# EC655 Introduction to R

Justin Smith

## Downloading and Installing R

### Basics

- **R** is a statistical program that does a wide variety of tasks
- It is a powerful program, but its user interface is not good
- Most people use **R** within an interface called **R Studio**
- Think of **R** as the engine, and **R Studio** as the vehicle console

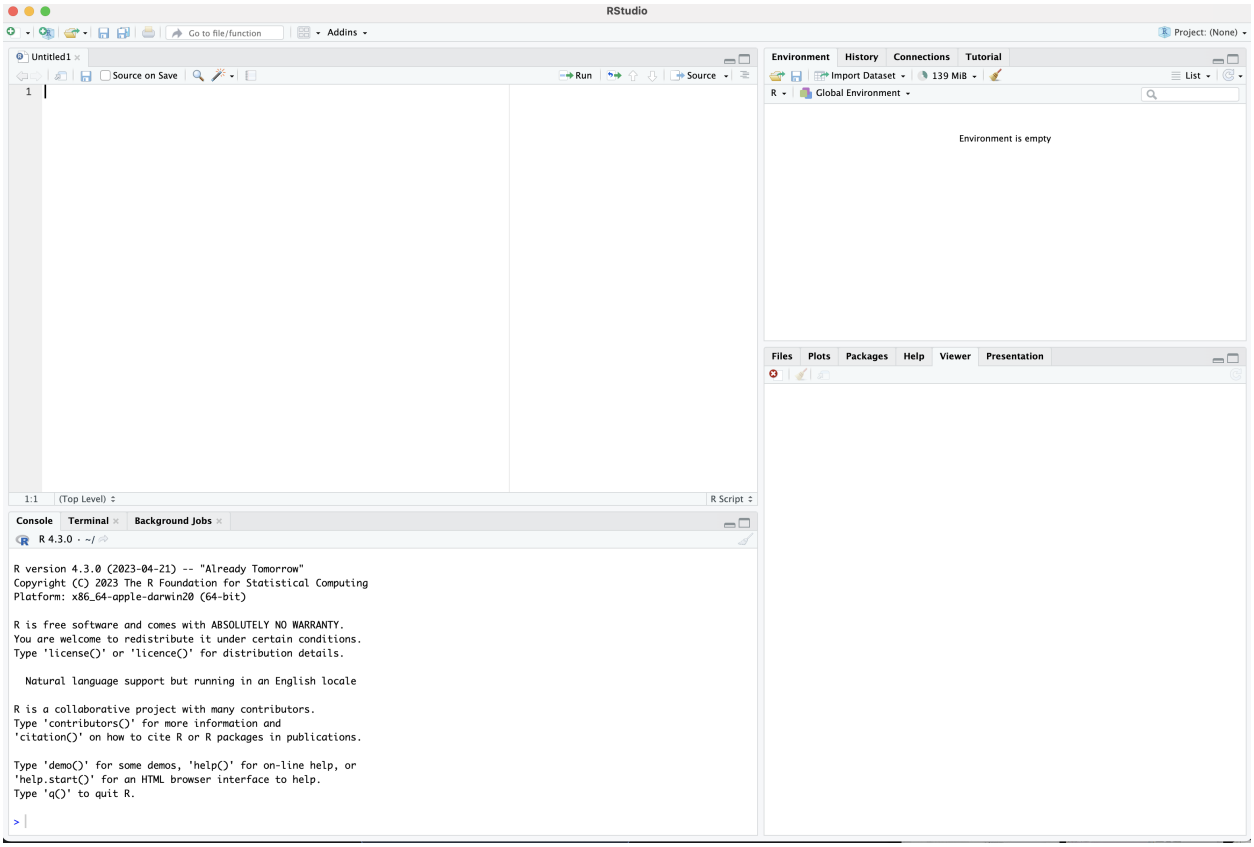
### Installing R

- Download **R** here: https://www.r-project.org
- Instructions on how to download and install https://www.youtube.com/watch?v=BuoAuRbt3qw)

### Installing R Studio

- Download **R Studio** here: https://posit.co/download/rstudio-desktop/
- Make note of your operating system. https://www.youtube.com/watch?v=iHrJTzYVFNw)

