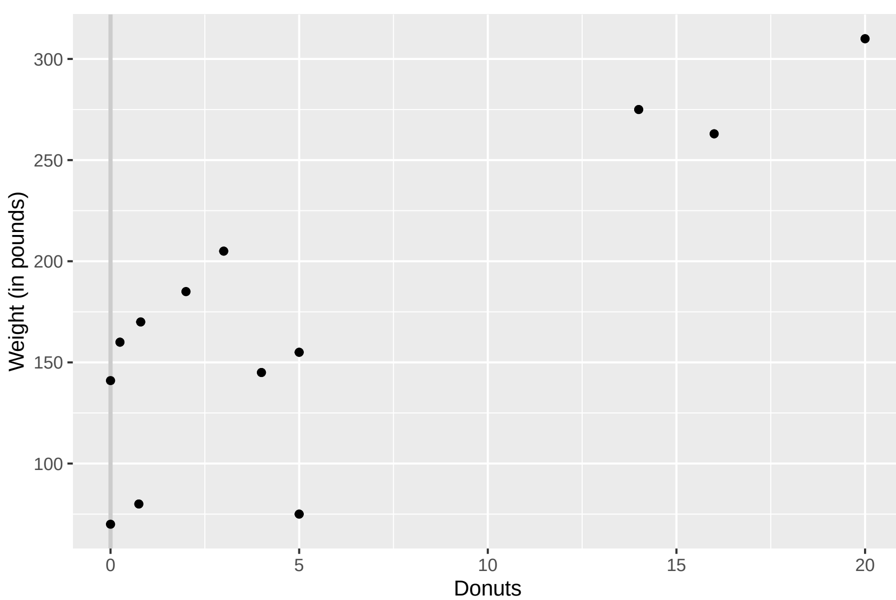


Figure 1.2: Weight and Donuts in Springfield

```
# Add the name labels
(p5 <- p4 + geom_text(aes(label = name)))
```

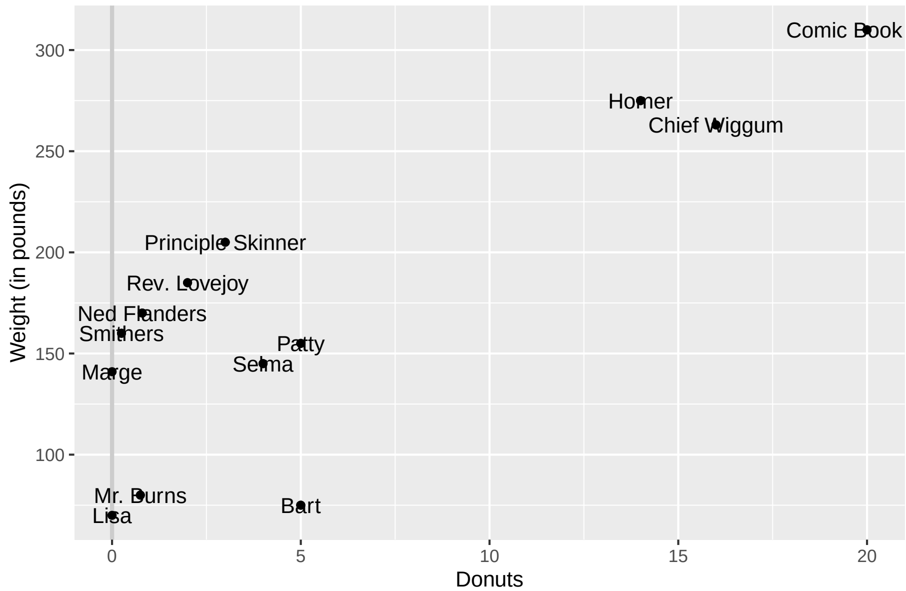


Figure 1.2: Weight and Donuts in Springfield

```
# Clean up the name labels with the ggrepel package
library(ggrepel)
(p6 <- p4 + geom_text_repel(aes(label = name)))
```

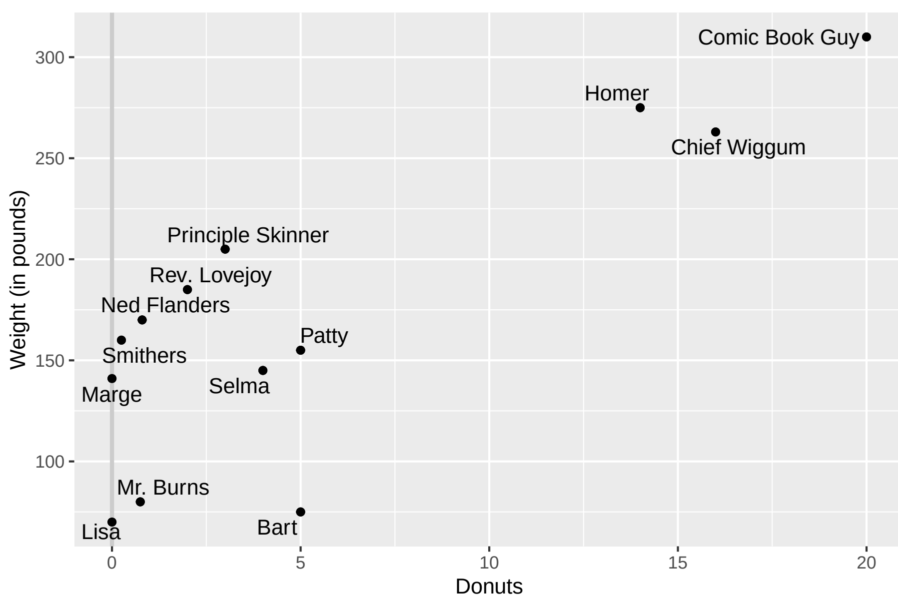


Figure 1.2: Weight and Donuts in Springfield

```r
# Use theme to adjust the non data elements
# \n in the y label creates a new line
(p7 <- p6 + labs(y = "Weight\n(in pounds)") +
    theme(axis.title.y = element_text(angle = 0), # change orientation of y-axis label
          panel.grid = element_blank(), # remove the background grid
          panel.background = element_blank(), # remove the background
          axis.line = element_line(), # add x and y axes
          plot.caption = element_text(hjust = 0))) #move the caption to the left.
```

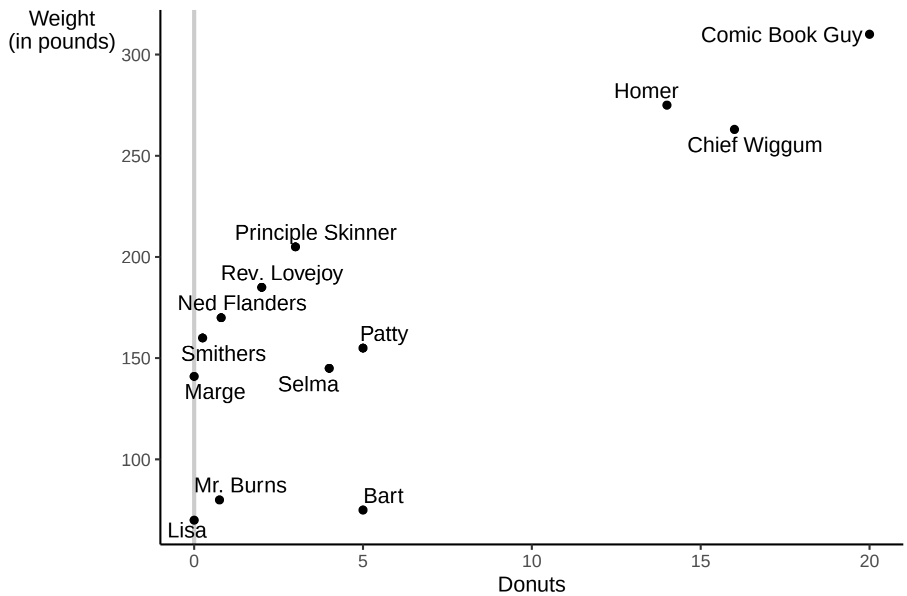


Figure 1.2: Weight and Donuts in Springfield

We can make the graph in one step, if we desire.

```r
p <- ggplot(data = donuts,
       mapping = aes(x = donuts_per_week, y = weight)) +
  geom_vline(xintercept = 0, color = "gray80", size = 1) +
  geom_point() +
  labs(x = "Donuts",
       y = "Weight\n(in pounds)", # \n creates a new line
       caption = "Figure 1.2: Weight and Donuts in Springfield") +
  geom_text_repel(aes(label = name)) +
  theme(axis.title.y = element_text(angle = 0),
          panel.grid = element_blank(),
          panel.background = element_blank(),
          axis.line = element_line(),
          plot.caption = element_text(hjust = 0))
p
```

Figure 1.2: Weight and Donuts in Springfield

again-1.bb

3.1.3 Figure 1.3

To create Figure 1.3, we start with the plot above we saved as an object, p. We add an additional geometric, geom_smooth to add the regression line.

```
p + labs(caption = "Figure 1.3: Regression Line for Weight and Donuts in Springfield")
  geom_smooth(method = "lm", se = F) + # method specifies the fit,  se = F turns off t
  annotate("text", label = expression(beta[1]*" (the slope)"), # text annotation
           y = 205, # position of the text
           x = 8,   # position of the text
           angle = 20, # angle of the text
           color = "Blue") + # color of the text
  geom_segment(aes(y = 121.613, x = 0, xend = 0, yend = 0),
               color = "blue",
               linetype = "dotted",
               size = 1) +
  annotate("text", label = expression(beta[0]*" = 121.613"),
           y = 115, x = 1.75, size = 3.5, color = "blue")
```

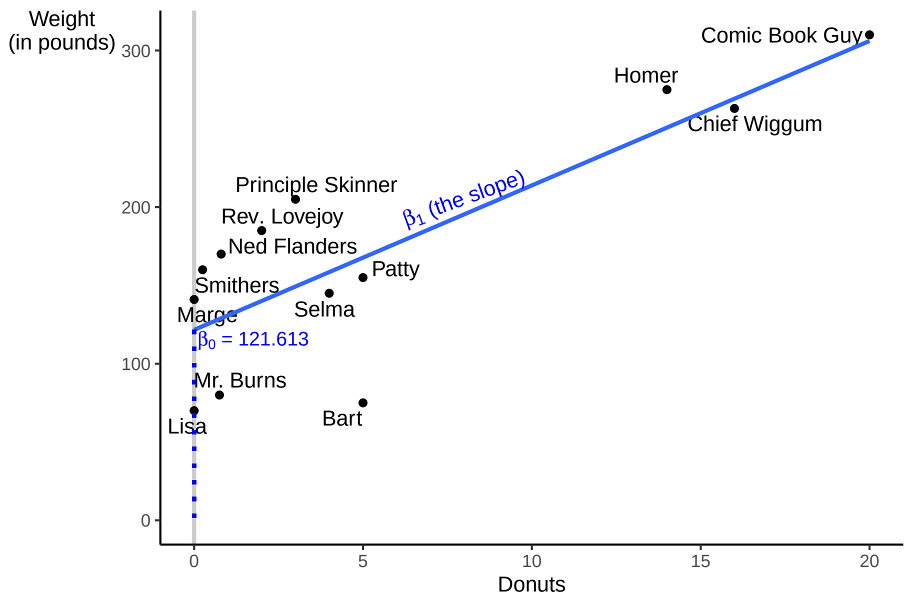


Figure 1.3: Regression Line for Weight and Donuts in Springfield

Chapter 4

Stats in the Wild: Good Data Practices

4.1 Introduction

I will introduce some additional R commands and packages through reproducing Table 2.1, Table 2.2, Table 2.5, and Figure 2.3. In addition, we will go through the *Computing Corner*.

4.2 Table and Figure Reproduction

4.2.1 Table 2.1

Since we saved the data frame we created in Chapter 1 as donuts.RData, we will `load` the file into global environment. We are only interested in the summary statistics for Weight and Donuts. We can get a basic set of summary statistics by calling `summary` on the data frame. But, `stargazer` from the stargazer package. The stargazer produces well formatted tables in LaTex code, HTML code, and ASCII text.

We will make use of the pipe operator from the `magrittr` package (also part of the `tidyverse`), as well. The pipe operator `%>%` (ctr-shift-m shortcut in R Studio) allows for more intuitive reading of code especially when nesting commands inside of each other. Take a simple example of finding calling the `str` command on a data frame, `df`. Without the pipe operator `%>%`, we would call the command like this `str(df)` and you might read this aloud alike this find the structure of df. With the pipe operator, call the command like this `df %>% str()`. Read aloud it might be something like this "take the `df` data and find its structure.” The pipe operator really shines when functions are nested together, as we shall see below.

```
load("donuts.RData")
library(tidyverse)
library(stargazer)
donuts %>%
  select("Weight" = weight, "Donuts" = donuts_per_week) %>% # choose and rename the co
  as.data.frame %>% # stargazer doesn't play nice with tibbles
    stargazer(type = "text", # tell stargazer to produce an ASCII text version of the
              title = "Table 2.1",
              omit.summary.stat = c("p25", "p75")) # omit the default 1st and 3rd quar
```

```
Table 2.1
=====================================
Statistic N   Mean   St. Dev. Min Max
-------------------------------------
Weight    13 172.000  76.200  70  310
Donuts    13  5.450   6.750    0  20
-------------------------------------
```

4.2.2 Table 2.2

To reproduce Table 2.2 we will need to add a variable named male which will take on the value 1 for each observation in the data that represents a male and a 0 otherwise.

```
load("donuts.RData")
donuts$name # this syntax reads print the variable name from the donuts data frame.
```

```
 [1] "Homer"            "Marge"            "Lisa"
 [4] "Bart"             "Comic Book Guy"   "Mr. Burns"
 [7] "Smithers"         "Chief Wiggum"     "Principle Skinner"
[10] "Rev. Lovejoy"     "Ned Flanders"     "Patty"
[13] "Selma"
```

Making use of donuts$name we see that the observations 1, 4, 5, 6, 7, 8, 9, 10, 11 are male and observations 2, 3, 12, 13 are not. We add the variable male to the donuts data frame as follows:

```
donuts$male <- c(1, 0, 0, 1, 1, 1, 1, 1, 1, 1, 1, 0, 0)
```

Call `table` to create a rudimentary version of Table 2.2

```
donuts$male %>%
  table()
```

```
.
0 1
4 9
```

4.2.3 Table 2.5

To reproduce Table 2.5 we must first retrieve the data. We will retrieve the data directly from the agencies responsible for their collection. You can retrieve the data as a comma-separated values (`csv`) file. A `csv` file is a plain text file in which the data are stored in a tabular format with values separated by a comma.

The crime data can be found on the U.S. Department of Justice Federal Bureau of Investigation Uniform Crime Reporting Statistics website. The single parent, urban, and poverty data can found on the U.S. Census website.

An investigation of the CrimeOneYearofData.csv file shows that there is meta information contained in the file along with the data. We could open the csv file in Excel and edited to remove the information or we could read it directly into R using `read_csv` from the `readr` package with options to ignore the meta information. The readr package has many advantages over the base R read functions, see `vignette("readr")` for more information. All of the text's data files are available in `csv` format, so we will make repeated use of `read-csv`. Investigation of the file with either Excel or text editor, shows the first nine rows are either blank or contain information about the data. Rows 63 to 183 contain footnotes and other additional information about the data. The names of the variables are in row ten of the csv file; so, we will skip the first nine rows using the option `skip`. We can choose the rows that contain the states and Washington, D.C., with the `n_max` option. Also, we need only the columns that contain the state names and the violent crime numbers. After reading in the data we will use `select` from the `dplyr` package.

Similar to `ggplot` being based on the grammar of graphics, `dplyr` is a grammar of data manipulation. dplyr consists of a set of "verbs" to help solve common data manipulation problems. To learn more about dplyr read `vignette("dplyr")`, visit dplyr, or for a good introduction visit the data import chapter in *R for Data Science*. We will make use of the pipe operator from the `magrittr` package (also part of the `tidyverse`), as well.

```
library(readr)
crime_one_year_of_data <- read_csv("Data/CrimeOneYearofData.csv", # read the data from
                              skip = 9, # start at row 10
                              n_max = 51) %>% # use only 51 records (ignores the US t
  select(c(1,3)) # select only the first and third columns
```

Using `glimpse` from dplyr, we see that we have a tibble with 51 observations and 2 variables. `glimpse` is similar to `str`.

```
crime_one_year_of_data %>%
  glimpse
```

```
Observations: 51
Variables: 2
$ State                 <chr> "Alabama", "Alaska", "Arizona", "Arkansas...
$ `Violent Crime rate` <dbl> 450, 633, 426, 516, 473, 339, 301, 645, 1...
```

We can see that State is a character vector and Violent Crime Rate is a numeric vector. Looking at the names of the variables we can see they do not adhere to the stylistic guidelines discussed above. The State variable begins with a capital letter and the Violent Crime Variable has capital letters and spaces in its name (the spaces are why you see the tick mark "‘" before and after the name). The state names are spelled out, but to reproduce Figure 2.3 we need to change those to two-letter abbreviations.

To bring the names into stylistic guidelines we can use `clean_names` from the janitor package, `snake case` is the default conversion. Note, the versatility of the `%>%` operator. If we did not use the `%>%` operator, the code would have been written as `glimpse(crime_one_year_of_data <- clean_names(crime_one_year_of_data))`

```
library(janitor)
crime_one_year_of_data <- crime_one_year_of_data %>%
  clean_names() %>%
  glimpse
```

```
Observations: 51
Variables: 2
$ state              <chr> "Alabama", "Alaska", "Arizona", "Arkansas",...
$ violent_crime_rate <dbl> 450, 633, 426, 516, 473, 339, 301, 645, 134...
```

We will read the other data in a similar fashion.

```
# Source: U.S. Census Bureau, 2009 American Community Survey, Table C23008
ACS_09_1YR_C23008_with_ann <- read_csv("Data/ACS_09_1YR_C23008_with_ann.csv",
    skip = 1,
    n_max = 51) %>%
  clean_names()
names(ACS_09_1YR_C23008_with_ann)[names(ACS_09_1YR_C23008_with_ann) == "geography"] <- "state"
# names(ACS_09_1YR_C23008_with_ann) looks at all the names
# [names(ACS_09_1YR_C23008_with_ann) == "geography"] extracts the column number of the name we wa
# <- "state" assigns the name state to that column number
ACS_09_1YR_C23008_with_ann %>% glimpse()
```

```
Observations: 51
Variables: 8
$ id                                          <chr> "0400000US01", "...
$ id2                                         <chr> "01", "02", "04"...
$ state                                       <chr> "Alabama", "Alas...
$ estimate_total                              <dbl> 1059528, 174634,...
$ estimate_under_6_years                      <dbl> 357122, 61489, 5...
$ estimate_under_6_years_living_with_one_parent <dbl> 141977, 20676, 2...
$ estimate_6_to_17_years                      <dbl> 702406, 113145, ...
$ estimate_6_to_17_years_living_with_one_parent <dbl> 270077, 32250, 3...
```

To create the percentage of children with single parents, add those under 6 living with one parent to those between 6 and 17 living with one parent and divide by the estimated total. We create the new variable with the `mutate` verb from dplyr and `select` state and percent with single parents into a new data frame.

```
single_parents <-
ACS_09_1YR_C23008_with_ann %>%
  mutate(percent_single_parents =
           (estimate_under_6_years_living_with_one_parent +
              estimate_6_to_17_years_living_with_one_parent) /
           estimate_total) %>%
  select(state, percent_single_parents) %>%
  glimpse()
```

```
Observations: 51
Variables: 2
$ state                  <chr> "Alabama", "Alaska", "Arizona", "Arkans...
$ percent_single_parents <dbl> 0.389, 0.303, 0.365, 0.378, 0.324, 0.28...
```

```
# Source: U.S. Census Bureau, 2009 American Community Survey, Table S1701
ACS_09_1YR_S1701_with_ann <- read_csv("Data/ACS_09_1YR_S1701_with_ann.csv",
    skip = 1,
```

```
    n_max = 51) %>%
  clean_names() %>%
  select("state" = geography, # directly name the variables when selected
         "percent_poverty" = percent_below_poverty_level_estimate_population_for_whom_p
  glimpse()
```

```
Observations: 51
Variables: 2
$ state           <chr> "Alabama", "Alaska", "Arizona", "Arkansas", "C...
$ percent_poverty <dbl> 17.5, 9.0, 16.5, 18.8, 14.2, 12.9, 9.4, 10.8, ...
```

To create the percent urban in 2009, we need to interpolate using the 2000 and 2010 censuses. After reading each set of data we will combine them into one data frame using `right_join` from the dplyr package.

```
# Source: U.S. Census Bureau, Table P002
DEC_00_SF1_P002_with_ann <- read_csv("Data/DEC_00_SF1_P002_with_ann.csv",
    skip = 1) %>%
  clean_names() %>%
  select("state" = geography, "total_00" = total , "urban_00" = urban) %>%
  glimpse()
```

```
Observations: 51
Variables: 3
$ state    <chr> "Alabama", "Alaska", "Arizona", "Arkansas", "Californ...
$ total_00 <dbl> 4447100, 626932, 5130632, 2673400, 33871648, 4301261,...
$ urban_00 <dbl> 2465673, 411257, 4523535, 1404179, 31989663, 3633185,...
```

```
# Source: U.S. Census Bureau, Table H2
DEC_10_SF1_P2_with_ann <- read_csv("Data/DEC_10_SF1_P2_with_ann.csv",
    skip = 1) %>%
  clean_names() %>%
  select("state" = geography, "total_10" = total , "urban_10" = urban) %>%
  glimpse()
```

```
Observations: 51
Variables: 3
$ state    <chr> "Alabama", "Alaska", "Arizona", "Arkansas", "Californ...
$ total_10 <chr> "4779736(r38235)", "710231(r38823)", "6392017", "2915...
$ urban_10 <dbl> 2821804, 468893, 5740659, 1637589, 35373606, 4332761,...
```

Note that total_10 from the 2010 census is a character vector. This means that there is at least one observation that includes characters. In fact, we can see at least 3 of the observations include parenthetical notes. `print` the variable to the screen to confirm each of the patterns is of the form "(some text)".

```
DEC_10_SF1_P2_with_ann$total_10
```

```
 [1] "4779736(r38235)"  "710231(r38823)"   "6392017"
 [4] "2915918(r39193)"  "37253956"         "5029196"
 [7] "3574097"          "897934"           "601723(r39494)"
[10] "18801310(r40184)" "9687653(r41102)"  "1360301"
[13] "1567582(r41542)"  "12830632"         "6483802"
[16] "3046355"          "2853118"          "4339367"
[19] "4533372"          "1328361"          "5773552(r42264)"
[22] "6547629"          "9883640(r45127)"  "5303925"
[25] "2967297"          "5988927"          "989415"
[28] "1826341"          "2700551"          "1316470"
[31] "8791894(r46246)"  "2059179(r46748)"  "19378102"
[34] "9535483"          "672591"           "11536504"
[37] "3751351"          "3831074"          "12702379"
[40] "1052567"          "4625364"          "814180(r48166)"
[43] "6346105"          "25145561(r48514)" "2763885"
[46] "625741"           "8001024"          "6724540"
[49] "1852994"          "5686986"          "563626"
```

We have confirmed that undesirable string has the same form in each position it exists. We must remove those comments and coerce the variable to numeric to proceed. We can determine how many instances of these comments occur using `str_detect` from the `stringr` package. `str_detect` will return a logical vector, so we need only sum the vector to count the number of times this occurs.

When calling `sum` on a logical vector, TRUE is treated as 1 and FALSE as 0, so summing the vector "counts" the number of TRUE occurrences. A regular expression, `regex` or `regexp`, is a sequence of characters that define a search pattern, to learn more visit regexr.com. The pattern we are looking for here is given by "\(.+\)". Since the parenthesis is a special character, it must be escaped with \, the . is a wild card, the + means 1 or more, so the .+ means find anything that appears 1 or more times. So the expression can be read as start with (find anything which occurs one or more times and end with).

```
str_detect(DEC_10_SF1_P2_with_ann$total_10, "\\(.+\\)") %>%
  sum()
```

```
[1] 13
```

The pattern occurs 13 times. We need to remove the string and coerce the character vector to a numeric vector. `str_replace_all` will remove all occurrences of the string. `as.numeric` will coerce the character vector to a numeric vector. We will make use of the "two-way" pipe operator `%<>%` in each function call. This operator takes the left hand side passes it to to the function and returns the result back to the original vector effectively overwriting it.

```
library(magrittr)
# the  %<>% operator is a "two way" pipe that sends the result back to the left hand s
DEC_10_SF1_P2_with_ann$total_10 %<>% str_replace_all("\\(.+\\)", "") # "" replaces the
DEC_10_SF1_P2_with_ann$total_10 %<>% as.numeric()
DEC_10_SF1_P2_with_ann %>% glimpse()
```

```
Observations: 51
Variables: 3
$ state    <chr> "Alabama", "Alaska", "Arizona", "Arkansas", "Californ...
$ total_10 <dbl> 4779736, 710231, 6392017, 2915918, 37253956, 5029196,...
$ urban_10 <dbl> 2821804, 468893, 5740659, 1637589, 35373606, 4332761,...
```

We see that total_10 is now a numeric vector.

We can now combine the two data frames using `right_join` from the dplyr package. Since each data frame has the state variable, `right_join` will add the columns from the 2010 census to the end (right) of the 2000 census matching observations by state. We will assign the result to percent_urban.

```
urban <- DEC_00_SF1_P002_with_ann %>%
  right_join(DEC_10_SF1_P2_with_ann) %>%
  glimpse()
```

```
Observations: 51
Variables: 5
$ state    <chr> "Alabama", "Alaska", "Arizona", "Arkansas", "Californ...
$ total_00 <dbl> 4447100, 626932, 5130632, 2673400, 33871648, 4301261,...
$ urban_00 <dbl> 2465673, 411257, 4523535, 1404179, 31989663, 3633185,...
$ total_10 <dbl> 4779736, 710231, 6392017, 2915918, 37253956, 5029196,...
$ urban_10 <dbl> 2821804, 468893, 5740659, 1637589, 35373606, 4332761,...
```

We can, now, interpolate the 2009 observations from the 2000 and 2010 observations. Since 2009 is nine tenths of the distance to 2010 from 2000, we will add 9/10 of the difference between the two observations to the 2000 observation.

```
urban %<>%
  mutate(percent_urban = (.9 * (urban_10 - urban_00) + urban_00) /
           (.9 * (total_10 - total_00) + total_00) * 100) %>%
  select(state, percent_urban)
```

We now have 4 data frames containing the information we need to create Table 2.5 and Figure 2.3. We will create a one data frame by joining the four data frames using the dplyr package.

```
crime_df <- crime_one_year_of_data %>%
  right_join(single_parents) %>%
  right_join(urban) %>%
  right_join(ACS_09_1YR_S1701_with_ann) %>%
  glimpse()
```

```
Observations: 51
Variables: 5
$ state                   <chr> "Alabama", "Alaska", "Arizona", "Arkans...
$ violent_crime_rate      <dbl> 450, 633, 426, 516, 473, 339, 301, 645,...
$ percent_single_parents <dbl> 0.389, 0.303, 0.365, 0.378, 0.324, 0.28...
$ percent_urban           <dbl> 58.7, 66.0, 89.7, 55.8, 94.9, 86.0, 88....
$ percent_poverty         <dbl> 17.5, 9.0, 16.5, 18.8, 14.2, 12.9, 9.4,...
```

Figure 2.3 includes state abbreviations rather than state names. We will change the names into abbreviations with the help of a built in character vector. state.name is character vector of state names, excluding Washington DC, built into R. We can concatenate that vector with the character string "District of Columbia", sort the new character vector alphabetically, convert the names to abbreviations with `state2abbr` from the `openintro` package, and assign the result to the state vector in the crime_one_year_of_data data frame.

```
library(openintro)
state_abb <- c(state.name, "District of Columbia") %>%
  sort() %>%
  state2abbr()
crime_df$state <- state_abb
crime_df %>% glimpse
```

```
Observations: 51
Variables: 5
$ state                   <chr> "AL", "AK", "AZ", "AR", "CA", "CO", "CT...
$ violent_crime_rate      <dbl> 450, 633, 426, 516, 473, 339, 301, 645,...
$ percent_single_parents <dbl> 0.389, 0.303, 0.365, 0.378, 0.324, 0.28...
$ percent_urban           <dbl> 58.7, 66.0, 89.7, 55.8, 94.9, 86.0, 88....
$ percent_poverty         <dbl> 17.5, 9.0, 16.5, 18.8, 14.2, 12.9, 9.4,...
```

We proceed as we did with Table 2.1 to reproduce Table 2.3.

```
crime_df %>%
  select("Violent crime rate (per 100,00 people)" = violent_crime_rate,
         "Percent single parents" = percent_single_parents,
         "Percent urban" = percent_urban,
         "Percent poverty" = percent_poverty) %>%
  as.data.frame() %>%
  stargazer(type = "text",
            title = "Table 2.3",
            omit.summary.stat = c("p25", "p75"))
```

```
Table 2.3
==========================================================================
Statistic                              N   Mean   St. Dev.   Min      Max
--------------------------------------------------------------------------
Violent crime rate (per 100,00 people) 51 407.000 206.000  120.000 1,349.000
Percent single parents                 51  0.332   0.064    0.185    0.608
Percent urban                          51 73.900   14.900  38.800   100.000
Percent poverty                        51 13.900   3.110    8.500   21.900
--------------------------------------------------------------------------
```

4.2.4 Figure 2.3

We will use `ggplot` from the `ggplot2` package to reproduce Figure 2.3. We will use the `plot_grd` from `cowplot` package to create a grid of the three individual plots after we create them individually.

```
plot_urban <-
  crime_df %>%
  ggplot(aes(x = percent_urban, y = violent_crime_rate)) +
  labs(x = "Percent urban\n(0-to-100 scale)", # \n creates a new line
       y = "Violent\ncrime\nrate\n(per\n100,000\npeople)") +
  geom_text(aes(label = state), color = "blue") +
  scale_y_continuous(breaks = seq(200, 1200, 200)) + # creates a sequence from 200 to
  scale_x_continuous(breaks = seq(40, 100, 10)) + # creates a sequence from 40 to 100
  theme(axis.title.y = element_text(angle = 0),
        panel.grid = element_blank(),
        panel.background = element_blank(),
        axis.line = element_line())

plot_single <-
  crime_df %>%
  ggplot(aes(x = percent_single_parents, y = violent_crime_rate)) +
  labs(x = "Percent single parent\n(0-to-1 scale)", # \n creates a new line
```

```
       y = "") +
  geom_text(aes(label = state), color = "blue") +
  scale_y_continuous(breaks = seq(200, 1200, 200)) +
  theme(axis.title.y = element_text(angle = 0),
        panel.grid = element_blank(),
        panel.background = element_blank(),
        axis.line = element_line())

plot_poverty <-
  crime_df %>%
  ggplot(aes(x = percent_poverty, y = violent_crime_rate)) +
  labs(x = "Percent poverty\n(0-to-100 scale)", # \n creates a new line
       y = "") +
  geom_text(aes(label = state), color = "blue") +
  scale_y_continuous(breaks = seq(200, 1200, 200)) +
  scale_x_continuous(breaks = seq(8, 22, 2)) +
  theme(axis.title.y = element_text(angle = 0),
        panel.grid = element_blank(),
        panel.background = element_blank(),
        axis.line = element_line())

library(cowplot)
plot_grid(plot_urban, plot_single, plot_poverty,  ncol = 3)
```

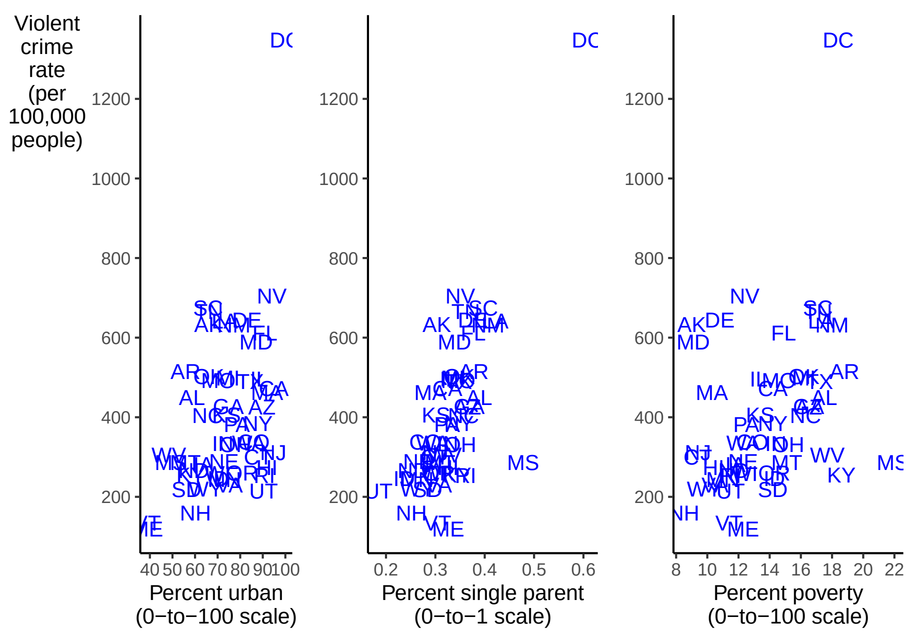


FIGURE 2.3: Scatterplots of Violent Crime against Percent Urban, Single Par-

ent, and Poverty

4.3 Computing Center

4.3.1 Reading Data

There are packages available to read data formatted in a variety of ways into R. Data can also be imported using the Import Dataset icon in the Environment/History pane. When learning to import data, this method can be useful as it will create the command line necessary to import the data, which you can then paste into your R Script of R Markdown file.

