Geospatial data analysis in R
Week 1 - Introduction

Josh Merfeld

KDI School

2024-09-08

Introductions

- Let's start with a little introduction

- Name, year, program, research interests, etc.
  - Why are you taking this class?

Course Overview

- Geospatial data analysis in R
- Major themes:
  - Geospatial data types
    - Shapefiles (vector files), rasters, etc.
  - Data visualization
    - ggplot, tidyterraarch
  - You will be doing assignments in R Markdown

Course Overview

- Today will just be a short introduction
- For next class, please come with R and R Studio installed on your computer
  - You can find instructions on the syllabus
  - You **must** bring a laptop to class. If you cannot do this, please speak with me.
- Course website: https://github.com/JoshMerfeld/geospatialdataR
  - You can find slides, assignments, and other materials here
  - It will be updated as we go throughout the semester
    - (I am still making slides!)

Course Overview

- This is a brand new class, so I will likely be making changes as we go
- Please check the course website regularly for updates

Detailed outline (tentative)

1. Linear regression (week 2)
    - Inference (confidence intervals, hypothesis testing, bootstrapping, etc.)
2. Maximum likelihood estimation (week 3)
    - Discrete choice (logit, probit, multinomial logit, etc.)
3. Introduction to causality (week 4)
    - Potential outcomes framework
    - Problems with simple regression
    - Why randomization works

Detailed outline (tentative)

4. Differences-in-differences (weeks 5 and 6)
    - Fixed effects, including two-way fixed effects
    - Event studies
    - Synthetic control
5. Instrumental variables (weeks 7 and 8)
    - Assumptions
    - IVs in RCTs (LATE)
    - Some examples
    - Weak instruments
    - Bartik (shift-share) instruments

Detailed outline (tentative)

6. Regression discontinuity (week 9)
   - Canonical regression discontinuity
   - Parametric vs. non-parametric
7. Machine learning in economics (week 10)
   - ML for prediction (lasso, ridge, elastic net)
   - Cross validation
   - Heterogeneous treatment effects
   - Brief introductionto other supervised ML (time dependent)

Grading

1. Homework - coding tasks (55%)

- The homeworks form the main grading component of the course
- The goal is to get you comfortable with coding *and writing* in R
  - I will also ask you to interpret things to make sure you understand what you are doing statistically
- I expect you to do your homeworks in R Markdown and turn in the code along with a pdf output[1]]
- I expect you will have four or five homeworks throughout the semester
- For those of you without a background in R, the first few weeks will take a bit of effort. It will get easier, I promise.

[1] Note: If you have a strong preference for using a different language (e.g. Python), please let me know and we can discuss it. However, you **must** be able to produce a pdf output with your code and results. Using Word is a no-go.

Grading

1. Final exam (35%)

- This will be a take-home exam with a mix of theory and coding.

2. Participation (10%)

- I expect everyone to participate in class. That means asking questions, answering questions, and participating in discussions.

TA sections

- The goal of TA sections is to help you with R and R Markdown
- For help with the actual material, please come to my office hours

Questions?

- Any questions about the course?

Next up: R and RStudio!

- We need to have R and RStudio installed for what's next
  - Another code editor is also acceptable: VS Code, for example

- Course website: https://github.com/JoshMerfeld/geospatialdataR

Goal for the rest of class

- The goal for today is to give you a brief introduction to R and R Markdown
- We will be using two small datasets to get you familiar with the program
  - Class website
- A note: if you are completely new to R, the first few weeks will be a slog
  - It will get better, I promise
- Much of the material covered today comes from two (free!) sources:
  - R for Data Science
  - R Markdown: The Definitive Guide

What are R and RStudio?

- R is a commonly used statistical program (and language)
  - It is free and open source, which means you can use this after graduation, without paying for it
  - R is CaSe SeNsItIvE
- To work with R, we want to use an accompaniment called RStudio
  - RStudio is what is referred to as an integrated development environment (IDE)
  - It is not the only option (I use VS Code, for example), but it is the most common
  - It makes working with R much easier
- Whenever you start R, you want to start RStudio
  - RStudio will start R for you

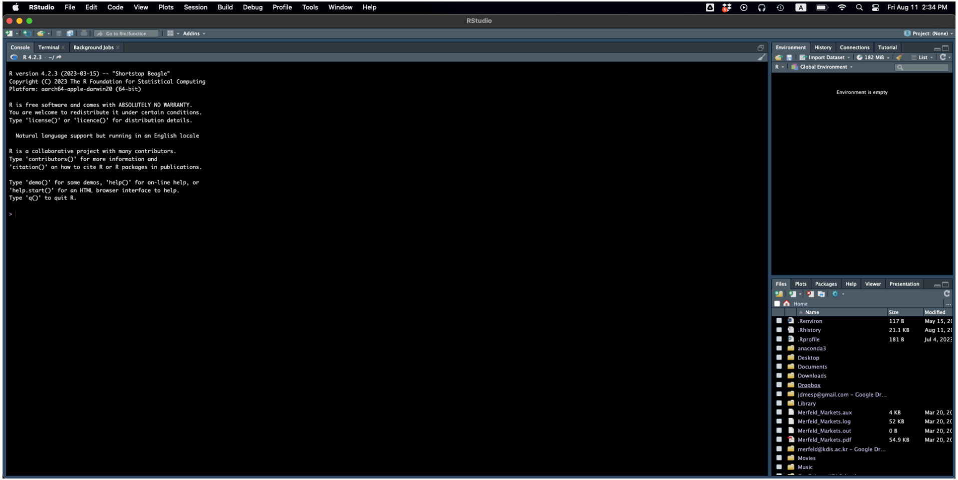
Some important considerations

- One of our goals is to make **reproducible** research
  - This means that we want to be able to share our code and have others be able to replicate our results
  - To do this, we will use “scripts” that contain our code

- A script should be self contained
  - This means that it should contain all of the code necessary to run the analysis
  - A well-written script should allow me to do everything without any additional information

- We will also use R Markdown to create documents
  - R Markdown is a way to combine text and code
  - This allows us to create documents that are reproducible
  - We will use R Markdown to create our homework assignments
  - More on this in a bit

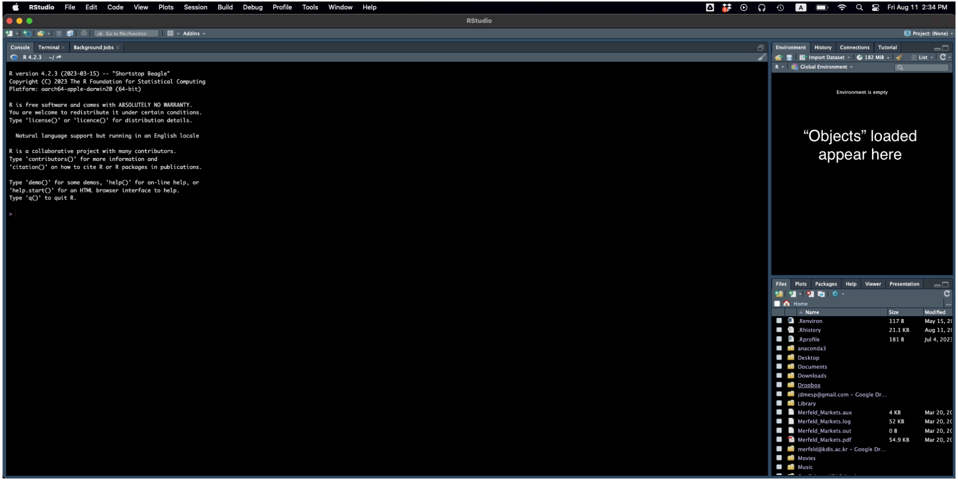
The RStudio interface



```
R version 4.2.3 (2023-03-15) -- "Shortstop Beagle"
Copyright (C) 2023 The R Foundation for Statistical Computing
Platform: aarch64-apple-darwin20 (64-bit)

R is free software and comes with ABSOLUTELY NO WARRANTY.
You are welcome to redistribute it under certain conditions.
Type 'license()' or 'licence()' for distribution details.

  Natural language support but running in an English locale

R is a collaborative project with many contributors.
Type 'contributors()' for more information and
'citation()' on how to cite R or R packages in publications.

Type 'demo()' for some demos, 'help()' for on-line help, or
'help.start()' for an HTML browser interface to help.
Type 'q()' to quit R.

>
```

Name	Size	Modified
.Renviron	117 B	May 15, 2(
.Rhistory	21.1 KB	Aug 11, 2(
.Rprofile	181 B	Jul 4, 2023
anaconda3		
Desktop		
Documents		
Downloads		
Dropbox		
jdmesp@gmail.com – Google Dr...		
Library		
Merfeld_Markets.aux	4 KB	Mar 20, 2(
Merfeld_Markets.log	52 KB	Mar 20, 2(
Merfeld_Markets.out	0 B	Mar 20, 2(
Merfeld_Markets.pdf	54.9 KB	Mar 20, 2(
merfeld@kdis.ac.kr – Google Dr...		
Movies		
Music		