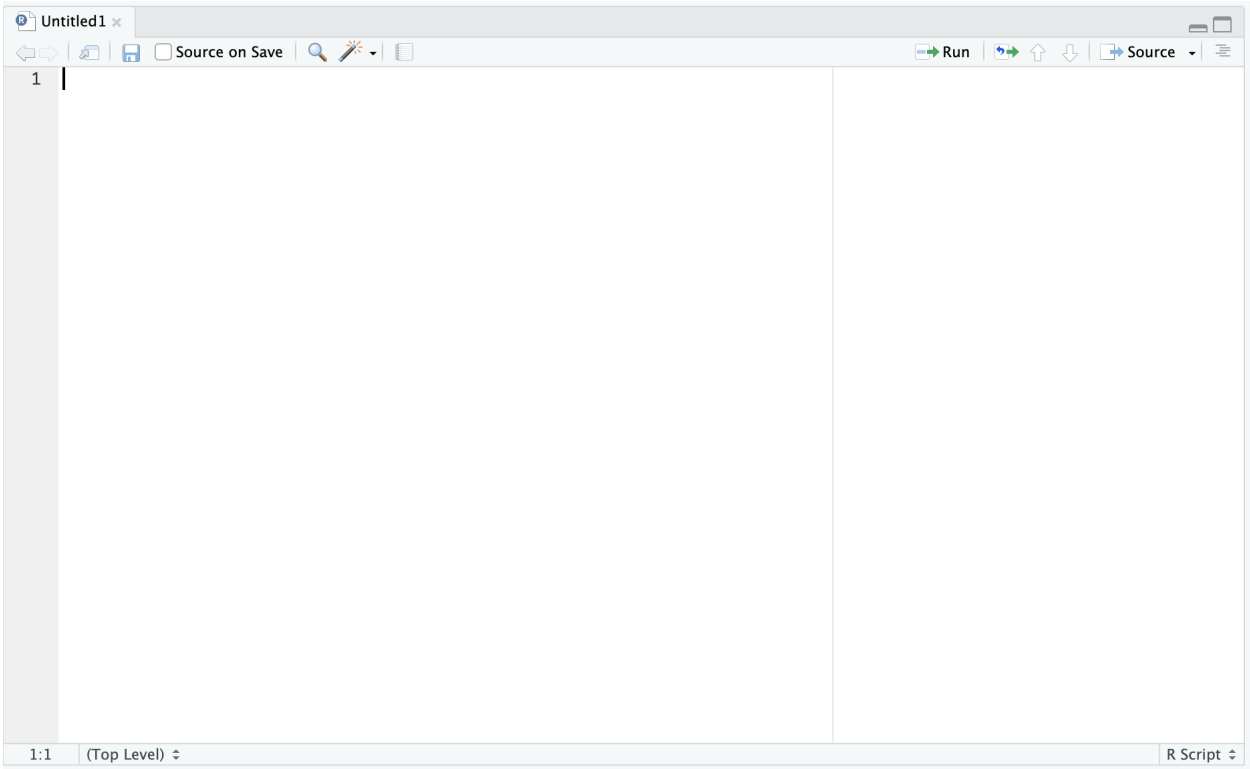
## The R Studio Interface




- **Top-Left**: Source
  - Input files for programs you are running
  - Similar to Stata's do-file
- **Bottom-Left**: Console
  - Where code is evaluated by R
  - You can put code directly into the console
  - Any error messages and other information will appear here
- **Top-Right**: Environment
  - Any objects in memory appear here
  - Mainly this will be data files
- **Bottom-Right**: Files/Plots/Packages/Help
  - Most used for viewing plots
  - Has list of available packages
  - Help for packages and commands is here

## Scripts

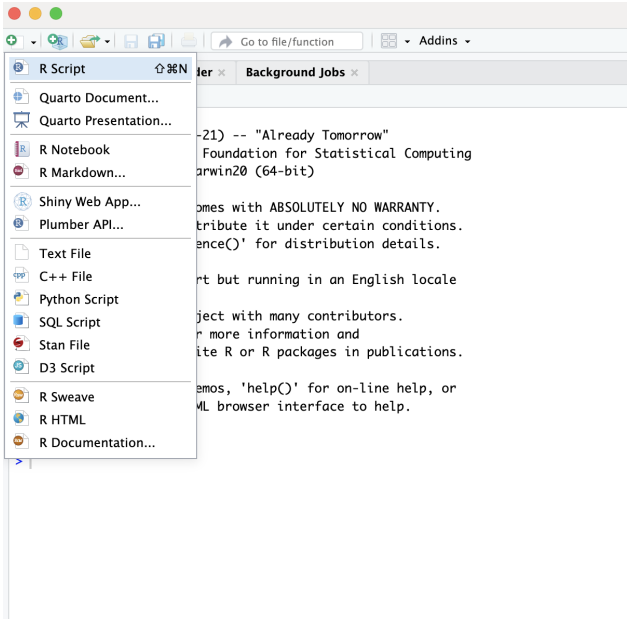
### What is a Script?



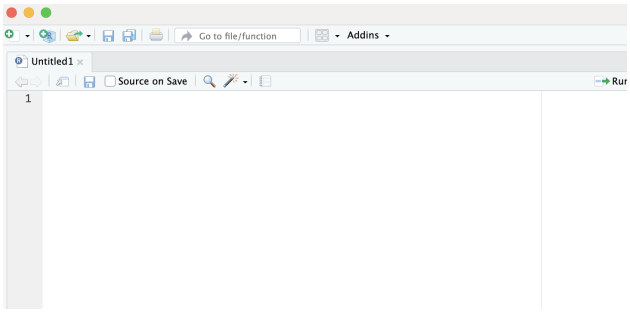

- Scripts are input files in **R** where you store your code
  - Same as do-files in Stata
- You should **always** code in a script
  - If you make a mistake, you can fix it
  - Others can reproduce your results
  - If R shuts down suddenly, you still have your code
- Try not to code directly in console
  - Unless you are testing commands

### Opening a New Script

- Navigate to the top left of the **R Studio** interface




- Below is what your new blank script will look like
- They have a ".R" extension




## Functions and Packages

### Functions

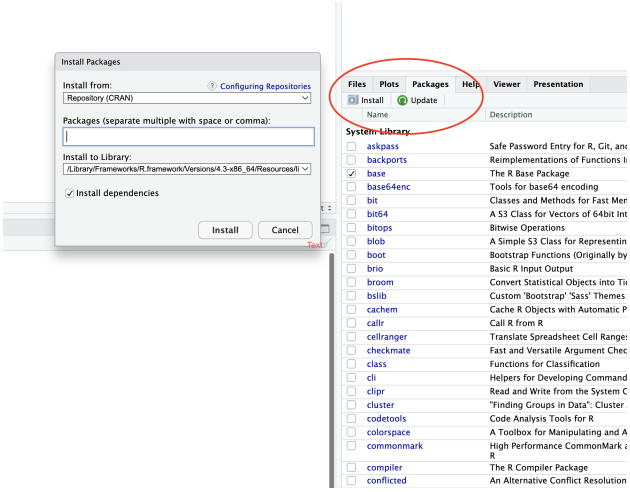
- Simple calculations can be done in **R** just like a calculator
- More complex ones use functions
  - Means, variances, logarithms, etc.
  - OLS estimator
  - Many, many others
- In **R** the function has a name, and a series of arguments in brackets
- We will many through the course

### Packages

- R has many built-in functions
- You can add many more through packages
  - Packages are collections of functions with a similar theme
- Since **R** is open-source, anyone can create packages and make them available
- To use a package, you must

1. *Install it*, which puts the package in your R user library. You only need to install once.
2. *Load it*, which makes the functions available in your R session. You need to load it every time you start a new R session.