4.3.2 Manually Entering Data

In Chapter 1 we saw that we can directly (manually) enter data into R as well. Below is the appropriate syntax for doing so.

```
name <- c("Homer", "Marge", "Lisa", "Bart", "Comic Book Guy", "Mr. Burns", "Smithers",
donuts_per_week <- c(14, 0, 0, 5, 20, 0.75, 0.25, 16, 3, 2, 0.8, 5, 4)
weight <- c(275, 141, 70, 75, 310, 80, 160, 263, 205, 185, 170, 155, 145)
```

We can combine this into a single "file" called a data frame as follows:

```
Donuts <- data.frame(name, donuts_per_week, weight)
Donuts %>%
  str()
```

```
'data.frame':   13 obs. of  3 variables:
 $ name           : Factor w/ 13 levels "Bart","Chief Wiggum",..: 4 6 5 1 3 7 13 2 10
 $ donuts_per_week: num  14 0 0 5 20 0.75 0.25 16 3 2 ...
 $ weight         : num  275 141 70 75 310 80 160 263 205 185 ...
```

The character vector "name" is coerced to a Factor by default. Factors in R are store as a vector of integer values with a corresponding set of character values. We will see that Factors are very useful, however, in this case we want name to remain a character vector. If we add the option `stringsAsFactors = FALSE` to our call of `data.frame` we can prevent the coercion. Our we can call `tibble` as described in Chapter 1.

```
Donuts <- data.frame(name, donuts_per_week, weight, stringsAsFactors = FALSE)
Donuts %>% str()
```

```
'data.frame':   13 obs. of  3 variables:
 $ name          : chr  "Homer" "Marge" "Lisa" "Bart" ...
 $ donuts_per_week: num  14 0 0 5 20 0.75 0.25 16 3 2 ...
 $ weight         : num  275 141 70 75 310 80 160 263 205 185 ...
```

You can see in the Global Environment tab of the Environment/History pane that you have an object named Donuts that has 13 observations on 3 variables. In addition, you can see under values you have each variable as well.

4.3.3 Simple Statistics

R has many built in calls to get basic statistics on data. For example, to get the mean of a variable call `mean()`. Be aware, that if there are missing values in the data the function call will return NA as it's result. The donuts data frame contains to missing values "NA", so it won't be a problem. Some simple exploratory data analysis will let you know if you have any issues in the data. We saw one of those problems above when we had a variable that we thought was numeric, but was read in as a character vector. `summary` is a good place to start.

```
donuts %>%
  summary()
```

```
 observation_number     name           donuts_per_week     weight
 Min.   : 1         Length:13          Min.   : 0.00   Min.   : 70
 1st Qu.: 4         Class :character   1st Qu.: 0.75   1st Qu.:141
 Median : 7         Mode  :character   Median : 3.00   Median :160
 Mean   : 7                            Mean   : 5.45   Mean   :172
 3rd Qu.:10                            3rd Qu.: 5.00   3rd Qu.:205
 Max.   :13                            Max.   :20.00   Max.   :310
      male
 Min.   :0.000
 1st Qu.:0.000
 Median :1.000
 Mean   :0.692
 3rd Qu.:1.000
 Max.   :1.000
```

We confirmed that there are no missing values in our data. If there were, we can easily deal with them with a option in the function call (I include the option below.)

```
donuts$weight %>%
  mean(na.rm = TRUE)
```

```
[1] 172
```

Call `var` or `sd` to return the sample variance or sample standard deviation. Of course the standard deviation can also be calculated by calling `sqrt` on the result of the `var` call. There are multiple ways to retrieve the number of observations of a variable or data frame. The minimum and maximum values are returned by calling `min` and `max`. As described in the text, we can call `sum` on the result of the call `is.finite`. `nrow` will return the number of observations in a data frame, `NROW` will return the number of observations of a vector or single variable.

```
donuts$weight %>% var()
```

```
[1] 5800
```

```
donuts$weight %>% var() %>% sqrt()
```

```
[1] 76.2
```

```
donuts$weight %>% sd()
```

```
[1] 76.2
```

```
donuts$weight %>% min()
```

```
[1] 70
```

```
donuts$weight %>% max()
```

```
[1] 310
```

```
donuts$weight %>% NROW()
```

```
[1] 13
```

To return a variety of descriptive statistics on a data frame or variable we can call `stargazer` from the stargazer package.

```
donuts %>%
  as.data.frame %>%
  stargazer(type = "text")
```

```
================================================================
Statistic          N   Mean   St. Dev. Min Pctl(25) Pctl(75) Max
----------------------------------------------------------------
observation_number 13  7.000   3.890    1     4        10     13
donuts_per_week    13  5.450   6.750    0    0.8       5      20
weight             13 172.000  76.200  70    141      205    310
male               13  0.692   0.480    0     0        1       1
----------------------------------------------------------------
```

Subsetting in R can be accomplished in a variety of ways. In Base R, use [] syntax. Use brackets can be used to call specific rows and columns from a matrix or data frame. To return the observation in the 12^{th} row and 3rd column call `donuts[12,3]`. To return all of the observations in a specific row or column, leave the row or column number out of the call. To return all of the observations from the 3rd column call `donuts[,3]`. To return the observations for an individual record, say the 4^{th} row, call `donuts[4,]`. To choose (subset) all of those records where, e.g„ donuts eaten per week is 0, call `donuts[donuts$donuts_per_week == 0,]`; to choose all those records where donuts donuts are not equal to 0, call `donuts[donuts$donuts_per_week != 0,]`. We can also subset using `filter` from `dplyr`. An advantage of subsetting with `dplyr` is that the resulting tibble can be piped into the another function call.

```
donuts[12,3]
```

```
# A tibble: 1 x 1
  donuts_per_week
            <dbl>
1               5
```

```
donuts[,3]
```

```
# A tibble: 13 x 1
   donuts_per_week
             <dbl>
 1              14
 2               0
 3               0
```

```
 4           5
 5          20
 6           0.75
 7           0.25
 8          16
 9           3
10           2
11           0.8
12           5
13           4
```

```
donuts[4,]
```

```
# A tibble: 1 x 5
  observation_number name  donuts_per_week weight  male
               <int> <chr>           <dbl>  <dbl> <dbl>
1                  4 Bart                5     75     1
```

```
donuts[donuts$donuts_per_week == 0,]
```

```
# A tibble: 2 x 5
  observation_number name  donuts_per_week weight  male
               <int> <chr>           <dbl>  <dbl> <dbl>
1                  2 Marge               0    141     0
2                  3 Lisa                0     70     0
```

```
donuts[donuts$donuts_per_week != 0,]
```

```
# A tibble: 11 x 5
   observation_number name              donuts_per_week weight  male
                <int> <chr>                       <dbl>  <dbl> <dbl>
 1                  1 Homer                       14       275     1
 2                  4 Bart                         5        75     1
 3                  5 Comic Book Guy              20       310     1
 4                  6 Mr. Burns                    0.75     80     1
 5                  7 Smithers                     0.25    160     1
 6                  8 Chief Wiggum                16       263     1
 7                  9 Principle Skinner            3       205     1
 8                 10 Rev. Lovejoy                 2       185     1
 9                 11 Ned Flanders                 0.8     170     1
10                 12 Patty                        5       155     0
11                 13 Selma                        4       145     0
```

```r
donuts %>%
  filter(donuts_per_week == 0)
```

```
# A tibble: 2 x 5
  observation_number name  donuts_per_week weight  male
               <int> <chr>           <dbl>  <dbl> <dbl>
1                  2 Marge               0    141     0
2                  3 Lisa                0     70     0
```

```r
donuts %>%
  filter(donuts_per_week != 0)
```

```
# A tibble: 11 x 5
   observation_number name              donuts_per_week weight  male
                <int> <chr>                       <dbl>  <dbl> <dbl>
 1                  1 Homer                       14       275     1
 2                  4 Bart                         5        75     1
 3                  5 Comic Book Guy              20       310     1
 4                  6 Mr. Burns                    0.75     80     1
 5                  7 Smithers                     0.25    160     1
 6                  8 Chief Wiggum                16       263     1
 7                  9 Principle Skinner            3       205     1
 8                 10 Rev. Lovejoy                 2       185     1
 9                 11 Ned Flanders                 0.8     170     1
10                 12 Patty                        5       155     0
11                 13 Selma                        4       145     0
```

We can subset on more than one variable as well. Using Base R we can choose all those males who consumed some donuts per week by calling `donuts[donuts$donuts_per_week != 0 & donuts$male == 1]`. We can choose all those observations where donut consumption per week is more than 15 or the person is female by using the or operator | in call `donuts[donuts$donuts_per_week > 15 | donuts$male != 1,]`. `filter` can be used as well.

```r
donuts[donuts$donuts_per_week != 0 & donuts$male == 1,]
```

```
# A tibble: 9 x 5
  observation_number name              donuts_per_week weight  male
               <int> <chr>                       <dbl>  <dbl> <dbl>
1                  1 Homer                       14       275     1
2                  4 Bart                         5        75     1
3                  5 Comic Book Guy              20       310     1
```

```
4                  6 Mr. Burns                      0.75     80      1
5                  7 Smithers                       0.25    160      1
6                  8 Chief Wiggum                  16       263      1
7                  9 Principle Skinner              3       205      1
8                 10 Rev. Lovejoy                   2       185      1
9                 11 Ned Flanders                   0.8     170      1
```

```
donuts %>%
  filter(donuts_per_week != 0 & male == 1)
```

```
# A tibble: 9 x 5
  observation_number name              donuts_per_week weight  male
               <int> <chr>                       <dbl>  <dbl> <dbl>
1                  1 Homer                       14       275      1
2                  4 Bart                         5        75      1
3                  5 Comic Book Guy              20       310      1
4                  6 Mr. Burns                    0.75     80      1
5                  7 Smithers                     0.25    160      1
6                  8 Chief Wiggum                16       263      1
7                  9 Principle Skinner            3       205      1
8                 10 Rev. Lovejoy                 2       185      1
9                 11 Ned Flanders                 0.8     170      1
```

```
# a slightly more intuitive alternative is:
donuts %>%
  filter(donuts_per_week != 0) %>%
  filter(male == 1)
```

```
# A tibble: 9 x 5
  observation_number name              donuts_per_week weight  male
               <int> <chr>                       <dbl>  <dbl> <dbl>
1                  1 Homer                       14       275      1
2                  4 Bart                         5        75      1
3                  5 Comic Book Guy              20       310      1
4                  6 Mr. Burns                    0.75     80      1
5                  7 Smithers                     0.25    160      1
6                  8 Chief Wiggum                16       263      1
7                  9 Principle Skinner            3       205      1
8                 10 Rev. Lovejoy                 2       185      1
9                 11 Ned Flanders                 0.8     170      1
```

```
donuts[donuts$donuts_per_week > 15 | donuts$male != 1,]
```

```
# A tibble: 6 x 5
```

```
  observation_number name           donuts_per_week weight  male
               <int> <chr>                    <dbl>  <dbl> <dbl>
1                  2 Marge                        0    141     0
2                  3 Lisa                         0     70     0
3                  5 Comic Book Guy              20    310     1
4                  8 Chief Wiggum                16    263     1
5                 12 Patty                        5    155     0
6                 13 Selma                        4    145     0
```

```
donuts %>%
  filter(donuts_per_week > 15 | male != 1)
```

```
# A tibble: 6 x 5
  observation_number name           donuts_per_week weight  male
               <int> <chr>                    <dbl>  <dbl> <dbl>
1                  2 Marge                        0    141     0
2                  3 Lisa                         0     70     0
3                  5 Comic Book Guy              20    310     1
4                  8 Chief Wiggum                16    263     1
5                 12 Patty                        5    155     0
6                 13 Selma                        4    145     0
```

We can modify Figure 2.2 to include only males by modifying our original code by piping the filtered results into `ggplot`.

```
library(ggrepel)
donuts %>%
  filter(male == 1) %>%
  ggplot(mapping = aes(x = donuts_per_week, y = weight)) +
  geom_vline(xintercept = 0, color = "gray80", size = 1) +
  geom_point(color = "blue", size = 2) +
  labs(x = "Donuts",
       y = "Weight\n(in pounds)", # \n creates a new line
       caption = "Figure 2.2: Weight and Donuts in Springfield") +
  geom_text_repel(aes(label = name), color = "blue") +
  theme(axis.title.y = element_text(angle = 0),
        panel.grid = element_blank(),
        panel.background = element_blank(),
        axis.line = element_line(),
        plot.caption = element_text(hjust = 0))
```

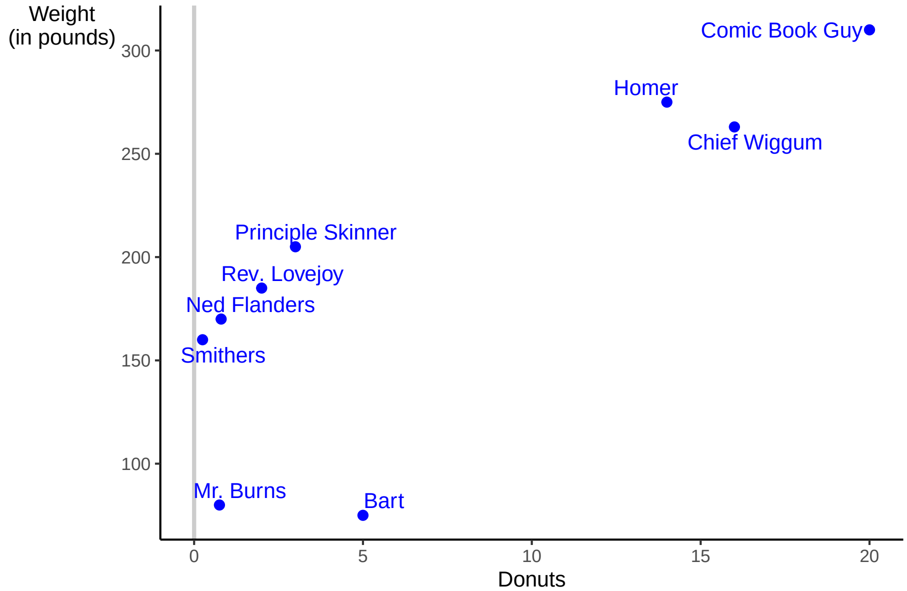


Figure 2.2: Weight and Donuts in Springfield

Chapter 5

Bivariate OLS: The Foundation of Econometric Analysis

We will work through *Computing Corner*.

5.1 Estimating a simple regression

To run a simple regression in R, make us of the function `lm`. Like all functions `lm` requires and argument list. You can see the arguments required and the defaualt values, if any, in a variety of ways in R Studio. `args(lm)` will return the arguments for the lm function. `?lm` will open the help page in the Files/Plots/Packages/Help Pane, which can also be accessed by typing lm in the search in the same pane. Estimating a regression using `lm` requires only two arguments: the formula and data arguments. If you provide those arguments in that order, R doesn't require that you use the argument's name. This is true for all functions in R, the default is that the arguments appear in order. We can see by calling `str` on ols_donuts that `lm` creates a list object which contains 12 elements. A list is an object that contains elements of several different types like, strings, vectors, matrices, lists, etc. Each of those elements can be extracted from the list in much the same way as accessing pieces of a data frame.

```
library(magrittr)
load("donuts.RData")
ols_donuts <- lm(formula = weight ~ donuts_per_week, data = donuts)
ols_donuts %>%
  str()
```

```
List of 12
 $ coefficients : Named num [1:2] 121.61 9.22
  ..- attr(*, "names")= chr [1:2] "(Intercept)" "donuts_per_week"
 $ residuals    : Named num [1:13] 24.26 19.39 -51.61 -92.73 3.92 ...
  ..- attr(*, "names")= chr [1:13] "1" "2" "3" "4" ...
 $ effects      : Named num [1:13] -619.6 215.66 -60.24 -99.06 4.49 ...
  ..- attr(*, "names")= chr [1:13] "(Intercept)" "donuts_per_week" "" "" ...
 $ rank         : int 2
 $ fitted.values: Named num [1:13] 251 122 122 168 306 ...
  ..- attr(*, "names")= chr [1:13] "1" "2" "3" "4" ...
 $ assign       : int [1:2] 0 1
 $ qr          :List of 5
  ..$ qr   : num [1:13, 1:2] -3.606 0.277 0.277 0.277 0.277 ...
  .. ..- attr(*, "dimnames")=List of 2
  .. .. ..$ : chr [1:13] "1" "2" "3" "4" ...
  .. .. ..$ : chr [1:2] "(Intercept)" "donuts_per_week"
  .. ..- attr(*, "assign")= int [1:2] 0 1
  ..$ qraux: num [1:2] 1.28 1.31
  ..$ pivot: int [1:2] 1 2
  ..$ tol  : num 0.0000001
  ..$ rank : int 2
  ..- attr(*, "class")= chr "qr"
 $ df.residual  : int 11
 $ xlevels      : Named list()
 $ call         : language lm(formula = weight ~ donuts_per_week, data = donuts)
 $ terms        :Classes 'terms', 'formula'  language weight ~ donuts_per_week
  .. ..- attr(*, "variables")= language list(weight, donuts_per_week)
  .. ..- attr(*, "factors")= int [1:2, 1] 0 1
  .. .. ..- attr(*, "dimnames")=List of 2
  .. .. .. ..$ : chr [1:2] "weight" "donuts_per_week"
  .. .. .. ..$ : chr "donuts_per_week"
  .. ..- attr(*, "term.labels")= chr "donuts_per_week"
  .. ..- attr(*, "order")= int 1
  .. ..- attr(*, "intercept")= int 1
  .. ..- attr(*, "response")= int 1
  .. ..- attr(*, ".Environment")=<environment: R_GlobalEnv>
  .. ..- attr(*, "predvars")= language list(weight, donuts_per_week)
  .. ..- attr(*, "dataClasses")= Named chr [1:2] "numeric" "numeric"
  .. .. ..- attr(*, "names")= chr [1:2] "weight" "donuts_per_week"
 $ model        :'data.frame':  13 obs. of  2 variables:
  ..$ weight         : num [1:13] 275 141 70 75 310 80 160 263 205 185 ...
  ..$ donuts_per_week: num [1:13] 14 0 0 5 20 0.75 0.25 16 3 2 ...
  ..- attr(*, "terms")=Classes 'terms', 'formula'  language weight ~ donuts_per_week
  .. .. ..- attr(*, "variables")= language list(weight, donuts_per_week)
  .. .. ..- attr(*, "factors")= int [1:2, 1] 0 1
  .. .. .. ..- attr(*, "dimnames")=List of 2
```

```
  .. .. .. .. ..$ : chr [1:2] "weight" "donuts_per_week"
  .. .. .. .. ..$ : chr "donuts_per_week"
  .. .. ..- attr(*, "term.labels")= chr "donuts_per_week"
  .. .. ..- attr(*, "order")= int 1
  .. .. ..- attr(*, "intercept")= int 1
  .. .. ..- attr(*, "response")= int 1
  .. .. ..- attr(*, ".Environment")=<environment: R_GlobalEnv>
  .. .. ..- attr(*, "predvars")= language list(weight, donuts_per_week)
  .. .. ..- attr(*, "dataClasses")= Named chr [1:2] "numeric" "numeric"
  .. .. .. ..- attr(*, "names")= chr [1:2] "weight" "donuts_per_week"
 - attr(*, "class")= chr "lm"
```

You can see the type for each of the 12 elements in the list. Each of those elements can be extracted by using the $ convention you use to get variables from a data frame (which is a type of list) as describe in the text. In addition, there are many commands that will extract a standard set of elements and present them in conventional ways. For example, to get the regression results, `summary` from base R, `stargazer` from the stargazer package, and `tidy` and `glance` from the `broom` package provide the output in useful ways. `augment` from the broom package creates a tibble of actual, fitted, residuals, etc. In fact, all of the commands in the `broom` package create tibbles which can useful for further analysis. `stargazer` can be modified to include a variety of statistics. So, you can extract fitted values or residuals, e.g., in the same way you retrieve any data from a data frame or tibble.

```
library(tidyverse)
library(broom)
library(stargazer)
ols_donuts %>%
  summary()
```

```

Call:
lm(formula = weight ~ donuts_per_week, data = donuts)

Residuals:
   Min     1Q Median     3Q    Max 
-92.73 -13.51   3.92  36.08  55.72 

Coefficients:
                Estimate Std. Error t value Pr(>|t|)    
(Intercept)       121.61      16.59    7.33 0.000015 ***
donuts_per_week     9.22       1.96    4.71  0.00064 ***
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

```
Residual standard error: 45.8 on 11 degrees of freedom
Multiple R-squared:  0.668, Adjusted R-squared:  0.638
F-statistic: 22.2 on 1 and 11 DF,  p-value: 0.000643
```

```r
ols_donuts %>%
  stargazer(type = "text")
```

```
===============================================
                        Dependent variable:
                    ---------------------------
                              weight
-----------------------------------------------
donuts_per_week              9.220***
                              (1.960)

Constant                    122.000***
                             (16.600)

-----------------------------------------------
Observations                    13
R2                             0.668
Adjusted R2                    0.638
Residual Std. Error      45.800 (df = 11)
F Statistic           22.200*** (df = 1; 11)
===============================================
Note:               *p<0.1; **p<0.05; ***p<0.01
```

```r
ols_donuts %>%
  tidy()
```

```
# A tibble: 2 x 5
  term            estimate std.error statistic   p.value
  <chr>              <dbl>     <dbl>     <dbl>     <dbl>
1 (Intercept)       122.       16.6       7.33 0.0000149
2 donuts_per_week     9.22      1.96      4.71 0.000643
```

```r
ols_donuts %>%
  glance()
```

```
# A tibble: 1 x 11
  r.squared adj.r.squared sigma statistic p.value    df logLik   AIC   BIC
```

```
      <dbl>         <dbl> <dbl>     <dbl>   <dbl> <int>   <dbl> <dbl> <dbl>
1     0.668         0.638  45.8      22.2 6.43e-4     2   -67.1  140.  142.
# ... with 2 more variables: deviance <dbl>, df.residual <int>
```

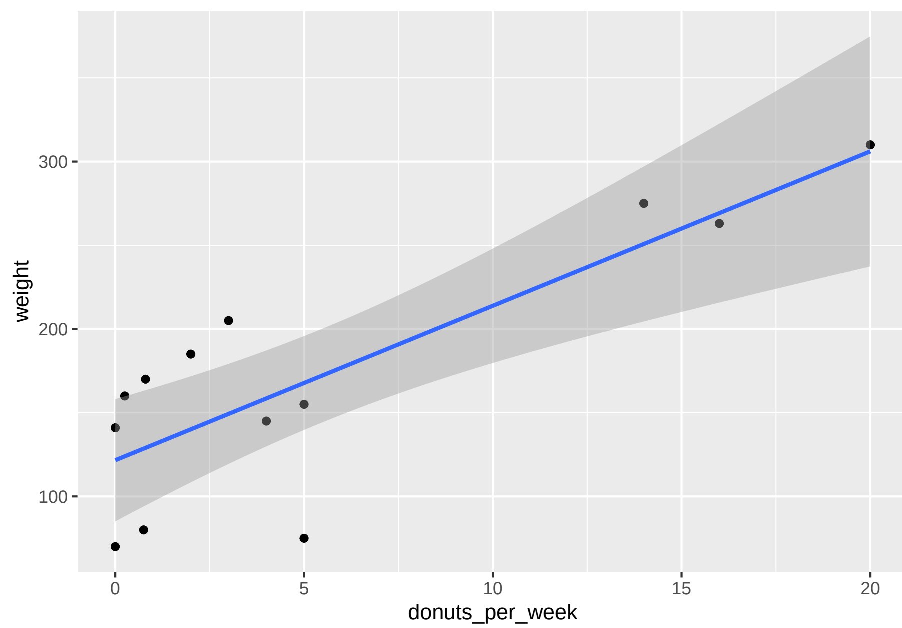
```r
ols_donuts %>%
  augment()
```

```
# A tibble: 13 x 9
   weight donuts_per_week .fitted .se.fit .resid   .hat .sigma .cooksd
    <dbl>           <dbl>   <dbl>   <dbl>  <dbl>  <dbl>  <dbl>   <dbl>
 1    275           14       251.    21.0  24.3  0.211    47.3 0.0474
 2    141            0       122.    16.6  19.4  0.131    47.6 0.0156
 3     70            0       122.    16.6 -51.6  0.131    44.7 0.110
 4     75            5       168.    12.7 -92.7  0.0773   37.1 0.186
 5    310           20       306.    31.2   3.92 0.464    48.0 0.00591
 6     80            0.75    129.    15.7 -48.5  0.117    45.2 0.0844
 7    160            0.25    124.    16.3  36.1  0.126    46.5 0.0513
 8    263           16       269.    24.3  -6.19 0.281    48.0 0.00495
 9    205            3       149.    13.6  55.7  0.0879   44.4 0.0781
10    185            2       140.    14.4  44.9  0.0986   45.7 0.0584
11    170            0.8     129.    15.6  41.0  0.116    46.0 0.0597
12    155            5       168.    12.7 -12.7  0.0773   47.9 0.00350
13    145            4       159.    13.0 -13.5  0.0807   47.8 0.00415
# ... with 1 more variable: .std.resid <dbl>
```

5.2 scatter Plot with Regression Line

ggplot2 makes adding a fitted regression line to a scatter plot very easy. You need only add a geometry called geom_smooth with the appropriate method to plot. The default is to include a confidence interval estimate around the fitted line. To remove the error band add the option `se = FALSE`.

```r
donuts %>%
  ggplot(aes(x = donuts_per_week, y = weight)) +
  geom_point() +
  geom_smooth(method = "lm")
```



5.3 Subsetting Data for Regressions

Subsetting can be directly done with the `subset` option in the `lm` call. To run a regression that excludes the Homer record, use the option `subset = (name != "Homer")`.

```
ols_no_homer <- lm(formula = weight ~ donuts_per_week, data = donuts, subset = (name !
ols_no_homer %>%
  tidy()
```

```
# A tibble: 2 x 5
  term            estimate std.error statistic   p.value
  <chr>              <dbl>     <dbl>     <dbl>     <dbl>
1 (Intercept)       122.        17.1       7.12 0.0000323
2 donuts_per_week     8.74       2.19      4.00 0.00252
```

```
ols_no_homer %>%
  summary()
```

```
Call:
lm(formula = weight ~ donuts_per_week, data = donuts, subset = (name !=
```

```
    "Homer"))

Residuals:
   Min     1Q Median     3Q    Max 
-90.58 -20.99   7.26  37.24  56.90 

Coefficients:
                Estimate Std. Error t value Pr(>|t|)    
(Intercept)       121.87      17.13    7.12 0.000032 ***
donuts_per_week     8.74       2.19    4.00   0.0025 ** 
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

Residual standard error: 47.3 on 10 degrees of freedom
Multiple R-squared:  0.615, Adjusted R-squared:  0.577 
F-statistic:   16 on 1 and 10 DF,  p-value: 0.00252
```

Alternatively we can make use of `filter` from the `dplyr` package. Recall, `filter` is the data manipulation verb that chooses observations in a data frame. I introduce the exposition operator **%$%** from the magrittr package. **%$%** is useful at the end of a pipeline to expose your manipulated data to function. You can use it with `subset` or `filter`.

```
donuts %>%
  subset(name != "Homer") %$%
  lm(weight ~ donuts_per_week)
```

```

Call:
lm(formula = weight ~ donuts_per_week)

Coefficients:
    (Intercept)  donuts_per_week  
         121.87             8.74  
```

```
donuts %>%
  filter(name != "Homer") %$%
  lm(weight ~ donuts_per_week)
```

```

Call:
lm(formula = weight ~ donuts_per_week)

Coefficients:
```

```
    (Intercept)  donuts_per_week
         121.87             8.74
```

To include those observations where weight is greater than 100:

```
donuts %>%
  subset(weight > 100) %$%
  lm(weight ~ donuts_per_week)
```

```

Call:
lm(formula = weight ~ donuts_per_week)

Coefficients:
    (Intercept)  donuts_per_week
         151.05             7.66
```

```
donuts %>%
  filter(weight > 100) %$%
  lm(weight ~ donuts_per_week)
```

```

Call:
lm(formula = weight ~ donuts_per_week)

Coefficients:
    (Intercept)  donuts_per_week
         151.05             7.66
```

5.4 Heteroscesdasticity-consistent standard errors.

The `estimatr` package allows you to directly calculate robust standard errors.

R Studio allows you to install packages in the Files/Plots/Packages/Help Pane by clicking on the Install icon on the Packages tab; as you type the name of the package, you will see completion suggestions. Choose the package you wish to install and R Studio will install it. You can load a package by checking the box next to its name in the Packages tab. Clicking on the packages name will bring up info about the pacakge.

Call `lm_robust()` to estimate an OLS model with robust standard errors with the `se_type = "HC0` option for the most common method of generating robust standard errors.