The RStudio interface

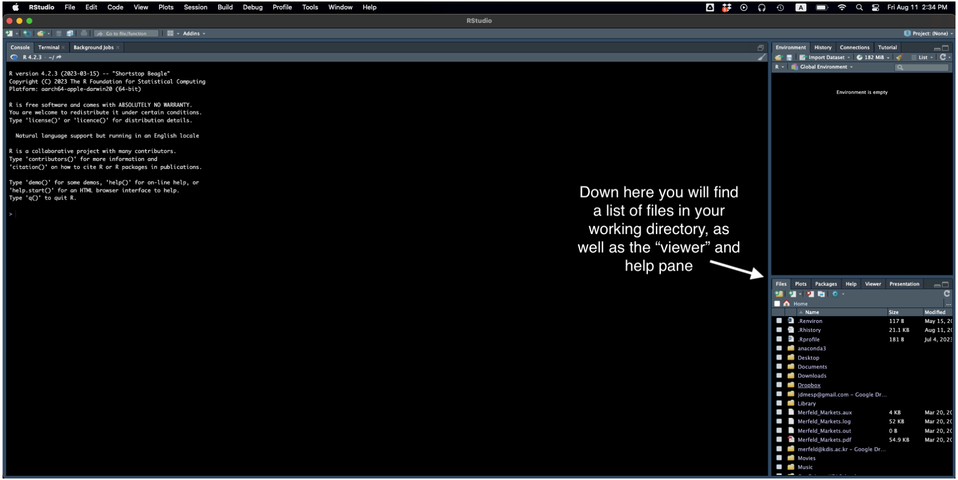


```
R version 4.2.3 (2023-03-15) -- "Shortstop Beagle"
Copyright (C) 2023 The R Foundation for Statistical Computing
Platform: aarch64-apple-darwin20 (64-bit)

R is free software and comes with ABSOLUTELY NO WARRANTY.
You are welcome to redistribute it under certain conditions.
Type 'license()' or 'licence()' for distribution details.

  Natural language support but running in an English locale

R is a collaborative project with many contributors.
Type 'contributors()' for more information and
'citation()' on how to cite R or R packages in publications.

Type 'demo()' for some demos, 'help()' for on-line help, or
'help.start()' for an HTML browser interface to help.
Type 'q()' to quit R.

>
```

Name	Size	Modified
.Renviron	117 B	May 15, 20
.Rhistory	21.1 KB	Aug 11, 20
.Rprofile	181 B	Jul 4, 2023
anaconda3		
Desktop		
Documents		
Downloads		
Dropbox		
jdmesp@gmail.com – Google Dr...		
Library		
Merfeld_Markets.aux	4 KB	Mar 20, 20
Merfeld_Markets.log	52 KB	Mar 20, 20
Merfeld_Markets.out	0 B	Mar 20, 20
Merfeld_Markets.pdf	54.9 KB	Mar 20, 20
merfeld@kdis.ac.kr – Google Dr...		
Movies		
Music		

“Objects” loaded appear here

The RStudio interface

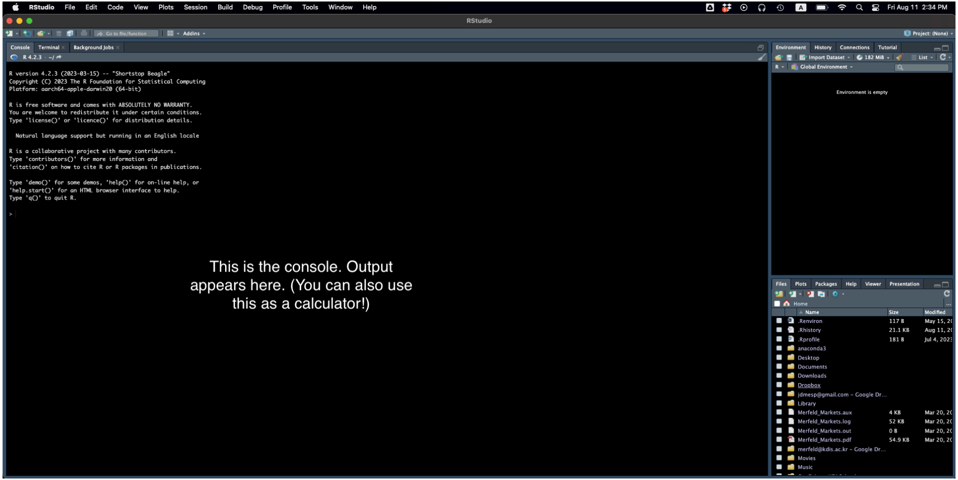


```
R version 4.2.3 (2023-03-15) -- "Shortstop Beagle"
Copyright (C) 2023 The R Foundation for Statistical Computing
Platform: aarch64-apple-darwin20 (64-bit)

R is free software and comes with ABSOLUTELY NO WARRANTY.
You are welcome to redistribute it under certain conditions.
Type 'license()' or 'licence()' for distribution details.

  Natural language support but running in an English locale

R is a collaborative project with many contributors.
Type 'contributors()' for more information and
'citation()' on how to cite R or R packages in publications.

Type 'demo()' for some demos, 'help()' for on-line help, or
'help.start()' for an HTML browser interface to help.
Type 'q()' to quit R.

>
```

Name	Size	Modified
.Renviron	117 B	May 15, 20
.Rhistory	21.1 KB	Aug 11, 20
.Rprofile	181 B	Jul 4, 2023
anaconda3		
Desktop		
Documents		
Downloads		
Dropbox		
jdmesp@gmail.com – Google Dr...		
Library		
Merfeld_Markets.aux	4 KB	Mar 20, 20
Merfeld_Markets.log	52 KB	Mar 20, 20
Merfeld_Markets.out	0 B	Mar 20, 20
Merfeld_Markets.pdf	54.9 KB	Mar 20, 20
merfeld@kdis.ac.kr – Google Dr...		
Movies		
Music		

Down here you will find a list of files in your working directory, as well as the "viewer" and help pane

The RStudio interface



```
R version 4.2.3 (2023-03-15) -- "Shortstop Beagle"
Copyright (C) 2023 The R Foundation for Statistical Computing
Platform: aarch64-apple-darwin20 (64-bit)

R is free software and comes with ABSOLUTELY NO WARRANTY.
You are welcome to redistribute it under certain conditions.
Type 'license()' or 'licence()' for distribution details.

  Natural language support but running in an English locale

R is a collaborative project with many contributors.
Type 'contributors()' for more information and
'citation()' on how to cite R or R packages in publications.

Type 'demo()' for some demos, 'help()' for on-line help, or
'help.start()' for an HTML browser interface to help.
Type 'q()' to quit R.

>
```

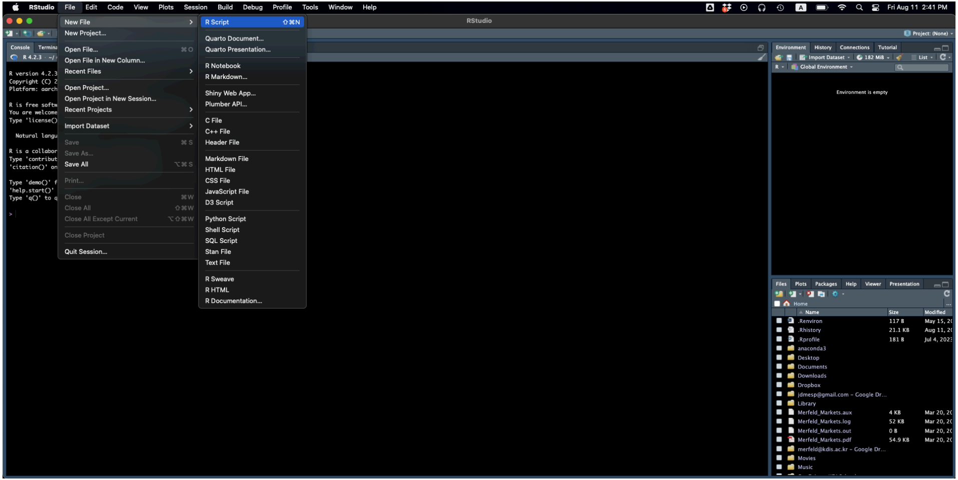
Environment is empty

Name	Size	Modified
.Renviron	117 B	May 15, 20
.Rhistory	21.1 KB	Aug 11, 20
.Rprofile	181 B	Jul 4, 2023
anaconda3		
Desktop		
Documents		
Downloads		
Dropbox		
jdmesp@gmail.com – Google Dr...		
Library		
Merfeld_Markets.aux	4 KB	Mar 20, 20
Merfeld_Markets.log	52 KB	Mar 20, 20
Merfeld_Markets.out	0 B	Mar 20, 20
Merfeld_Markets.pdf	54.9 KB	Mar 20, 20
merfeld@kdis.ac.kr – Google Dr...		
Movies		
Music		

This is the console. Output appears here. (You can also use this as a calculator!)

But we're missing something... what is it?