- The easiest way to install is to navigate to the packages tab, then click install
- Search for the package, click install




- To load a package, use the `library()` function
- Code below loads the `tidyverse` package, assuming you have already installed it

```
library(tidyverse)
```

## Objects

### What is an Object?

- All stored data is an **object** in R
  - Datasets are objects
  - Variables inside datasets are also objects
  - So are many other things
  - This is different from programs like Stata

### How to Assign Data to an Object

- Assign data to an object with a left arrow and dash "<-"
- As a simple example, you can create object "x" with the value "10"

```
x <- 10
x
```

```
## [1] 10
```

- A slightly more complicated example is

```
x <- 10
y <- 5
z <- x + y
z
```

```
## [1] 15
```

- There are different **data types** you can add to an object

  - Numeric - numbers
  - Character - text
  - Factor - numbers with text labels
  - Logical - true/false
- There are also different **data structures**
  - Vectors - a one-dimensional array of data
  - Matrix - a multidimensional array of a single data type
  - Data Frame - a multidimensional array of possibly multiple data types
  - Other types we will discuss later

### Vectors

- One of the simpler data structures is a vector
- To create one, we use the `c()` function
- Below we create a 3-element vector with values 1,2,3
- Then display the object by typing its name

```
vec <- c(1,2,3)
vec
```

```
## [1] 1 2 3
```

- You can make vectors out of any data type

```
vec <- c("a","character","vector")
vec
```

```
## [1] "a"         "character" "vector"
```

```
vec2 <- c(TRUE, FALSE, TRUE)
vec2
```

```
## [1]  TRUE FALSE  TRUE
```

### Data Frames

- In econometrics, you will mostly work with data frames
- This data structure is similar to datasets in Stata
- It is a collection of vectors of varying types but equal lengths
- One way to create a dataframe is to use the `data.frame()` function
- Below we combine two vectors into a data frame, then view it

```
vec1 <- c("a","character","vector")
vec2 <- c(TRUE, FALSE, TRUE)
df <- data.frame(vec1, vec2)
df
```

```
##        vec1  vec2
## 1         a  TRUE
## 2 character FALSE
## 3    vector  TRUE
```

- In a data frame:
  - Each column is a variable
  - Each row is an observation

## Loading and Saving Data

### Loading R Data

- **R** has two data storage types that you can use
  - ".RDS" files can contain a single **R** object
  - ".RData" files can contain multiple objects
- Most of the time your data will not come in this format
- We will need to know how to load different types

### Loading CSV files

- Comma-separated values (.csv) files are common
- R can load these with the `read.csv()` function (part of the `readr` package)
- The loaded object is a data frame
- The code below loads a small .csv file

```
csvdata <- read.csv("data/exampcsv.csv")
csvdata
```

```
##   a b c
## 1 1 2 3
## 2 4 5 6
## 3 7 8 9
```

### Loading Excel Files

- You can load excel files (.xls, .xlsx) using the `readxl` package
- There are a few different functions
- The easiest is `read_excel()`

```
exceldata <- read_excel("data/exampexcel.xlsx")
exceldata
```

```
## # A tibble: 3 x 3
##       a     b     c
##   <dbl> <dbl> <dbl>
## 1     1     2     3
## 2     4     5     6
## 3     7     8     9
```

### Loading Stata Files

- Most economists use Stata data files (.dta)
- So you might need to load a .dta file
- The `haven` package can do this directly
- Use the `read_stata()` function

```
statadata <- read_stata("data/exampstata.dta")
statadata
```

```
## # A tibble: 3 x 3
##       a     b     c
##   <dbl> <dbl> <dbl>
## 1     1     2     3
## 2     4     5     6
## 3     7     8     9
```

### Other Loading Information

- R can open other types of data too
- We have used the simplest application of each function
  - You might need to specify some additional arguments in your context
- Unlike Stata, you can have many data files open in memory at once
  - They will all be different objects in the **R** environment

### Saving Data

- You can export data into each of the formats listed above
- But if you are using **R** you will probably want to save in .RDS or .RData
- To save in .RData format, use the `save` function
- The basic syntax is to list the objects you want to save, and the file name
- Example below saves the three objects above into a .RData file

```
csvdata <- read.csv("data/exampcsv.csv")
exceldata <- read_excel("data/exampexcel.xlsx")
statadata <- read_stata("data/exampstata.dta")

save(csvdata, exceldata, statadata, file = "data/exampRsave.RData")
```

## The Tidyverse

### Introduction

- As noted, R has a series of built-in **R** functions
- These are called **base R**
- You can do a lot with these, but they can be complicated
- A group of packages called the **tidyverse** simplifies data analysis in **R**
- They simplify things like
  - Graphing
  - Data manipulation
  - Cleaning data
  - Applying functions
  - Working with character and factor variables

### Loading the Tidyverse

- The **tidyverse** contains many packages
- You can load the **core tidyverse** by loading the `tidyverse` package
- This will load the following packages: `ggplot2`, `dplyr`, `tidyr`, `readr`, `purrr`, `tibble`, `stringr`, `forcats`, `lubridate`
- You can also load them individually

### `filter()` and `select()`

- Often you need to subset your data
  - Pull out specific columns or rows
- To subset the columns, use `select()`
- To subset the rows, use `filter()`
- As an example, we will use **mtcars**, a dataset in **R** memory
- First, list the first few rows of the data with the `head()` function

```
mtcars
```

```
##                     mpg cyl  disp  hp drat    wt  qsec vs am gear carb
```

```
## Mazda RX4           21.0   6 160.0 110 3.90 2.620 16.46  0  1    4    4
## Mazda RX4 Wag       21.0   6 160.0 110 3.90 2.875 17.02  0  1    4    4
## Datsun 710          22.8   4 108.0  93 3.85 2.320 18.61  1  1    4    1
## Hornet 4 Drive      21.4   6 258.0 110 3.08 3.215 19.44  1  0    3    1
## Hornet Sportabout   18.7   8 360.0 175 3.15 3.440 17.02  0  0    3    2
## Valiant             18.1   6 225.0 105 2.76 3.460 20.22  1  0    3    1
## Duster 360          14.3   8 360.0 245 3.21 3.570 15.84  0  0    3    4
## Merc 240D           24.4   4 146.7  62 3.69 3.190 20.00  1  0    4    2
## Merc 230            22.8   4 140.8  95 3.92 3.150 22.90  1  0    4    2
## Merc 280            19.2   6 167.6 123 3.92 3.440 18.30  1  0    4    4
## Merc 280C           17.8   6 167.6 123 3.92 3.440 18.90  1  0    4    4
## Merc 450SE          16.4   8 275.8 180 3.07 4.070 17.40  0  0    3    3
## Merc 450SL          17.3   8 275.8 180 3.07 3.730 17.60  0  0    3    3
## Merc 450SLC         15.2   8 275.8 180 3.07 3.780 18.00  0  0    3    3
## Cadillac Fleetwood  10.4   8 472.0 205 2.93 5.250 17.98  0  0    3    4
## Lincoln Continental 10.4   8 460.0 215 3.00 5.424 17.82  0  0    3    4
## Chrysler Imperial   14.7   8 440.0 230 3.23 5.345 17.42  0  0    3    4
## Fiat 128            32.4   4  78.7  66 4.08 2.200 19.47  1  1    4    1
## Honda Civic         30.4   4  75.7  52 4.93 1.615 18.52  1  1    4    2
## Toyota Corolla      33.9   4  71.1  65 4.22 1.835 19.90  1  1    4    1
## Toyota Corona       21.5   4 120.1  97 3.70 2.465 20.01  1  0    3    1
## Dodge Challenger    15.5   8 318.0 150 2.76 3.520 16.87  0  0    3    2
## AMC Javelin         15.2   8 304.0 150 3.15 3.435 17.30  0  0    3    2
## Camaro Z28          13.3   8 350.0 245 3.73 3.840 15.41  0  0    3    4
## Pontiac Firebird    19.2   8 400.0 175 3.08 3.845 17.05  0  0    3    2
## Fiat X1-9           27.3   4  79.0  66 4.08 1.935 18.90  1  1    4    1
## Porsche 914-2       26.0   4 120.3  91 4.43 2.140 16.70  0  1    5    2
## Lotus Europa        30.4   4  95.1 113 3.77 1.513 16.90  1  1    5    2
## Ford Pantera L      15.8   8 351.0 264 4.22 3.170 14.50  0  1    5    4
## Ferrari Dino        19.7   6 145.0 175 3.62 2.770 15.50  0  1    5    6
## Maserati Bora       15.0   8 301.0 335 3.54 3.570 14.60  0  1    5    8
## Volvo 142E          21.4   4 121.0 109 4.11 2.780 18.60  1  1    4    2
```