```
library(estimatr)
ols_robust <- lm_robust(weight ~ donuts_per_week, donuts, se_type = "HC0")
ols_robust %>%
  tidy()
```

```
             term estimate std.error statistic    p.value conf.low
1     (Intercept)   121.61     15.87      7.66 0.00000983    86.68
2 donuts_per_week     9.22      1.02      9.07 0.00000194     6.99
  conf.high df outcome
1     156.5 11  weight
2      11.5 11  weight
```

5.5 Generating Random Numbers

Random numbers can be useful in variety of applications in econometrics. One application is simulation, where we simulate observations to demonstrate properties of OLS estimators, eg. Once you've decided the distribution from which your random numbers will be drawn and the number of draws you wish to make, you will create a vector of those observations. The most intuitive form of random number generation is `sample`. Suppose you wanted to simulate the role of a single die, use `sample(1:6,1)` or using the pipe operator `1:6 %>% sample(1)`. Read the command aloud like this "from the integers 1, 2, 3, 4, 5, 6, choose a sample of size 1." You can choose larger samples by changing the size argument. The size argument can not be larger than the number of integers unless the default option of replace = FALSE, is changed to replace = TRUE. To generate a simulation of 100 rolls of a single die call `1:6 %>% sample(100, replace = TRUE)`.

Random numbers may be generate from any probability distribution. The random number generator function for a given probability distribution begins with the letter r followed by the name of the distribution in r. To generate uniform random numbers between 0 and 1, use `runif`, from a normal distribution use `rnorm`, etc. Use `args(distribution name)` or `?distribution name` to find out more about the necessary arguments for individual distributions.

5.6 Simulations

Monte Carlo simulations are a useful tool for understanding how the value of an estimator changes as the sample data changes. Consider the example of rolling a single die n times and calculating the average number of pips on the side up face of the die. We know that $\bar{X}$ is an ubiased estimator of μ. Recall that an estimator, $\hat{\theta}$ is unbiased if $E(\hat{\theta}) = \theta$. We can show that $E(\bar{X}) = \mu$. Let

$$\bar{X} = \frac{\sum x_i}{n}$$

Then,

$$\begin{aligned} E(\bar{X}) &= E\left(\frac{\sum x_i}{n}\right) \\ &= \frac{1}{n}\sum E(x_i) \\ &= \frac{1}{n}\sum \mu \\ &= \frac{1}{n} n\mu \\ &= \mu \end{aligned}$$

So, we would expect $\bar{X} = 3.5$ since $\mu = 3.5$. Simulating 100 rolls of a single die 1000 times would allow us to look at the sampling distribution of the sample mean. This will allow us to see the range of values that $\bar{X}$ might take on.

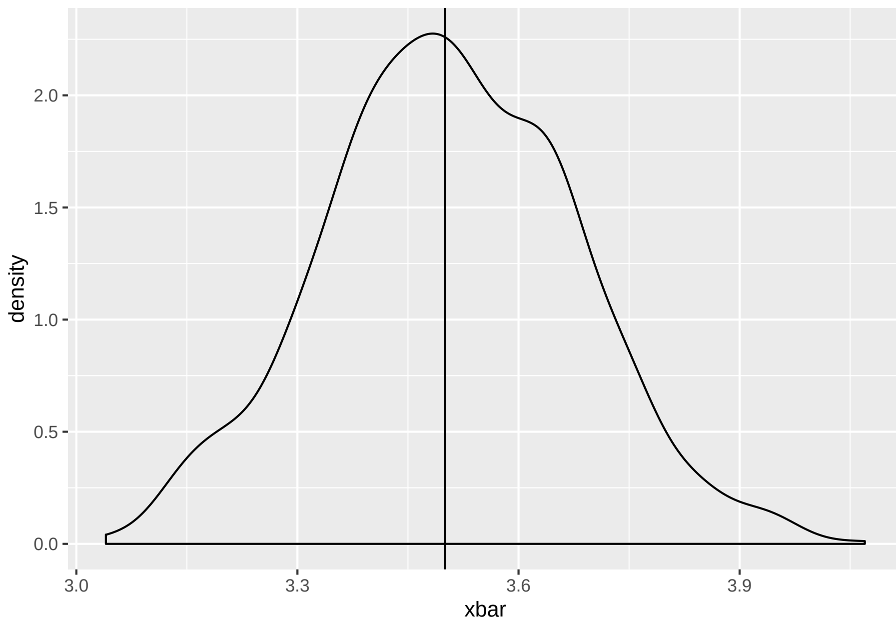
Perform a Monte Carlo simulation by generating many samples, find the value of the estimator, and investigate it's distribution. We could do this by generating a single sample, calculating the value of the estimator, and repeating the desired number of times. This would be tedious. We can instead make use of the concept of a loop in R. A loop evaluates the same code repeatedly until some threshold is met.

There are two types of loops in R, for loops and while loops. A for loop runs the code a specific number of times; a while loop runs the code until a logical condition is met. We will use a for loop to run our simulation. First, instruct R on the number of times to run through the loop. The loop itself is contained between the braces {}.

```
# library(tidyverse)
xbar <- 1 # initialize the vector to store the observations of x bar
for(i in 1:1000) {
  x <- 1:6 %>% sample(100, replace = T)
  xbar[i] <- mean(x)
}
xbar %>%
  mean() # find the mean of the 1000
```

```
[1] 3.51
```

```
xbar %>%
  as.data.frame() %>% # coerce xbar to a data frame
  ggplot(aes(x = xbar)) + # map xbar to x
  geom_density() + # geom_density creates a "probability distribution"
  geom_vline(xintercept = 3.5) # place a verticle line at the mean.
```



We could do the same thing with the simple linear regression $Y_i = \beta_0 + \beta_1 X_i + \epsilon_i$. We know the OLS estimator of β_1 is $\hat{\beta}_1$. The value of the estimator, called the estimate, depends upon the particular sample that is drawn. Monte Carlo simulation will allows to see how the estimate changes across many samples.

For $\hat{\beta}_j$ to be an unbiased esitmator of β_j, $E(\hat{\beta}_j) = \beta_j$. The proof is beyond the scope of this manual, but you will see or have seen the proof.

Suppose we perform a Monte Carlo simulation with know values of β_0 and β_1 where the error term ϵ_i is drawn from a normal distribution with a mean of zero and a constant variance, i.e., $\epsilon_i\ N(0, \sigma^2)$, will the estimates be statistically the same as the known parameters. Let's find out. Suppose the population regression function is $y_i = 10 + 3x_i$,

```
n <- 50
N <- 1000 # of simulations
beta_0 <- 10
beta_1 <- 3
beta_hat_0 <- 0
```

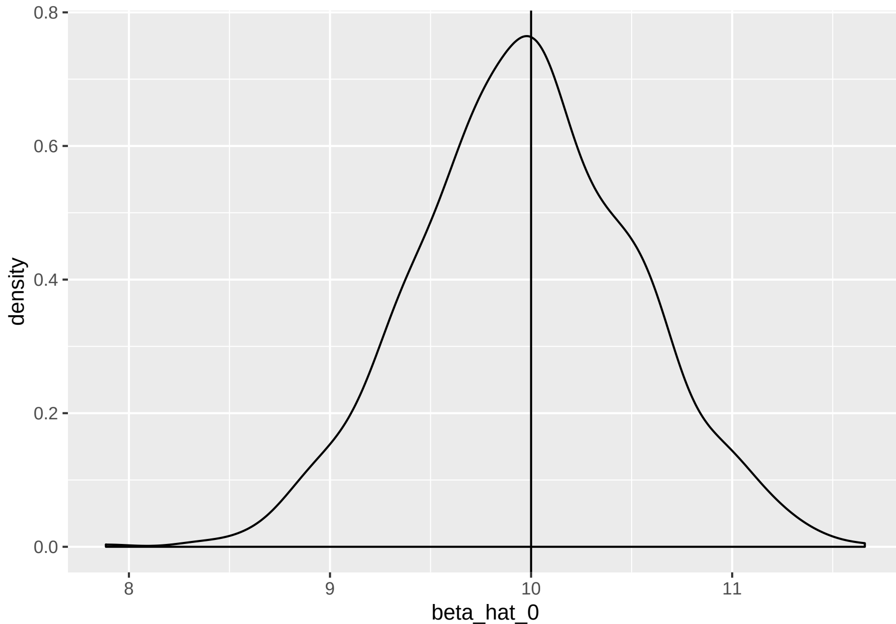
```
beta_hat_1 <- 0
y <- 0
x <- 1:10 %>% sample(n, replace = T) # we would determine x here if x were fixed in re
for(i in 1:N) {
 x <- 0:10 %>% sample(n, replace = T)
 epsilon <- rnorm(n, 0 , 2)
 y <-  beta_0 + beta_1*x + epsilon
 beta_hat_0[i] <- lm(y ~ x)$coef[1]
 beta_hat_1[i] <- lm(y ~ x)$coef[2]
}
#
beta_hat_0 %>%
  mean()
```

```
[1] 9.98
```

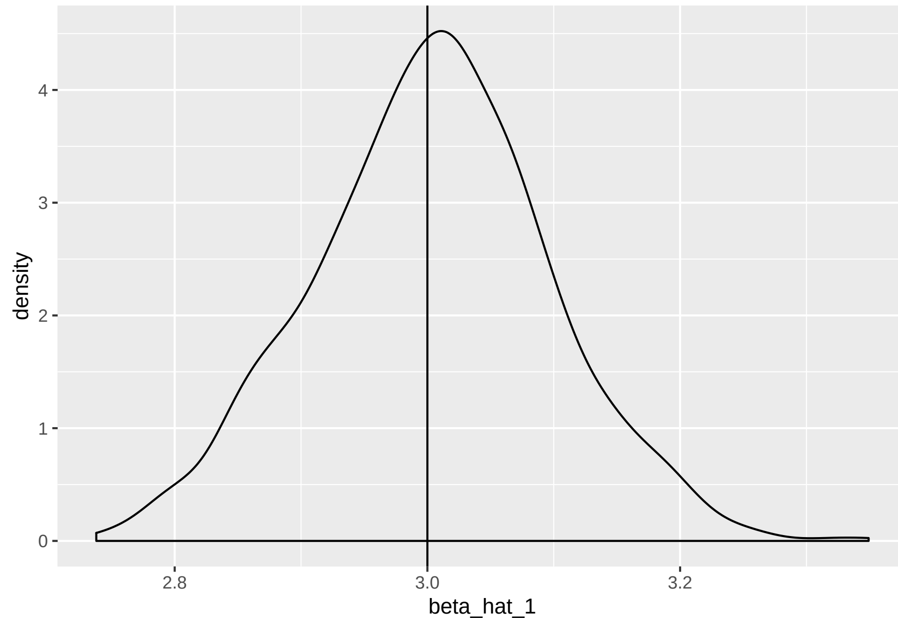
```
beta_hat_1 %>%
  mean()
```

```
[1] 3
```

```
beta_hat_0 %>%
as.data.frame() %>%
ggplot(aes(x = beta_hat_0)) +
geom_density() +
geom_vline(xintercept = 10)
```




```
beta_hat_1 %>%
as.data.frame() %>%
ggplot(aes(x = beta_hat_1)) +
geom_density() +
geom_vline(xintercept = 3)
```



density
4
3
2
1
0
2.8
3.0
3.2
beta_hat_1

Chapter 6

Hypothesis Testing and Interval Estimation; Answering Research Questions

6.1 Computing Corner

We will learn the basisc for hypothesis testing in R.

6.1.1 Probability Distributions in R

For every probability distribution there are four commands. These command for each distribution are prepended by a letter to indicate the functionality.

- "d" returns the height of the probability "d"ensity function
- "p" returns the cummulative density function or the "p"robability of being being between two values of the random variable.
- "q" returns the inverse density function or the value of the random variable ("q"uantile) given a probability.
- "r" returns a "r"andomly generated number from the probability distribution

The distributions you are most likely to encounter in econometrics are the normal (norm), the F distribution (f), the chi-square distribution (chisq), and Student's t-distribution (t). Others include the uniform (unif), binomial (binom), Poisson (pois), etc. Use of the help tab in the Files/Plots/Packages/Help pane or use of `args` will list the arguments necessary to extract value for each distribution.

6.1.2 Critical Values in R

To calculate critical values to perform a hypothesis test use the "q" version of the probability distribution. This will return the quantile for the given probability. The probability under the curve will be cummulative from $-\infty$ to the quantile returned. The "q" version will return the critical value for a one-tail test. Suppose you'd like to test the following hypothesis about μ:

$$H_0 : \mu = 0$$
$$H_1 : \mu < 0$$

at the $\alpha = .05$ level of significance. To calculate the critical t-stastic call `qt(p = .05, df = n-1)`. You know from `args(qt)` the default value of the argument lower.tail is TRUE. Suppose, instead, you'd like to test the following hypothesis about μ

$$H_0 : \mu = 0$$
$$H_1 : \mu > 0$$

at the $\alpha = .10$ level of significance. You can call `qt` in two ways:

1. `qt(p = .10, df = n-1, lower.tail = FALSE)` or
2. `qt(p = .90, df = n-1)`

Finally, suppose you'd like to test the following hypothesis about μ

$$H_0 : \mu = 0$$
$$H_1 : \mu \neq 0$$

at the $\alpha = .01$ level of significance. Since the t-distribution is symmetric you can use the lower tail or upper tail value and -1 times it. You can call `qt` in three ways:

1. `qt(p = .005, df = n-1)` or
2. `qt(p = .005, df = n-1, lower.tail = FALSE)` or
3. `qt(p = .995, df = n-1)`

You can find crtical values for the normal, F, and χ^2 distributions with similar function calls.

6.1.2.1 *p* values in R

To calculate p values in R, use the "p" version of the distribution call. So suppose we test the following hypothesis:

$$H_0 : \sigma_1^2 = \sigma_2^2$$
$$H_0 : \sigma_1^2 \neq \sigma_2^2$$

at the $\alpha = .05$ level of significance. We could use an F test of the form

$$F = \frac{s_x^2}{s_y^2}$$

where s_x^2 and s_y^2 are the sample variances with n-1 and m-1 degrees of freedom. To calculate the p value, call `pf(F, n-1, m-1)` where F is the value calculated above.

6.1.3 Confidence Intervals for OLS estimates

In addition to `confint()`, `confint_tidy()` from the broom package will create a tibble of the low and high values for each estimate. The default level of confidence is 95%.

6.1.4 Power Curves

The power curve represents the probability of making Type II error under alternative null hypotheses. We can generate the power of the test with the `pwr.norm.test(d = NULL, n = NULL, sig.level =.05, power = NULL, alternative = c("two-sided", "less", "greater"))` call from the `pwr` package and plot the power with `ggplot`. To estimate the power we need the effect size $d = \beta_i - \beta$ where β is the hypothesised paramater. We will use

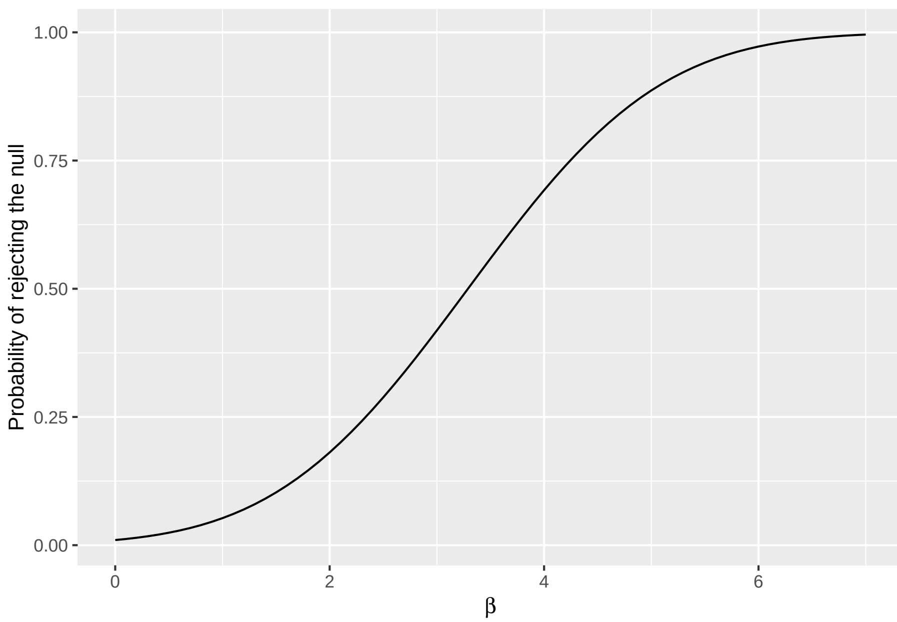
$$H_0 : \beta = 0$$

$$H_1 : \beta > 0$$

The β_i represent alternative null hypothseses for β. Let's let $0 < beta < 7$. Let the significance level be $\alpha = .01$ and $se_\beta = 1$.

```
library(tidyverse)
library(pwr)

beta_i <- seq(0, 7, .1)
```

```
se_beta <- 1 # to keep se_beta = 1 we will set n = 1 below.

pwr <- pwr.2p.test(beta_i, n = 1, sig.level = .01, alternative = "greater")

#the output is a list we need to extract, h and power from pwr

data <- tibble(beta = pwr$h, power =  pwr$power)

data %>%
ggplot(aes(x = beta, y = power)) +
  geom_line() +
  ylab("Probability of rejecting the null") +
  xlab(expression(beta))
```



Chapter 7

Multivariate OLS: Where the Action Is

7.1 Computing Corner

Packages needed for this chapter.

```
library(magrittr)
library(car)
library(broom)
library(estimatr)
library(lm.beta)
```

In this chapter you will learn the basics of estimating multivariate OLS models.

7.1.1 Multiple Regression

To estitmate a multiple regression (a regression with more than one independent variable) use the same function `lm` but change the formula argument to include the additional variables. In a simple regression, the formula argument was of the form `y ~ x`. In a multiple regression, the formula argument takes the form `y ~ x1 + x2`. To include additional variables, extend the argument in a similar manner `y ~ x1 + x2 + x3 + ...`. The remaining arguments are the same as in the simple regression. You can assign the results to an object just as with a simple regression. The output will be the list of 12, but the objects in the list will change to reflect the additional variable(s).

To make use of the results, you can use any of the functions described in Chapter 3 of this manual. You can also make use of any of the subsetting commands as well.

Estimate a regression with robust standard errors with `lm_robust` with the modified function argument.

7.1.2 Multicollinearity

You can directly estimate the VIF's with the `vif()` function from the car package. To esitmate the VIF's call `ols %>% vif()` where ols is the object you created with the `lm` call.

7.1.3 Standardized Coefficients

Estimate standardized regression coefficients with `lm.beta()` from the lm.beta package. `ols %>% lm.beta()`.

7.1.4 *F* tests

F tests in econometrics are generally about the joint significance of multiple variables. Suppose, we estimate the regression on $i = 1, 2, \ldots n$ observations.

$$y_i = \beta_0 + \beta_1 x_{1,i} + \beta_2 x_{2,i} + \cdots + \beta_m x_{i,m} + \beta_{m+1} x_{m+1,i} + \cdots + \beta_k x_{i,k} + \epsilon_i$$

To test the joint significance of the $\beta_1, \ldots, \beta_m$ in the model we would use an F test to perform the following hypothesis test:

$$H_0 : \beta_1 = \beta_2 = \cdots = \beta_m = 0$$

$$H_1 : \text{@ least one } \beta_j \neq 0$$

An F test essentially compares the difference in the residual sum of squares under the null and alternative hypotheses. If this difference in large enough relative to the unrestricted standard error, we have evidence to reject the null hypothesis in favor of the alternative hypothesis. The mechanics of the test are as follows:

1. Estimate the model that does not hold under the null hypothesis, that is, the model above and call it the unsrestricted model and retrieve the residual sum of squares. Retrieve the residual sum of squares, rss_u. The residuals from unrestricted model will have $n - k - 1$ degrees of freedom. The unrestricted model, U, is:

   U: $y_i = \beta_0 + \beta_1 x_{1,i} + \beta_2 x_{2,i} + \cdots + \beta_m x_{i,m} + \beta_{m+1} x_{m+1,i} + \cdots + \beta_k x_{i,k} + \epsilon_i$

2. Estimate the model that holds under the null hypothesis Restrict the model so that the null hypothesis holds. That restricted model, R, is

$$\text{R: } y_i = \beta_0 + \beta_{m+1}x_{m+1,i} + \beta_{m+2}x_{m+2,i} + \cdots + \beta_k x_{k,i} + \eta_i$$

. Retrieve the residual sum of squares rss_r The residual from restricted model will have $n - m - 1$ degrees of freedom.

3. Calculate the difference in the residual sum of squares $rss_r - rss_u$ and divide by its degrees of of freedom $q = (n - m - 1) - (n - k - 1) = k - m$. So, q is the number of restrictions. A simple way to calculate the number of restrictions is to count the number of equal signs, =, in the null hypothesis.

4. Calculate $rss_u/(n - k - 1)$

5. Divide the result from 3 by the result from 4. This will give you an F statistic with k-m and n-k-1 degrees of freedom.

$$F_c = \frac{\frac{rss_r - rss_u}{q}}{\frac{rss_u}{n-k-1}}$$

The F-test (Wald test) can be used for any number of restrictions on the unrestricted model. For example, suppose we would like to know if a production function with a Cobb-Douglas form has constant returns to scale. The Cobb-Douglas function for output as a function of labor and capital takes the form

$$q = al^{\alpha}k^{\beta}\epsilon$$

. If constant returns to scale hold, $\alpha + \beta = 1$. So we test the following hypothesis:

$$H_0 : \alpha + \beta = 1$$

$$H_1 : \alpha + \beta \neq 1$$

To test this hypothesis form the unrestricted and restricted forms of the model, estimate the models, retrieve the sum of squared residuals, and calculate the F statistic. In the form presented above, the Cobb-Douglas model is not linear in the parameters so it can't be estimated with OLS. We can make it linear in the parameters by taking the logarithm of both sides.

$$\ln(q) = \ln(al^{\alpha}k^{\beta}\epsilon)$$

$$\text{U: } \ln(q) = \gamma + \alpha \ln(l) + \beta \ln(k) + \eta$$

.

Form the restricted model by emposing the null hypothesis on the paramaters. From the null hypothesis, $\beta = 1 - \alpha$. Substituting for β in the restricted model yields the restricted model.

$$\text{R: } \ln(q) - \ln(k) = \gamma + \alpha[\ln(l) - \ln(k)] + \eta$$

The F-stat is:

$$F_c = \frac{rss_r - rss_u}{\frac{rss_u}{n-k-1}}$$

The degrees of freedom are $q = 1$ (the number of equal signs in the null hypothesis) and $n - k - 1$.

7.1.4.1 *F*-test for overall significance.

Estimate the model $y_i = \beta_0 + \beta_1 x_{1,i} + \beta_2 x_{2,i} + \cdots + \beta_k x_{k,i} + \epsilon_i$. Test the hypothesis

$$H_0 : \beta_1 = \beta_2 = \cdots = \beta_k = 0$$
$$H_1 : \text{@ least one } \beta_j \neq 0$$

If we reject the null hypothesis, we can say that we have explained some varation in y with variation in at least one of the $x's$. In other words, we have a model that is significant. If we fail to reject the null hypothesis, our model has no explanatory power. There is no need to calculate the F-statistic to perform this test because it is reported as a matter of course in the base R call `summary` or in `glance` from the broom package. The degrees of freedom are $q = k$ (the number of coefficients estimated - 1) and $n - k - 1$.

`summary` will report the F-statistic, its degrees of freedom (numerator and denominator), and the p-value. `glance` reports the F as "statistic", the p-value as "p.value", $k + 1$ as "df", and $n - k - 1$ as "df.residual". Note that this test is also a test for the significance of R^2.

7.1.4.2 *F*-test of linear restrictions

The test we performed above are tests of linear restrictions of the parameters. These hypotheses can be tested directly using `linearHypothesis` from the car package. Performing a test of linear restrictions using `linearHypothesis` requires two arguments: model and hypothesis.matrix.

Let the unrestricted model be

$$y = \beta_0 + \beta_1 x_1 + \beta_2 x_2 + \beta_3 x_3 + \epsilon$$

Estimate the model as `ols_u <- df %$% lm(y ~ x1 + x2 + x3)`, where df is the data frame containing the data.

Let's test the hypothesis $\beta_2 = \beta_3 = 0$ versus at that one of the $\beta's \neq 0$ using `linearHypothesis(model = ols_u, hypothesis.matrix = c("x2 = 0", "x3 = 0")`. The result will be an F-test on the restrictions. The F-statistic, its degrees of freedom, and p-value will be returned.

Let's test the linear restriction for the Cobb-Douglas model above. Estimate the model as `ols_u <- df %$% lm(log(q) ~ log(l) + log(k))`. To test the hypothesis $\alpha = \beta$ pipe ols_u into `linearHypothesis` with the argument $c(log(l) = log(k))$: `ols_u %>% linearHypothesis(c("log(l) = log(k)"))`. Again, the F-statistic, its degrees of freedom, and p-value will be returned.

7.1.5 Examples

The Motor Trend Car Road Test (mtcars) data set is part of the datasets in base R. The data were extracted from the 1974 Motor Trend US magazine, and comprises fuel consumption and 10 aspects of automobile design and performance for 32 automobiles (1973-74 models). See `?mtcars` for more information on the data. `data(mtcars)` will load the data into your global environment as mtcars. We will perform each of the F-tests described above: overall significance, joint signficance of a subset of variables, and equality of two coefficients.

7.1.5.1 Multiple Regression

Suppose we want to estimate the mpg as a function of the number of cylinders, the displacement, and the gross horsepower, then our (unrestricted) model is

$$mpg = \beta_0 + \beta_1 cyl + \beta_2 disp + \beta_3 hp + \epsilon$$

.

Let's estimate the unrestricted model using the expose pipe `%$%` both with and without robust errors.

```
# estimate model without reobust standard errors
ols <- mtcars %$% lm(mpg ~ cyl + disp + hp)
ols %>% tidy()
```

```
# A tibble: 4 x 5
  term        estimate std.error statistic  p.value
  <chr>          <dbl>     <dbl>     <dbl>    <dbl>
1 (Intercept)  34.2       2.59        13.2  1.54e-13
2 cyl          -1.23      0.797       -1.54 1.35e- 1
3 disp         -0.0188    0.0104      -1.81 8.09e- 2
4 hp           -0.0147    0.0147      -1.00 3.25e- 1
```

```
# estimate model with robust standard erros
ols_robust <- mtcars %$% lm_robust(mpg ~ cyl + disp + hp)
ols_robust %>%  tidy()
```

```
         term estimate std.error statistic           p.value conf.low
1 (Intercept)  34.1849    2.4700     13.84 0.000000000000048  29.1253
2         cyl  -1.2274    0.5967     -2.06 0.049121075438813  -2.4498
3        disp  -0.0188    0.0083     -2.27 0.031138440490781  -0.0358
4          hp  -0.0147    0.0109     -1.34 0.190818678697032  -0.0371
  conf.high df outcome
1  39.24451 28     mpg
2  -0.00506 28     mpg
3  -0.00183 28     mpg
4   0.00775 28     mpg
```

7.1.5.2 Multicollinearity

Using the model above

$$mpg = \beta_0 + \beta_1 cyl + \beta_2 disp + \beta_3 hp + \epsilon$$

.

We can calcualte the VIF's as follows:

```
ols %>% vif()
```

```
 cyl disp   hp
6.73 5.52 3.35
```

```
ols_robust %>% vif()
```

```
 cyl disp   hp
3.67 2.90 2.71
```

7.1.5.3 Standardize Regression Coefficients

Using the model

$$mpg = \beta_0 + \beta_1 cyl + \beta_2 disp + \beta_3 hp + \epsilon$$

, estimate standardized regression coefficients as follows:

```r
ols %>% lm.beta()
```

```

Call:
lm(formula = mpg ~ cyl + disp + hp)

Standardized Coefficients::
(Intercept)         cyl        disp          hp 
      0.000      -0.364      -0.387      -0.167 
```

7.1.5.4 *F*-test for Overall significance

Suppose we want to estimate the mpg as a function of the number of cylinders, the displacement, and the gross horsepower, then our (unrestricted) model is

$$mpg = \beta_0 + \beta_1 cyl + \beta_2 disp + \beta_3 hp + \epsilon$$

.

Let's estimate the unrestricted model using the expose pipe `%$%`

```r
ols_u <- mtcars %$% lm(mpg ~ cyl + disp + hp)
```

The test for overall significance is:

$$H_0 : \beta_1 = \beta_2 = \beta_3 = 0$$

$$H_1 : @ \text{ least one } \beta_j \neq 0$$

Recall that *the* F-test is reported as a matter of course in `summary` from base R and `glance` from the broom package.