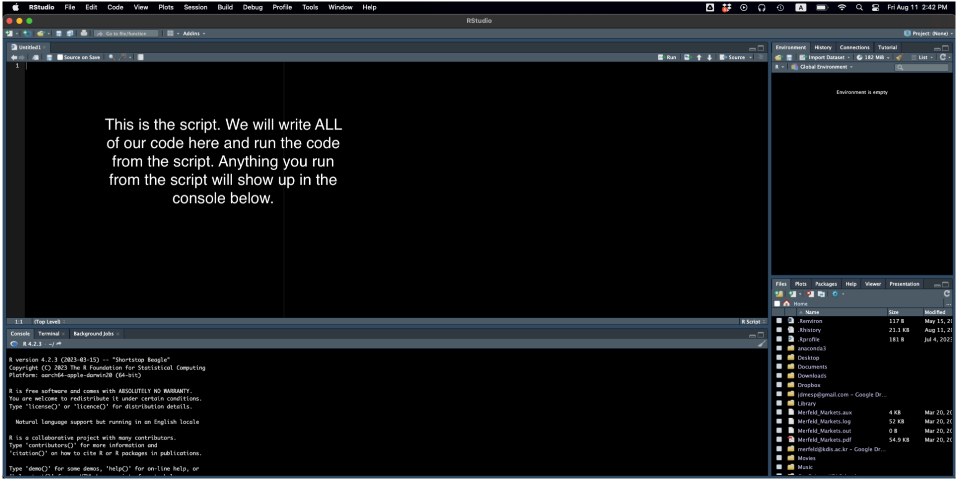
But we're missing something... what is it?



```
R version 4.2.3
Copyright (C) 2
Platform: aarch

R is free softw
You are welcome
Type 'license()

  Natural langu

R is a collabor
Type 'contribut
'citation()' on

Type 'demo()' f
'help.start()'
Type 'q()' to q

>
```

Name	Size	Modified
.Renviron	117 B	May 15, 20
.Rhistory	21.1 KB	Aug 11, 20
.Rprofile	181 B	Jul 4, 2023
anaconda3		
Desktop		
Documents		
Downloads		
Dropbox		
jdmesp@gmail.com – Google Dr...		
Library		
Merfeld_Markets.aux	4 KB	Mar 20, 20
Merfeld_Markets.log	52 KB	Mar 20, 20
Merfeld_Markets.out	0 B	Mar 20, 20
Merfeld_Markets.pdf	54.9 KB	Mar 20, 20
merfeld@kdis.ac.kr – Google Dr...		
Movies		
Music		

The script



This is the script. We will write ALL of our code here and run the code from the script. Anything you run from the script will show up in the console below.

```
R version 4.2.3 (2023-03-15) -- "Shortstop Beagle"
Copyright (C) 2023 The R Foundation for Statistical Computing
Platform: aarch64-apple-darwin20 (64-bit)

R is free software and comes with ABSOLUTELY NO WARRANTY.
You are welcome to redistribute it under certain conditions.
Type 'license()' or 'licence()' for distribution details.

  Natural language support but running in an English locale

R is a collaborative project with many contributors.
Type 'contributors()' for more information and
'citation()' on how to cite R or R packages in publications.

Type 'demo()' for some demos, 'help()' for on-line help, or
```

Name	Size	Modified
.Renviron	117 B	May 15, 20
.Rhistory	21.1 KB	Aug 11, 20
.Rprofile	181 B	Jul 4, 2023
anaconda3		
Desktop		
Documents		
Downloads		
Dropbox		
jdmesp@gmail.com – Google Dr…		
Library		
Merfeld_Markets.aux	4 KB	Mar 20, 20
Merfeld_Markets.log	52 KB	Mar 20, 20
Merfeld_Markets.out	0 B	Mar 20, 20
Merfeld_Markets.pdf	54.9 KB	Mar 20, 20
merfeld@kdis.ac.kr – Google Dr…		
Movies		
Music		

Some notes

- You can add comments to your script using a hashtag (#)
  - At the top of ALL my scripts, I have a comment that says what the script does.
  - At the top of your script, write a comment. It should say "# Week 1 - Introduction to R"
  - I put LOTS of comments in my scripts. This is good practice.
- You can run a line of code by clicking the "Run" button
  - There are also shortcuts. On Mac it is command + enter. On windows it is control + enter. You can change these if you want.
- You can run multiple lines of code by highlighting them and clicking the "Run" button (or the shortcut)
- We will practice these later

Object types

- R has a few different types of objects
    - The most common are vectors, matrices, and data frames
        - A “tibble” is a type of data frame used by the tidyverse package (more below)
    - We will use data frames almost exclusively since we are working with datasets, but vectors are common, too
- You can create a vector using the c() function:
    - Note how we create a new object using the assignment operator, <-. You can also use =.

```r
vec <- c(1, 2, 3, 4)
vec
```

```
[1] 1 2 3 4
```

Object types

- You can check what type of object something is by using the class() function
  - For example, if I want to check what type of object vec is, I would write class(vec)
  - Note that the output is "numeric"
  - This is because vec is a vector of numbers
- If I want to check whether it is a vector, I can write is.vector(vec)
  - Note that the output is TRUE

```r
vec <- c(1, 2, 3, 4)
class(vec)
```

```
[1] "numeric"
```

```r
is.vector(vec)
```

```
[1] TRUE
```

First things first: the working directory

- The working directory is the folder that R is currently working in
  - This is where R will look for files
  - This is where R will save files
  - This is where R will create files
- You can always write out an entire file path, but this is tedious
  - More importantly, it makes your code less reproducible since the path is specific to YOUR computer
- One nice thing about R is that the working directory will automatically be where you open the script from
  - Let's try this. Save your script to a folder on your computer, then open the script from that folder.

First things first: the working directory

The working directory should be where you opened the file from. Check it like this:

```r
getwd()
```

```
[1] "/Users/Josh/Dropbox/KDIS/Classes/geospatialdataR"
```

R packages

- R is a language that is built on packages
  - Packages are collections of functions that do specific things
  - R comes with a set of “base” packages that are installed automatically

- We are going to use one package consistently, called the “tidyverse”
  - This consists of a set of packages that are designed to work together, with data cleaning in mind

R packages

The one exception to always using a script? I install packages in the CONSOLE. You can install packages like this:

```r
install.packages("tidyverse")
```

Loading R packages in your script

We need to load any R packages we want to use at the very top of the script. You should have a comment on line one, so on line two write:

```
library("tidyverse")
```

This will load the tidyverse package.

Loading data

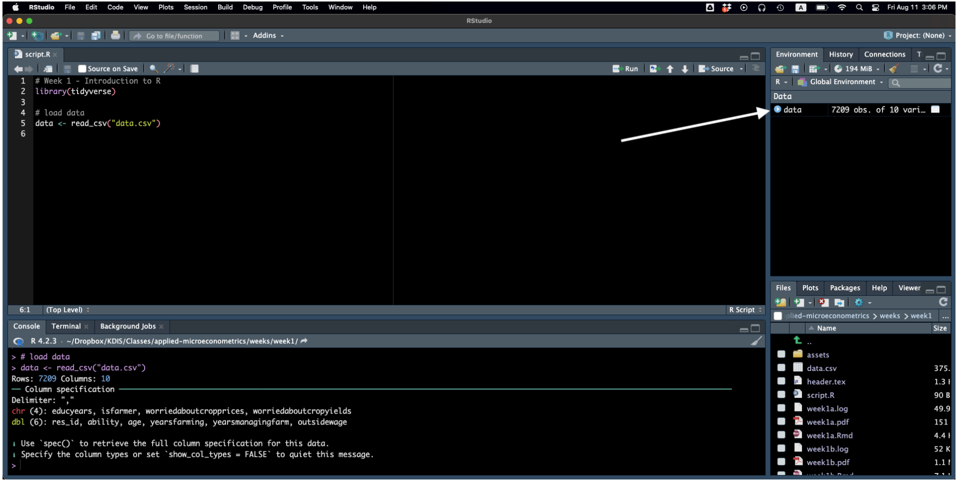
- Go to the class website and download the data for today.
  - Put it in your WORKING DIRECTORY (where the script is)

- We will use the read_csv() function to load the data
  - This function is part of the tidyverse package
  - It will create a data frame
  - We need to NAME the object (data frame). As before, note the assignment operator (<-). You can actually use = though.

```r
library(tidyverse)

# read in the data
data <- read_csv("data.csv")
```

Objects in memory

The data frame should show up in the upper right hand corner of RStudio.



```
# Week 1 - Introduction to R
library(tidyverse)

# load data
data <- read_csv("data.csv")

```

Data

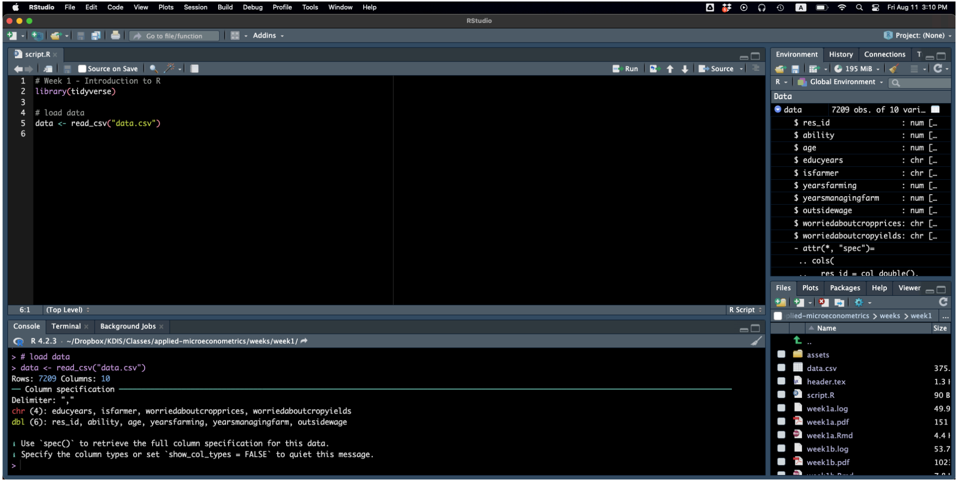
data	7209 obs. of 10 vari…

```
> # load data
> data <- read_csv("data.csv")
Rows: 7209 Columns: 10
── Column specification ────────────────────────────────────────
Delimiter: ","
chr (4): educyears, isfarmer, worriedaboutcropprices, worriedaboutcropyields
dbl (6): res_id, ability, age, yearsfarming, yearsmanagingfarm, outsidewage

ℹ Use `spec()` to retrieve the full column specification for this data.
ℹ Specify the column types or set `show_col_types = FALSE` to quiet this message.
>
```