- Use `select()` to keep only *mpg* and *cyl*

```
select(mtcars, mpg, cyl)
```

```
##                      mpg cyl
## Mazda RX4           21.0   6
## Mazda RX4 Wag       21.0   6
## Datsun 710          22.8   4
## Hornet 4 Drive      21.4   6
## Hornet Sportabout   18.7   8
## Valiant             18.1   6
## Duster 360          14.3   8
## Merc 240D           24.4   4
## Merc 230            22.8   4
## Merc 280            19.2   6
## Merc 280C           17.8   6
## Merc 450SE          16.4   8
## Merc 450SL          17.3   8
## Merc 450SLC         15.2   8
## Cadillac Fleetwood  10.4   8
## Lincoln Continental 10.4   8
## Chrysler Imperial   14.7   8
```

```
## Fiat 128            32.4   4
## Honda Civic         30.4   4
## Toyota Corolla      33.9   4
## Toyota Corona       21.5   4
## Dodge Challenger    15.5   8
## AMC Javelin         15.2   8
## Camaro Z28          13.3   8
## Pontiac Firebird    19.2   8
## Fiat X1-9           27.3   4
## Porsche 914-2       26.0   4
## Lotus Europa        30.4   4
## Ford Pantera L      15.8   8
## Ferrari Dino        19.7   6
## Maserati Bora       15.0   8
## Volvo 142E          21.4   4
```

- Use `filter()` to keep data with *mpg* less than or equal to 20

```
filter(mtcars, mpg <= 20)
```

```
##                     mpg cyl  disp  hp drat    wt  qsec vs am gear carb
## Hornet Sportabout   18.7   8 360.0 175 3.15 3.440 17.02  0  0    3    2
## Valiant             18.1   6 225.0 105 2.76 3.460 20.22  1  0    3    1
## Duster 360          14.3   8 360.0 245 3.21 3.570 15.84  0  0    3    4
## Merc 280            19.2   6 167.6 123 3.92 3.440 18.30  1  0    4    4
## Merc 280C           17.8   6 167.6 123 3.92 3.440 18.90  1  0    4    4
## Merc 450SE          16.4   8 275.8 180 3.07 4.070 17.40  0  0    3    3
## Merc 450SL          17.3   8 275.8 180 3.07 3.730 17.60  0  0    3    3
## Merc 450SLC         15.2   8 275.8 180 3.07 3.780 18.00  0  0    3    3
## Cadillac Fleetwood  10.4   8 472.0 205 2.93 5.250 17.98  0  0    3    4
## Lincoln Continental 10.4   8 460.0 215 3.00 5.424 17.82  0  0    3    4
## Chrysler Imperial   14.7   8 440.0 230 3.23 5.345 17.42  0  0    3    4
## Dodge Challenger    15.5   8 318.0 150 2.76 3.520 16.87  0  0    3    2
## AMC Javelin         15.2   8 304.0 150 3.15 3.435 17.30  0  0    3    2
## Camaro Z28          13.3   8 350.0 245 3.73 3.840 15.41  0  0    3    4
## Pontiac Firebird    19.2   8 400.0 175 3.08 3.845 17.05  0  0    3    2
## Ford Pantera L      15.8   8 351.0 264 4.22 3.170 14.50  0  1    5    4
## Ferrari Dino        19.7   6 145.0 175 3.62 2.770 15.50  0  1    5    6
## Maserati Bora       15.0   8 301.0 335 3.54 3.570 14.60  0  1    5    8
```