```r
ols_u %>% summary()
```

```

Call:
lm(formula = mpg ~ cyl + disp + hp)

Residuals:
   Min     1Q Median     3Q    Max 
-4.089 -2.085 -0.774  1.397  6.918 

Coefficients:
            Estimate Std. Error t value         Pr(>|t|)
```

```
(Intercept)  34.1849     2.5908   13.19 0.00000000000015 ***
cyl          -1.2274     0.7973   -1.54            0.135    
disp         -0.0188     0.0104   -1.81            0.081 .  
hp           -0.0147     0.0147   -1.00            0.325    
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

Residual standard error: 3.06 on 28 degrees of freedom
Multiple R-squared:  0.768, Adjusted R-squared:  0.743 
F-statistic: 30.9 on 3 and 28 DF,  p-value: 0.00000000505
```

```r
ols_u %>% glance()
```

```
# A tibble: 1 x 11
  r.squared adj.r.squared sigma statistic p.value    df logLik   AIC   BIC
      <dbl>         <dbl> <dbl>     <dbl>   <dbl> <int>  <dbl> <dbl> <dbl>
1     0.768         0.743  3.06      30.9 5.05e-9     4  -79.0  168.  175.
# ... with 2 more variables: deviance <dbl>, df.residual <int>
```

So we see that $F = 30.877$, $q = 3$, and $df2 = 28$. The critical F with $\alpha = .05$ is 2.947. Since the calculated F-stat is greater than the critical F-stat, we reject H_0 in favor of H_1. That is, the explanatory power of the model is statistical significant.

7.1.5.5 *F*-test of Joint Significance

Suppose we'd like to add the weight (wt), number of gears (gear), and number of carburetors (carb) together increase the explanatory power of the model at the $\alpha = .05$, level of significance. Our unrestricted model becomes:

$$mpg = \beta_0 + \beta_1 cyl + \beta_2 disp + \beta_3 hp + \beta_4 wt + \beta_5 gear + \beta_6 carb + \eta$$

.

The null and alternative hypotheses are:

$$H_0 : \beta_4 = \beta_5 = \beta_6 = 0$$

$$H_1 : @ \text{ least one } \beta_j \neq 0$$

7.1.5.6 Perform the test "manually"

```
# estimate the unrestricted model
ols_u <- mtcars %$% lm(mpg ~ cyl + disp + disp + hp + wt + gear + carb)
# generate the residual sum of squares
rss_u <- ols_u$residuals^2 %>%
  sum()
# retrive the degrees of freedom for the unrestricted model
df_u <- ols_u$df.residual
# estimate the restricted model
ols_r <- mtcars %$% lm(mpg ~ cyl + disp + disp + hp)
# generate the residual sum of squares
rss_r <- ols_r$residuals^2 %>%
  sum()
# retrive the degrees of freedom for the restricted model
df_r <- ols_r$df.residual
# calculate the number of restrictions
q <- df_r - df_u
# calculate F
(F_stat <- ((rss_r-rss_u)/q)/(rss_u/df_u)) # () around the call prints the result to the screen
```

```
[1] 4.8
```

```
# retrieve the critical F
(F_crit <- qf(.95, q, df_u))
```

```
[1] 2.99
```

```
# p-value
(F_stat %>% pf(q, df_u, lower.tail = F))
```

```
[1] 0.00897
```

Since 4.796 is greater than 2.991 we can reject H_0 in favor of H_1 and conclude that wt, am, and carb add signficant explanatory power to the model. We can also see that the p-vallue for our calculated F-statistic is 0.009. Since this is less than $\alpha = .05$ we reject H_0.

7.1.5.7 Perform the test with `linearHypothesis`

```
linearHypothesis(ols_u, c("wt", "gear", "carb"))
```

```
Linear hypothesis test

Hypothesis:
wt = 0
gear = 0
carb = 0

Model 1: restricted model
Model 2: mpg ~ cyl + disp + disp + hp + wt + gear + carb

  Res.Df RSS Df Sum of Sq   F Pr(>F)
1     28 261
2     25 166  3      95.5 4.8  0.009 **
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

Of course, we have the same result.

7.1.5.8 Test of Linear Restrictions

Let the model be

$$\ln(mpg) = \beta_0 + \beta_1 \ln(cyl) + \beta_2 \ln(wt) + \epsilon$$

. Suppose we'd like to test

$$H_0 : \beta_1 + \beta_2 = -1$$

against

$$H_0 : \beta_1 + \beta_2 \neq -1$$

7.1.5.8.1 Perform the Test "Manually"

Form the restricted model under H_0. If H_0 holds, $\beta_2 = -1 - \beta_1$. Substituting into the unrestricted model yields the restricted model:

$$\text{R: } \ln(mpg) + \ln(wt) = \beta_0 + \beta_1(\ln(cyl) - \ln(wt)) + \eta$$

```r
# estimate the unrestricted model
ols_u <- mtcars %$% lm(log(mpg) ~ log(cyl) + log(wt))
# generate the residual sum of squares
rss_u <- ols_u$residuals^2 %>%
  sum()
# retrive the degrees of freedom for the unrestricted model
df_u <- ols_u$df.residual
```

```r
# estimate the restricted model
ols_r <- mtcars %$% lm(I(log(mpg)+log(wt)) ~ I(log(cyl) - log(wt)))
# generate the residual sum of squares
rss_r <- ols_r$residuals^2 %>%
  sum()
# retrive the degrees of freedom for the restricted model
df_r <- ols_r$df.residual
# calculate the number of restrictions
q <- df_r - df_u
# calculate F
(F_stat <- ((rss_r-rss_u)/q)/(rss_u/df_u)) # () around the call prints the result to the screen
```

```
[1] 1.29
```

```r
# retrieve the critical F
(F_crit <- qf(.95, q, df_u))
```

```
[1] 4.18
```

```r
# p-value
(F_stat %>% pf(q, df_u, lower.tail = F))
```

```
[1] 0.266
```

Since 1.289 is less than 4.183 we can failt to reject H_0 and conclude that we have no evidence to suggest that $\beta_1 + \beta_2 \neq 1$. We can also see that the p-vallue for our calculated F-statistic is 0.266. Since this is greater than $\alpha = .05$ we fail to reject H_0.

7.1.5.9 Perform the test with `linearHypothesis`

```r
ols_u %>% linearHypothesis(c("log(cyl) + log(wt) = -1"))
```

```
Linear hypothesis test

Hypothesis:
log(cyl)  + log(wt) = - 1

Model 1: restricted model
Model 2: log(mpg) ~ log(cyl) + log(wt)
```

```
  Res.Df   RSS Df Sum of Sq    F Pr(>F)
1     30 0.419
2     29 0.401  1    0.0178 1.29   0.27
```

Chapter 8

Dummy Variables: Smarter than You Think

In this chapter we will learn how R handles dummy variables.

We will need the following libraries.

```
library(tidyverse)
library(magrittr)
library(broom)
library(estimatr)
library(forcats)
```

8.1 Dummy Variables in R

R uses factor vectors to to represent dummy or categorical data. Factors can be ordered or unordered. Factor vectors are built on top of integer vectors and include a unique label for each integer.

8.1.1 Factors

R uses factors to handle categorical variables. Categorical variables have fixed and known set of possible values. The package forcats as part of the tidyverse offers a suite of tools for that solve common problems with factors. See the vignette on forcats for more information on the forcats package to learn more about using factors in R.

8.1.2 Character Vectors as Dummies

Character vectors are one of the six atomic vector types in R. *Atomic* means that the vector contains only data of a single type, in this case all of the observations are characters. Categorical data or dummy variables though they are typically coded as numeric are character vectors. For example, a dummy varialbe for sex may contain male and female, but be coded as 0 and 1 and named female. If you use a character vector as an argument in `lm`, R will treat the vector as a set of dummy variables. The number of dummy variables will be the number of characteristics (unique observations) minus 1.

The student admissions at UC Berkeley data set has aggregate data on graduate school applicants for the six largest departments, `?UCBAdmissions` for more information. There are four variables in the data set, Admit (whether the cadidate was admitted or rejected), Gender (the gender of the candidate: Male or Female), Dept (department to which the candidate applied coded as A, B, C, D, E, F), and n (the number of applicants). n is a numeric vector. Admit, Gender, and Dept, are character vectors. Since the data are store as a table, to read them into R as a data frame call `as_tibble` from the dplyr package with the argument UCBAdmissions.

```
UCB_Admissions <- UCBAdmissions %>%
  as_tibble() %>%
  glimpse()
```

```
Observations: 24
Variables: 4
$ Admit  <chr> "Admitted", "Rejected", "Admitted", "Rejected", "Admitt...
$ Gender <chr> "Male", "Male", "Female", "Female", "Male", "Male", "Fe...
$ Dept   <chr> "A", "A", "A", "A", "B", "B", "B", "B", "C", "C", "C", ...
$ n      <dbl> 512, 313, 89, 19, 353, 207, 17, 8, 120, 205, 202, 391, ...
```

Suppose we wisht to estimate the difference in difference model $n_i = \beta_0 + \beta_1 Admit_i + \epsilon_i$. If we use Admit as an argument in `lm`, R will correctly treat Admit as single dummy variable with two categories.

```
UCB_Admissions %$%
  lm(n ~ Admit)
```

```
Call:
lm(formula = n ~ Admit)

Coefficients:
  (Intercept)  AdmitRejected
        146.2           84.7
```

R has coded Rejected as 1 and Admitted as 0. The regression indicates that mean of admitted is 146.25 while the mean number rejected is 230.92. We can confirm that directly as well.

```
# Using dplyr verbs
UCB_Admissions %>%
  filter(Admit == "Admitted") %>%
  summary(mean)
```

```
    Admit              Gender              Dept                 n
 Length:12          Length:12          Length:12          Min.   : 17
 Class :character   Class :character   Class :character   1st Qu.: 46
 Mode  :character   Mode  :character   Mode  :character   Median :107
                                                          Mean   :146
                                                          3rd Qu.:154
                                                          Max.   :512
```

```
UCB_Admissions %>%
  filter(Admit == "Rejected") %>%
  summary(mean)
```

```
    Admit              Gender              Dept                 n
 Length:12          Length:12          Length:12          Min.   :  8
 Class :character   Class :character   Class :character   1st Qu.:188
 Mode  :character   Mode  :character   Mode  :character   Median :262
                                                          Mean   :231
                                                          3rd Qu.:314
                                                          Max.   :391
```

```
# Directly selecting observations based on other values
UCB_Admissions %$%
  mean(n[Admit == "Admitted"])
```

```
[1] 146
```

```
UCB_Admissions %$%
  mean(n[Admit == "Rejected"])
```

```
[1] 231
```

Similarly, if we want to calculate the mean number of applicants by department, R will treat Dept as 5 dummy variables.

```
UCB_Admissions %$%
  lm(n ~ Dept)
```

```
Call:
lm(formula = n ~ Dept)

Coefficients:
(Intercept)        DeptB        DeptC        DeptD        DeptE  
     233.25       -87.00        -3.75       -35.25       -87.25  
      DeptF  
     -54.75  
```

The mean number of applicants in Department A is 233.25. To find the mean number of applicants for each department add the appropriate coefficient to 233.25.

We can confirm these results as we did above.

8.2 Difference in Means Test

Using the UCB Admissions data, let's conduct a difference of means test for number of applications by Gender. We will test the following hypothesis:

$$H_0 : \mu_{Male} = \mu_{Female} H_1 : \mu_{Male} \neq \mu_{Female}$$

at the $\alpha = .05$ level of significance. We can use `t.test` in two different ways, `lm`, or `lm_robust`. First, we will test the hypothesis with `t.test` assuming, in turn, equal and unequal variances.

8.2.1 Using `t.test`

```
# Assume equal variances
# Use t.test default method
UCB_Admissions %$%
  t.test(n[Gender == "Female"], n[Gender == "Male"], var.equal = TRUE)
```

```
    Two Sample t-test

data:  n[Gender == "Female"] and n[Gender == "Male"]
```

```
t = -1, df = 22, p-value = 0.2
alternative hypothesis: true difference in means is not equal to 0
95 percent confidence interval:
 -188.4   45.7
sample estimates:
mean of x mean of y
      153       224
```

```
# Use t.test for class 'formula`
UCB_Admissions %$%
  t.test(n ~ Gender, var.equal = TRUE)
```

```

    Two Sample t-test

data:  n by Gender
t = -1, df = 22, p-value = 0.2
alternative hypothesis: true difference in means is not equal to 0
95 percent confidence interval:
 -188.4   45.7
sample estimates:
mean in group Female   mean in group Male
                 153                  224
```

```
# Assume unequal variances
# unequal variances is the default
UCB_Admissions %$%
  t.test(n[Gender == "Female"], n[Gender == "Male"])
```

```

    Welch Two Sample t-test

data:  n[Gender == "Female"] and n[Gender == "Male"]
t = -1, df = 22, p-value = 0.2
alternative hypothesis: true difference in means is not equal to 0
95 percent confidence interval:
 -188.4   45.8
sample estimates:
mean of x mean of y
      153       224
```

```
# Use t.test for class 'formula`
UCB_Admissions %$%
  t.test(n ~ Gender)
```

```
	Welch Two Sample t-test

data:  n by Gender
t = -1, df = 22, p-value = 0.2
alternative hypothesis: true difference in means is not equal to 0
95 percent confidence interval:
 -188.4   45.8
sample estimates:
mean in group Female   mean in group Male
                 153                  224
```

8.2.2 Using `lm` and `lm_robust`

```
# Assume equal variances
UCB_Admissions %$%
  lm(n ~ Gender) %>%
  tidy()
```

```
# A tibble: 2 x 5
  term        estimate std.error statistic  p.value
  <chr>          <dbl>     <dbl>     <dbl>    <dbl>
1 (Intercept)    153.       39.9      3.83 0.000911
2 GenderMale      71.3      56.5      1.26 0.220
```

```
# Assume unequal variances
UCB_Admissions %$%
  lm_robust(n ~ Gender) %>%
  tidy()
```

```
         term estimate std.error statistic  p.value conf.low conf.high df
1 (Intercept)    152.9      38.7      3.95 0.000679     72.7       233 22
2  GenderMale     71.3      56.5      1.26 0.219606    -45.7       188 22
  outcome
1       n
2       n
```

8.3 Integer and Numerical Vectors as Dummy Variables

`lm` treated the character vectors as factors. For most of what we will do, that is enough. If the categorical (dummy) variable is coded as a numeric vector or integer vector, we my have coerce the variable to a factor for `lm` to interpret it correctly. If the variable is coded as 0 and 1, we can use it as it is. For example, conisder the the mtcars data.

```
data(mtcars)
mtcars %>% glimpse()
```

```
Observations: 32
Variables: 11
$ mpg  <dbl> 21.0, 21.0, 22.8, 21.4, 18.7, 18.1, 14.3, 24.4, 22.8, 19....
$ cyl  <dbl> 6, 6, 4, 6, 8, 6, 8, 4, 4, 6, 6, 8, 8, 8, 8, 8, 8, 4, 4, ...
$ disp <dbl> 160.0, 160.0, 108.0, 258.0, 360.0, 225.0, 360.0, 146.7, 1...
$ hp   <dbl> 110, 110, 93, 110, 175, 105, 245, 62, 95, 123, 123, 180, ...
$ drat <dbl> 3.90, 3.90, 3.85, 3.08, 3.15, 2.76, 3.21, 3.69, 3.92, 3.9...
$ wt   <dbl> 2.62, 2.88, 2.32, 3.21, 3.44, 3.46, 3.57, 3.19, 3.15, 3.4...
$ qsec <dbl> 16.5, 17.0, 18.6, 19.4, 17.0, 20.2, 15.8, 20.0, 22.9, 18....
$ vs   <dbl> 0, 0, 1, 1, 0, 1, 0, 1, 1, 1, 1, 0, 0, 0, 0, 0, 0, 1, 1, ...
$ am   <dbl> 1, 1, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 1, ...
$ gear <dbl> 4, 4, 4, 3, 3, 3, 3, 4, 4, 4, 4, 3, 3, 3, 3, 3, 3, 4, 4, ...
$ carb <dbl> 4, 4, 1, 1, 2, 1, 4, 2, 2, 4, 4, 3, 3, 3, 4, 4, 4, 1, 2, ...
```

The type of transmission, am, takes on two values 1 if the transmission is automatic and 0 if it is manual. Suppose we'd like to know if the mpg is different for the two types of transmissions. We can test the hypothesis

$$H_0 : \mu_a = \mu_m$$

$$H_1 : \mu_a \neq \mu_m$$

d at the $\alpha = .05$ level of significance.

```
mtcars %$%
  lm_robust(mpg ~ am) %>%
  tidy()
```

```
         term estimate std.error statistic               p.value conf.low
1 (Intercept)    17.15      0.88     19.50 0.0000000000000000138    15.35
2          am     7.24      1.92      3.77 0.00072109506857981581     3.32
  conf.high df outcome
1      18.9 30     mpg
2      11.2 30     mpg
```

If, however, the categorical variable is not coded as 0 and 1, we will have to coerce it to a factor. The forcats package simplifies this process. Suppose we'd like to know if the average mpg is different for 4, 6, and 8 cylinder cars.

$$H_0 : \mu_4 = \mu_6 = \mu_8$$

$$H_1 : @ \text{ least one } \mu \text{ is not equal}$$

If we estimate a model of mpg on cyl, the coefficient on cyl will give us the marginal effect on mpg of adding a cylinder. A signficant coefficient in this model will not answer our question. To do that, we must coerce cyl into a categorical variable with `as.factor`.

```
mtcars %$%
  lm(mpg ~ as.factor(cyl)) %>%
  summary()
```

```
Call:
lm(formula = mpg ~ as.factor(cyl))

Residuals:
   Min     1Q Median     3Q    Max 
-5.264 -1.836  0.029  1.389  7.236 

Coefficients:
                Estimate Std. Error t value             Pr(>|t|)    
(Intercept)       26.664      0.972   27.44 < 0.0000000000000002 ***
as.factor(cyl)6   -6.921      1.558   -4.44              0.00012 ***
as.factor(cyl)8  -11.564      1.299   -8.90        0.00000000086 ***
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

Residual standard error: 3.22 on 29 degrees of freedom
Multiple R-squared:  0.732, Adjusted R-squared:  0.714 
F-statistic: 39.7 on 2 and 29 DF,  p-value: 0.00000000498
```

The *F-stat* for overall significance of the model is significant at the $\alpha = .05$ level of significance so we reject the null hypothesis in favor of the alternative and conclude that at least one average mpg is different.

The base case is cars with 4 cylinders with an average mpg of 26.7 mpg. 6 cylinder cars average a statistically significant 6.9 mpg less than 4 cylinder cars. 8 cylinder cars average a statistically siginficant 11.6 mpg less than 4 cylider cars. These averages are statistically signficantly different.

Had we estimated the model without coercing cylinders into a factor our results would have been

```
mtcars %$%
  lm(mpg ~ cyl) %>%
  tidy()
```

```
# A tibble: 2 x 5
  term        estimate std.error statistic  p.value
  <chr>          <dbl>     <dbl>     <dbl>    <dbl>
1 (Intercept)    37.9      2.07      18.3  8.37e-18
2 cyl            -2.88     0.322     -8.92 6.11e-10
```

$\hat{\beta}_1 = -2.88$ tells us that for each additional cylinder fuel mileage will fall by 2.88 mpg.

8.4 Manipulating Factors

The forcats package provides a set of tools for the simple manipulation of factors like renaming factors, re-ordering factors, combining factors, etc. Using the mtcars data, lets coerce the number of cylinders to a factor and look at ways to manipulate in ways to aid in understanding. The compound pipe operator `%<>%` is used to update a value by first piping into one or more expressions and then assigning the result.

```
data(mtcars)
### Coerce cyl to a factor
mtcars$cyl %<>%
  as.character() %>% # forcats will not coerce integer or numeric vectors to factors
  as_factor()
mtcars$cyl %>% str()
```

```
 Factor w/ 3 levels "6","4","8": 1 1 2 1 3 1 3 2 2 1 ...
```

cyl is now a factor with 3 levels, 6, 4, 8. Suppose we estimate the model $mpg = \beta_0 + \beta_1 mpg + \epsilon$.

```
mtcars %$%
  lm(mpg ~ cyl) %>%
  tidy()
```

```
# A tibble: 3 x 5
  term        estimate std.error statistic  p.value
  <chr>          <dbl>     <dbl>     <dbl>    <dbl>
1 (Intercept)    19.7       1.22     16.2  4.49e-16
2 cyl4            6.92      1.56      4.44 1.19e- 4
3 cyl8           -4.64      1.49     -3.11 4.15e- 3
```

This model indicates that cars with 6 cylinder engines average 19.74 mpg, cars with 4 cylinders average 6.9 mpg more than cars with 6 cylinders, and cars with 8 cylinders average 4.64 mpg less than cars with 6 cylinders. Suppose, instead, you'd prefere 4 cylinder cars to be the base case. We can reorder the factor with `fct_relevel` from the forcats package. `fct_revel` changes the order of a factor by hand.

For some factors the order doesn't or won't matter, for others there is "natural" ordering suggested by the data, for others you may have an ordering that you prefer. `fct_relevel()` from the forcats package handles that task. If we call `fct_relevel` within `lm` the releveling will be *ad hoc*.

```
mtcars %$%
  lm(mpg ~ fct_relevel(cyl, levels = c("4", "6", "8"))) %>%
  tidy()
```

```
# A tibble: 3 x 5
  term                                   estimate std.error statistic  p.value
  <chr>                                     <dbl>     <dbl>     <dbl>    <dbl>
1 (Intercept)                               26.7      0.972     27.4  2.69e-22
2 "fct_relevel(cyl, levels = c(\"4\"~     -6.92     1.56      -4.44 1.19e- 4
3 "fct_relevel(cyl, levels = c(\"4\"~    -11.6      1.30      -8.90 8.57e-10
```

We can permantly relevel cylinders

```
# relevel the factor
mtcars$cyl  <-
  fct_relevel(mtcars$cyl, levels = c("4", "6", "8"))
# alternatively
mtcars$cyl <-
  fct_relevel(mtcars$cyl, "6", after = 1) # move 6 to the second position
# re-estimate the model
mtcars %$%
  lm(mpg ~ cyl) %>%
  tidy()
```

```
# A tibble: 3 x 5
  term        estimate std.error statistic  p.value
  <chr>          <dbl>     <dbl>     <dbl>    <dbl>
1 (Intercept)    26.7      0.972     27.4  2.69e-22
2 cyl6           -6.92     1.56      -4.44 1.19e- 4
3 cyl8          -11.6      1.30      -8.90 8.57e-10
```

See Reorder factor levels by hand for a more ways to relevel factors.

The transmission variable (am) is a numeric vector coded as 0 and 1. Suppose we'd like to coerce it to a factor coded with the levels named "automatic" and "manual" rather than 0 and 1.

```
mtcars$am %<>%
  factor(levels = c(0,1), labels = c("automatic", "manual"))
```

If we re-estimate the model $mpg = \beta_0 + \beta_1 am$ we see the results are the same, but the variable is labeled more clearly.

```
mtcars %$%
  lm_robust(mpg ~ am) %>%
  tidy()
```

```
         term estimate std.error statistic                p.value conf.low
1 (Intercept)    17.15      0.88     19.50 0.00000000000000000138    15.35
2    ammanual     7.24      1.92      3.77 0.00072109506857981581     3.32
  conf.high df outcome
1      18.9 30     mpg
2      11.2 30     mpg
```

8.5 Dummy Interaction Variables

Dummy interactions $x_i D_i$ can be created in lm as an argument. Let's esitmate the the model $mpg = \beta_0 + \beta_1 am + \beta_2 hp + \beta_3 hp * am + \epsilon$.

```
mtcars %$%
  lm(mpg ~ hp*am)
```

```
Call:
lm(formula = mpg ~ hp * am)

Coefficients:
(Intercept)           hp     ammanual  hp:ammanual
  26.624848    -0.059137     5.217653     0.000403
```

Notice that R assumed that you wanted to calculate $\hat{\beta}_1$, $\hat{\beta}_2$, and $\hat{\beta}_3$. By including `hp*am` as an argument in `lm` R estimated the continuous coefficients for the continuous variable, the dummy variable, and the interactions. If, on the other hand, you wanted just the interaction term, i.e., $mpg = \alpha_0 + \alpha_1 hp * am + \eta$, use the "AsIs" function `I()` as follows:

```r
data(mtcars)
mtcars %$%
  lm(mpg ~ I(hp*am))
```

```
Call:
lm(formula = mpg ~ I(hp * am))

Coefficients:
(Intercept)   I(hp * am)
    19.5696       0.0101
```

`I()` is used to inhibit the interpretation of operators in formulas, so they are used as arithmetic operators.

Chapter 9

Specifying Models

In addition to its role in limiting endogeneity, model specification plays an important role in modeling in economics. The defining characteristic of economic reasoning is marginalism so having functional forms that make marginal sense is important. In this chapter we will learn to estimate some important functional forms in economics.

We will use the following libraries.

```
library(tidyverse)
library(magrittr)
library(broom)
library(estimatr)
library(forcats)
```

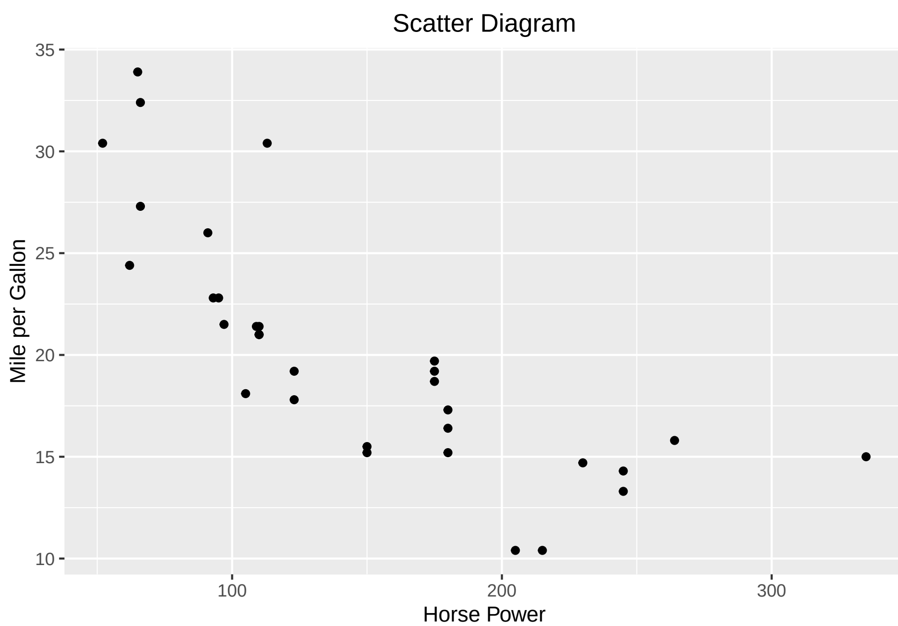
9.1 Polynomial Models

Estimating a model of the form $y = \beta_0 + \beta_1 x + \beta_2 x^2 + \beta_3 x^3 + \cdots + \beta_k x^k + \epsilon$ is straightforward in R. There is no need to create new variable within the data frame since the variables can be created directly as arguments within a function, e.g., `lm`.

Create a scatter diagram of miles per gallon vs horse power from the mtcars built in data set.

```
p <-
mtcars %>%
  ggplot(aes(x= hp, y = mpg)) +
  geom_point() +
```

```r
  labs(y = "Mile per Gallon", x = "Horse Power", title = "Scatter Diagram") +
  theme(plot.title = element_text(hjust = 0.5))
p
```



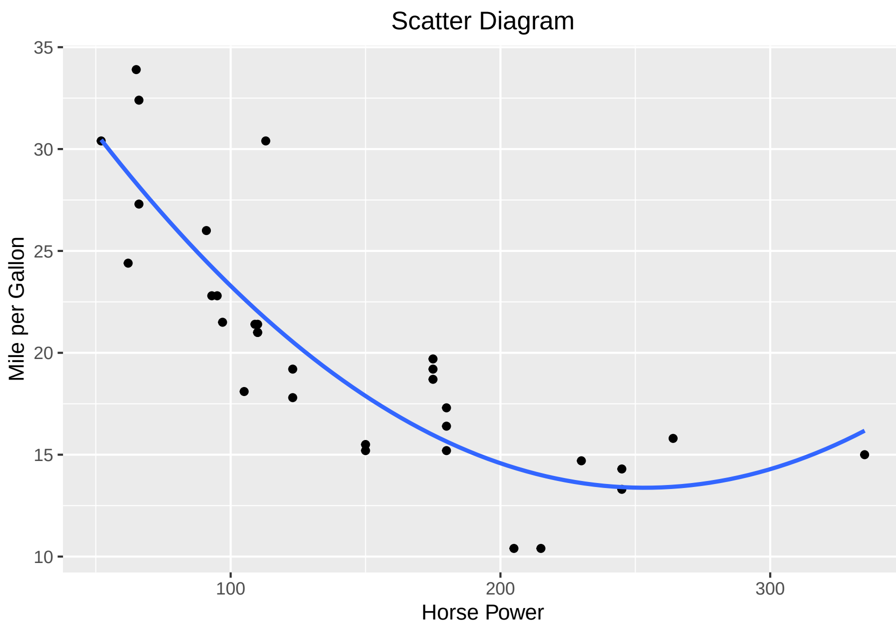
It appears as if mpg falls at a diminishing rate as horse power increases. There are a number of functional forms that will gives us that basic shape. One such form is quadratic. Let's estimate a quadratic form of the model $mpg = \beta_0 + \beta_1 hp + \beta_2 hp + \epsilon$

```r
mtcars %$%
  lm(mpg ~ hp + I(hp^2)) # uses the AsIs function I() to create hp squared
```

```

Call:
lm(formula = mpg ~ hp + I(hp^2))

Coefficients:
(Intercept)           hp      I(hp^2)
  40.409117    -0.213308     0.000421
```

Add the quadratic fit to the scatter diagram

```
p +
  geom_smooth(method = "lm", formula = y ~  x + I(x^2), se = F)
```



Adding higher order polynomials is accomplished by adding additional AsIs functions `I()` to the model.

9.2 Logarithmic

Logarithms have a wide variety of uses in Econometrics.

9.2.1 Constant Elasticity (log-log or double log)

Log-log models are of the form $\ln y = \beta_0 + \beta_1 \ln x$. So $\beta_1 = \frac{d \ln y}{d \ln x}$

Recall that $dlnX = \frac{dX}{X}$; that is, the change in the logarithm is a percent change in a variable. For example, ΔlnP is the % change in P.

So, β_1 is the percent change in y resultant from a 1% change in x. Or, a one *percent* change in x will induce a β_1 *percent* change in y.

Constant elasticity demand functions are estimated using log-log models. Let, $q = \alpha p^\beta$ where $\beta < 0$ be the demand function. Recall he elasticity of demand is given by $\eta = \frac{\%\Delta q}{\%\Delta p} = \frac{dq/q}{dp/p} = \frac{dq}{dp}\frac{p}{q}$. The elasticity of demand for our demand function is

$$\eta = \beta\alpha p^{\beta-1}\frac{p}{q} = \frac{\beta\alpha p^{\beta}}{q}$$

Substitute for $q = \alpha p^{\beta}$

$$\eta = \frac{\beta\alpha p^{\beta}}{\alpha p^{\beta}} = \beta$$

The elasticity of demand is β which is invariant. To make $q = \alpha p^{\beta}$ estimable with OLS take the logarithm of both sides:

$$\ln(q) = \ln(\alpha p^{\beta}\epsilon)$$

$$\ln(q) = \ln(\alpha) + \ln(p^{\beta}) + \ln(\epsilon)$$

$$\ln(q) = \gamma + \beta\ln(p) + \delta$$

is now in estimable form and can estimated by `lm(log(q) ~ log(p), data = df)`

Cobb-Douglas production functions are also estimated as log-log models. Let's estimate a Cobb-Douglas model for horsepower as a function of number of cylinders and displacement in cubic inches from the mtcars data. The production function is:

$$hp = \beta_0 cyl^{\beta_1} disp^{\beta_2}\epsilon$$

The estimable form of the model is

$$\ln hp = \alpha_0 + \beta_1 \ln cyl + \beta_2 \ln disp + \gamma$$

```
mtcars %$%
  lm(log(hp) ~ log(cyl) + log(disp))
```

```
Call:
lm(formula = log(hp) ~ log(cyl) + log(disp))

Coefficients:
(Intercept)     log(cyl)    log(disp)
      1.851        0.817        0.299
```

$\hat{\beta}_1$ indicates that a 1% increase in the number of cylinders, *ceteris paribus*, increases horsepower by 0.8170%. $\hat{\beta}_2$ indicates that a 1% increase in the displacement, *ceteris paribus* increases horsepower by 0.2986%.