Name	Size
..	
assets	
data.csv	375.
header.tex	1.3
script.R	90 B
week1a.log	49.9
week1a.pdf	151
week1a.Rmd	4.4
week1b.log	52 K
week1b.pdf	1.1

Objects in memory

Click on the arrow and it will show more information.



```
# Week 1 - Introduction to R
library(tidyverse)

# load data
data <- read_csv("data.csv")

```

```
R 4.2.3 · ~/Dropbox/KDIS/Classes/applied-microeconometrics/weeks/week1/
> # load data
> data <- read_csv("data.csv")
Rows: 7209 Columns: 10
── Column specification ────────────────────────────────────────────────────
Delimiter: ","
chr (4): educyears, isfarmer, worriedaboutcropprices, worriedaboutcropyields
dbl (6): res_id, ability, age, yearsfarming, yearsmanagingfarm, outsidewage

ℹ Use `spec()` to retrieve the full column specification for this data.
ℹ Specify the column types or set `show_col_types = FALSE` to quiet this message.
>
```

```
Data
data        7209 obs. of 10 vari…
   $ res_id                : num […
   $ ability               : num […
   $ age                   : num […
   $ educyears             : chr […
   $ isfarmer              : chr […
   $ yearsfarming          : num […
   $ yearsmanagingfarm     : num […
   $ outsidewage           : num […
   $ worriedaboutcropprices: chr […
   $ worriedaboutcropyields: chr […
   - attr(*, "spec")=
    .. cols(
    ..   res_id = col_double(),
```

plied-microeconometrics > weeks > week1

Name	Size
..	
assets	
data.csv	375.
header.tex	1.3 I
script.R	90 B
week1a.log	49.9
week1a.pdf	151
week1a.Rmd	4.4 I
week1b.log	53.7
week1b.pdf	102:

Objects in memory

- The data frame is a matrix
  - Each row is an observation and each column is a variables
- We can also see the names of the columns like this:

```r
colnames(data)
```

```
 [1] "res_id"                 "ability"                "age"
 [4] "educyears"              "isfarmer"               "yearsfarming"
 [7] "yearsmanagingfarm"      "outsidewage"            "worriedaboutcropprices"
[10] "worriedaboutcropyields"
```

- This is the kind of thing I might do in the console since it's not really required for the script.

Calling variables in R

- Some of you might be used to Stata
- One big difference between the two is that Stata generally only has one data frame in memory at a time
  - This means that you can call a variable *without referencing the data frame*
- In R, if you want to look at a variable, you have to tell R which data frame it is in
  - This is done with the $ operator
  - For example, if I want to look at the variable “age” in the data frame “data”, I would write data$age
  - Let’s look at summary statistics for age:

```
summary(data$age)
```

```
   Min. 1st Qu.  Median    Mean 3rd Qu.    Max. 
  18.00   34.00   42.00   43.54   52.00   87.00 
```

Summary statistics for the entire data frame

- You can also use summary on the data frame instead of a single column
  - It helps to think of a data frame as rows and columns. For variables, you want to call specific columns.
- Look at the difference here (it cuts off because of the size of the slide):

```
summary(data)
```

```
    res_id        ability            age          educyears
Min.   : 501   Min.   : 10.00   Min.   :18.00   Length:7209
1st Qu.:2783   1st Qu.: 51.00   1st Qu.:34.00   Class :character
Median :4714   Median : 59.00   Median :42.00   Mode  :character
Mean   :4775   Mean   : 58.66   Mean   :43.54
3rd Qu.:6764   3rd Qu.: 67.00   3rd Qu.:52.00
Max.   :8955   Max.   :100.00   Max.   :87.00

  isfarmer          yearsfarming   yearsmanagingfarm  outsidewage
Length:7209        Min.   :-9.00   Min.   :-9.00     Min.   :2.000e+03
Class :character   1st Qu.:18.00   1st Qu.: 6.00     1st Qu.:3.500e+06
Mode  :character   Median :26.00   Median :14.00     Median :1.000e+10
                   Mean   :28.02   Mean   :15.94     Mean   :7.156e+09
                   3rd Qu.:38.00   3rd Qu.:22.00     3rd Qu.:1.000e+10
                   Max.   :70.00   Max.   :70.00     Max.   :1.000e+10
                   NA's   :219     NA's   :219       NA's   :216
worriedaboutcropprices worriedaboutcropyields
Length:7209            Length:7209
Class :character       Class :character
Mode  :character       Mode  :character
```

Calling rows/columns of a data frame (matrix)

- Think about how we refer to rows and columns in a matrix.
  - We use the row and column number, in that order.
  - For example, if I want the first row and second column of a matrix X, mathematically I could write $X_{1,2}$
- We do *the same thing in R*
- If I want the first row and second column of the data frame “data”, I would write data[1,2]
  - Note that we use square brackets instead of parentheses
  - Note that we use a comma to separate the row and column

```r
data[1,2]
```

```
# A tibble: 1 x 1
  ability
    <dbl>
1      74
```

Calling columns of a data frame (matrix)

- We can call entire columns of a data frame by leaving the row blank
  - For example, if I want the second column of the data frame “data”, I would write data[,2]
  - Note that the second column is the ability variable

```r
colnames(data)
```

```
 [1] "res_id"                 "ability"                "age"
 [4] "educyears"              "isfarmer"               "yearsfarming"
 [7] "yearsmanagingfarm"      "outsidewage"            "worriedaboutcropprices"
[10] "worriedaboutcropyields"
```

```r
data[,2]
```

```
# A tibble: 7,209 x 1
   ability
     <dbl>
 1      74
 2      42
 3      67
 4      54
 5      57
 6      72
 7      51
 8      65
 9      54
10      24
```

Missing variables R

- Missing variables are denoted by NA
  - This is different from Stata, which uses a period (.)
- Note that this is only how the PROGRAM stores missing variables. Sometimes the data itself has different missing values. PAY ATTENTION!
- For example, take a look at the first ten rows of the data frame (also note how I call the first ten rows and leave out the first column!):

```r
data[1:10,-1]
```

```
# A tibble: 10 x 9
   ability   age educyears isfarmer yearsfarming yearsmanagingfarm outsidewage
     <dbl> <dbl> <chr>     <chr>           <dbl>             <dbl>       <dbl>
 1      74    83 16        Yes                60                46     3000000
 2      42    27 7         Yes                17                17  9999999999
 3      67    49 7         Yes                20                 5     6000000
 4      54    50 7         Yes                15                10  9999999999
 5      57    70 4         Yes                40                26  9999999999
 6      72    45 7         Yes                15                15      800000
 7      51    58 7         Yes                30                25     2000000
 8      65    41 7         Yes                20                15  9999999999
 9      54    45 7         Yes                20                10      300000
10      24    70 <NA>      Yes                60                50  9999999999
# i 2 more variables: worriedaboutcropprices <chr>,
#   worriedaboutcropyields <chr>
```

Variable types

- R also has a few different types of variables
  - The most common are numeric, character, and logical
- Look at the previous code again:

```r
data[1:10,-1]
```

```
# A tibble: 10 x 9
   ability   age educyears isfarmer yearsfarming yearsmanagingfarm outsidewage
     <dbl> <dbl> <chr>     <chr>           <dbl>             <dbl>       <dbl>
 1      74    83 16        Yes                60                46     3000000
 2      42    27 7         Yes                17                17  9999999999
 3      67    49 7         Yes                20                 5     6000000
 4      54    50 7         Yes                15                10  9999999999
 5      57    70 4         Yes                40                26  9999999999
 6      72    45 7         Yes                15                15      800000
 7      51    58 7         Yes                30                25     2000000
 8      65    41 7         Yes                20                15  9999999999
 9      54    45 7         Yes                20                10      300000
10      24    70 <NA>      Yes                60                50  9999999999
# i 2 more variables: worriedaboutcropprices <chr>,
#   worriedaboutcropyields <chr>
```

Variable types

- dbl is short for double, which is a numeric variable (the “type” of numeric variable is about how much memory is needed to store it)
- chr is short for character, which is a string of characters (text)
    - Surprisingly, in our previous example, educyears was a character string even though it seemed to be a number
    - Let’s look at the possible values of educyears using the unique() function, which outputs a vector:

```r
unique(data$educyears)
```

```
 [1] "16"            "7"             "4"             NA             
 [5] "11"            "6"             "13"            "5"            
 [9] "8"             "10"            "12"            "9"            
[13] "2"             "3"             "15"            "14"           
[17] "20"            "18"            "17"            "1"            
[21] "Not Mentioned" "19"           
```

Variable types

- Interesting! It seems that there is a “Not Mentioned” value.
    - What if we want to replace those with missing, instead?
- Let’s talk through the following code
    - First note how it refers to a specific column and then a specific row
    - Also note how it uses two equal signs (==) to check whether the value is “Not Mentioned”
        - This is similar to Stata!

```
# replace "Not Mentioned" with NA
data$educyears[data$educyears == "Not Mentioned"] <- NA
# check that it worked by looking at the unique values
unique(data$educyears)
```

```
 [1] "16" "7"  "4"  NA   "11" "6"  "13" "5"  "8"  "10" "12" "9"  "2"  "3"  "15"
[16] "14" "20" "18" "17" "1"  "19"
```

```
# turn into numeric
data$educyears <- as.numeric(data$educyears)
class(data$educyears)
```

```
[1] "numeric"
```