### Mutate

- Working with data often means creating new variables
- In the **tidyverse** the main way to do this is `mutate()`
- Suppose we want a variable that measures kilometers per gallon
- One mile is about 1.6 kilometers, so to make this variable we would type

```
mtcars <- mutate(mtcars, kpg = 1.6*mpg)
mtcars
```

```
##                     mpg cyl  disp  hp drat    wt  qsec vs am gear carb   kpg
## Mazda RX4           21.0   6 160.0 110 3.90 2.620 16.46  0  1    4    4 33.60
## Mazda RX4 Wag       21.0   6 160.0 110 3.90 2.875 17.02  0  1    4    4 33.60
## Datsun 710          22.8   4 108.0  93 3.85 2.320 18.61  1  1    4    1 36.48
## Hornet 4 Drive      21.4   6 258.0 110 3.08 3.215 19.44  1  0    3    1 34.24
## Hornet Sportabout   18.7   8 360.0 175 3.15 3.440 17.02  0  0    3    2 29.92
```

```
## Valiant             18.1   6 225.0 105 2.76 3.460 20.22  1  0    3    1 28.96
## Duster 360          14.3   8 360.0 245 3.21 3.570 15.84  0  0    3    4 22.88
## Merc 240D           24.4   4 146.7  62 3.69 3.190 20.00  1  0    4    2 39.04
## Merc 230            22.8   4 140.8  95 3.92 3.150 22.90  1  0    4    2 36.48
## Merc 280            19.2   6 167.6 123 3.92 3.440 18.30  1  0    4    4 30.72
## Merc 280C           17.8   6 167.6 123 3.92 3.440 18.90  1  0    4    4 28.48
## Merc 450SE          16.4   8 275.8 180 3.07 4.070 17.40  0  0    3    3 26.24
## Merc 450SL          17.3   8 275.8 180 3.07 3.730 17.60  0  0    3    3 27.68
## Merc 450SLC         15.2   8 275.8 180 3.07 3.780 18.00  0  0    3    3 24.32
## Cadillac Fleetwood  10.4   8 472.0 205 2.93 5.250 17.98  0  0    3    4 16.64
## Lincoln Continental 10.4   8 460.0 215 3.00 5.424 17.82  0  0    3    4 16.64
## Chrysler Imperial   14.7   8 440.0 230 3.23 5.345 17.42  0  0    3    4 23.52
## Fiat 128            32.4   4  78.7  66 4.08 2.200 19.47  1  1    4    1 51.84
## Honda Civic         30.4   4  75.7  52 4.93 1.615 18.52  1  1    4    2 48.64
## Toyota Corolla      33.9   4  71.1  65 4.22 1.835 19.90  1  1    4    1 54.24
## Toyota Corona       21.5   4 120.1  97 3.70 2.465 20.01  1  0    3    1 34.40
## Dodge Challenger    15.5   8 318.0 150 2.76 3.520 16.87  0  0    3    2 24.80
## AMC Javelin         15.2   8 304.0 150 3.15 3.435 17.30  0  0    3    2 24.32
## Camaro Z28          13.3   8 350.0 245 3.73 3.840 15.41  0  0    3    4 21.28
## Pontiac Firebird    19.2   8 400.0 175 3.08 3.845 17.05  0  0    3    2 30.72
## Fiat X1-9           27.3   4  79.0  66 4.08 1.935 18.90  1  1    4    1 43.68
## Porsche 914-2       26.0   4 120.3  91 4.43 2.140 16.70  0  1    5    2 41.60
## Lotus Europa        30.4   4  95.1 113 3.77 1.513 16.90  1  1    5    2 48.64
## Ford Pantera L      15.8   8 351.0 264 4.22 3.170 14.50  0  1    5    4 25.28
## Ferrari Dino        19.7   6 145.0 175 3.62 2.770 15.50  0  1    5    6 31.52
## Maserati Bora       15.0   8 301.0 335 3.54 3.570 14.60  0  1    5    8 24.00
## Volvo 142E          21.4   4 121.0 109 4.11 2.780 18.60  1  1    4    2 34.24
```

- Note how we assigned the result to the original object *mtcars*
  - You need to do this for the new variable to add it as a column in the original data
  - If you did not, it would just display the result but not add it to the data
- You can create multiple new variables within the same `mutate()` function
- Below we also create the natural log of weight

```
mtcars <- mutate(mtcars, kpg = 1.6*mpg, lwt = log(wt))
mtcars
```

```
##                     mpg cyl  disp  hp drat    wt  qsec vs am gear carb   kpg
## Mazda RX4          21.0   6 160.0 110 3.90 2.620 16.46  0  1    4    4 33.60
## Mazda RX4 Wag      21.0   6 160.0 110 3.90 2.875 17.02  0  1    4    4 33.60
## Datsun 710         22.8   4 108.0  93 3.85 2.320 18.61  1  1    4    1 36.48
## Hornet 4 Drive     21.4   6 258.0 110 3.08 3.215 19.44  1  0    3    1 34.24
## Hornet Sportabout  18.7   8 360.0 175 3.15 3.440 17.02  0  0    3    2 29.92
## Valiant            18.1   6 225.0 105 2.76 3.460 20.22  1  0    3    1 28.96
## Duster 360         14.3   8 360.0 245 3.21 3.570 15.84  0  0    3    4 22.88
## Merc 240D          24.4   4 146.7  62 3.69 3.190 20.00  1  0    4    2 39.04
## Merc 230           22.8   4 140.8  95 3.92 3.150 22.90  1  0    4    2 36.48
## Merc 280           19.2   6 167.6 123 3.92 3.440 18.30  1  0    4    4 30.72
## Merc 280C          17.8   6 167.6 123 3.92 3.440 18.90  1  0    4    4 28.48
## Merc 450SE         16.4   8 275.8 180 3.07 4.070 17.40  0  0    3    3 26.24
## Merc 450SL         17.3   8 275.8 180 3.07 3.730 17.60  0  0    3    3 27.68
## Merc 450SLC        15.2   8 275.8 180 3.07 3.780 18.00  0  0    3    3 24.32
## Cadillac Fleetwood 10.4   8 472.0 205 2.93 5.250 17.98  0  0    3    4 16.64
## Lincoln Continental 10.4  8 460.0 215 3.00 5.424 17.82  0  0    3    4 16.64
## Chrysler Imperial  14.7   8 440.0 230 3.23 5.345 17.42  0  0    3    4 23.52
```

```
## Fiat 128           32.4   4  78.7  66 4.08 2.200 19.47  1  1    4    1 51.84
## Honda Civic        30.4   4  75.7  52 4.93 1.615 18.52  1  1    4    2 48.64
## Toyota Corolla     33.9   4  71.1  65 4.22 1.835 19.90  1  1    4    1 54.24
## Toyota Corona      21.5   4 120.1  97 3.70 2.465 20.01  1  0    3    1 34.40
## Dodge Challenger   15.5   8 318.0 150 2.76 3.520 16.87  0  0    3    2 24.80
## AMC Javelin        15.2   8 304.0 150 3.15 3.435 17.30  0  0    3    2 24.32
## Camaro Z28         13.3   8 350.0 245 3.73 3.840 15.41  0  0    3    4 21.28
## Pontiac Firebird   19.2   8 400.0 175 3.08 3.845 17.05  0  0    3    2 30.72
## Fiat X1-9          27.3   4  79.0  66 4.08 1.935 18.90  1  1    4    1 43.68
## Porsche 914-2      26.0   4 120.3  91 4.43 2.140 16.70  0  1    5    2 41.60
## Lotus Europa       30.4   4  95.1 113 3.77 1.513 16.90  1  1    5    2 48.64
## Ford Pantera L     15.8   8 351.0 264 4.22 3.170 14.50  0  1    5    4 25.28
## Ferrari Dino       19.7   6 145.0 175 3.62 2.770 15.50  0  1    5    6 31.52
## Maserati Bora      15.0   8 301.0 335 3.54 3.570 14.60  0  1    5    8 24.00
## Volvo 142E         21.4   4 121.0 109 4.11 2.780 18.60  1  1    4    2 34.24
##                        lwt
## Mazda RX4          0.9631743
## Mazda RX4 Wag      1.0560527
## Datsun 710         0.8415672
## Hornet 4 Drive     1.1678274
## Hornet Sportabout  1.2354715
## Valiant            1.2412686
## Duster 360         1.2725656
## Merc 240D          1.1600209
## Merc 230           1.1474025
## Merc 280           1.2354715
## Merc 280C          1.2354715
## Merc 450SE         1.4036430
## Merc 450SL         1.3164082
## Merc 450SLC        1.3297240
## Cadillac Fleetwood 1.6582281
## Lincoln Continental 1.6908336
## Chrysler Imperial  1.6761615
## Fiat 128           0.7884574
## Honda Civic        0.4793350
## Toyota Corolla     0.6070445
## Toyota Corona      0.9021918
## Dodge Challenger   1.2584610
## AMC Javelin        1.2340169
## Camaro Z28         1.3454724
## Pontiac Firebird   1.3467736
## Fiat X1-9          0.6601073
## Porsche 914-2      0.7608058
## Lotus Europa       0.4140944
## Ford Pantera L     1.1537316
## Ferrari Dino       1.0188473
## Maserati Bora      1.2725656
## Volvo 142E         1.0224509
```