9.2.2 Constant Growth in a Dependent Variable (log-lin or semilog)

The log-lin model has the form $\ln y = \beta_0 + \beta_1 x$. $\beta_1 = \frac{d \ln y}{dx}$ or the percent change in y resultant from a 1 unit change in x. Or, a *unit* change in x will induce a $100 * \beta_1$ *percent* change in y.

Suppose we have a variable P that is growing at a constant rate of g per period t such that $P_t = (1+g)P_{t-1}$. By repeated substitution we get $P_t = P_0(1+g)^t$ where P_0 is the initial value of Y. ^[Note the relationship to the time value of money. Let $FV = PV(1+r)^t$. Suppose we'd like to calculate the average annual rate of return, r, on $\$PV$, given a future value of $\$FV$. After appropriate algebraic gymnastics,

$$r = \sqrt[t]{\frac{FV}{PV}} - 1$$

. We could calculate g without regression as

$$g = \sqrt[t]{\frac{P_t}{P_0}} - 1$$

. We could calculate g with a regression with the aid of logarithms. Let the model be $P_t = P_0(1+g)^t\epsilon$. Taking the logarithm of both sides yields

$$\ln(P_t) = \ln(P_0) + t\ln(1+g) + \ln(\epsilon)$$

Which we can estimate as

$$Y_t = \beta_0 + \beta_1 t + u_t$$

Where $\beta_0 = \ln(P_0)$, $u_t = \ln(\epsilon_t)$, and, most importantly, $\beta_1 = \ln(1+g)$. We can get our estimate of g from $\hat{\beta}_1 = \ln\widehat{(1+g)}$ by exponentiating both sides and solving for g.

$$e^{\hat{\beta}_1} = e^{\ln\widehat{(1+g)}}$$

$$\hat{g} = e^{\hat{\beta}_1} - 1$$

[1]

Suppose we'd like to know the percent change in fuel mileage resultant from adding a cylinder to the car using the mtcars data. Estimate the model $\ln mpg = \beta_0 + \beta_1 cyl + \epsilon$

```
mtcars %$%
  lm(log(mpg) ~ cyl)
```

[1] We know that $\beta_1 = \frac{d(\ln P_t)}{dt}$. Since $d(\ln P_t) = \frac{dP_t}{P_t}$ or the percent change in P_t, β_1 is known as the **instantaneous rate of growth**.

```
Call:
lm(formula = log(mpg) ~ cyl)

Coefficients:
(Intercept)          cyl
      3.839       -0.143
```

$\hat{\beta}_1$ indicates that adding one cylinder reduces fuel mileage by 14.25%.

9.2.3 Constant % Change in Dependent Variable (lin-log)

The lin-log model has the form $y = \beta_0 + \beta_1 \ln x$. $\beta_1 = \frac{dy}{d \ln x}$ or the change in y from a 1 percent change in x. Or, one *percent* change in x will induce a $\frac{\beta_1}{100}$ *unit* change in y.

Suppose we'd like to know the change in the number of miles per gallon as a function of the percent change in displacement. Estimate the model $mpg = \beta_0 + beta_1 \ln disp + \epsilon$

```
mtcars %$%
  lm(mpg ~ log(disp))
```

```
Call:
lm(formula = mpg ~ log(disp))

Coefficients:
(Intercept)    log(disp)
      69.21        -9.29
```

$\hat{\beta}_1$ indicates that a one percent change in displacement will decrease fuel mileage by .093 mpg.

9.3 Other Useful Functional Forms

We know that marginal analysis is the *sine qua non* of economic reasoning. So our economic models should be built with an eye toward what we think the marginal relationship should look like from theory. Use a sort "a poor man's" differential equations to work from a marginal model to the functional form. For example, consider production as a function of labor holding capital constant $Q = Q(\bar{K}, L)$ with K fixed at $\bar{K}$. From microeconomic theory we know that the marginal product of labor, $MP_L = \frac{dQ}{dl}$ at some point diminishes. This suggests $MP_L = \beta_1 + \beta_2 L + \beta_3 L^2$ where $\beta_1 \geq 0, \beta_2 > 0$, and $\beta_3 < 0$ as the model product of labor. The production function that will yield this marginal product is $Q = \beta_0 + \beta_1 L + \beta_2 L^2 + \beta_3 L^3$. Thus the usefulness of a polynomial functional form in economics.

There are other functional forms that will yield diminishing marginal functions. For example, another functional form that will yield a diminishing marginal is a rectangular hyperbola $y = \beta_0 + \frac{\beta_1}{x}$. The marginal effect of x on y is $\frac{dy}{dx} = \frac{-\beta_1}{x^2}$. A one unit change in X leads to a $\frac{1}{x}$ change in y. Notice as x increases the change in y decreases, i.e., diminishing marginal.

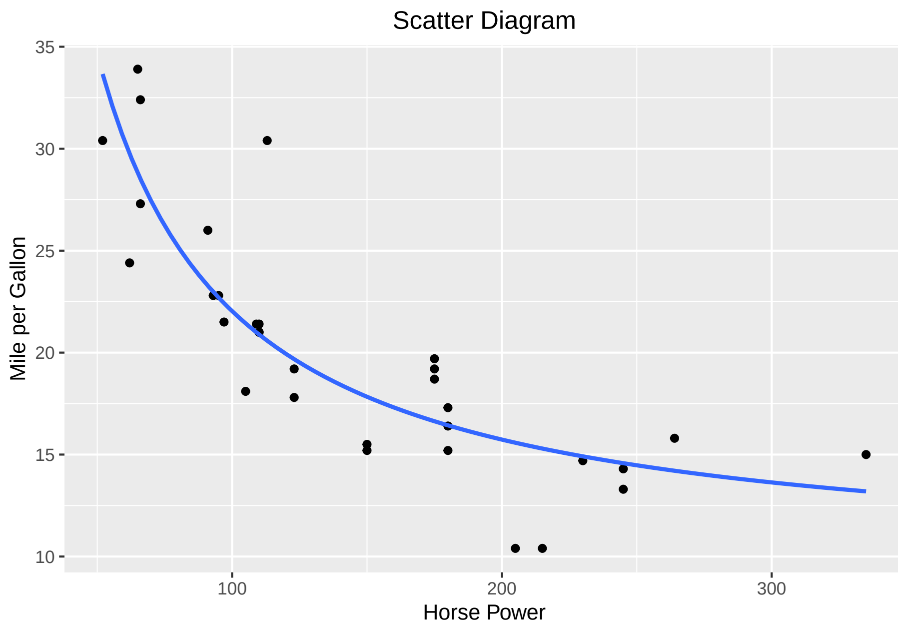
The plot of miles per gallon vs. horsepower above suggests a negative diminishing relationship between the variables. Suppose we postulate that miles per gallon is a function of the inverse of horsepower $mpg = \beta_0 + \frac{\beta_1}{hp} + \epsilon$.

```
mtcars %$%
  lm(mpg ~ I(1/hp))
```

```
Call:
lm(formula = mpg ~ I(1/hp))

Coefficients:
(Intercept)      I(1/hp)
       9.43      1259.88
```

Add the fitted model to scatter diagram.

```
p +
  geom_smooth(method = lm, formula = y ~ I(1/x), se = F)
```


Scatter Diagram
Mile per Gallon
35
30
25
20
15
10
100
200
300
Horse Power

Chapter 10

Using Fixed Effects Models to Fight Endogeneity in Panel Data and Difference–in–Difference Models

In this chapter we will learn to deal with panel data in R. Panel data are data that include observations in and through time. Panel data combine aspects of cross–sectional data with time–series data. The libraries necessary for this chapter are:

```
library(tidyverse)
library(magrittr)
library(broom)
library(estimatr)
library(carData)
```

10.1 Simpson's Paradox

Simpson's paradox - Simpson (1951) is phenomenon where an apparent relationship between two variables reverses itself when the data are dis-aggregated. For example, let's look at the admissions rate for men and women in the University of California at Berkeley admissions data.

UCBAdmissions is a cross-tabulation of 4526 applicants by 3 variables: Admit, Gender, and Dept, the number of observations for each is n stored as 3-dimensional array.

```
UCBAdmissions
```

```
, , Dept = A

          Gender
Admit      Male Female
  Admitted  512     89
  Rejected  313     19

, , Dept = B

          Gender
Admit      Male Female
  Admitted  353     17
  Rejected  207      8

, , Dept = C

          Gender
Admit      Male Female
  Admitted  120    202
  Rejected  205    391

, , Dept = D

          Gender
Admit      Male Female
  Admitted  138    131
  Rejected  279    244

, , Dept = E

          Gender
Admit      Male Female
  Admitted   53     94
  Rejected  138    299

, , Dept = F

          Gender
Admit      Male Female
  Admitted   22     24
```

```
  Rejected  351    317
```

To calculate admission rates, we need to create a new variable, apps, that is the sum of admitted and rejected apps for both men and women.

```
UCBAdmissions %>%
  as_tibble() %>% # convert the table to a data frame
  group_by(Dept, Gender) %>%  # allows us to sum admitted and rejected by department
  mutate(apps = sum(n)) %>% # create number of applicants by department
  ungroup() %>% # return the full data frame
  filter(Admit == "Admitted") %>% # select only those applicants admitted
  group_by(Gender) %>% # allows us to calculate acceptance rates by gender
  summarize(rate = sum(n)/sum(apps))
```

```
# A tibble: 2 x 2
  Gender  rate
  <chr>  <dbl>
1 Female 0.304
2 Male   0.445
```

Males are accepted at rate of 44.5% while females are accepted at lower rate of 30.4%.

```
UCBAdmissions %>%
  as_tibble() %>%
  group_by(Dept, Gender) %>%
  mutate(apps = sum(n)) %>%
  ungroup() %>%
  filter(Admit == "Admitted") %>%
  group_by(Dept, Gender) %>%
  summarize(n/apps)
```

```
# A tibble: 12 x 3
# Groups:   Dept [6]
   Dept  Gender `n/apps`
   <chr> <chr>     <dbl>
 1 A     Female   0.824
 2 A     Male     0.621
 3 B     Female   0.68
 4 B     Male     0.630
 5 C     Female   0.341
 6 C     Male     0.369
 7 D     Female   0.349
 8 D     Male     0.331
```

```
 9 E     Female   0.239
10 E     Male     0.277
11 F     Female   0.0704
12 F     Male     0.0590
```

We now see that females are admitted at higher rates to four of the six departments.

10.2 Figures 8.1-8.3

We see a similar effect in Figures 8.1-8.3 in the text. We can reproduce those graphs with the code below. The crime data set contains observations on 19 variables from 58 cities over the period 1972 to 1993. First choose observations for only the California cities of Fresno, Los Angeles, Oakland, Sacramento, and San Francisco. Next convert the robbery and police to numbers per 1000 persons. The data frame crime contains the data.# the %in% operator means match the elements in one vector with elements in another.

```
crime %>%
  select(cityname, policesworn, robbery, popcity) %>% # choose relevant variables
  filter(cityname %in% c("fresno", "losangel", "oakland", "sacramen", "sanfran")) %>% #
  mutate(robbery=robbery/popcity*1000, policesworn = policesworn/popcity*1000) %>% # c
  ggplot(aes(x = policesworn, y = robbery)) +
  geom_point(na.rm = T) +
  geom_smooth(method = lm, na.rm = T, se = F) +
  xlab("Police per 1000 People") +
  ylab("Robberies per 1000 People") +
  labs(caption = "Figure 8.1: Robberies and Police for Large Cities in California") + #
  theme(plot.caption = element_text(hjust = 0)) # left justify the caption
```

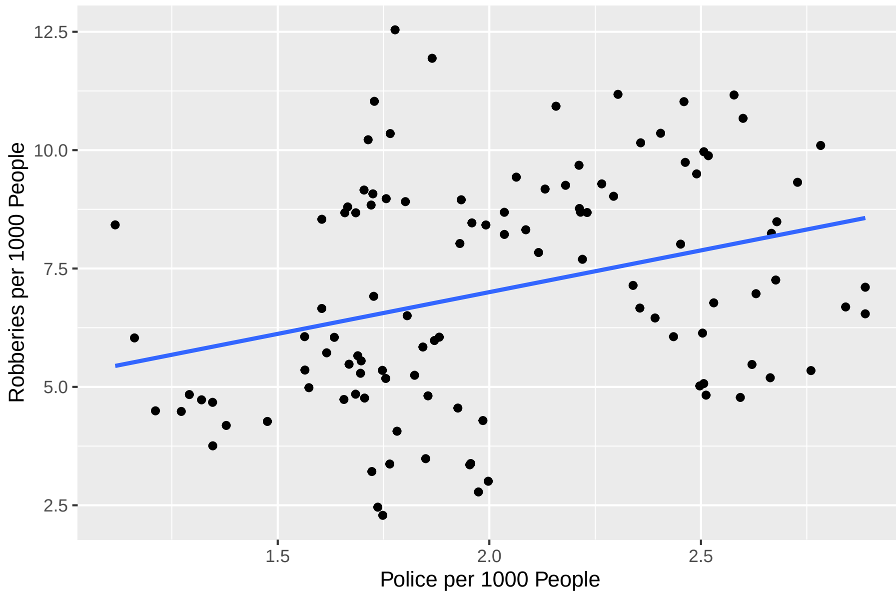


Figure 8.1: Robberies and Police for Large Cities in California

```
crime %>%
  select(cityname, policesworn, robbery, popcity) %>%
  filter(cityname %in% c("fresno", "losangel", "oakland", "sacramen", "sanfran")) %>%
  mutate(robbery=robbery/popcity*1000, policesworn = policesworn/popcity*1000) %>%
  ggplot(aes(x = policesworn, y = robbery, color = cityname)) +
  geom_point(na.rm = T) +
  xlab("Police per 1000 People") +
  ylab("Robberies per 1000 People") +
  labs(caption = "Figure 8.2: Robberies and Police for Specified Cities in California") +
  theme(plot.caption = element_text(hjust = 0), legend.position = "none") + # remove legend
  # place city names with corresponding colors.
  annotate(geom = "text", x = 1.6, y = 10, label = "Oakland", col = "#00BF7D") +
  annotate(geom = "text", x = 2, y = 5, label = "Sacramento", col = "#00B0F6") +
  annotate(geom = "text", x = 2.58, y = 4.5, label = "Los Angeles", col = "#A3A500") +
  annotate(geom = "text", x = 2.7, y = 7.8, label = "San Francisco", col = "#E76BF3") +
  annotate(geom = "text", x = 1.25, y = 3.5, label = "Fresno", col = "#F8766D")
```

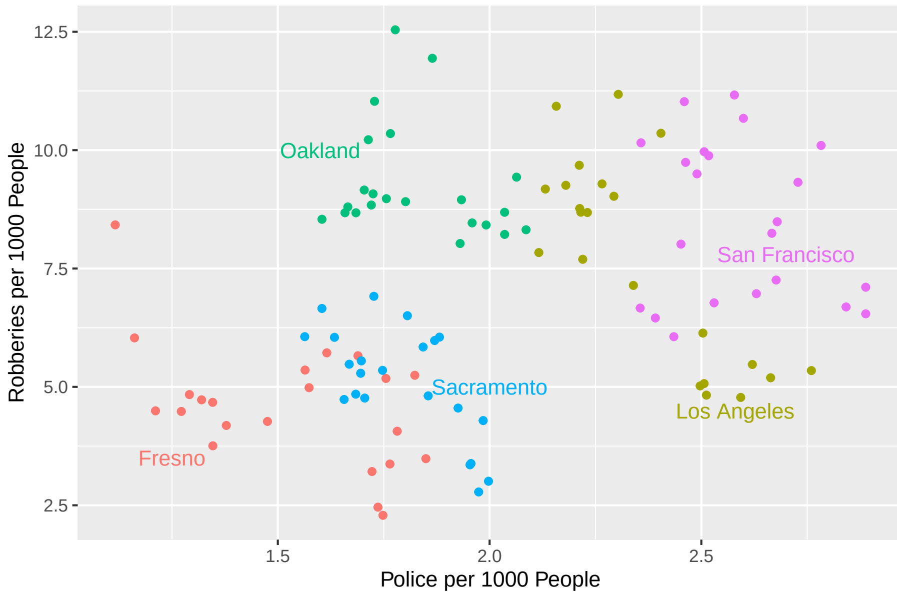


Figure 8.2: Robberies and Police for Specified Cities in California

```r
crime %>%
  select(cityname, policesworn, robbery, popcity) %>%
  filter(cityname %in% c("fresno", "losangel", "oakland", "sacramen", "sanfran")) %>%
  mutate(robbery=robbery/popcity*1000, policesworn = policesworn/popcity*1000) %>%
  ggplot(aes(x = policesworn, y = robbery, color = cityname)) +
  geom_point(na.rm = T) +
  xlab("Police per 1000 People") +
  ylab("Robberies per 1000 People") +
  labs(caption = "Figure 8.3: Robberies and Police for Specified Cities in California w
  theme(plot.caption = element_text(hjust = 0), legend.position = "none") +
  annotate(geom = "text", x = 1.6, y = 10, label = "Oakland", col = "#00BF7D") +
  annotate(geom = "text", x = 2, y = 5, label = "Sacramento", col = "#00B0F6") +
  annotate(geom = "text", x = 2.58, y = 4.5, label = "Los Angeles", col = "#A3A500") +
  annotate(geom = "text", x = 2.7, y = 7.8, label = "San Francisco", col = "#E76BF3") +
  annotate(geom = "text", x = 1.25, y = 3.5, label = "Fresno", col = "#F8766D") +
  geom_smooth(method = "lm", se = F) # add regression lines. the addition of the color
```

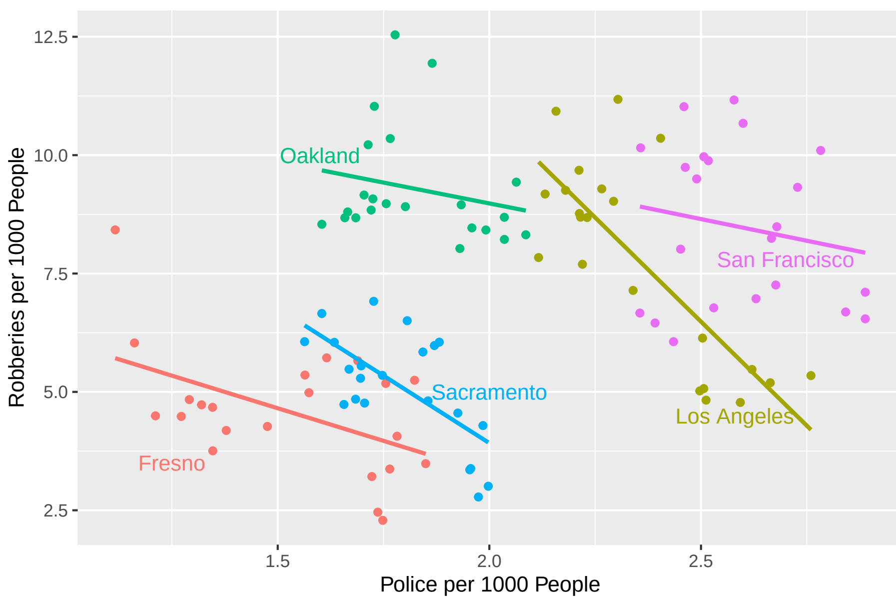


Figure 8.3: Robberies and Police for Specified Cities in California with City−Specific Regression Lines

10.3 One-Way Fixed Effects Models

10.3.1 LSDV Approach

The least squares dummy variable approach allows us to account for the fixed effects by including a dummy variable for each unit. First, let's calculate the pooled model.

```
crime %>%
  select(cityname, policesworn, robbery, popcity) %>%
  filter(cityname %in% c("fresno", "losangel", "oakland", "sacramen", "sanfran")) %>%
  mutate(robbery=robbery/popcity*1000, policesworn = policesworn/popcity*1000) %$%
  lm(robbery ~ policesworn) %>%
  tidy()
```

```
# A tibble: 2 x 5
  term        estimate std.error statistic  p.value
  <chr>          <dbl>     <dbl>     <dbl>    <dbl>
1 (Intercept)     3.48     1.05       3.31 0.00129
2 policesworn     1.76     0.509      3.46 0.000771
```

We can see that the coefficient on the police variable is positive and significantly different than zero.

To apply LSDV approach in R, we add cityname as an explanatory variable. Since cityname is a character vector, R will treat it as a factor.

```
crime %>%
  select(cityname, policesworn, robbery, popcity) %>%
  filter(cityname %in% c("fresno", "losangel", "oakland", "sacramen", "sanfran")) %>%
  mutate(robbery=robbery/popcity*1000, policesworn = policesworn/popcity*1000) %$%
  lm(robbery ~ policesworn + cityname)
```

```
Call:
lm(formula = robbery ~ policesworn + cityname)

Coefficients:
     (Intercept)       policesworn  citynamelosangel   citynameoakland
           10.93             -4.16              6.60              5.96
citynamesacramen   citynamesanfran
            1.63              8.32
```

We can confirm that below.

```
crime %>%
  select(cityname, policesworn, robbery, popcity) %>%
  filter(cityname %in% c("fresno", "losangel", "oakland", "sacramen", "sanfran")) %>%
  mutate(robbery=robbery/popcity*1000,
         policesworn = policesworn/popcity*1000,
         cityname = as_factor(cityname)) %$% # coerce cityname to a factor
  lm(robbery ~ policesworn + cityname)
```

```
Call:
lm(formula = robbery ~ policesworn + cityname)

Coefficients:
     (Intercept)       policesworn  citynamelosangel   citynameoakland
           10.93             -4.16              6.60              5.96
citynamesacramen   citynamesanfran
            1.63              8.32
```

The equation for each city is:

$$\text{Fresno: } Robbery = 8.79 - 2.75 Police$$

$$\text{Los Angeles: } Robbery = 17.53 - 2.75 Police$$

$$\text{Oakland: } Robbery = 16.89 - 2.75 Police$$

$$\text{Sacramento: } Robbery = 12.56 - 2.75 Police$$

$$\text{San Francisco: } Robbery = 19.25 - 2.75 Police$$

We see the effect of Simpson's Paradox in the slope variable here. The slope variable is now negative and significant. It should be noted that these results are not consistent with Figure 8.3. Here we have only one slope coefficient with five different intercepts; Figure 8.3 shows five different slope coefficients along with five different intercepts. We can return results consistent with Figure 8.3 as below. We can show the equation for each of the five cities by adding the coefficient on the dummy variable to the intercept with the base case being Fresno[1]

```
crime %>%
  select(cityname, policesworn, robbery, popcity) %>%
  filter(cityname %in% c("fresno", "losangel", "oakland", "sacramen", "sanfran")) %>%
  mutate(robbery=robbery/popcity*1000,
         policesworn = policesworn/popcity*1000,
         cityname = as_factor(cityname)) %$% # coerce cityname to a factor
  lm(robbery ~ policesworn * cityname)
```

```

Call:
lm(formula = robbery ~ policesworn * cityname)

Coefficients:
                 (Intercept)                   policesworn
                       8.786                        -2.754
            citynamelosangel               citynameoakland
                      19.679                         3.720
            citynamesacramen                citynamesanfran
                       6.522                         4.442
policesworn:citynamelosangel   policesworn:citynameoakland
                      -6.038                         0.992
policesworn:citynamesacramen   policesworn:citynamesanfran
                      -2.940                         0.923
```

Now the equation for each city requires that we add the slope dummy coefficient to the intercept coefficient and the interaction coefficient to the coefficient on policesworn. So the equation for each city is:[2]

[1] The base case can be changed from the default with appropriate arguments see the forcats package for more.

[2] Please note that not all of the coefficients are significant at the 5% level. This is ignored in the equations derived for expository purposes. In fact, we can see that the slope coefficients for Oakland, Sacramento, and San Francisco are not significantly different from the slope coefficient for Fresno, since each of those interaction effects are not significant.

$$\text{Fresno: } Robbery = 8.79 - 2.75Police$$

$$\text{Los Angeles: } Robbery = 28.46 - 8.79Police$$

$$\text{Oakland: } Robbery = 12.51 - 1.76Police$$

$$\text{Sacramento: } Robbery = 15.31 - 5.69Police$$

$$\text{San Francisco: } Robbery = 13.23 - 1.83Police$$

The above equation are consistent with the regression lines in Figure 8.3.

10.3.2 *F*-test for significance of fixed effects.

The unrestricted model is given by:

$$Y_{it} = \beta_0 + \beta_1 X_{1it} + \beta_2 D_{1i} + \beta_3 D_{2i} + \cdots + \beta_P D_{P-1,i} + \nu_{it}$$

To test for the significance of fixed effects we test the following hypothesis:

$$H_0 : \beta_2 = \beta_3 = \cdots = \beta_P$$

$$H_1 : \text{@ least one } \beta \neq 0$$

As in Chapter 5, we will make use of `linearHypothesis` from the car package.

```
library(car)
crime %>%
  select(cityname, policesworn, robbery, popcity) %>%
  filter(cityname %in% c("fresno", "losangel", "oakland", "sacramen", "sanfran")) %>%
  mutate(robbery=robbery/popcity*1000,
         policesworn = policesworn/popcity*1000,
         cityname = as_factor(cityname)) %$% # coerce cityname to a factor
  lm(robbery ~ policesworn + cityname) %>%
  linearHypothesis(c("citynamelosangel = 0", # use the variable names from the lm obje
                     "citynameoakland = 0",
                     "citynamesacramen = 0",
                     "citynamesanfran = 0" ))
```

```
Linear hypothesis test

Hypothesis:
citynamelosangel = 0
citynameoakland = 0
citynamesacramen = 0
citynamesanfran = 0
```

```
Model 1: restricted model
Model 2: robbery ~ policesworn + cityname

  Res.Df RSS Df Sum of Sq    F               Pr(>F)
1    108 571
2    104 194  4       377 50.6 <0.0000000000000002 ***
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

Since the reported F-stat is 50.626 with a *p-value* of 0, we will reject the null hypothesis of no fixed effects in favor of the alternative suggesting that fixed effects exist.

10.4 De-Meaned approach

10.4.1 Manually De-Mean

We can estimate the fixed-model with a de-meaned approach with the model:

$$Y_{it} - \bar{Y}_{i.} = \beta_1(X_{it} - \bar{X}_{i.})$$

`scale` will de-mean the data with the argument `scale = F`. Learn more about `scale` by calling `?scale`. Do de-mean the data by city will use `group_by` in our pipeline to group the data by city, then we will `mutate` the crime and police variables with `scale` to de-mean them. We should end up the same estimate of the slope coefficient from the LSDV approach.

```
crime %>%
  select(cityname, policesworn, robbery, popcity) %>%
  filter(cityname %in% c("fresno", "losangel", "oakland", "sacramen", "sanfran")) %>%
  mutate(robbery=robbery/popcity*1000,
         policesworn = policesworn/popcity*1000,
         cityname = as_factor(cityname)) %>%
  group_by(cityname) %>%
  mutate(robbery = scale(robbery, scale = F),
         policesworn = scale(policesworn, scale = F)) %$%
  lm(robbery ~ policesworn)
```

```
Call:
lm(formula = robbery ~ policesworn)

Coefficients:
          (Intercept)            policesworn
 0.000000000000000609   -4.160101927269632682
```

The slope coefficient is the same as the slope coefficient estimated by LSDV.

10.4.2 Using the `plm` package

We can estimate the fixed effects model with `plm` from the `plm` package. The `plm` package was created to make the estimation of linear panel models straightforward. To learn more read the `vignette(plmPackage)`. To estimate the one-way fixed effects model with `plm`, we need four arguments`formula`, `data`, `index`, and `model`. The `formula` and `data` arguments are the same as those in the `lm` call. `index` is a vector of the units and the type of variation is invoked with `model`. We estimate the model below:

```
library(plm)
crime %>%
  select(cityname, policesworn, robbery, popcity) %>%
  filter(cityname %in% c("fresno", "losangel", "oakland", "sacramen", "sanfran")) %>%
  mutate(robbery=robbery/popcity*1000,
         policesworn = policesworn/popcity*1000,
         cityname = as_factor(cityname)) -> # the %$% pipe does not function with plm
  cali # the modified data are assigned to the object cali
  plm(robbery ~ policesworn, data = cali, index = "cityname", model = "within")
```

```
Model Formula: robbery ~ policesworn

Coefficients:
policesworn
      -4.16
```

Again, we get the same estimate of the slope coefficient.

10.5 Two-Way Fixed Effects Models

The two-way fixed effects model is given by:

$$Y_{it} = \beta_0 + \beta_1 X_{1it} + \alpha_i + \tau_t + \nu_{it}$$

So we need to incorporate time into the one-way fixed effects model. This can be accomplished in one of two ways. Time can be treated as a factor (dummy variable) or set the effect in `plm` to `"twoways"`. The results will be the same.

10.5.1 Time as a factor

```
crime %>%
  select(cityname, policesworn, robbery, popcity, year) %>%
  filter(cityname %in% c("fresno", "losangel", "oakland", "sacramen", "sanfran")) %>%
  mutate(robbery=robbery/popcity*1000,
         policesworn = policesworn/popcity*1000,
         cityname = as_factor(cityname)) -> # the %$% pipe does not function with plm
  cali # the modified data are assigned to the object cali
  plm(robbery ~ policesworn + factor(year), data = cali, index = "cityname", model = "within")
```

```
Model Formula: robbery ~ policesworn + factor(year)

Coefficients:
   policesworn factor(year)72 factor(year)73 factor(year)74 factor(year)75
        -1.945         -0.493         -0.220         -0.158          0.647
factor(year)76 factor(year)77 factor(year)78 factor(year)79 factor(year)80
         0.872          0.789          1.476          1.545          2.831
factor(year)81 factor(year)82 factor(year)83 factor(year)84 factor(year)85
         2.509          1.940          1.151          0.782          1.005
factor(year)86 factor(year)87 factor(year)88 factor(year)89 factor(year)90
         1.361          0.109          0.140          0.574          1.432
factor(year)91 factor(year)92
         2.394          3.373
```

10.5.2 effect = "twoways"

```
crime %>%
  select(cityname, policesworn, robbery, popcity, year) %>%
  filter(cityname %in% c("fresno", "losangel", "oakland", "sacramen", "sanfran")) %>%
  mutate(robbery=robbery/popcity*1000,
         policesworn = policesworn/popcity*1000,
         cityname = as_factor(cityname)) -> # the %$% pipe does not function with plm
  cali # the modified data are assigned to the object cali
  plm(robbery ~ policesworn, data = cali, index = "cityname", model = "within", effect = "twoways
```

```
Model Formula: robbery ~ policesworn

Coefficients:
policesworn
      -1.94
```

As expected, the coefficient on the police variable is the same in each case.

10.6 Difference-in-Difference Models

In 1992 New Jersey raised it's minimum wage from \$4.25 to \$5.05 while neighboring Pennsylvania did not. We can use a difference-in-difference model to investigate the effect of the treatment (increase in minimum wage) on the effect full time employment. The `PoEdata`[3] package contains a data set named `njmin3` that has 820 observations on 14 variables, call `?njmin` for more information.

Estimate the basic model

$$fte_{it} = \beta_0 + \beta_1 nj_i + \beta_2 d_i + \beta_3 (nj_i \times d_i) + \epsilon_{it}$$

where fte_i is full-time equivalent employees, nj_i is the treatment[4], and d_i is the after dummy[5]. Since β_3 is the difference in differences of treated and control states, test the hypothesis:

$$H_0 : \beta_3 = 0$$

$$H_1 : \beta_3 \neq 0$$

```
# Call the following only once.
# install.packages("devtools") # required to insall GitHub packages do this only once
# devtools::install_git("https://github.com/ccolonescu/PoEdata") # install the package
```

```
library(PoEdata)
data(njmin3)
njmin3 %$%
  lm(fte ~ nj*d) %>%
  summary()
```

```
Call:
lm(formula = fte ~ nj * d)

Residuals:
   Min     1Q Median     3Q    Max
-21.17  -6.44  -1.03   4.47  64.56

Coefficients:
```

[3]The PoEdata package is not housed at CRAN, instead it is house at GitHub, so installing it requires an extra step.

[4]nj_i takes the value 1 for New Jersey where the minimum wage was increased and the value 0 for Pennsylvania where the minimum wage was not changed

[5]d_1 takes the value 1 after the minimum wage is changed and the value 0 before the change.