Pipes

- One of the most useful things in R is the pipe operator (%>%)
  - This is part of the tidyverse package
  - It allows you to chain commands together
  - It makes your code much easier to read
  - It makes your code much easier to write
  - It makes your code much easier to debug
  - It makes your code much easier to share
  - It makes your code much easier to reproduce

- It's easy to use but it will take some time for you to get used to the names of the functions we can use with it
  - This also goes for other tasks in R, not just with the pipe operator

Pipes example

Here is an example of how we can use pipes with the mutate() function in tidyverse

- We are also going to use ifelse() to make this work

```
data <- data %>%
          mutate(educyears = ifelse(educyears == "Not Mentioned", NA, educyears), # if educyears=="Not Mentioned", replace
                educyears = as.numeric(educyears))    # replace educyears as numeric (instead of character)
summary(data$educyears)
```

```
   Min. 1st Qu.  Median    Mean 3rd Qu.    Max.    NA's 
  1.000   7.000   7.000   6.735   7.000  20.000    3113 
```

Pipes example

Here is an example of how we can use pipes with the mutate() function in tidyverse

- We are also going to use ifelse() to make this work

```
data <- data %>%
          mutate(educyears = ifelse(educyears == "Not Mentioned", NA, educyears), # if educyears=="Not Mentioned", replace
                educyears = as.numeric(educyears))    # replace educyears as numeric (instead of character)
summary(data$educyears)
```

```
   Min. 1st Qu.  Median    Mean 3rd Qu.    Max.    NA's 
  1.000   7.000   7.000   6.735   7.000  20.000    3113 
```

Note that we could wrap as.numeric() around the ifelse() command to do it on one line!

```
data <- data %>%
          mutate(educyears = as.numeric(ifelse(educyears == "Not Mentioned", NA, educyears))) # wrapped into one line
summary(data$educyears)
```

```
   Min. 1st Qu.  Median    Mean 3rd Qu.    Max.    NA's 
  1.000   7.000   7.000   6.735   7.000  20.000    3113 
```

Missings and functions in R

In Stata, by default, functions ignore missing values

- R does not do this by default. Look at this:

```r
data <- data %>%
     mutate(educyears = as.numeric(ifelse(educyears == "Not Mentioned", NA, educyears))) # wrapped into one line
mean(data$educyears)
```

```
[1] NA
```

If there are any missing values, the function will evalute to missing!

- But we can also do this:

```r
data <- data %>%
     mutate(educyears = as.numeric(ifelse(educyears == "Not Mentioned", NA, educyears))) # wrapped into one line
mean(data$educyears, na.rm = TRUE) # BE CAREFUL WITH THIS! Make sure it is indeed what you want to do.
```

```
[1] 6.735107
```

Functions and storing values

The mean() function in the previous slide outputs a single value - That means we could store that value as an object:

```
data <- data %>%
          mutate(educyears = as.numeric(ifelse(educyears == "Not Mentioned", NA, educyears))) # wrapped into one line
meaneduc <- mean(data$educyears, na.rm = TRUE)
sdeduc <- sd(data$educyears, na.rm = TRUE)
meaneduc
```

```
[1] 6.735107
```

```
sdeduc
```

```
[1] 2.404086
```

How is this helpful? We can use these values later in our script!

Functions and mutate()

We can combine the mean() and sd() functions within mutate to create a new, standardized variable:

```
data <- data %>%
          mutate(educyears = as.numeric(ifelse(educyears == "Not Mentioned", NA, educyears)), # wrapped into one line
                 educyears_std = (educyears - mean(educyears))/sd(educyears))
summary(data$educyears_std)
```

```
   Min. 1st Qu.  Median    Mean 3rd Qu.    Max.    NA's
     NA      NA      NA     NaN      NA      NA    7209
```

Oh no! what happened?

Functions and mutate()

We can combine the mean() and sd() functions within mutate to create a new, standardized variable:

```r
data <- data %>%
          mutate(educyears = as.numeric(ifelse(educyears == "Not Mentioned", NA, educyears)), # wrapped into one line
                 educyears_std = (educyears - mean(educyears, na.rm = T))/sd(educyears, na.rm = T))
summary(data$educyears_std)
```

```
   Min. 1st Qu.  Median    Mean 3rd Qu.    Max.    NA's 
-2.3856  0.1102  0.1102  0.0000  0.1102  5.5176    3113 
```

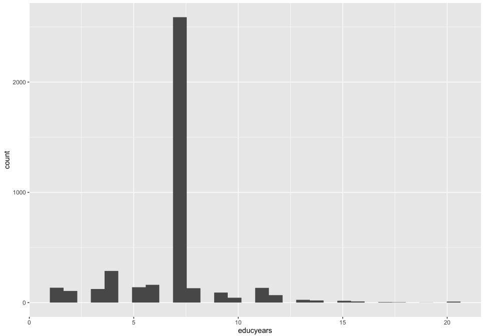
Note that we can shorten TRUE to T (or FALSE to F).

Visualizations with ggplot2

- ggplot2 is a flexible way to create visualizations in R
- The basic idea is that you create a plot object and then add layers to it
- Let's create a histogram of educyears

Visualizations with ggplot2

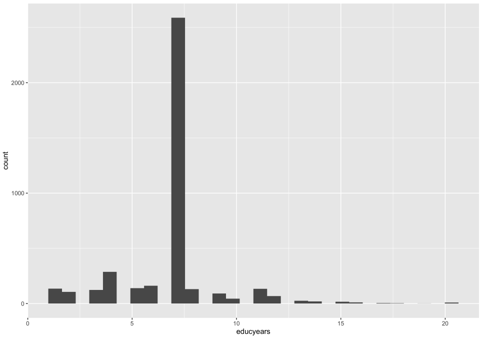
```
data <- data %>%
          mutate(educyears = as.numeric(ifelse(educyears == "Not Mentioned", NA, educyears)))
# we call ggplot() and NOT ggplot2()
ggplot() +   # note how we use + here, NOT the pipe operator
  geom_histogram(data = data, aes(x = educyears)) # the histogram with geom_histogram
```



```
# data = data tells R to use the data frame "data", and the aes() is the aesthetic
# only an x value here since a histogram uses just a SINGLE value
```

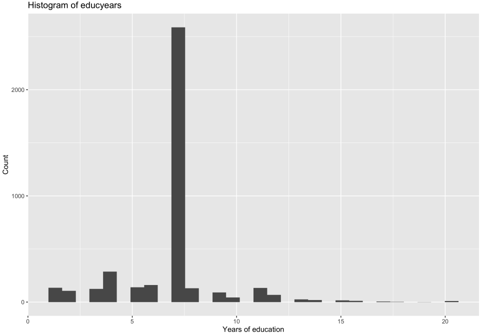
Visualizations with ggplot2

```r
data <- data %>%
          mutate(educyears = as.numeric(ifelse(educyears == "Not Mentioned", NA, educyears)))
# we can save the plot as an object
g1 <- ggplot() +
        geom_histogram(data = data, aes(x = educyears))
g1
```



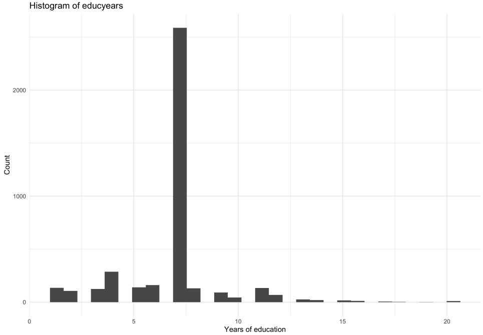
Visualizations with ggplot2

```
data <- data %>%
          mutate(educyears = as.numeric(ifelse(educyears == "Not Mentioned", NA, educyears)))
# lots of ways to change the plot
g1 <- ggplot() +
        geom_histogram(data = data, aes(x = educyears)) +
        labs(title = "Histogram of educyears",
             x = "Years of education",
             y = "Count")
g1
```



One more example

```
data <- data %>%
          mutate(educyears = as.numeric(ifelse(educyears == "Not Mentioned", NA, educyears)))
g1 <- ggplot() +
        geom_histogram(data = data, aes(x = educyears)) +
        labs(title = "Histogram of educyears",
             x = "Years of education",
             y = "Count") +
        theme_minimal()
g1
```



Let's try this with a NEW dataset

First install a new package that has a dataset we will use (you can do this in the console):

```r
install.packages("nycflights13")
```

Now let's see:

```r
library(nycflights13)
glimpse(flights)
```

```
Rows: 336,776
Columns: 19
$ year           <int> 2013, 2013, 2013, 2013, 2013, 2013, 2013, 2013, 2013, 2~
$ month          <int> 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1~
$ day            <int> 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1~
$ dep_time       <int> 517, 533, 542, 544, 554, 554, 555, 557, 557, 558, 558, ~
$ sched_dep_time <int> 515, 529, 540, 545, 600, 558, 600, 600, 600, 600, 600, ~
$ dep_delay      <dbl> 2, 4, 2, -1, -6, -4, -5, -3, -3, -2, -2, -2, -2, -2, -1~
$ arr_time       <int> 830, 850, 923, 1004, 812, 740, 913, 709, 838, 753, 849,~
$ sched_arr_time <int> 819, 830, 850, 1022, 837, 728, 854, 723, 846, 745, 851,~
$ arr_delay      <dbl> 11, 20, 33, -18, -25, 12, 19, -14, -8, 8, -2, -3, 7, -1~
$ carrier        <chr> "UA", "UA", "AA", "B6", "DL", "UA", "B6", "EV", "B6", "~
$ flight         <int> 1545, 1714, 1141, 725, 461, 1696, 507, 5708, 79, 301, 4~
$ tailnum        <chr> "N14228", "N24211", "N619AA", "N804JB", "N668DN", "N394~
$ origin         <chr> "EWR", "LGA", "JFK", "JFK", "LGA", "EWR", "EWR", "LGA",~
$ dest           <chr> "IAH", "IAH", "MIA", "BQN", "ATL", "ORD", "FLL", "IAD",~
$ air_time       <dbl> 227, 227, 160, 183, 116, 150, 158, 53, 140, 138, 149, 1~
```