### The Pipe Operator

- One of the most useful parts of the **tidyverse** is the pipe operator
- Sometimes you need to create an object with a sequence of operations, but don't want to keep the intermediate steps

- The pipe operator %>% allows you to keep the final object without the intermediate ones
- Suppose we want to select some columns from the data and create a new variable
- We can do this using the pipe

```
newdata <- mtcars %>% select(mpg, wt) %>% mutate(kpg = 1.6*mpg, lwt = log(wt))
newdata
```

```
##                      mpg    wt   kpg       lwt
## Mazda RX4           21.0 2.620 33.60 0.9631743
## Mazda RX4 Wag       21.0 2.875 33.60 1.0560527
## Datsun 710          22.8 2.320 36.48 0.8415672
## Hornet 4 Drive      21.4 3.215 34.24 1.1678274
## Hornet Sportabout   18.7 3.440 29.92 1.2354715
## Valiant             18.1 3.460 28.96 1.2412686
## Duster 360          14.3 3.570 22.88 1.2725656
## Merc 240D           24.4 3.190 39.04 1.1600209
## Merc 230            22.8 3.150 36.48 1.1474025
## Merc 280            19.2 3.440 30.72 1.2354715
## Merc 280C           17.8 3.440 28.48 1.2354715
## Merc 450SE          16.4 4.070 26.24 1.4036430
## Merc 450SL          17.3 3.730 27.68 1.3164082
## Merc 450SLC         15.2 3.780 24.32 1.3297240
## Cadillac Fleetwood  10.4 5.250 16.64 1.6582281
## Lincoln Continental 10.4 5.424 16.64 1.6908336
## Chrysler Imperial   14.7 5.345 23.52 1.6761615
## Fiat 128            32.4 2.200 51.84 0.7884574
## Honda Civic         30.4 1.615 48.64 0.4793350
## Toyota Corolla      33.9 1.835 54.24 0.6070445
## Toyota Corona       21.5 2.465 34.40 0.9021918
## Dodge Challenger    15.5 3.520 24.80 1.2584610
## AMC Javelin         15.2 3.435 24.32 1.2340169
## Camaro Z28          13.3 3.840 21.28 1.3454724
## Pontiac Firebird    19.2 3.845 30.72 1.3467736
## Fiat X1-9           27.3 1.935 43.68 0.6601073
## Porsche 914-2       26.0 2.140 41.60 0.7608058
## Lotus Europa        30.4 1.513 48.64 0.4140944
## Ford Pantera L      15.8 3.170 25.28 1.1537316
## Ferrari Dino        19.7 2.770 31.52 1.0188473
## Maserati Bora       15.0 3.570 24.00 1.2725656
## Volvo 142E          21.4 2.780 34.24 1.0224509
```

- The pipe feeds the result from the left of %>% as the first argument in the function to the right of %>%
- As in the example above, you can pipe in a long sequence
- It only keeps the result from the very end of the pipe

### Summarize

- Summary statistics are a key part of data analysis
- `summarize()` is a convenient way to reduce a dataset into summary statistics
- This function uses other functions as inputs
- Suppose we want to create a new object containing the mean and standard deviation of *mpg*

```
sumstats <- mtcars %>% summarize(mmpg = mean(mpg), sdmpg = sd(mpg))
sumstats
```

```
##       mmpg    sdmpg
## 1 20.09062 6.026948
```

### Grouping

- There are some operations on data you want to perform within groups
- For example, if you had income data on men and women and wanted the gender-specific mean
- To perform opertations within groups you can use `group_by()`
- This defines groups in the data, and downstream functions will use that grouping
- Below we compute the mean and sd for mpg for cars with the same number of cylinders

```
sumstats <- mtcars %>% group_by(cyl) %>% summarize(mmpg = mean(mpg), sdmpg = sd(mpg))
sumstats
```

```
## # A tibble: 3 x 3
##     cyl  mmpg sdmpg
##   <dbl> <dbl> <dbl>
## 1     4  26.7  4.51
## 2     6  19.7  1.45
## 3     8  15.1  2.56
```

- Once you define a group within a data frame, it stays there
- So functions will make use of that grouping
- To remove a grouping, use the `ungroup()` function

## Data Visualization

### Introduction

- One of the main strengths of *R* is its graphics capabilities
- You can produce very nice looking, fully customized visualizations
- The main graphics package `ggplot2` is part of the **tidyverse**
- Simple graphs are straightforward, but customization can get very complicated