```
            Estimate Std. Error t value            Pr(>|t|)
(Intercept)    23.33       1.07   21.77 <0.0000000000000002 ***
nj             -2.89       1.19   -2.42               0.016 *
d              -2.17       1.52   -1.43               0.154
nj:d            2.75       1.69    1.63               0.103
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

Residual standard error: 9.41 on 790 degrees of freedom
  (26 observations deleted due to missingness)
Multiple R-squared:  0.0074,    Adjusted R-squared:  0.00363
F-statistic: 1.96 on 3 and 790 DF,  p-value: 0.118
```

At the $\alpha = .05$ level of significance the t-statistic with 790 degrees of freedom is $\pm$ 1.963. The calculated t-statistic is 1.631 so we fail to reject the null hypothesis and conclude that there is no evidence to suggest that the change in the minimum wage changed full-time employment.

We control for other variables below

```
njmin3 %$%
  lm(fte ~ nj*d + co_owned) %>%
  summary()
```

```

Call:
lm(formula = fte ~ nj * d + co_owned)

Residuals:
   Min     1Q Median     3Q    Max
-22.06  -6.34  -1.06   4.56  66.29

Coefficients:
            Estimate Std. Error t value             Pr(>|t|)
(Intercept)   24.293      1.093   22.24 < 0.0000000000000002 ***
nj            -2.939      1.184   -2.48              0.01323 *
d             -2.234      1.503   -1.49              0.13759
co_owned      -2.645      0.696   -3.80              0.00015 ***
nj:d           2.820      1.674    1.68              0.09255 .
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

Residual standard error: 9.33 on 789 degrees of freedom
  (26 observations deleted due to missingness)
Multiple R-squared:  0.0253,    Adjusted R-squared:  0.0203
F-statistic: 5.11 on 4 and 789 DF,  p-value: 0.000454
```

Chapter 11

Instrumental Variables: Using Exogenous Variation to Fight Endogeneity

In this chapter we will learn to use R to instrumental variables and two–stage least squares models. We will use the libraries below.

```
library(tidyverse)
library(magrittr)
library(broom)
```

11.1 2 Stage Least Squares

To estimate a 2SLS, use `ivreg` from the `AER` package. `ivreg`, at a minimum, requires a formula that specifies the dependent and independent variables, instruments that identify instrumental variables, and the data. So the form of the call is, for example, `ivreg`(Y ~ X1 + X2 | Z1 + Z2 + X2, dataframe).[1] Where X1 is the endogenous variable, X2 is exogenous and Z1 and Z2 are instruments for X1.

```
library(AER)
```

The classic example of endogeneity in economics is that of a demand equation, that is of quantity demanded as a function of price, $Q = Q(P)$. There is no reason we can't write $P = P(Q)$ because a price determines quantity demanded,

[1] The data argument can be called with the expose operator %$%.

but we can't have a quantity without a price. That is, price depends on quantity demanded which depends on price. To solve this problem we need an instrument that is exogenous to the demand equation but related to supply. This variable will induce changes in supply along the demand curve and thus changes in price. Since changes in supply will be correlated (cause) with changes in price, this new variable can serve as an instrument for price.

Let the demand equation be given by

$$q_d = \beta_0 + \beta_1 p + u,$$

supply by

$$q_s = \alpha_0 + \alpha_1 p + v,$$

and the market clearing equation by

$$q_d = q_s = q$$

These are known as the *structural equations.* Solving for p and q separately gives us the *reduced form* equations. Using the market clearing equation we know:

$$\beta_0 + \beta_1 p + u = \alpha_0 + \alpha_1 p + v$$

so,

$$p = \frac{\alpha_0 - \beta_0}{\beta_1 - \alpha_1} + \frac{v - u}{\beta_1 - \alpha_1} = \lambda_0 + \epsilon_1$$

and

$$q = \frac{\beta_1 \alpha_0 - \beta_0 \alpha_1}{\beta_1 - \alpha_1} + \frac{\beta_1 v - \alpha_1 u}{\beta_1 - \alpha_1} = \mu_0 + \epsilon_2$$

Notice that we have two estimable equations now. We can obtain OLS estimates for the reduced form parameters as $\hat{\lambda}_0$ and $\hat{\mu}_0$ as

$$\hat{\lambda}_0 = \bar{p} = \frac{\alpha_0 - \beta_0}{\beta_1 - \alpha_1}$$

and

$$\hat{\mu}_o = \bar{q} = \frac{\beta_1 \alpha_0 - \beta_0 \alpha_1}{\beta_1 - \alpha_1}$$

where $\bar{p}$ and $\bar{q}$ are the sample means of p and q.

What we want, however, are estimates of the structural parameters β_0, β_1, α_0, and α_1. We have two equations and four unknowns; we cannot estimate the four parameters from the the two OLS estimates, $\hat{\lambda}_0$ and $\hat{\mu}_0$. That is, we cannot derive unique values for structural parameters from our estimates of the reduced form parameters. This is the essence of what's known as the identification problem. If we can find a unique solution to the structural parameters from the OLS estimates of the reduced form parameters, then the equation is identified. The parameters of an identified equation are estimable.

Suppose the supply is now given by

$$q_s = \alpha_0 + \alpha_1 p + \alpha_2 r + v$$

where r is an exogenous variable. Solving for p and q yields the *reduced form* equations

$$p = \lambda_0 + \lambda_1 r + \epsilon_1$$

and

$$q = \mu_0 + \mu_1 r + \epsilon_2$$

where $\lambda_0 = \frac{\alpha_0 - \beta_0}{\beta_1 - \alpha_1}$, $\lambda_1 = \frac{\alpha_2}{\beta_1 - \alpha_1}$, $\mu_0 = \beta_0 + \beta_1 \lambda_0$, and $\mu_1 = \beta_1 \lambda_1$. We can solve for unique values of $\hat{\beta}_0 = \hat{\mu}_0 - \frac{\hat{\mu}_1}{\hat{\lambda}_1}\hat{\lambda}_0$ and $\hat{\beta}_1 = \frac{\hat{\mu}_1}{\hat{\lambda}_1}$. So the demand equation is identified. We can not obtain unique parameter estimates for the supply equation, however, so because $\hat{\mu}_0 = \frac{\hat{\beta}_1 \alpha_1 - \hat{\beta}_0 \alpha_1}{\hat{\beta}_1 - \alpha_1}$ and $\hat{\mu}_1 = \frac{\hat{\beta}_1 \alpha_2}{\hat{\beta}_1 - \alpha_1}$ are only two equations with three unknowns. If we add an exogenous variable to the demand equation, both equations would be identified.[2]

This method for obtaining parameter estimates is called indirect least squares (ILS). Let's use the truffles data set from the `PoEdata` package.[3] Truffles is a data frame with 30 observations on 5 variables. p is the price per ounce of premium truffles in \$, q is the quantity of truffles traded in ounces, ps is the price per ounce of choice truffles in \$, di is monthly per capita disposable income in \$1000 per month, and pf is the hourly rental fee in \$ of a truffle pig.

```
library(PoEdata)
data("truffles")
```

Let the demand function be

$$q = \beta_0 + \beta_1 p + u$$

and the supply function be

$$q = \alpha_0 + \alpha_1 p + \alpha_2 pf + v$$

Estimate the two reduced form equations as follows:

```
truffles %$%
  lm(p ~ pf)
```

[2]The reader can verify that be saying adding the exogenous variable y to the demand equation to yield $q_d = \beta_0 + \beta_1 p + \beta_2 y + u$ and solving for the reduced form equations.

[3]Install the `PoEdata` package as follows: Install the `remotes` package with `install.packages("remotes")`. The remotes package allows you to install R packages from remote repositories such as GitHub. Install the `PoEdata` package by calling `remotes::install_github("ccolonescu/PoEdata")`. Finally, load the truffles data by calling `data("truffles")`.

```
Call:
lm(formula = p ~ pf)

Coefficients:
(Intercept)           pf
       4.34         2.57
```

```
truffles %$%
  lm(q ~ pf)
```

```
Call:
lm(formula = q ~ pf)

Coefficients:
(Intercept)           pf
     21.501       -0.134
```

The reduced form parameter estimates are $\hat{\lambda}_0 = 3.343$, $\hat{\lambda}_1 = 2.566$, $\hat{\mu}_0 = 21.5006$, and $\hat{\mu}_1 = -0.1337$. The structural from parameter estimates for the demand equation are $\hat{\beta}_1 = \frac{-0.1337}{2.566} = -0.0521$ and $\hat{\beta}_0 = 21.5006 - (-0.0521) * 4.343 = 21.7269$. So are demand equation is $q_d = 21.7269 - 0.0521p$.

Below we see the two stage least square estimates are the same.

```
truffles %$%
  ivreg(q ~ p | pf) %>%
  summary()
```

```
Call:
ivreg(formula = q ~ p | pf)

Residuals:
    Min      1Q  Median      3Q     Max
-13.350  -2.662   0.148   3.931   9.152

Coefficients:
            Estimate Std. Error t value Pr(>|t|)
(Intercept)  21.7269     4.6046    4.72  0.00006 ***
p            -0.0521     0.0718   -0.73     0.47
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

```
Residual standard error: 5.29 on 28 degrees of freedom
Multiple R-Squared: -0.268, Adjusted R-squared: -0.313
Wald test: 0.527 on 1 and 28 DF,  p-value: 0.474
```

11.2 Explanatory power of the instruments

Now, let the demand for premium truffles be a function of the price premium truffles, disposable income, and the price of choice truffles. Let the supply of premium truffles be a function the price of premium truffles and the rental rate of a truffle pig. Suppose we'd like to estimate the demand equation. In this case, *pf* is the lone instrument for *p*. Assess the explanatory power of *pf* as an instrument as follows:

```
truffles %$%
  lm(p ~ pf + di + ps) %>%
  tidy()
```

```
# A tibble: 4 x 5
  term        estimate std.error statistic   p.value
  <chr>          <dbl>     <dbl>     <dbl>     <dbl>
1 (Intercept)   -32.5      7.98      -4.07 0.000387
2 pf              1.35     0.299      4.54 0.000115
3 di              7.60     1.72       4.41 0.000160
4 ps              1.71     0.351      4.87 0.0000476
```

The t statistic exceeds 3, so *pf* is a good instrument for *p*.

Similarly we can estimate the supply of premium truffles as a function of the price of premium truffles and the rental rate of a truffle pig. Using the demand function from above, we now have two instruments for p in the supply equation, *ps* and *di*. Since there is only one exogenous variable in the supply equation, the F test for the instruments is simply the F test for overall significance for the regression $pf = \beta_0 + \beta_1 ps + \beta_2 di + \epsilon$.

```
truffles %$%
  lm(pf ~ ps + di) %>%
  glance()
```

```
# A tibble: 1 x 11
  r.squared adj.r.squared sigma statistic p.value    df logLik   AIC   BIC
      <dbl>         <dbl> <dbl>     <dbl>   <dbl> <int>  <dbl> <dbl> <dbl>
1     0.407         0.363  4.25      9.27 8.63e-4     3  -84.4  177.  182.
# ... with 2 more variables: deviance <dbl>, df.residual <int>
```

The F statistic is 9.27 which is slightly below the rule of thumb of 10 for multiple instruments.

11.3 Estimating Simultaneous Equation Model

We can estimate the model posed above by estimating each equation as follows:

```
truffles %$%
  ivreg(q ~ p + ps + di | p + ps + di + pf) %>%
  summary()
```

```

Call:
ivreg(formula = q ~ p + ps + di | p + ps + di + pf)

Residuals:
   Min     1Q Median     3Q    Max 
-7.155 -1.936 -0.374  2.396  6.335 

Coefficients:
            Estimate Std. Error t value Pr(>|t|)   
(Intercept)   1.0910     3.7116    0.29   0.7711   
p             0.0233     0.0768    0.30   0.7642   
ps            0.7100     0.2143    3.31   0.0027 **
di            0.0764     1.1909    0.06   0.9493   
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

Residual standard error: 3.46 on 26 degrees of freedom
Multiple R-Squared: 0.496,  Adjusted R-squared: 0.438 
Wald test: 8.52 on 3 and 26 DF,  p-value: 0.000416 
```

```
truffles %$%
  ivreg(q ~ p + pf | p + ps + di + pf) %>%
  summary()
```

```

Call:
ivreg(formula = q ~ p + pf | p + ps + di + pf)

Residuals:
   Min     1Q Median     3Q    Max 
-3.783 -0.853  0.227  0.758  3.347 
```

```
Coefficients:
            Estimate Std. Error t value           Pr(>|t|)
(Intercept)  20.0328     1.2220    16.4 0.0000000000000015 ***
p             0.3380     0.0217    15.5 0.0000000000000054 ***
pf           -1.0009     0.0764   -13.1 0.0000000000003235 ***
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

Residual standard error: 1.5 on 27 degrees of freedom
Multiple R-Squared: 0.902,  Adjusted R-squared: 0.895
Wald test:  124 on 2 and 27 DF,  p-value: 0.0000000000000245
```

Chapter 12

Experiments: Dealing with Real–World Challenges

We will learn to assess balance with R in this chapter. We need the following libraries

```
library(tidyverse)
library(broom)
```

12.1 Assess Balance

Let's use the `ProgramEffectiveness` data set from the `AER` package to assess balance. The `ProgramEffectiveness` data set contains 32 observations on four variables[1]. The data are used to examine whether a new method of teaching economics improved performance in later economics courses. The variables are *grade* coded as a factor with levels "increase" and "decrease", *average* (grade point average), *testscore* (test score on economics test), and *participation* coded as a factor with levels "no" and "yes". *participation* is the treatment in this case. We assess the balance below:

```
library(AER)
data("ProgramEffectiveness")
ProgramEffectiveness %$%
  lm(average ~ participation) %>%
  tidy()
```

[1] `?AER::ProgramEffectiveness for more information`

```
# A tibble: 2 x 5
  term             estimate std.error statistic  p.value
  <chr>               <dbl>     <dbl>     <dbl>    <dbl>
1 (Intercept)        3.10       0.112    27.8   5.97e-23
2 participationyes   0.0367     0.169     0.218 8.29e- 1
```

```
ProgramEffectiveness %$%
  lm(testscore ~ participation) %>%
  tidy()
```

```
# A tibble: 2 x 5
  term             estimate std.error statistic  p.value
  <chr>               <dbl>     <dbl>     <dbl>    <dbl>
1 (Intercept)        21.6       0.929    23.2   1.01e-20
2 participationyes    0.873     1.40      0.622 5.39e- 1
```

For each variable, we can conclude that the treatment is balanced.

12.2 Estimate ITT Model

We estimate the ITT model below:

```
ProgramEffectiveness %$%
  lm(as.numeric(grade) ~ participation) %>%
  summary()
```

```

Call:
lm(formula = as.numeric(grade) ~ participation)

Residuals:
   Min     1Q Median     3Q    Max 
-0.571 -0.167 -0.167  0.429  0.833 

Coefficients:
                 Estimate Std. Error t value        Pr(>|t|)    
(Intercept)         1.167      0.105   11.13 0.0000000000035 ***
participationyes    0.405      0.158    2.56           0.016 *  
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

Residual standard error: 0.445 on 30 degrees of freedom
Multiple R-squared:  0.179, Adjusted R-squared:  0.151 
F-statistic: 6.53 on 1 and 30 DF,  p-value: 0.0159
```

We can reject the null hypothesis of no effect and conclude that participation increased the test score on later tests.

Chapter 13

Regression Discontinuity: Looking for Jumps in Data

We will learn techniques in R to deal with "jumps" in the data. We will use the following libraries

```
library(tidyverse)
library(broom)
```

13.1 Same slope

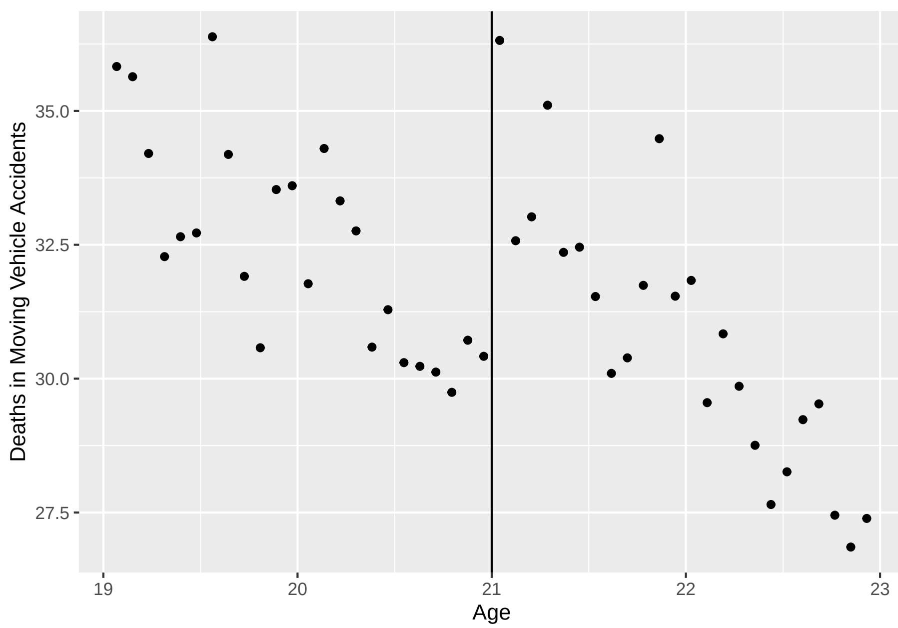
To estimate an RD model where the slope is the same before and after the cutoff value make use of the `ifelse` call in R. `ifelse` returns one value if the test condition holds and another when it doesn't. For example suppose the we create a variable, T that takes on the value 1 when another variable say X is greater than 10. Create T with the call `T -> ifelse(X > 10, 1, 0)`[1].

Let's estimate an RD model using the data from a 2009 paper by Carpenter and Dobkin about the effect of increasing the drinking age on mortality rates.[2]. Let's just look at motor vehicle deaths as a function of age.

[1]Using T as variable name is not good practice as T is an abbreviation for TRUE, so we will use D throughout our code.

[2]Carpenter, Christopher and Carlos Dobkin. "The Effects of Alcohol Consumption on Mortality: Regression Discontinuity from the Minimum Drinking Age," *American Economic Journal: Applied Econometrics*, 2009, 1:1, 164-182.
The data are available at https://github.com/jrnold/masteringmetrics/tree/master/masteringmetrics/data in mlda.rda

```
load("Data/mlda.rda")
mlda %>%
  ggplot(aes(x = agecell, y = mva)) +
  geom_point() +
  geom_vline(xintercept = 21) +
  labs(y = "Deaths in Moving Vehicle Accidents", x = "Age")
```



There appears to be a discontinuity at age 21. Let's estimate the RD model

$$mva = \beta_0 + \beta_1 T + \beta_2(agecell - 21) + \epsilon$$

where

$$T = 1 \text{ if agecell} \geq 21$$
$$T = 0 \text{ if agecell} < 21$$

We will make use of the `tidyverse` verb `mutate` and pipe operators from the `magrittr` package to create D[3].

```
mlda %>%
  mutate(D = ifelse(agecell >= 21, 1, 0)) %$%
  lm(mva ~ D + I(agecell - 21)) %>%
  summary()
```

[3]Recall we will use D to avoid the ambiguity of T as a variable name.

```
Call:
lm(formula = mva ~ D + I(agecell - 21))

Residuals:
   Min     1Q Median     3Q    Max 
-2.532 -0.849 -0.180  0.758  3.309 

Coefficients:
                Estimate Std. Error t value             Pr(>|t|)    
(Intercept)       29.356      0.429   68.39 < 0.0000000000000002 ***
D                  4.534      0.768    5.90        0.0000004338310 ***
I(agecell - 21)   -3.149      0.337   -9.34        0.0000000000043 ***
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

Residual standard error: 1.33 on 45 degrees of freedom
  (2 observations deleted due to missingness)
Multiple R-squared:  0.703, Adjusted R-squared:  0.689 
F-statistic: 53.1 on 2 and 45 DF,  p-value: 0.00000000000142
```

13.2 Varying Slopes

Let's estimate the relationship described above with a varying slopes RD model. The model now has the form:

$$mva = \beta_0 + \beta_1 T + \beta_2(agecell - 21) + \beta_3(agecell - 21)T + \epsilon$$

where

$$T = 1 \text{ if agecell} \geq 21$$

$$T = 0 \text{ if agecell} < 21$$

```
mlda %>%
  mutate(D = ifelse(agecell >= 21, 1, 0)) %$%
  lm(mva ~ D * I(agecell - 21)) %>%
  summary()
```

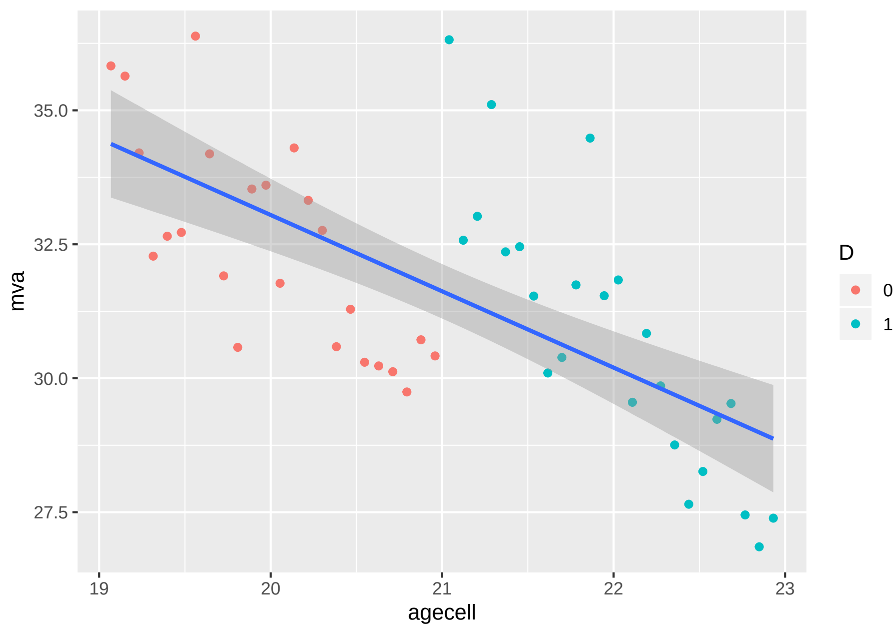
```
Call:
lm(formula = mva ~ D * I(agecell - 21))
```

```
Residuals:
   Min     1Q Median     3Q    Max 
-2.412 -0.777 -0.291  0.850  3.238 

Coefficients:
                  Estimate Std. Error t value             Pr(>|t|)    
(Intercept)         29.929      0.531   56.39 < 0.0000000000000002 ***
D                    4.534      0.751    6.04           0.00000029 ***
I(agecell - 21)     -2.568      0.466   -5.51           0.00000177 ***
D:I(agecell - 21)   -1.162      0.659   -1.76                0.085 .  
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

Residual standard error: 1.3 on 44 degrees of freedom
  (2 observations deleted due to missingness)
Multiple R-squared:  0.722, Adjusted R-squared:  0.703 
F-statistic: 38.1 on 3 and 44 DF,  p-value: 0.00000000000267
```

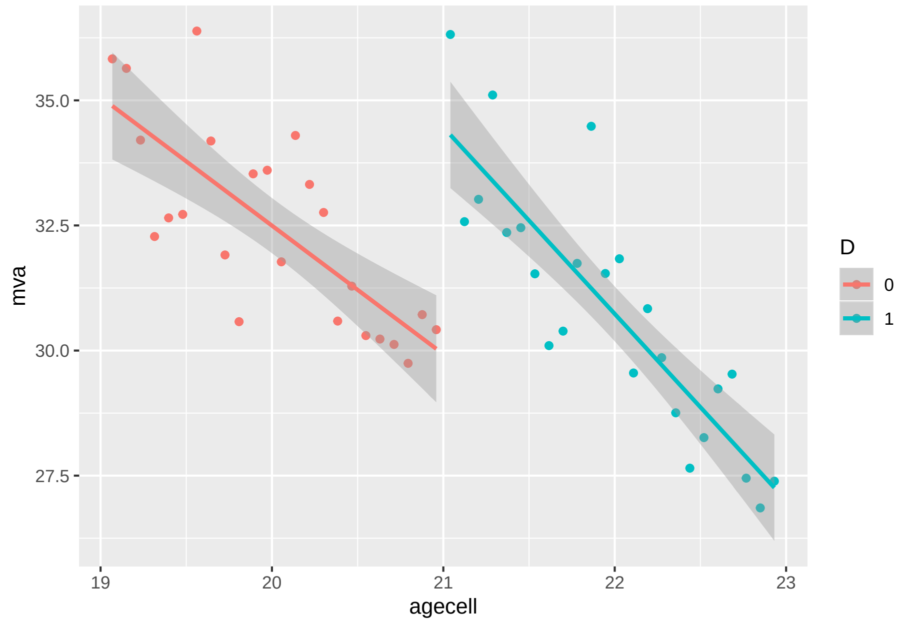
13.3 Plot RD Model

Use `ggplot` to plot the RD model. We include plots with an simple regression and an RD model.

```r
mlda %>%
  select(agecell, mva) %>%
  mutate(D = as.factor(ifelse(agecell >= 21, 1, 0))) %>%
  ggplot(aes(x = agecell, y = mva)) +
  geom_point(aes(color = D)) +
  geom_smooth(method = "lm")
```




```r
mlda %>%
  select(agecell, mva) %>%
  mutate(D = as.factor(ifelse(agecell >= 21, 1, 0))) %>%
  ggplot(aes(x = agecell, y = mva, color = D)) +
  geom_point() +
  geom_smooth(method = "lm")
```



13.4 rddtools package

We can estimate RD models with the `rddtools` package[4]. To estimate an RD model with `rddtools` first create an `rdd_data` object as follows: `rdd_data(y = df$y, x = df$x, cutpoint = C)`. Use the `rdd_data` object with `rdd_reg_lm` to estimate the model.

13.4.1 Same slope

To estimate an RD model with a constant slope call the argument `rdd_reg_lm(rdd_object, slope = "same")`

```
library(rddtools)
rdd_data(mlda$mva, mlda$agecell, cutpoint = 21) %>%
  rdd_reg_lm(slope = "same") %>%
  summary()
```

```
Call:
lm(formula = y ~ ., data = dat_step1, weights = weights)
```

[4] `?rddtools` for more.

```
Residuals:
   Min     1Q Median     3Q    Max 
-2.532 -0.849 -0.180  0.758  3.309 

Coefficients:
            Estimate Std. Error t value             Pr(>|t|)    
(Intercept)   29.356      0.429   68.39 < 0.0000000000000002 ***
D              4.534      0.768    5.90      0.0000004338310 ***
x             -3.149      0.337   -9.34      0.0000000000043 ***
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

Residual standard error: 1.33 on 45 degrees of freedom
  (2 observations deleted due to missingness)
Multiple R-squared:  0.703, Adjusted R-squared:  0.689 
F-statistic: 53.1 on 2 and 45 DF,  p-value: 0.00000000000142
```

Note the results are the same as above.

13.4.2 Varying Slopes

To estimate an RD model with varying slopes, change the slope argument to “separate”.