Dates with lubridate

There's a nice package called lubridate that makes working with dates much easier.

```r
library(lubridate)
# create a date variable
flights$date <- as_date(paste0(flights$year, "-", flights$month, "-", flights$day))
head(flights$date)
```

```
[1] "2013-01-01" "2013-01-01" "2013-01-01" "2013-01-01" "2013-01-01"
[6] "2013-01-01"
```

Dates with lubridate

Departure time/arrival time is in the format HHMM (e.g., 1530 is 3:30pm). We can add this to the date

```r
flights$dep_time_new <- hm(paste0(flights$dep_time %/% 100, ":", flights$dep_time %% 100))
head(flights$dep_time_new, n = 20)
```

```
 [1] "5H 17M 0S" "5H 33M 0S" "5H 42M 0S" "5H 44M 0S" "5H 54M 0S" "5H 54M 0S"
 [7] "5H 55M 0S" "5H 57M 0S" "5H 57M 0S" "5H 58M 0S" "5H 58M 0S" "5H 58M 0S"
[13] "5H 58M 0S" "5H 58M 0S" "5H 59M 0S" "5H 59M 0S" "5H 59M 0S" "6H 0M 0S"
[19] "6H 0M 0S"  "6H 1M 0S"
```

Dates with lubridate

Let's fix that!

```r
flights$dep_time_new <- hm(sprintf("%02d:%02d", flights$dep_time %/% 100, flights$dep_time %% 100))

head(flights$dep_time_new, n = 20)
```

```
 [1] "5H 17M 0S" "5H 33M 0S" "5H 42M 0S" "5H 44M 0S" "5H 54M 0S" "5H 54M 0S"
 [7] "5H 55M 0S" "5H 57M 0S" "5H 57M 0S" "5H 58M 0S" "5H 58M 0S" "5H 58M 0S"
[13] "5H 58M 0S" "5H 58M 0S" "5H 59M 0S" "5H 59M 0S" "5H 59M 0S" "6H 0M 0S" 
[19] "6H 0M 0S"  "6H 1M 0S" 
```

One more example

Lubridate also lets us work with “periods”

```
flights$dep_delay_new <- as.period(flights$dep_delay, unit = "minute")
# NOTE: You have to be very careful with taking means/medians, etc.
head(flights$dep_delay_new)
```

```
[1] "2M 0S"  "4M 0S"  "2M 0S"  "-1M 0S" "-6M 0S" "-4M 0S"
```

Let’s look at some new tidyverse functions

Let’s get the average departure delay by NYC airport:

```
# Remember I said be careful with means of periods/durations! Using the original value here.
flights %>%
    group_by(origin) %>% # this groups ROWS based on their origin value
    summarize(avg_dep_delay = mean(dep_delay, na.rm = T)) # this summarizes the data, creating means absed on the grouping!
```

```
# A tibble: 3 x 2
  origin avg_dep_delay
  <chr>          <dbl>
1 EWR             15.1
2 JFK             12.1
3 LGA             10.3
```

Note that this does not create a single value. Instead it creates a tibble (a data frame) summarizing the data by our grouping variable.

Let's look at some new tidyverse functions

What if we want to save that tibble instead?

```
summat <- flights %>%
            group_by(origin) %>% # this groups ROWS based on their origin value
            summarize(avg_dep_delay = mean(dep_delay, na.rm = T)) # this summarizes the data, creating means based on groups!
summat # print the 3x2 matrix in the console
```

```
# A tibble: 3 x 2
  origin avg_dep_delay
  <chr>          <dbl>
1 EWR             15.1
2 JFK             12.1
3 LGA             10.3
```

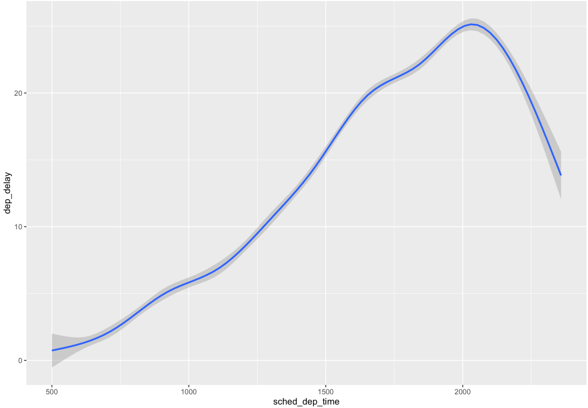
I could then output this to a table if I wanted to (using Markdown, more on this later):

origin	avg_dep_delay
EWR	15.10795
JFK	12.11216
LGA	10.34688

Let's look at a new plot

How does departure delay vary by time of day?

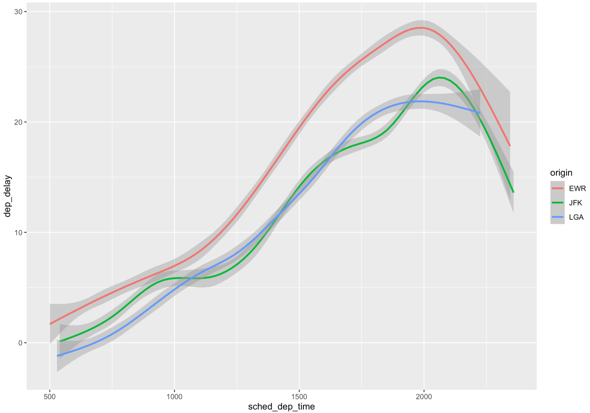
```
ggplot() +
  geom_smooth(data = flights, aes(x = sched_dep_time, y = dep_delay))
```



Let's look at a new plot

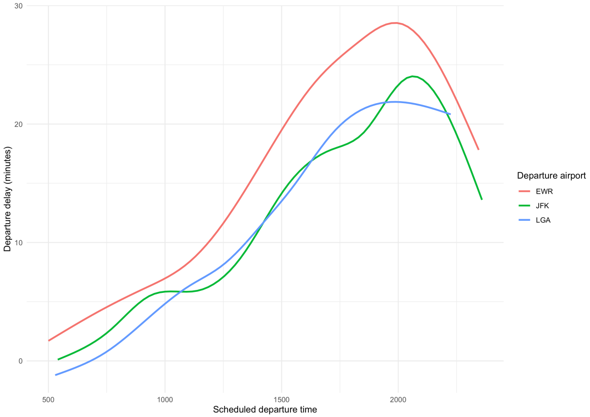
We can color code by origin, too!

```
ggplot() +
  geom_smooth(data = flights, aes(x = sched_dep_time, y = dep_delay, color = origin))
```



Make it prettier

```
ggplot() +
  geom_smooth(data = flights, aes(x = sched_dep_time, y = dep_delay, color = origin), se = FALSE) +
  labs(x = "Scheduled departure time",
       y = "Departure delay (minutes)") +
  theme_minimal() + guides(color = guide_legend(title = "Departure airport"))
```



What is R Markdown?

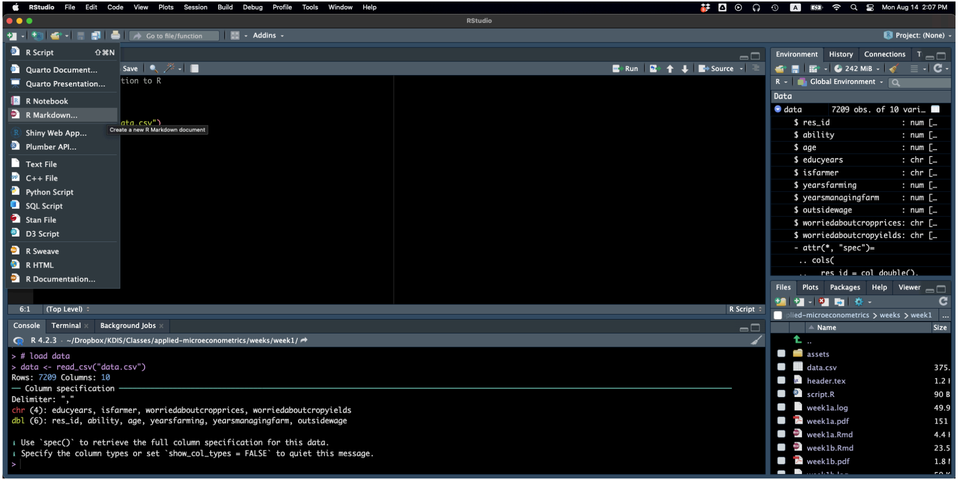
- R Markdown is a way to combine text and code
  - This allows us to create documents that are reproducible
  - We will use R Markdown to create our homework assignments
- These slides were all created in R Markdown
- My papers are written in R Markdown (well, some of them are, anyway)
  - **Here** is an example
- Yihui Xie, J. J. Allaire, and Garrett Grolemund have an awesome – free! – resource on R Markdown, **R Markdown: The Definitive Guide**