### The Structure of ggplot

- Plots in *ggplot* take the following general structure

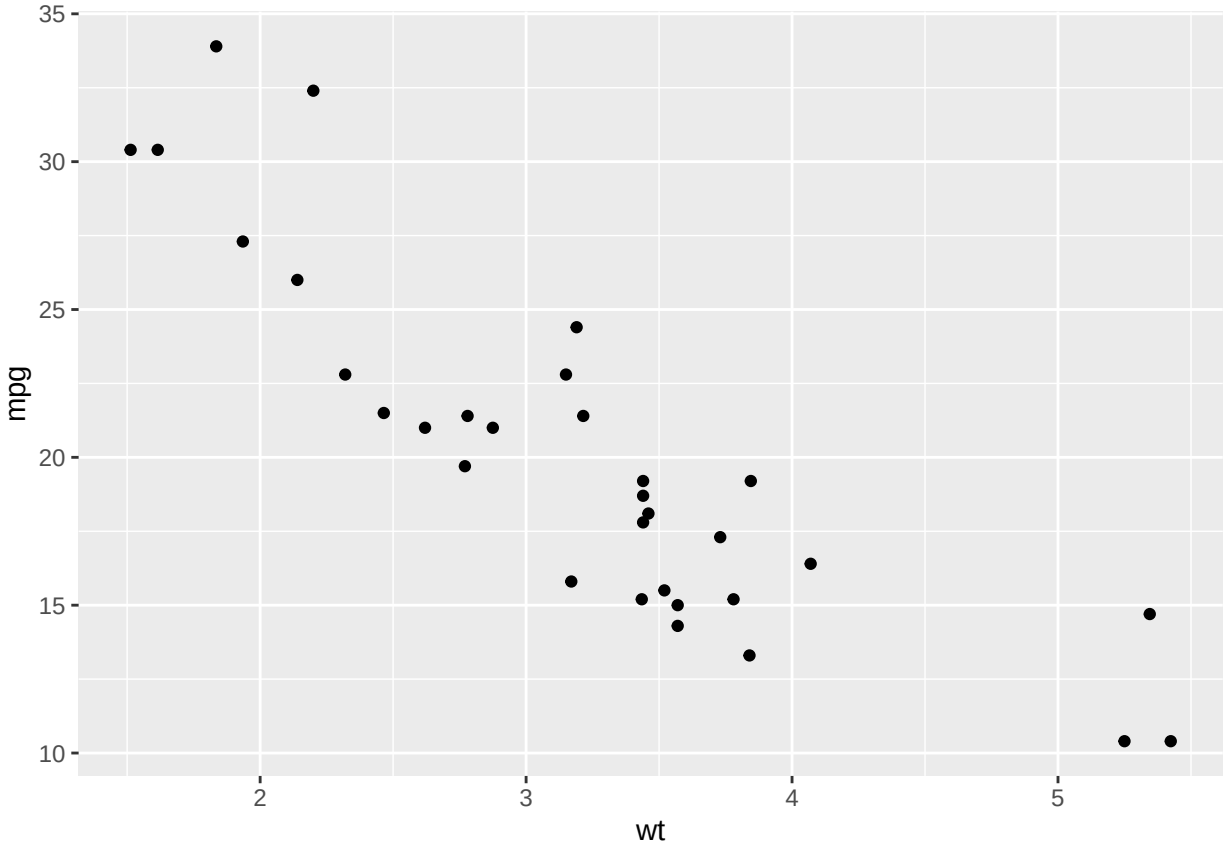
1. Declare a plot with `ggplot()` along with its data and *aesthetics*
   - Data are usually a dataframe
   - Aesthetics refers to the variables on the axes, data groupings, and characteristics like size, shape, color, etc.
2. Layers (aka geoms)
   - Layers are the types of plots you want to see, like scatter, line, bar, etc.
   - Also includes labels, fills, color scales, other formatting
   - You can add multiple layers to the plot
   - You can also add other aesthetics specific to each layer if necessary

- As noted, this can get complicated depending on the level of customization

### A Simple Scatterplot

- Below we use the *mtcars* data to do a quick scatterplot

```
ggplot(mtcars, aes(x = wt, y = mpg)) +
  geom_point()
```