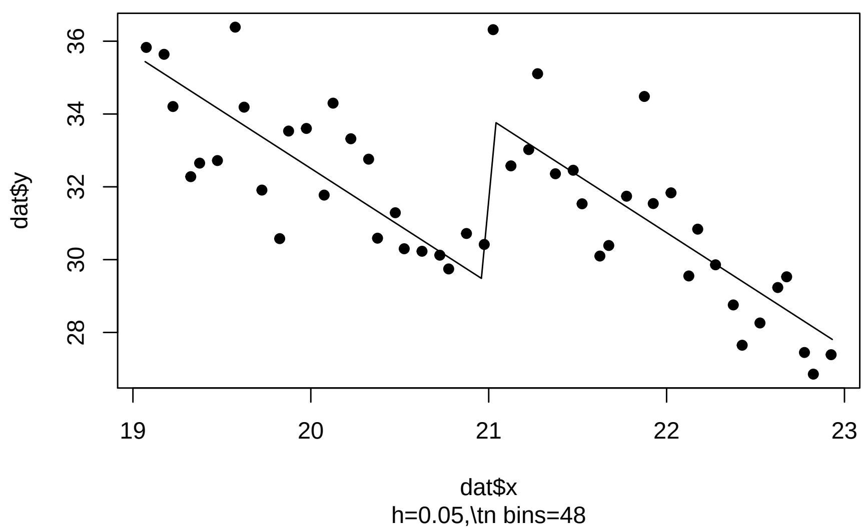
```r
rdd_data(mlda$mva, mlda$agecell, cutpoint = 21) %>%
  rdd_reg_lm(slope = "separate") %>%
  summary()
```

```
Call:
lm(formula = y ~ ., data = dat_step1, weights = weights)

Residuals:
   Min     1Q Median     3Q    Max 
-2.412 -0.777 -0.291  0.850  3.238 

Coefficients:
            Estimate Std. Error t value             Pr(>|t|)    
(Intercept)   29.929      0.531   56.39 < 0.0000000000000002 ***
D              4.534      0.751    6.04           0.00000029 ***
x             -2.568      0.466   -5.51           0.00000177 ***
x_right       -1.162      0.659   -1.76                0.085 .  
---
```

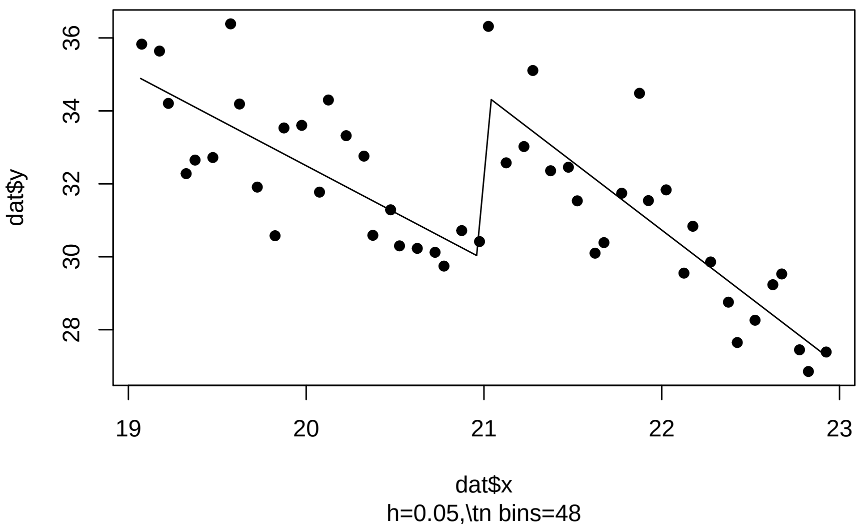
```
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

Residual standard error: 1.3 on 44 degrees of freedom
  (2 observations deleted due to missingness)
Multiple R-squared:  0.722, Adjusted R-squared:  0.703
F-statistic: 38.1 on 3 and 44 DF,  p-value: 0.00000000000267
```

13.4.3 Scatter Plot

```r
rdd_data(mlda$mva, mlda$agecell, cutpoint = 21) %>%
  rdd_reg_lm(slope = "same") %>%
  plot()
```




```r
rdd_data(mlda$mva, mlda$agecell, cutpoint = 21) %>%
  rdd_reg_lm(slope = "separate") %>%
  plot()
```



dat$y
36
34
32
30
28
19
20
21
22
23
dat$x
h=0.05,\tn bins=48

Chapter 14

Dummy Dependent Variables

We will learn techniques in R to estimate and interpret models in which the dependent variable is categorical. In particular we will learn to estimate linear probability models, probit models, and logit models. We will use the libraries below.

```
data(mtcars)
library(tidyverse)
library(magrittr)
library(broom)
```

14.1 Probit Estimation

The probit model is given by

$$Pr(Y_i = 1) = \Phi(\beta_0 + \beta_1 X_{1i})$$

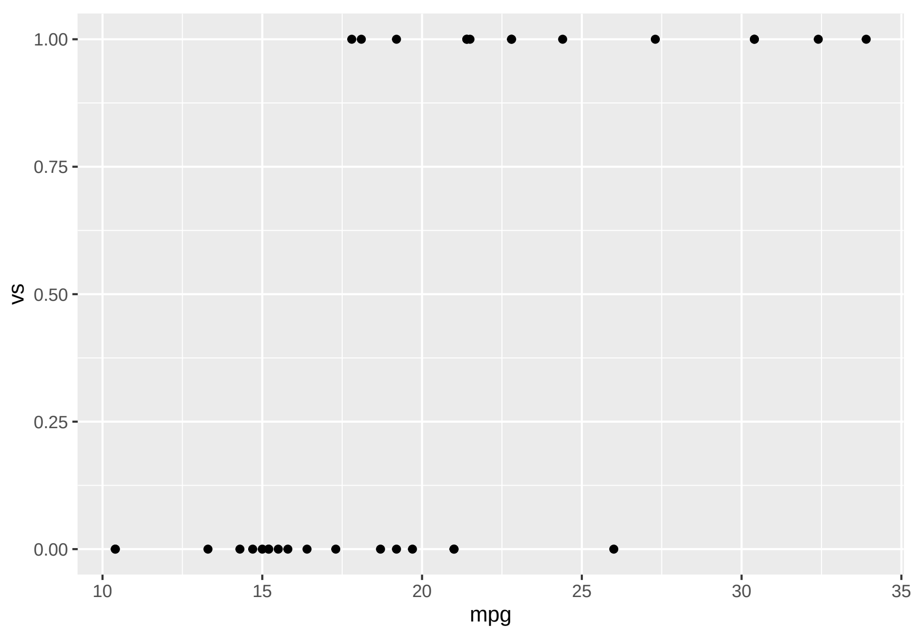
where $\Phi()$ is the standard normal CDF. Let's make use of the `mtcars`[1] data set to estimate a probit model to determine engine type as a function of mpg. Engine type, *vs*, is coded as 0 for V-shaped and 1 for straight.

14.1.1 EDA

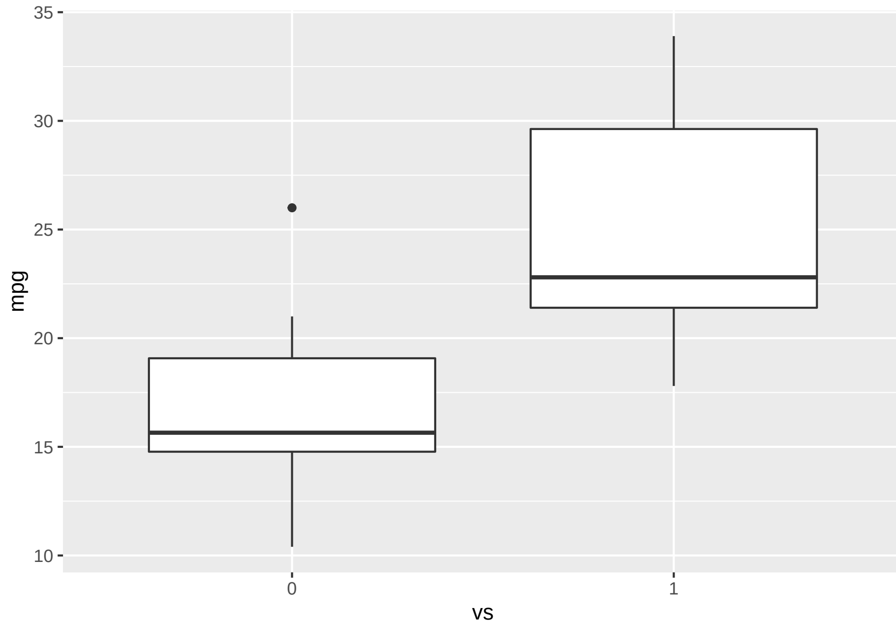
Let's look at a scatter plot and a box plot of *mpg* vs *vs*.

[1] `?mtcars` for a reminder.

```r
mtcars %>%
  ggplot(aes(x = mpg, y = vs)) +
  geom_point()
```




```r
mtcars %>%
  mutate(vs = as.factor(vs)) %>%
  ggplot(aes(x = vs, y = mpg)) +
  geom_boxplot()
```



The boxplot indicates that there is difference in mpg between straight vs v-shaped engines. Note the difference in the code between the two similar calls. We need to treat *vs* as a factor in the boxplot but not in the scatter diagram.

14.1.2 Estimate the model

Let's estimate the probit model $Pr(vs_i = 1) = \Phi(\beta_0 + \beta_1 mpg_i)$. `glm` is used to fit dummy dependent variable models.[2] To estimate the probit model `glm` requires three arguments: `formula`, `family`, and `data`. You are familiar with the data and formula arguments. The family argument is description of the error distribution. In this case our family argument will be `binomial(link = "probit")`.

```
mtcars %$%
  glm(vs ~ mpg, family = binomial(link = "probit")) %>%
  tidy()
```

```
# A tibble: 2 x 5
  term        estimate std.error statistic p.value
  <chr>          <dbl>     <dbl>     <dbl>   <dbl>
1 (Intercept)   -5.09     1.64       -3.10 0.00193
2 mpg            0.246    0.0825      2.98 0.00286
```

[2] These models are also known as limited dependent variable models or limdep models.

14.1.3 Estimated Effects

14.1.3.1 Discrete Difference

14.1.3.1.1 X_1 is Continuous

To estimate the effect of a change in the independent variable on the probability of observing the dependent variable we need to calculate the average difference between the fitted values of the model, *P1*, and the predicted values of the model when the independent variable we are interested in is changed by one standard deviation, *P2*.

Fitted values, *P1*, are easily obtained from the glm object as follows:

```
vs_glm <-
  mtcars %$%
  glm(vs ~ mpg, family = binomial(link = "probit"))
P1 <- vs_glm$fitted
```

The fitted variables have had `pnorm()` applied to the linear estimates, so they are *P1*.

To obtain marginal effects, we need to let *mpg* vary by one standard deviation and obtain the predicted values from the estimated equation. To find *P2*, the predicted values resulting from a one standard deviation change in the independent variable, we will make use of `predict.glm`. `predict.glm`[3] will require two arguments to estimate *P2*, the equation object and the newdata `predict.glm(object, newdata = df`. Unfortunately the expose pipe `%$%` does not function with `predict.glm`, so we will have to create a data frame of the changed independent variable. We will use the `dplyr` verbs `select` and `mutate` to create the new data frame. We calculate *P2* below:

```
# Create the new data
newdata <-
mtcars %>%
  dplyr::select(mpg) %>% #I used this form to avoid the conflict with select in the MA
  mutate(mpg = mpg + sd(mpg))
# Create P2
P2 <-
  predict.glm(vs_glm, newdata) %>%
  pnorm()
# Marginal Effect
mean(P2-P1)
```

```
[1] 0.339
```

[3] `?predict.glm` for more information.

So, a one standard deviation increase in *mpg* will yield a 33.89% increase in the probability that the car has straight engine.

14.1.3.1.2 Independent variable is a dummy.

Let's add *am*, transmission type, to the model. *am* is coded as 0 if the car has an automatic transmission and 1 if it has a manual transmission. First, estimate the model $Pr(vs_i = 1) = \Phi(\beta_0 + \beta_1 am + \beta_2 mpg_i)$.

```
mtcars %$%
  glm(vs ~ am + mpg, family = binomial(link = "probit"))
```

```
Call:  glm(formula = vs ~ am + mpg, family = binomial(link = "probit"))

Coefficients:
(Intercept)           am          mpg
      -7.47        -1.84         0.40

Degrees of Freedom: 31 Total (i.e. Null);  29 Residual
Null Deviance:      43.9
Residual Deviance: 20.3     AIC: 26.3
```

We will follow similar steps as those above to interpret a change from automatic to manual transmission on the probability that the engine is straight. We will estimate *P0*, the fitted values, when $am = 0$, and *P1*, the fitted values when $am = 1$.

```
# Estimate the model
vs_am_glm <-
  mtcars %$%
  glm(vs ~ am + mpg, family = binomial(link = "probit"))
# P0
newdata <-
  mtcars %>%
  dplyr::select(am, mpg) %>%
  mutate(am = 0)
P0 <-
  predict.glm(vs_am_glm, newdata) %>%
  pnorm()
# P1
newdata <-
  mtcars %>%
  dplyr::select(am, mpg) %>%
  mutate(am = 1)
```

```
P1 <-
  predict.glm(vs_am_glm, newdata) %>%
  pnorm()
mean(P1-P0)
```

```
[1] -0.269
```

A car with an manual transmission is 26.9% less likely, on average, to have a straight engine, *ceteris paribus*.

14.1.3.2 Marginal Effects

If X_1 is continuous we can estimate the marginal effects of a change in X_1 as $\phi(\hat{\beta}_0 + \hat{\beta}_1 X_{1i} + \hat{\beta}_2 X_{2i})\hat{\beta}_1$. Where $\phi()$ is the normal PDF. Let's estimate the marginal effect of *mpg* on *vs* using the model above.

```
marg_effect <-
dnorm(vs_am_glm$coef[1] + vs_am_glm$coef[2]*mtcars$am + vs_am_glm$coef[3]*mtcars$mpg)
mean(marg_effect)
```

```
[1] 0.0692
```

The marginal effect of *mpg* on type of engine is 0.069.

14.1.3.2.1 `mfx` and `margins` Packages

We can use the `mfx` and `margins` packages to estimate the marginal effect of a continuous variable directly from the model we estimate. `mfx::probitmfx(formula, data, atmean = F)` and `margins::margins(model)` are the respective function calls to estimate marginal effects from the two packages.

```
# mfx
mfx::probitmfx(vs_am_glm, mtcars, atmean = F)
```

```

Call:
mfx::probitmfx(formula = vs_am_glm, data = mtcars, atmean = F)

Marginal Effects:
       dF/dx Std. Err.     z              P>|z|
am  -0.26942   0.09154 -2.94             0.0032 **
mpg  0.06925   0.00765  9.05 <0.0000000000000002 ***
```

```
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

dF/dx is for discrete change for the following variables:

[1] "am"
```

Note that these values are identically to the ones calculated by hand above.

```
# margins
margins::margins(vs_am_glm, data = mtcars)
```

```
      am     mpg
 -0.3185 0.06925
```

The marginal effect of *mpg* is the same, while the effect of *am* is similar. `?margins` or An Introduction to 'margins' for more on the `margins` package.

14.2 Logit Estimation

The logit model takes the form $Pr(Y_i = 1) = \frac{e^{\beta_0+\beta_1 X_{1i}}}{1+e^{\beta_0+\beta_1 X_{1i}}}$ An alternative form of the logit model might be easier to interpret. With appropriate algebraic gymnastics we can write the logistic model as $\ln(\frac{p_i}{1-p_i}) = \beta_0 + \beta_1 X_{1i}$, where $\ln(\frac{p_1}{1-p_i})$ is the log of the odds ratio.

Let's estimate the model from above as a logit rather than a probit. All we need to do is change the link argument to logit to estimate the model.

```
mtcars %$%
  glm(vs ~ mpg + am, family = binomial(link = "logit"))
```

```
Call:  glm(formula = vs ~ mpg + am, family = binomial(link = "logit"))

Coefficients:
(Intercept)          mpg           am
    -12.705        0.681       -3.007

Degrees of Freedom: 31 Total (i.e. Null);  29 Residual
Null Deviance:      43.9
Residual Deviance: 20.6     AIC: 26.6
```

Suppose we'd like to know the probability that a vehicle with automatic transmission that gets 25 mpg has a straight engine. Calculate the odds ratio as $\ln(\frac{p_1}{1-p_i}) = -12.7051+0.6809*25-3.0073*0 = 4.9474$. Exponentiate both sides and solve for p. $e^{\ln(\frac{p_i}{1-p_i})} = e^{4.9474}$. We know that an exponentiated natural log is just itself so we have $\frac{p_i}{1-p_i} = 140.808$. Solving for p yields $p_i = \frac{140.808}{141.808} = .9925$. The probability we are looking for is 99.25%. So, $\hat{p} = \frac{e^{\hat{\beta}_0+\hat{\beta}_1 X_1}}{1+e^{\hat{\beta}_0+\hat{\beta}_1 X_1}}$.

14.2.1 Discrete Differences

The discrete-difference can be calculated as the difference in two probabilities. We can estimate the mean change in probability from an increase in *mpg* of 1.

```
vs_logit <-
  mtcars %$%
  glm(vs ~ mpg + am, family = binomial(link = "logit"))
# p1 are the fitted values of the regression
p1 <- vs_logit$fitted
# to calculate p2 add one to mpg and find the predicted values
newdata <-
  mtcars %>%
  dplyr::select(mpg, am) %>%
  mutate(mpg = mpg + 1)
p2 <- exp(predict(vs_logit, newdata))/(1+exp(predict(vs_logit, newdata)))
# calcualte the mean difference between the p2 and p1
mean(p2-p1)
```

```
[1] 0.0727
```

On average an increase of 1 *mpg* will increase the probability the car has straight engine by 7.3%.

14.2.2 Marginal Effects

Use the `mfx` or `margins` package to estimate the marginal effects of a change in an independent variable.

```
# mfx
mfx::logitmfx(vs_logit, mtcars, atmean = F)
```

```
Call:
mfx::logitmfx(formula = vs_logit, data = mtcars, atmean = F)
```

```
Marginal Effects:
      dF/dx Std. Err.      z  P>|z|
mpg  0.0692    0.0453  1.53 0.1267
am  -0.2618    0.0941 -2.78 0.0054 **
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

dF/dx is for discrete change for the following variables:

[1] "am"
```

```
# margins
margins::margins(vs_logit, mtcars)
```

```
     mpg      am
 0.06923 -0.3057
```

14.3 Testing Hypotheses

Let's estimate a new probit model $Pr(vs_i = 1) = \Phi(\beta_0 + \beta_1 am + \beta_2 mpg_i + \beta_3 hp_i)$ using the `mtcars` data set and test the hypothesis that our model has overall explanatory power.

$$H_0 : \beta_1 = \beta_2 = \beta_3 = 0$$

vs.

$$\text{@ least one } \beta \neq 0$$

We an estimate a restricted model and compare the likelihood ratios to the likelihood ratio of the unrestricted model and perform the LR test where $LR = 2(\log L_{UR} - \log L_R)\ \chi^2_{df}$. Where the df is equal to the number of restrictions or number of equal signs in H_0.

```
ur_model <-
  mtcars %$%
  glm(vs ~ am + mpg + hp, family = binomial(link = "probit"))
r_model <-
  mtcars %$%
  glm(vs ~ 1, family = binomial(link = "probit"))
lr <- 2*(logLik(ur_model)[1]-logLik(r_model)[1])
1 - pchisq(lr, 3)
```

```
[1] 0.000000302
```

We can reject H_0.

Instead, let's use `lrtest` from the `lmtest` package to test hypotheses about our limited dependent variable models. We can specify the restrictions as an argument in the call.

```
lmtest::lrtest(ur_model, c("am", "mpg", "hp"))
```

```
Likelihood ratio test

Model 1: vs ~ am + mpg + hp
Model 2: vs ~ 1
  #Df LogLik Df Chisq Pr(>Chisq)
1   4  -5.36
2   1 -21.93 -3  33.1  0.0000003 ***
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

Let's test the null hypothesis

$$H_0 : \beta_2 = \beta_3$$

$$H_1 : \beta_2 \neq \beta_3$$

The restricted model becomes $Pr(vs_i = 1) = \Phi(\beta_0 + \beta_1 am + \beta_2(mpg_i + hp_i))$

```
r_model <-
  mtcars %$%
  glm(vs ~ am + I(mpg + hp), family = binomial(link = "probit"))
lmtest::lrtest(ur_model, r_model)
```

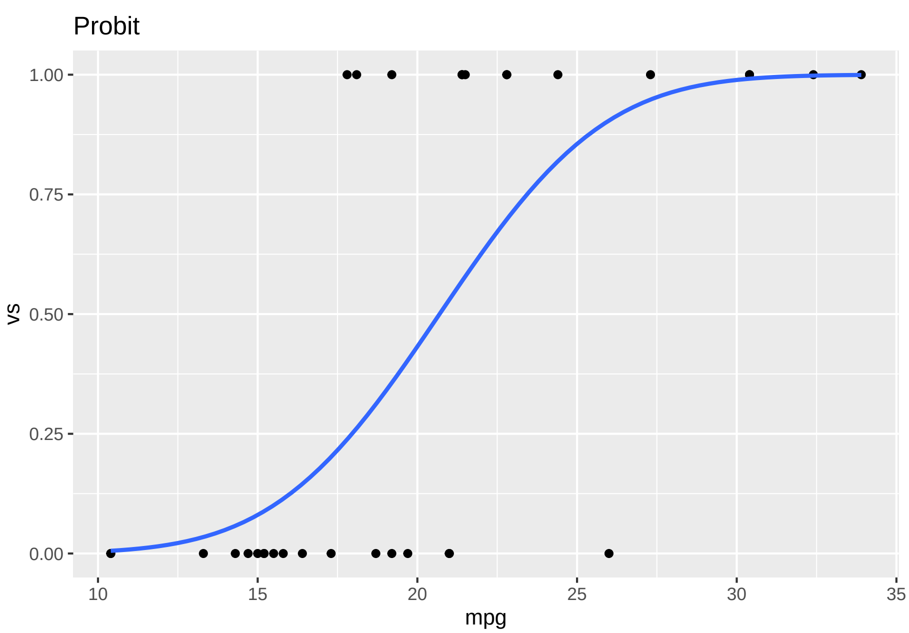
```
Likelihood ratio test

Model 1: vs ~ am + mpg + hp
Model 2: vs ~ am + I(mpg + hp)
  #Df LogLik Df Chisq Pr(>Chisq)
1   4  -5.36
2   3  -6.63 -1  2.53       0.11
```

We fail to reject H_0 and conclude that we have no evidence to believe that $\beta_2 \neq \beta_3$.

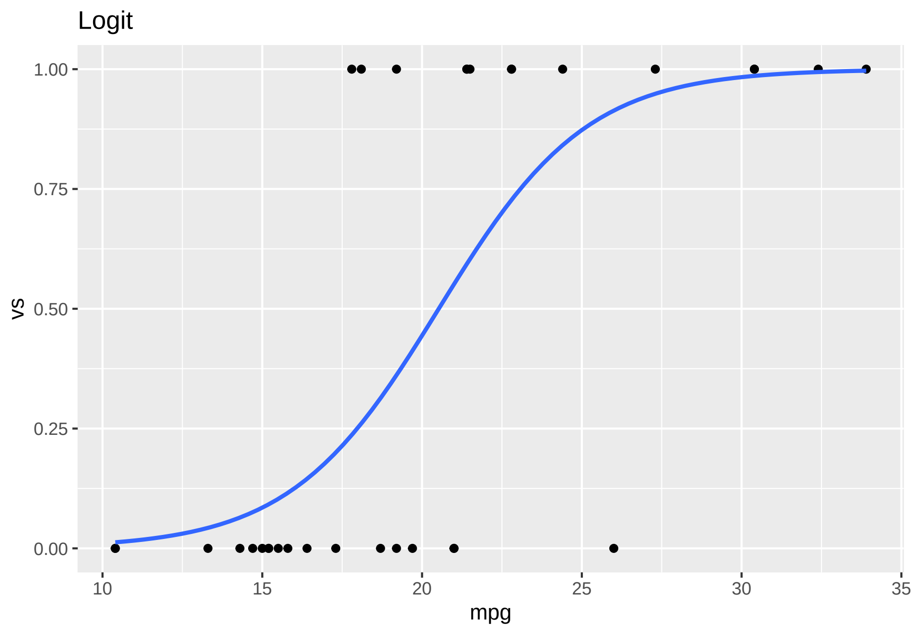
We would test hypotheses concerning logit models in same way.

14.4 Graphing Probit and Logit Models

```
mtcars %>%
  ggplot(aes(x = mpg, y = vs)) +
  geom_point() +
  geom_smooth(method = "glm", method.args=list(family=binomial(link = "probit")), se = F) +
  ggtitle("Probit")
```




```
mtcars %>%
  ggplot(aes(x = mpg, y = vs)) +
  geom_point() +
  geom_smooth(method = "glm", method.args=list(family=binomial(link = "logit")), se = F) +
  ggtitle("Logit")
```


Logit
1.00
0.75
0.50
0.25
0.00
vs
10
15
20
25
30
35
mpg

Chapter 15

Time Series: Dealing with Stickiness over Time

In this chapter we will learn to work with time–series data in R.

15.1 Time Series Objects in R

Working with time-series data in R is simplified if the data are structured as a *time-series object*. There are a host of functions and packages dedicated to time-series data. A time-series object is an R structure that contains the observations, the start and end date of the series, and information about the frequency or periodicity.

We will use the Bike Sharing Dataset[1] from the UCI Machine Learning Repository.

- The data set has 731 observations on 17 variables
  - instant: record index
  - dteday : date
  - season : season (1:spring, 2:summer, 3:fall, 4:winter)
  - yr : year (0: 2011, 1:2012)
  - mnth : month (1 to 12)
  - hr : hour (0 to 23)
  - holiday : weather day is holiday or not (extracted from http://dchr.dc.gov/page/holiday-schedule)

[1] Fanaee-T, Hadi, and Gama, Joao, 'Event labeling combining ensemble detectors and background knowledge', Progress in Artificial Intelligence (2013): pp. 1-15, Springer Berlin Heidelberg

- weekday : day of the week
- workingday : if day is neither weekend nor holiday is 1, otherwise is 0.
- weathersit :
  - 1: Clear, Few clouds, Partly cloudy, Partly cloudy
  - 2: Mist + Cloudy, Mist + Broken clouds, Mist + Few clouds, Mist
  - 3: Light Snow, Light Rain + Thunderstorm + Scattered clouds, Light Rain + Scattered clouds
  - 4: Heavy Rain + Ice Pallets + Thunderstorm + Mist, Snow + Fog
- temp : Normalized temperature in Celsius. The values are divided to 41 (max)
- atemp: Normalized feeling temperature in Celsius. The values are divided to 50 (max)
- hum: Normalized humidity. The values are divided to 100 (max)
- windspeed: Normalized wind speed. The values are divided to 67 (max)
- casual: count of casual users
- registered: count of registered users
- cnt: count of total rental bikes including both casual and registered

```
library(tidyverse)
library(broom)
library(xts)
library(magrittr)
bike_share <- read_csv("Data/day.csv")
head(bike_share)
```

```
# A tibble: 6 x 16
  instant dteday     season    yr  mnth holiday weekday workingday
    <dbl> <date>      <dbl> <dbl> <dbl>   <dbl>   <dbl>      <dbl>
1       1 2011-01-01      1     0     1       0       6          0
2       2 2011-01-02      1     0     1       0       0          0
3       3 2011-01-03      1     0     1       0       1          1
4       4 2011-01-04      1     0     1       0       2          1
5       5 2011-01-05      1     0     1       0       3          1
6       6 2011-01-06      1     0     1       0       4          1
# ... with 8 more variables: weathersit <dbl>, temp <dbl>, atemp <dbl>,
#   hum <dbl>, windspeed <dbl>, casual <dbl>, registered <dbl>, cnt <dbl>
```

We can convert `bike_share` to a time-series object using the `xts` package. A time-series object requires an index to identify each observation by its date. We create the index with `seq(as.date("YYYY-MM-DD"), by = "period",` `length.out = n`), our start date is 2011-01-01, by "days", with the number of observations as the length. After we create the date index, we will use `dplyr:select` to choose the variables (we don't need *dteday*, *yr*, or *mnth*). As always, we will make use of the pipe operator to complete the code.

```
dates <- seq(as.Date("2011-01-01"), by = "days", length.out = 731)
bike_ts <-
bike_share %>%
  select(instant, season, holiday, weekday, workingday, weathersit, temp, atemp, hum, windspeed,
  xts(dates)
bike_ts %>%
  head()
```

```
           instant season holiday weekday workingday weathersit  temp
2011-01-01       1      1       0       6          0          2 0.344
2011-01-02       2      1       0       0          0          2 0.363
2011-01-03       3      1       0       1          1          1 0.196
2011-01-04       4      1       0       2          1          1 0.200
2011-01-05       5      1       0       3          1          1 0.227
2011-01-06       6      1       0       4          1          1 0.204
           atemp   hum windspeed casual registered  cnt
2011-01-01 0.364 0.806    0.1604    331        654  985
2011-01-02 0.354 0.696    0.2485    131        670  801
2011-01-03 0.189 0.437    0.2483    120       1229 1349
2011-01-04 0.212 0.590    0.1603    108       1454 1562
2011-01-05 0.229 0.437    0.1869     82       1518 1600
2011-01-06 0.233 0.518    0.0896     88       1518 1606
```

We see that the data now have a date index indicating to which date each observations belongs.

15.2 Detecting Autocorrelation

Let's estimate the total number of riders as a function of time, $cnt = \beta_0 + \beta_1 instant + \epsilon$, and test the residuals for first order auto correlation using the auxiliary regression approach.

```
bike_ts %$%
  lm(cnt ~ instant)
```

```
Call:
lm(formula = cnt ~ instant)
```

```
Coefficients:
(Intercept)      instant
    2392.96         5.77
```

```
# retrieve the residuals as e
e <-
  bike_ts %$%
  lm(cnt ~ instant)$residuals
# auxiliary regression
lm(e ~ lag(e,1)) %>%
  tidy()
```

```
# A tibble: 2 x 5
  term        estimate std.error statistic   p.value
  <chr>          <dbl>     <dbl>     <dbl>     <dbl>
1 (Intercept)   -2.06    37.0      -0.0558 9.56e-  1
2 lag(e, 1)      0.752    0.0247   30.5    2.92e-132
```

The *t-statistic* on $\hat{\rho}_{t-1}$ is 30.50 so we can reject the null hypothesis of no autocorrelation in the error term.

15.3 Correcting Autocorrelation

15.3.1 Newey-West

The `sandwich` package contains a function to estimate Newey-West standard errors.[2] The output of the function call is the corrected variance-covariance matrix. We still need to calculate *t-statistics* based on the corrected variance-covariances. We will use the `lmtest` package to perform this test. `coeftest(lm_object, vcov = variance-covariance_matrix)`.

```
# estimate the model (the lm_object)
bike_lm <- lm(cnt ~ instant, bike_ts)
bike_lm %>%
  tidy()
```

```
# A tibble: 2 x 5
  term        estimate std.error statistic  p.value
  <chr>          <dbl>     <dbl>     <dbl>    <dbl>
1 (Intercept)  2393.     112.         21.4 1.91e-79
2 instant         5.77     0.264      21.8 1.02e-81
```

[2]Unfortunately the pipe operator does not play nice with `NeweyWest`.

```
# determine the number of lags for the Newey-West correction
lags <- length(bike_ts$instant)^(.25)
nw_vcov <- sandwich::NeweyWest(bike_lm, lag = lags, prewhite = F, adjust = T)
# model with corrected errors
lmtest::coeftest(bike_lm, vcov. = nw_vcov)
```

```

t test of coefficients:

            Estimate Std. Error t value            Pr(>|t|)
(Intercept) 2392.961    221.782   10.79 <0.0000000000000002 ***
instant        5.769      0.646    8.93 <0.0000000000000002 ***
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

15.3.2 Cochrane-Orcutt

The `orcutt` package allows us to use the Cochrane-Orcutt method to ρ difference the data to produce corrected standard errors using `cochrane.orcutt(lm_oject)`

```
bike_lm %>%
  orcutt::cochrane.orcutt() %>%
  tidy()
```

```
# A tibble: 2 x 5
  term        estimate std.error statistic  p.value
  <chr>          <dbl>     <dbl>     <dbl>    <dbl>
1 (Intercept)  2457.     301.         8.17 1.39e-15
2 instant         5.57     0.707      7.88 1.22e-14
```

15.4 Dynamic Models

Using a time-series object makes running dynamic models as easy as calling the argument `lag(variable_name, number_of_lags)`. Suppose we'd like to estimate the a lagged version of the model we have been using with the form $cnt_t = \beta_0 + \beta_1 instant_t + \beta_2 temp_{t-1} + \epsilon$. We want to see if yesterday's weather affects today's rentals.

```
bike_lm_dyn <- lm(cnt ~ instant + lag(temp, 1), bike_ts)
bike_lm_dyn %>%
 tidy
```

```
# A tibble: 3 x 5
  term         estimate std.error statistic   p.value
  <chr>           <dbl>     <dbl>     <dbl>     <dbl>
1 (Intercept)   -189.     128.        -1.48 1.39e-  1
2 instant          5.02     0.193     26.0  4.77e-106
3 lag(temp, 1)  5769.     222.        25.9  1.31e-105
```

15.5 Dickey-Fuller Test

The `tseries` package includes an augmented Dickey-Fuller test, `adf.test(time_series)`.

```
bike_ts$cnt %>%
  tseries::adf.test()
```

```

    Augmented Dickey-Fuller Test

data:  .
Dickey-Fuller = -2, Lag order = 9, p-value = 0.7
alternative hypothesis: stationary
```

We conclude that *cnt* is non-stationary.