Installing R Markdown

You'll need to install R Markdown. You can do this in the console:

```
install.packages("rmarkdown")
```

Creating an R Markdown document in RStudio



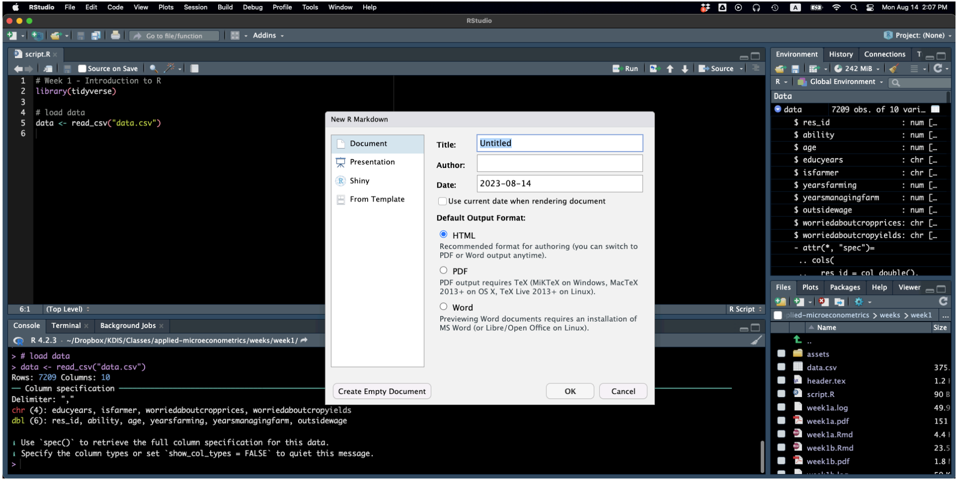
```
> # load data
> data <- read_csv("data.csv")
Rows: 7209 Columns: 10
── Column specification ────────────────────────────────────────────────────────
Delimiter: ","
chr (4): educyears, isfarmer, worriedaboutcropprices, worriedaboutcropyields
dbl (6): res_id, ability, age, yearsfarming, yearsmanagingfarm, outsidewage

ℹ Use `spec()` to retrieve the full column specification for this data.
ℹ Specify the column types or set `show_col_types = FALSE` to quiet this message.
>
```

Data

```
data    7209 obs. of 10 vari…
    $ res_id                : num […
    $ ability               : num […
    $ age                   : num […
    $ educyears             : chr […
    $ isfarmer              : chr […
    $ yearsfarming          : num […
    $ yearsmanagingfarm     : num […
    $ outsidewage           : num […
    $ worriedaboutcropprices: chr […
    $ worriedaboutcropyields: chr […
    - attr(*, "spec")=
     .. cols(
     ..   res_id = col_double(),
```

Name	Size
..	
assets	
data.csv	375.
header.tex	1.2
script.R	90 B
week1a.log	49.9
week1a.pdf	151
week1a.Rmd	4.4
week1b.Rmd	23.5
week1b.pdf	1.8

Creating an R Markdown document in RStudio



RStudio File Edit Code View Plots Session Build Debug Profile Tools Window Help
Mon Aug 14 2:07 PM
RStudio
Go to file/function
Addins
Project: (None)
script.R
Source on Save
Run
Source
6:1
(Top Level)
R Script
Environment History Connections
242 MiB
Global Environment
Files Plots Packages Help Viewer
Console Terminal Background Jobs
Create Empty Document
OK
Cancel

```
# Week 1 - Introduction to R
library(tidyverse)

# load data
data <- read_csv("data.csv")

```

New R Markdown

- Document
- Presentation
- Shiny
- From Template

Title: Untitled

Author:

Date: 2023-08-14

□ Use current date when rendering document

Default Output Format:

● HTML
Recommended format for authoring (you can switch to PDF or Word output anytime).

○ PDF
PDF output requires TeX (MiKTeX on Windows, MacTeX 2013+ on OS X, TeX Live 2013+ on Linux).

○ Word
Previewing Word documents requires an installation of MS Word (or Libre/Open Office on Linux).

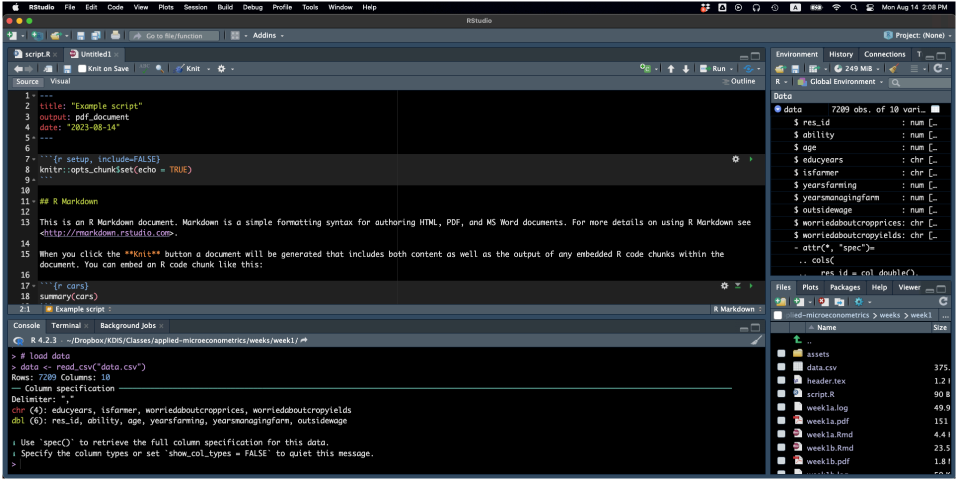
```
R 4.2.3 · ~/Dropbox/KDIS/Classes/applied-microeconometrics/weeks/week1/
> # load data
> data <- read_csv("data.csv")
Rows: 7209 Columns: 10
── Column specification ──────────────────────────────
Delimiter: ","
chr (4): educyears, isfarmer, worriedaboutcropprices, worriedaboutcropyields
dbl (6): res_id, ability, age, yearsfarming, yearsmanagingfarm, outsidewage

ℹ Use `spec()` to retrieve the full column specification for this data.
ℹ Specify the column types or set `show_col_types = FALSE` to quiet this message.
>
```

```
Data
data          7209 obs. of 10 vari…
  $ res_id                : num […
  $ ability               : num […
  $ age                   : num […
  $ educyears             : chr […
  $ isfarmer              : chr […
  $ yearsfarming          : num […
  $ yearsmanagingfarm     : num […
  $ outsidewage           : num […
  $ worriedaboutcropprices: chr […
  $ worriedaboutcropyields: chr […
  - attr(*, "spec")=
   .. cols(
   ..   res_id = col_double(),
```

plied-microeconometrics > weeks > week1

Name	Size
..	
assets	
data.csv	375.
header.tex	1.2
script.R	90 B
week1a.log	49.9
week1a.pdf	151
week1a.Rmd	4.4
week1b.Rmd	23.5
week1b.pdf	1.8

Creating an R Markdown document in RStudio



~~~
---
title: "Example script"
output: pdf_document
date: "2023-08-14"
---

```{r setup, include=FALSE}
knitr::opts_chunk$set(echo = TRUE)
```

## R Markdown

This is an R Markdown document. 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. You can embed an R code chunk like this:

```{r cars}
summary(cars)
~~~

```
R 4.2.3 · ~/Dropbox/KDIS/Classes/applied-microeconometrics/weeks/week1/
> # load data
> data <- read_csv("data.csv")
Rows: 7209 Columns: 10
── Column specification ────────────────────────────────────────────────
Delimiter: ","
chr (4): educyears, isfarmer, worriedaboutcropprices, worriedaboutcropyields
dbl (6): res_id, ability, age, yearsfarming, yearsmanagingfarm, outsidewage

ℹ Use `spec()` to retrieve the full column specification for this data.
ℹ Specify the column types or set `show_col_types = FALSE` to quiet this message.
>
```

```
Data
data          7209 obs. of 10 vari…
  $ res_id                : num […
  $ ability               : num […
  $ age                   : num […
  $ educyears             : chr […
  $ isfarmer              : chr […
  $ yearsfarming          : num […
  $ yearsmanagingfarm     : num […
  $ outsidewage           : num […
  $ worriedaboutcropprices: chr […
  $ worriedaboutcropyields: chr […
  - attr(*, "spec")=
   .. cols(
   ..   res_id = col_double(),
```

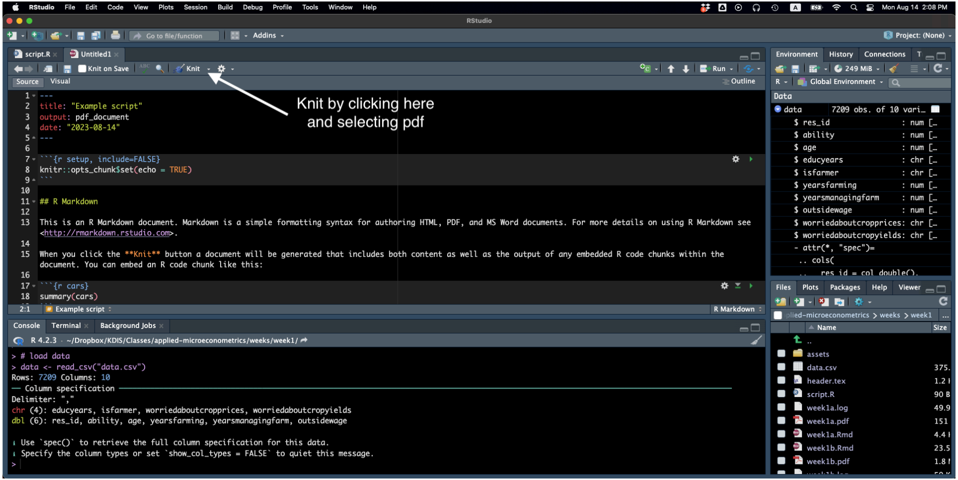
plied-microeconometrics > weeks > week1

Name	Size
..	
assets	
data.csv	375.
header.tex	1.2 I
script.R	90 B
week1a.log	49.9
week1a.pdf	151
week1a.Rmd	4.4 I
week1b.Rmd	23.5
week1b.pdf	1.8 I

Go ahead and save this document

- Go ahead and save this document in your working directory.
    - One think about Markdown files is that it will ALWAYS set the working directory to where the file is saved whenever you "knit" the document.
- What is "knitting"?
    - Knitting is the process of turning your R Markdown document into a pdf, html, or word document.
    - We will just focus on pdfs for now.

Knit it!



Knit by clicking here and selecting pdf