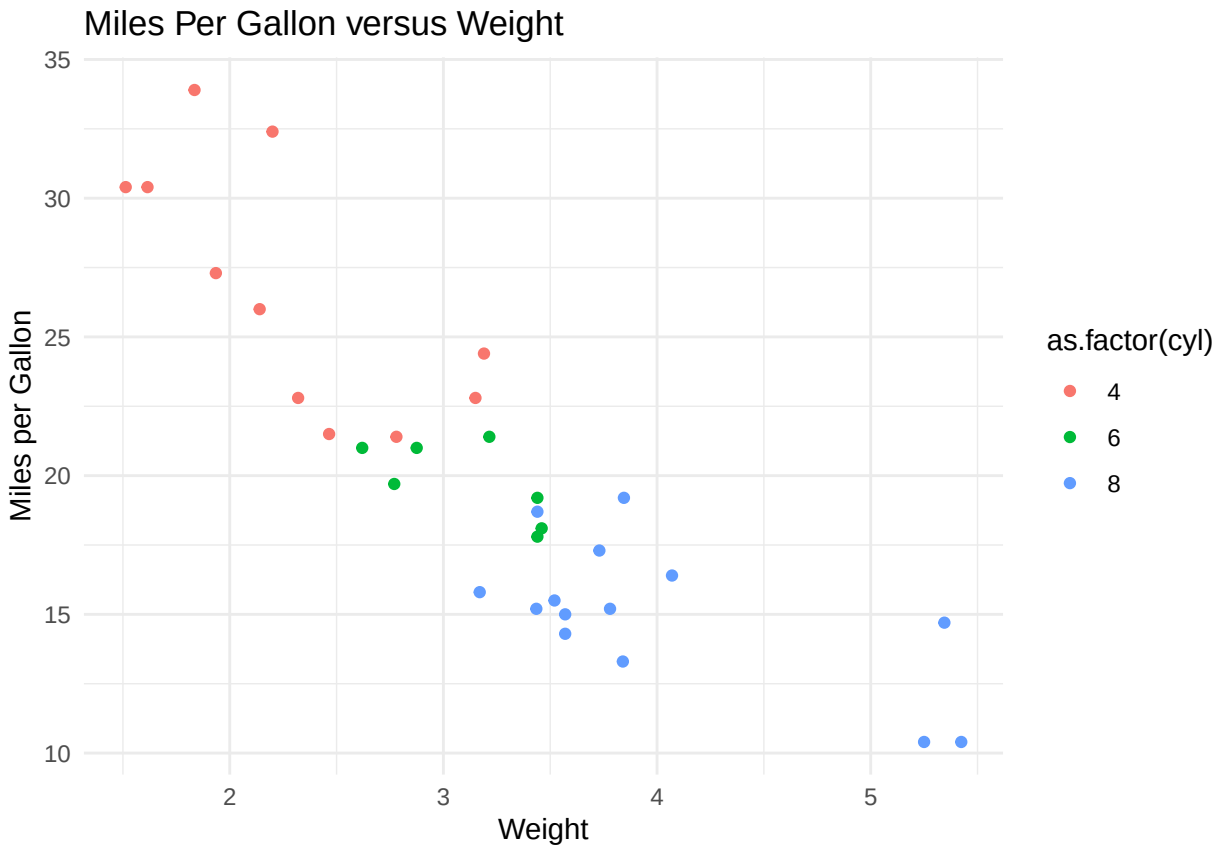



- Notice the structure
  - Declare a new plot using `ggplot()` and the *mtcars* data
  - Aesthetics include weight on the x-axis, mpg on the y-axis
  - `geom_point()` is the layer that tells **R** to do a scatterplot
- Here we have done no customization
  - No title
  - Variable names are on the axes
  - The size of the plot is the default

### Customizing the Simple Scatter

- Most graphs will require some customization
  - For example, a title
- Below we add
  - A title
  - Custom axis labels
  - Specific colours for cars with different numbers of cylinders
  - A mimimal theme to streamline the look

```
ggplot(mtcars, aes(x = wt, y = mpg, color = as.factor(cyl))) +
  geom_point() +
  labs(title = "Miles Per Gallon versus Weight", x = "Weight", y = "Miles per Gallon") +
  theme_minimal()
```

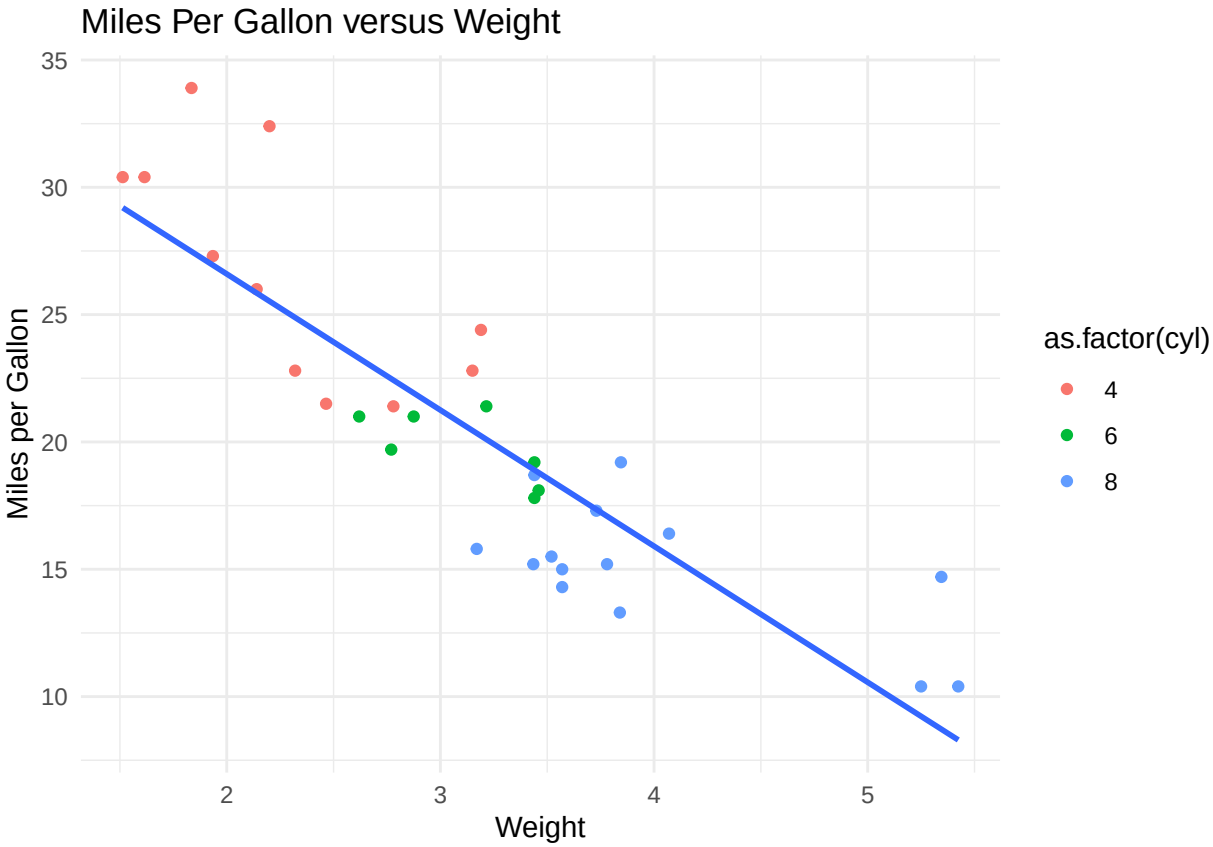



### Multiple Plot Layers

- Many intersting plots have multiple components
  - Two or more lines
  - A scatterplot with regression on top
- You can do this in *ggplot* by just adding another layer
- Below we add a linear fit to the scatterplot

```
ggplot(mtcars, aes(x = wt, y = mpg)) +
  geom_point(aes(color = as.factor(cyl))) +
  geom_smooth(se = FALSE, method = lm) +
  labs(title = "Miles Per Gallon versus Weight", x = "Weight", y = "Miles per Gallon") +
  theme_minimal()
```

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




- Notice that we moved the color aesthetic inside of `geom_point()`
  - Declaring it inside `ggplot()` will apply that aesthetic to **all layers**
  - Leaving it there would create a separate linear regression for each cylinder type
  - By putting it inside `geom_point()`, it applies the aesthetic to that layer only

## Comments on Plotting

- These examples just scratch the surface of plotting in *ggplot*
- You will need to reference google, youtube, other sources for other types of plots
- Things can get very complicated and might take awhile