15.6 First Differencing

A simple solution to non-stationarity is to use first differences of values, i.e., $\Delta y_t = y_t - y_{t-1}$. `diff(x, ...)` makes this easy with a time-series object. Let's test Δy_t for stationarity.

```
bike_ts$cnt %>%
  diff() %>%
  tseries::na.remove() %>% # first differencing introduces NA's into the data
  tseries::adf.test()
```

```

    Augmented Dickey-Fuller Test

data:  .
Dickey-Fuller = -14, Lag order = 8, p-value = 0.01
alternative hypothesis: stationary
```

We can reject the null-hypothesis of non-stationarity. So let's estimate the model $\Delta cnt_t = \beta_0 + \beta_1 \Delta temp_t + \eta_t$

```
lm(diff(cnt) ~ diff(temp), bike_ts) %>%
  tidy()
```

```
# A tibble: 2 x 5
  term        estimate std.error statistic  p.value
  <chr>          <dbl>     <dbl>     <dbl>    <dbl>
1 (Intercept)     3.24      38.0    0.0852 9.32e- 1
2 diff(temp)   4838.       651.     7.43   3.09e-13
```

```
# Compare to same equation in the levels.
lm(cnt ~ temp, bike_ts) %>%
  tidy()
```

```
# A tibble: 2 x 5
  term        estimate std.error statistic  p.value
  <chr>          <dbl>     <dbl>     <dbl>    <dbl>
1 (Intercept)    1215.      161.      7.54 1.43e-13
2 temp           6641.      305.     21.8  2.81e-81
```

Chapter 16

Advanced OLS

We will prove the Gauss–Markov Theorem with matrix algebra and learn how to generate random numbers in R.

16.1 Derive OLS estimator (Matrix Form)

Suppose we have a linear statistical model $y = XB + e$. Let y is an n x 1 vector of observations on the dependent variable

$$y = \begin{bmatrix} y_1 \\ y_2 \\ \vdots \\ y_n \end{bmatrix}$$

.

Let X be an n x k matrix of observations on k - 1 independent variables

$$X = \begin{bmatrix} 1 & X_{21} & X_{31} & \cdots & X_{k1} \\ 1 & X_{22} & X_{32} & \cdots & X_{k2} \\ & & \ddots & & \\ 1 & X_{2n} & X_{3n} & \cdots & X_{kn} \end{bmatrix}$$

Let $\hat{\beta}$ be a k x 1 vector of estimators for B.

$$\hat{\beta} = \begin{bmatrix} \hat{\beta}_1 \\ \hat{\beta}_2 \\ \vdots \\ \hat{\beta}_k \end{bmatrix}$$

Let e be an n x 1 matrix of residuals.

$$e = \begin{bmatrix} e_1 \\ e_2 \\ \vdots \\ e_n \end{bmatrix}$$

We want to find $\hat{\beta}$ such that $\sum e^2$ is a minimum. The estimated equation is

$$\hat{y} = X\hat{\beta} + \hat{e}$$

The ordinary least squares estimator is $\hat{\beta}$ such that $\hat{e}^T\hat{e}$ is minimized. Solving for $\hat{e}$ yields.

$$\hat{e} = y - X\hat{\beta}$$

So,

$$\begin{aligned}
\hat{e}^T\hat{e} &= (y - X\hat{\beta})^T(y - X\hat{\beta}) \\
&= y^Ty - \hat{\beta}X^Ty - y^TX\hat{\beta} + \hat{\beta}X^TX\hat{\beta} \\
&= y^Ty - 2\hat{\beta}X^Ty + \hat{\beta}X^TX\hat{\beta}
\end{aligned}$$

Take the partial derivative of $\hat{e}^T\hat{e}$ with respect to $\hat{\beta}$ and set it equal to 0.

$$\begin{aligned}
\frac{\partial \hat{e}^T\hat{e}}{\partial \hat{\beta}} &= -2X^Ty + 2X^TX\hat{\beta} = 0 \\
&= -X^Ty + X^TX\hat{\beta} = 0 \\
X^TX\hat{\beta} &= X^Ty
\end{aligned}$$

Pre-multiple both sides by $(X^TX)^{-1}$

$$\begin{aligned}
(X^TX)^{-1}(X^TX)\hat{\beta} &= (X^TX)^{-1}X^Ty \\
I\hat{\beta} &= (X^TX)^{-1}X^Ty \\
\hat{\beta} &= (X^TX)^{-1}X^Ty
\end{aligned}$$

16.1.1 Example

Suppose we have 14 observations on the dependent y:

$$\begin{bmatrix} 1065 \\ 1254 \\ 1300 \\ 1577 \\ 1600 \\ 1750 \\ 1800 \\ 1870 \\ 1935 \\ 1948 \\ 2254 \\ 2600 \\ 2800 \\ 3000 \end{bmatrix}$$

We also have 14 observations on a single independent variable

$$X = \begin{bmatrix} 1 & 199.9 \\ 1 & 228 \\ 1 & 235 \\ 1 & 285 \\ 1 & 239 \\ 1 & 293 \\ 1 & 285 \\ 1 & 365 \\ 1 & 295 \\ 1 & 290 \\ 1 & 385 \\ 1 & 505 \\ 1 & 425 \\ 1 & 425 \\ 1 & 415 \end{bmatrix}$$

Let's find the $\begin{bmatrix} \hat{\beta}_0 \\ \hat{\beta}_1 \end{bmatrix}$ step by step using matrix operators in R. The matrix operators we need are in the table below.

Operator	What it does
`%*%`	matrix multiplication
`t()`	transposes a matrix
`solve()`	inverts a matrix
`crossprod()`	performs t(x) %*% x

Let's step through the calculations one at a time.

16.1.1.1 create X and y

```
# create the 1 x 14 column vector y
y <- c(199.9, 228, 235, 285, 239, 293, 285, 365, 295, 290, 385, 505, 425, 415)
# create the 2 x 14 matrix X
# cbind combines vectors by columns into a matrix
X <- cbind(c(rep(1,14)), # rep() repeats a value a given number of times
          c(1065, 1254, 1300,1577,1600,1750,1800,1870,1935,1948,2254, 2600,2800,3000))
y
```

```
 [1] 200 228 235 285 239 293 285 365 295 290 385 505 425 415
```

```
X
```

```
      [,1] [,2]
 [1,]    1 1065
 [2,]    1 1254
 [3,]    1 1300
 [4,]    1 1577
 [5,]    1 1600
 [6,]    1 1750
 [7,]    1 1800
 [8,]    1 1870
 [9,]    1 1935
[10,]    1 1948
[11,]    1 2254
[12,]    1 2600
[13,]    1 2800
[14,]    1 3000
```

16.1.1.2 create X transpose

```
X_T <- t(X)
X_T
```

```
     [,1] [,2] [,3] [,4] [,5] [,6] [,7] [,8] [,9] [,10] [,11] [,12] [,13]
[1,]    1    1    1    1    1    1    1    1    1     1     1     1     1
[2,] 1065 1254 1300 1577 1600 1750 1800 1870 1935  1948  2254  2600  2800
     [,14]
[1,]     1
[2,]  3000
```

16.1.1.3 create X transpose X

```
X_t_X <- X_T %*% X
X_t_X
```

```
      [,1]     [,2]
[1,]    14    26753
[2,] 26753 55462515
```

```
# alternatively we could call crossprod
X_T_X <- crossprod(X)
X_T_X
```

```
      [,1]     [,2]
[1,]    14    26753
[2,] 26753 55462515
```

16.1.1.4 invert X transpose X

```
X_T_X_inverse <- solve(X_T_X)
X_T_X_inverse
```

```
         [,1]        [,2]
[1,]  0.91293 -0.00044036
[2,] -0.00044  0.00000023
```

16.1.1.5 X transpose X inverse X Transpose

```
X_T_X_inverse_X_T <- X_T_X_inverse %*% X_T
X_T_X_inverse_X_T
```

```
          [,1]      [,2]      [,3]      [,4]       [,5]       [,6]
[1,]  0.443944  0.360715  0.340459  0.218478  0.2083499  0.1422955
[2,] -0.000195 -0.000151 -0.000141 -0.000077 -0.0000717 -0.0000371
           [,7]        [,8]       [,9]      [,10]      [,11]     [,12]
[1,]  0.1202774  0.08945199 0.06082841 0.05510370 -0.0796473 -0.232013
[2,] -0.0000256 -0.00000943 0.00000555 0.00000854  0.0000791  0.000159
         [,13]     [,14]
[1,] -0.320085 -0.408158
[2,]  0.000205  0.000251
```

16.1.1.6 X transpose X inverse X Transpose y

```
beta <- X_T_X_inverse %*% X_T %*% y
beta
```

```
       [,1]
[1,] 52.351
[2,]  0.139
```

This is the matrix of our estimates for B. So, the equation we have estimated is $\hat{y} = 52.351 + 0.139X$

16.2 Gauss–Markov Theorem

The Gauss-Markov theorem proves that among the class of linear estimators of B, the ordinary least squares estimator has the minimum variance. That is, the OLS estimator is BLUE: the **B**est, **L**inear, **U**nbiased, **E**stimator. Below is the proof.

16.2.1 OLS estimator is linear

Since $(X^TX)^{-1}X^T$ is a matrix of fixed numbers, $\hat{\beta}$ is linear combination of X and y.

16.2.2 OLS estimator is unbiased

$\hat{\beta}$ is an unbiased estimator of B if $E(\hat{\beta}) = B$

$$E(\hat{\beta}) = E\left[(X^TX)^{-1}X^Ty)\right]$$

Substituting for $y = XB + e$

$$\begin{aligned}
E(\hat{\beta}) &= E\left[(X^TX)^{-1}X^T(XB + e)\right] \\
&= E\left[(X^TX)^{-1}X^T(XB + e)\right] \\
&= E\left[(X^TX)^{-1}(X^TX)B + (X^TX)^{-1}X^Te\right] \\
&= E\left[B + (X^TX)^{-1}X^Te\right] \\
&= E(B) + E\left[(X^TX)^{-1}X^Te\right]
\end{aligned}$$

Since B is a matrix of parameters it is equal to its expected value so

$$E(\hat{\beta}) = B + E\left[(X^TX)^{-1}X^Te\right]$$

For $\hat{\beta}$ to be an unbiased estimator of B, $E\left[(X^TX)^{-1}X^Te\right]$ must be 0. If the X is a matrix of non-stochastic observations on the independent variables, then

$$E\left[(X^TX)^{-1}X^Te\right] = (X^TX)^{-1}X^TE(e)$$

Since $E(e) = 0$, $\hat{\beta}$ is an unbiased estimator of B. If we assume that X is fixed in repeated samples, X is non-stochastic.

In the wild X is not fixed in repeated samples, therefore X is stochastic. So if $E\left[(X^TX)^{-1}X^Te\right] \neq 0$ X and e are correlated. This is the problem of endogeneity.

16.2.3 Variance-Covariance is a minimum

Let's find the "variance" of the OLS estimators.[1] var-cov$(\hat{\beta})$.

$$\begin{aligned}
\text{var-cov}(\hat{\beta}) &= E\left[(\hat{\beta}-\beta)(\hat{\beta}-\beta)^T\right] \\
\text{recall from above} & \\
\hat{\beta} &= B + (X^TX)^{-1}X^Te \\
\text{so} & \\
\hat{\beta} - B &= (X^TX)^{-1}X^Te \\
(\hat{\beta}-\beta)(\hat{\beta}-\beta)^T &= \left[(X^TX)^{-1}X^Te\right]\left[(X^TX)^{-1}X^Te\right]^T \\
&= (X^TX)^{-1}X^Tee^TX(X^TX)^{-1} \\
\text{thus} & \\
\text{var-cov}(\hat{\beta}) &= E\left[(X^TX)^{-1}X^Tee^TX(X^TX)^{-1}\right] \\
\text{if X is exogenous} & \\
&= (X^TX)^{-1}X^TE(ee^T)X(X^TX)^{-1} \\
\text{since } E(ee^T) = \sigma^2 I & \\
&= (X^TX)^{-1}X^T\sigma^2IX(X^TX)^{-1} \\
&= \sigma^2(X^TX)^{-1}X^TIX(X^TX)^{-1} \\
&= \sigma^2(X^TX)^{-1}X^TX(X^TX)^{-1} \\
\text{var-cov}(\hat{\beta}) &= \sigma^2(X^TX)^{-1}
\end{aligned}$$

To prove that this variance is the minimum variance among the class of linear estimators, we will show that any other unbiased linear estimator must have a

[1] Recall that the variance of a random variable, X, is essentially the mean of the squared deviations. Or $\text{Var}(X) = E(X-\mu)^2$

larger variance. Let $\tilde{\beta}$ be any other linear estimator of B, which can be written as$\tilde{\beta} = \left[(X^TX)^{-1}X^T + C)\right] y$ where C is a matrix of constants. Substituting $y = X\beta + e$ yields

$$\begin{aligned}\tilde{\beta} &= \left[(X^TX)^{-1}X^T + C)\right] [XB + e] \\ &= (X^TX)^{-1}X^TXB + CXB + (X^TX)^{-1}X^Te + Ce \\ &= B + CXB + (X^TX)^{-1}Xe + Ce\end{aligned}$$

16.3 Probability Distributions in R

Every distribution that R handles has four functions. There is a root name, for example, the root name for the normal distribution is `norm`. This root is prefixed by one of the letters

- `p` for "probability", the cumulative distribution function (c. d. f.)
- `q` for "quantile", the inverse c. d. f.
- `d` for "density", the density function (p. f. or p. d. f.)
- `r` for "random", a random variable having the specified distribution

For the normal distribution, these functions are `pnorm`, `qnorm`, `dnorm`, and `rnorm`.

For a continuous distribution (like the normal), the most useful functions for doing problems involving probability calculations are the "p" and "q" functions (c. d. f. and inverse c. d. f.), because the the density (p. d. f.) calculated by the "d" function can only be used to calculate probabilities via integrals and R doesn't do integrals.

For a discrete distribution (like the binomial), the "d" function calculates the density (p. f.), which in this case is a probability $f(x) = P(X = x)$ and hence is useful in calculating probabilities.

R has functions to handle many probability distributions. The table below gives the names of the functions for a few of the distributions.

Distribution	Functions
Binomial	pbinom qbinom dbinom rbinom
Chi-Square	pchisq qchisq dchisq rchisq
F	pf qf df rf
Normal	rnorm qnorm dnorm rnorm
Student t	pt qt dt rt
Uniform	punif qunif dunif runif

You can find the specific argumnets for each with `?args(pnorm)`, for example. Or help with `?pt`, for example.

16.3.1 Obtaining Critical Statistics

Make use of the functions above to obtain critical statistics for hypothesis testing. For example, suppose we wanted to perform a two-tail *t-test* at the the $\alpha = 5\%$ level of significance with $df = 132$ degrees of freedom. We would call `qt(p = .975, df = 132, lower.tail = TRUE)`. This would return $t_{.05,132} = 1.978$

16.3.2 Generating Random Numbers

Supposed we'd like to generate a sample of size $n = 10$ random values of X such that $X \sim N(12, 5)$, we would call `rnorm(n = 10, mean = 12, sd = 5)`. This would return $11.039, 7.76, 21.698, 0.138, 7.215, 8.637, 17.865, 13.343, 10.565, 11.444$.

Chapter 17

Advanced Panel Data

In this chapter we will learn techinques in R for panel data where there might be serially correlated errors, temporal dependence with a lagged dependent variable, and random effects models.

17.1 The Data

We will make use of the `Cigar` dataset from the `plm` package for this chapter. `Cigar` is a panel of 46 U.S. states over the period 1963-1992. The variables are:

- state - State number
- year
- price - the price per pack of cigarettes in cents
- pop - state population in thousands
- pop16 - state population over the age of 16 in thousands
- cpi - consumer price index (1983=100)
- ndi - per capita disposable income in dollars
- sales - cigarette sales per capita in packs
- pimin - minimum price in adjoining states per pack of cigarettes in cents

```
library(plm)
data("Cigar")
```

The `plm` packages offers many functions to simplify the handling of advanced panel data.

17.2 Variation within Units

`dplyr` verbs make checking for variation within units across multiple variables relatively simple. First we use `group-by` so that any functions will be applied to each state individually. `summarize_all` will apply a function to each variable.

```
library(tidyverse)
library(broom)
library(magrittr)
# Check for variaton by state.
Cigar %>%
  group_by(state) %>% # ensures that subsequent functions will be performed by state
  select(price, pop, pop16, cpi, ndi, sales, pimin) %>%
  summarise_all(sd) # sd is standard deviation
```

```
# A tibble: 46 x 8
   state price    pop  pop16   cpi   ndi sales pimin
   <int> <dbl>  <dbl>  <dbl> <dbl> <dbl> <dbl> <dbl>
 1     1  39.6  269.   319.   37.1 3944.  10.7  38.0
 2     3  40.7  748.   622.   37.1 4364.  13.3  40.8
 3     4  41.5  189.   198.   37.1 3814.  10.8  37.6
 4     5  48.3 3995.  3531.   37.1 5264.  20.6  40.9
 5     7  48.0  139.   218.   37.1 6624.  18.2  43.7
 6     8  40.6   58.1   66.6  37.1 4921.  13.8  38.3
 7     9  45.4   79.5   34.7  37.1 6223.  59.3  38.2
 8    10  45.4 2562.  2229.   37.1 4889.  10.3  37.6
 9    11  37.6  783.   734.   37.1 4478.  10.0  35.9
10    13  41.6  136.   113.   37.1 3900.  14.3  37.5
# ... with 36 more rows
```

```
# Check for variation by year.
Cigar %>%
  group_by(year) %>%
  select(-year, -state) %>% # the "-" indicates variables to be removed
  summarise_all(sd)
```

```
# A tibble: 30 x 8
    year price   pop pop16   cpi   ndi sales pimin
   <int> <dbl> <dbl> <dbl> <dbl> <dbl> <dbl> <dbl>
 1    63  1.86 4116. 2855.     0  387.  33.1  1.21
 2    64  1.89 4185. 2903.     0  413.  32.8  1.41
 3    65  1.97 4253. 2953.     0  417.  33.1  1.68
 4    66  2.43 4285. 2982.     0  438.  41.6  2.25
 5    67  2.32 4329. 3024.     0  460.  37.9  2.10
```

```
 6    68  2.39 4372. 3069.     0  486.  33.4  2.23
 7    69  3.13 4420. 3139.     0  504.  32.0  2.28
 8    70  3.65 4463. 3198.     0  537.  32.0  3.24
 9    71  3.71 4513. 3260.     0  559.  33.5  3.27
10    72  4.45 4536. 3306.     0  555.  36.2  4.07
# ... with 20 more rows
```

CPI is the only variable with a standard deviation of 0 for all units. As would be expected, CPI should not vary within year.

`pvar` from the `plm` package will perform the task of checking for variation.

```
pvar(Cigar)
```

```
no time variation:       state
no individual variation: year cpi
```

17.3 Two-Way Fixed Effects Model

Let's estimate cigarette demand as:

$$sales_{it} = \beta_0 + \beta_1 price_{it} + \beta_2 pop16_{it} + \beta_3 ndi_{it} + \alpha_i + \tau_t + \nu_{it}$$

We would expect $\beta_1 < 0$, $\beta_2 > 0$, and $\beta_3 < 0$ if cigarettes are an inferior good[1].

```
cigar_plm <- plm(sales ~ price + pop16 + ndi,
                 data = Cigar, # recall plm does not play nice with the expose pipe, %$%
                 index = c("state", "year"),
                 model = "within",
                 effect = "twoways")
cigar_plm %>%
  tidy()
```

```
# A tibble: 3 x 5
  term  estimate std.error statistic  p.value
  <chr>    <dbl>     <dbl>     <dbl>    <dbl>
1 price -0.841    0.0750      -11.2  6.35e-28
2 pop16  0.00114  0.000547      2.07 3.83e- 2
3 ndi   -0.00557  0.000445    -12.5  5.72e-34
```

Each of the coefficients has the expected sign and is significant at the 5% level.

[1]If cigarettes are a normal good we'd expect $\beta_3 > 0$

17.4 Testing for autocorrelation

Testing for autocorrelation is done by testing the following hypothesis:

$$H_0 : \rho = 0$$

$$H_1 : \rho \neq 0$$

```
Cigar %>%
  glimpse()
```

```
Observations: 1,380
Variables: 9
$ state <int> 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1,...
$ year  <int> 63, 64, 65, 66, 67, 68, 69, 70, 71, 72, 73, 74, 75, 76, ...
$ price <dbl> 28.6, 29.8, 29.8, 31.5, 31.6, 35.6, 36.6, 39.6, 42.7, 42...
$ pop   <dbl> 3383, 3431, 3486, 3524, 3533, 3522, 3531, 3444, 3481, 35...
$ pop16 <dbl> 2236, 2277, 2328, 2370, 2394, 2405, 2412, 2395, 2444, 24...
$ cpi   <dbl> 30.6, 31.0, 31.5, 32.4, 33.4, 34.8, 36.7, 38.8, 40.5, 41...
$ ndi   <dbl> 1558, 1684, 1810, 1915, 2024, 2202, 2377, 2591, 2785, 30...
$ sales <dbl> 93.9, 95.4, 98.5, 96.4, 95.5, 88.4, 90.1, 89.8, 95.4, 10...
$ pimin <dbl> 26.1, 27.5, 28.9, 29.5, 29.6, 32.0, 32.8, 34.3, 35.8, 37...
```

Our data are organized by unit by year, so we can estimate $\hat{\rho}$ directly. First, obtain the residuals, e, from the estimated equation. Estimate the equation $e = \rho e_{i,t-1} + \eta_{it}$.

```
# Obtain the residuals
Cigar$e <- cigar_plm$residuals
# test of rho hat
aux_1 <-
  Cigar %$%
  lm(e ~ -1 + lag(e)) # -1 removes the constant.
aux_1 %>%
  tidy()
```

```
# A tibble: 1 x 5
  term   estimate std.error statistic p.value
  <chr>     <dbl>     <dbl>     <dbl>   <dbl>
1 lag(e)    0.888    0.0126      70.3       0
```

We can reject the null hypothesis at the 1% level.

We can also check for autocorrelation with the LM test by estimating the model

$$\hat{\epsilon}_{it} = \rho\hat{\epsilon}_{i,t-1} + \gamma_1 price_{it} + \gamma_2 pop16_{it} + \gamma_3 ndi_{it} + \eta_{it}$$

where $nR^2 \sim \chi^2_{df=1}$.

```
aux_2 <-
plm(e ~ lag(e) + price + pop16 + ndi,
    data = Cigar,
    index = c("state", "year"),
    model = "within",
    effect = "twoways")
nR2 <-
  aux_2 %>%
  r.squared *
  aux_2$df.residual
nR2 %>%
  pchisq(1, lower.tail = F)
```

```
[1] 0.0000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000
```

Again, we can reject the null hypothesis of no autocorrelation.

`pwartest` from the `lpm` package allows us to test for autocorrelation (`?pwartest` for relevant arguments).

```
pwartest(cigar_plm)
```

```

    Wooldridge's test for serial correlation in FE panels

data:  cigar_plm
F = 10212, df1 = 1, df2 = 1332, p-value <0.0000000000000002
alternative hypothesis: serial correlation
```

We reject the null hpothesis of no autocorrelation.

17.5 Estimating $\hat{\rho}$

To correct for autocorrelation we need an estimate of $\hat{\rho}$. We can estimate $\hat{\rho}$ using either auxiliary regression from above.

```
aux_1 %>%
   tidy()
```

```
# A tibble: 1 x 5
  term   estimate std.error statistic p.value
  <chr>     <dbl>     <dbl>     <dbl>   <dbl>
1 lag(e)    0.888    0.0126      70.3       0
```

Our estimate of $\hat{\rho}$ is `0.888` is 0.888.

```
aux_2 %>%
  tidy()
```

```
# A tibble: 4 x 5
  term    estimate std.error statistic    p.value
  <chr>      <dbl>     <dbl>     <dbl>      <dbl>
1 lag(e)  0.894     0.0127      70.2   0
2 price   0.156     0.0339       4.59  0.00000482
3 pop16  -0.000203  0.000254    -0.802 0.423
4 ndi     0.000517  0.000201     2.57  0.0102
```

Our estimate of $\hat{\rho}$ is 0.894.

17.6 Estimate a ρ-Transformed Model

We can manually transform the data and compare the transformed model to the non-transformed model.

```
rho_hat <- aux_2$coefficients[1] # set rho_hat to the coef of lagged e in aux_2
plm(I(sales - rho_hat*lag(sales)) ~
      I(price - rho_hat*lag(price)) +
      I(pop - rho_hat*lag(pop16)) +
      I(ndi - rho_hat*lag(ndi)),
    data = Cigar,
    index = c("state", "year"),
    model = "within",
    effect = "twoways") %>%
  summary()
```

```
Twoways effects Within Model

Call:
plm(formula = I(sales - rho_hat * lag(sales)) ~ I(price - rho_hat *
    lag(price)) + I(pop - rho_hat * lag(pop16)) + I(ndi - rho_hat *
    lag(ndi)), data = Cigar, effect = "twoways", model = "within",
```

```
    index = c("state", "year"))

Balanced Panel: n = 46, T = 29, N = 1334

Residuals:
   Min. 1st Qu.  Median 3rd Qu.    Max.
-27.611  -1.992   0.208   1.956  61.268

Coefficients:
                                 Estimate Std. Error t-value
I(price - rho_hat * lag(price)) -0.329602   0.047324   -6.96
I(pop - rho_hat * lag(pop16))   -0.000730   0.000594   -1.23
I(ndi - rho_hat * lag(ndi))      0.000988   0.000682    1.45
                                       Pr(>|t|)
I(price - rho_hat * lag(price)) 0.0000000000053 ***
I(pop - rho_hat * lag(pop16))              0.22
I(ndi - rho_hat * lag(ndi))                0.15
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

Total Sum of Squares:    35700
Residual Sum of Squares: 34300
R-Squared:      0.0414
Adj. R-Squared: -0.0166
F-statistic: 18.0908 on 3 and 1257 DF, p-value: 0.000000000017
```

```r
cigar_plm %>%
  tidy()
```

```
# A tibble: 3 x 5
  term  estimate std.error statistic  p.value
  <chr>    <dbl>     <dbl>     <dbl>    <dbl>
1 price -0.841    0.0750      -11.2  6.35e-28
2 pop16  0.00114  0.000547      2.07 3.83e- 2
3 ndi   -0.00557  0.000445    -12.5  5.72e-34
```

Now only $\hat{\beta}_1$ is significantly different than zero.

We can use the **`panelAR`** package to directly estimate a corrected model[2]. **`?panelAR`** for arguments necessary to estimate the corrected model.

```r
library(panelAR)
panelAR(sales ~ price + pop +ndi,
```

[2]Note the slight differences, because **`panelAR`** also corrects for heteroscedasticity.

```
        data = Cigar,
        panelVar = "state",
        timeVar = "year",
        autoCorr = "ar1",
        panelCorrMethod = "pcse") %>%
  summary()
```

```
Panel Regression with AR(1) Prais-Winsten correction and panel-corrected standard erro

Balanced Panel Design:
 Total obs.:       1380 Avg obs. per panel 30
 Number of panels: 46   Max obs. per panel 30
 Number of times:  30   Min obs. per panel 30

Coefficients:
              Estimate Std. Error t value             Pr(>|t|)
(Intercept) 137.461557   5.974766   23.01 < 0.0000000000000002 ***
price        -0.386976   0.060172   -6.43        0.00000000017 ***
pop          -0.000446   0.000338   -1.32               0.1872
ndi           0.001879   0.000719    2.61               0.0091 **
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

R-squared: 0.5899
Wald statistic: 57.2869, Pr(>Chisq(3)): 0
```

17.7 Lagged Dependent Variable Panel Data Model

Let's estimate the lagged-depdendent variable model

$$sales_{it} = \gamma sales_{i,t-1} + \beta_0 + \beta_1 price_{it} + \beta_2 pop16_{it} + \beta_3 ndi_{it} + \epsilon_{it}$$

```
cigar_lag_plm <- plm(sales ~ lag(sales) + price + pop16 + ndi,
                 data = Cigar, # recall plm does not play nice with the expose pipe, %
                 index = c("state", "year"),
                 model = "within",
                 effect = "twoways")
cigar_lag_plm %>%
  summary()
```

```
Twoways effects Within Model
```

```
Call:
plm(formula = sales ~ lag(sales) + price + pop16 + ndi, data = Cigar,
    effect = "twoways", model = "within", index = c("state",
        "year"))

Balanced Panel: n = 46, T = 29, N = 1334

Residuals:
   Min. 1st Qu.  Median 3rd Qu.    Max.
-29.476  -1.900   0.241   2.002  60.188

Coefficients:
             Estimate Std. Error t-value            Pr(>|t|)
lag(sales)  0.8977592  0.0119755   74.97 < 0.0000000000000002 ***
price      -0.1349955  0.0332910   -4.06            0.000053 ***
pop16       0.0000834  0.0002409    0.35                0.73
ndi        -0.0002809  0.0002022   -1.39                0.16
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

Total Sum of Squares:    244000
Residual Sum of Squares: 35000
R-Squared:      0.857
Adj. R-Squared: 0.848
F-statistic: 1876.56 on 4 and 1256 DF, p-value: <0.0000000000000002
```

17.8 Random Effects Model

```r
plm(sales ~ price + pop16 + ndi,
    data = Cigar,
    index = c("state", "year"),
    model = "random",
    effect = "twoways"
    ) %>%
  summary()
```

```
Twoways effects Random Effect Model
   (Swamy-Arora's transformation)

Call:
plm(formula = sales ~ price + pop16 + ndi, data = Cigar, effect = "twoways",
    model = "random", index = c("state", "year"))
```

```
Balanced Panel: n = 46, T = 30, N = 1380

Effects:
                 var std.dev share
idiosyncratic 156.13   12.50  0.26
individual    438.16   20.93  0.73
time            6.70    2.59  0.01
theta: 0.892 (id) 0.42 (time) 0.419 (total)

Residuals:
   Min. 1st Qu.  Median 3rd Qu.    Max.
-57.175  -7.171   0.235   5.785 128.022

Coefficients:
              Estimate Std. Error z-value             Pr(>|z|)
(Intercept) 139.028985   3.995359   34.80 < 0.0000000000000002 ***
price        -0.197951   0.044962   -4.40             0.000011 ***
pop16         0.000357   0.000526    0.68                 0.50
ndi          -0.000356   0.000404   -0.88                 0.38
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

Total Sum of Squares:    325000
Residual Sum of Squares: 280000
R-Squared:      0.138
Adj. R-Squared: 0.136
Chisq: 220.089 on 3 DF, p-value: <0.0000000000000002
```