~~~
---
title: "Example script"
output: pdf_document
date: "2023-08-14"
---

```{r setup, include=FALSE}
knitr::opts_chunk$set(echo = TRUE)
~~~

R Markdown

This is an R Markdown document. 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. You can embed an R code chunk like this:

```{r cars}
summary(cars)
```

2:1

```
R 4.2.3 · ~/Dropbox/KDIS/Classes/applied-microeconometrics/weeks/week1/
> # load data
> data <- read_csv("data.csv")
Rows: 7209 Columns: 10
── Column specification ──────────────────────────────────────────────
Delimiter: ","
chr (4): educyears, isfarmer, worriedaboutcropprices, worriedaboutcropyields
dbl (6): res_id, ability, age, yearsfarming, yearsmanagingfarm, outsidewage

ℹ Use `spec()` to retrieve the full column specification for this data.
ℹ Specify the column types or set `show_col_types = FALSE` to quiet this message.
>
```

```
Data
data          7209 obs. of 10 vari…
   $ res_id                : num […
   $ ability               : num […
   $ age                   : num […
   $ educyears             : chr […
   $ isfarmer              : chr […
   $ yearsfarming          : num […
   $ yearsmanagingfarm     : num […
   $ outsidewage           : num […
   $ worriedaboutcropprices: chr […
   $ worriedaboutcropyields: chr […
   - attr(*, "spec")=
    .. cols(
    ..   res_id = col_double(),
```

plied-microeconometrics > weeks > week1

Name	Size
..	
assets	
data.csv	375.
header.tex	1.2 I
script.R	90 B
week1a.log	49.9
week1a.pdf	151
week1a.Rmd	4.4 I
week1b.Rmd	23.5
week1b.pdf	1.8 I

Check out the document you just created

- Go to your working directory and open the pdf to see what it looks like.
    - It will always create the pdf in the same folder as the .Rmd file.

YAML header

- At the very top of the document is some information about the document
  - This is called the YAML header
  - It tells R Markdown what kind of document to create
  - It also allows you to set some options
  - DO NOT DELETE THE — AT THE TOP AND BOTTOM OF THE YAML HEADER!
- You can change the title and date as you please
  - For today's date, you can use Sys.Date() within R inline code (more in a second):

```
date: "`r Sys.Date()`"
```

The setup chunk

- Just below the YAML header you'll see a "code chunk" called "setup" (r setup, include = FALSE)
- Note how it has ``` and ``` at the top and bottom. This differentiates the "code chunk" from the rest of the document.
    - Whenever you want to add a code chunk, you *must* have the ``` at the top and bottom of it, at the beginning of the line.
- Use the setup code chunk to load any packages or data that you want to use in the rest of the document.
    - Later code chunks are "local": they will be able to access things from the setup chunk *but not from other code chunks*.

The setup chunk

This is an example of what the setup chunk looks like.

~~~
```{r setup, include=FALSE}
# universal chunk options.
# echo = TRUE will show the code in the document.
# echo = FALSE will not.
knitr::opts_chunk$set(echo = TRUE)

# load any packages you want to use throughout the document.
library(tidyverse)

# load any data you want to use throughout the document.
data <- read_csv("data.csv")
```
~~~

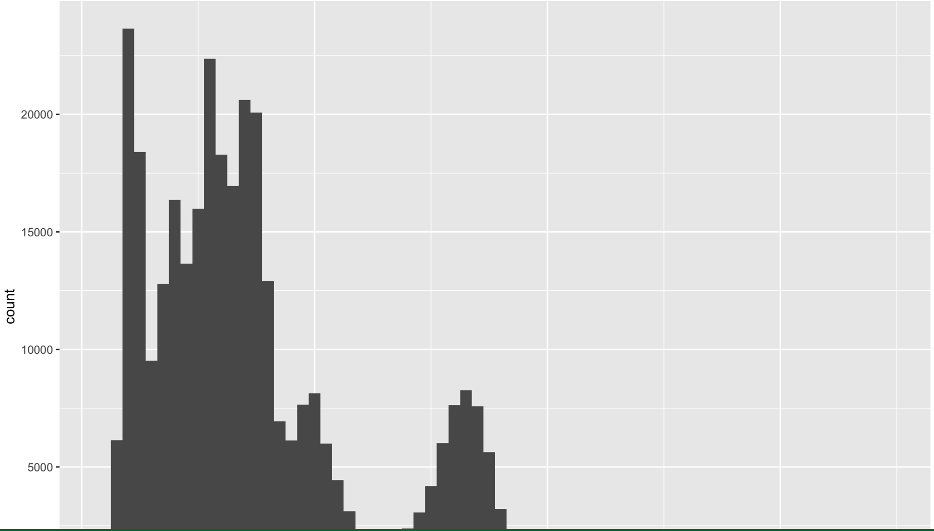
Code chunks

Here is an example of a regular code chunk.

~~~
```{r chunkexample, include = TRUE}
# note that I named the chunk.
# all chunks must have a UNIQUE name.
# you will get an error if they don't

# I already loaded by data above
ggplot(flights) +
  geom_histogram(aes(x = air_time), binwidth = 10)
```
~~~

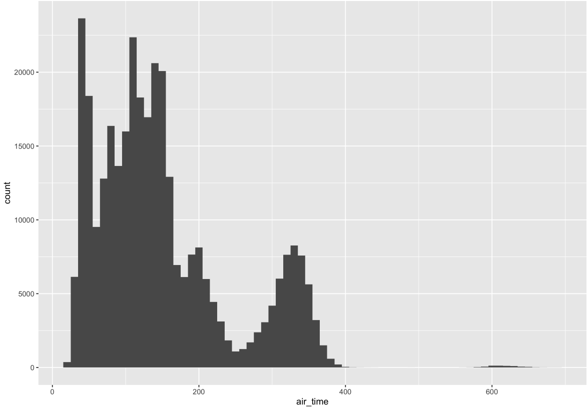
Code chunks

Here is the output of that chunk:



Code chunks

Oh no! It looks bad! Changes:



Code chunks

How did I do that?

~~~
```{r chunkexample, include = TRUE, warning = FALSE, out.width = "55%",
          fig.align = "center"}
# note that I named the chunk.
# all chunks must have a UNIQUE name.
# you will get an error if they don't

# I already loaded by data above
ggplot(flights) +
  geom_histogram(aes(x = air_time), binwidth = 10)
```
~~~

- NOTE: The start of the chunk must be on ONE line. It is wrapped here just for presentation.

Code chunks options

- There are lots of code chunk options.
- You can find the code chunk options here (https://rpubs.com/Lingling912/870659)
- This will get easier to use as you get more and more practice.

Starting new sections/subsections

```
# This will create a new section

## This will create a new sub-section

### This will create a new sub-sub-section
Don't do this.
```

You can add R inline code

- You can add R inline code using the $`r'$ operator.
  - For example, if I want to add the date, I can write 2024-09-08
  - There are 22 columns in the flights data.
  - There are 336776 columns in the flights data.

```
- You can add R inline code using the $`r'$ operator.
  - For example, if I want to add the date, I can write 2024-09-08
  - There are 22 columns in the flights data.
  - There are 336776 columns in the flights data.
```

Enumerated lists/bullets

- I like lists.
  - With indentations.

1. Really, I do.
   1. Indent! It’s just a tab.

      ““markdown

- I like lists.
  - With indentations.

1. Really, I do.
   1. Indent! It’s just a tab. ““

latex

- R Markdown uses latex to create pdfs. This allows you to do some cool things.
- For example, it is easy to add equations with latex, using $:

$y = x + \varepsilon$

```
- For example, it is easy to add equations with latex, using \$:

$y = x + \varepsilon$
```

latex

- I can center it, too:

$$y = x + \varepsilon$$

```
- I can center it, too:

$$y = x + \varepsilon$$
```

latex

$$y = x + \varepsilon \tag{1}$$

- In Equation 1

```
\begin{gather}\label{eq1} y = x + \varepsilon \end{gather}
- In Equation \autoref{eq1}
```

- You might think, so what? Well what's cool is that if we add equations before it, the number will automatically update!

latex

- latex is particularly helpful for rendering math
- You can find a handy reference guide here (https://icl.utk.edu/~mgates3/docs/latex.pdf)

Creating tables

- There are lots of ways to create tables in R Markdown.
  - I will show you how using the kable() function in the knitr package.
  - You do not need to download this package, it is already installed with R Markdown.
    - There is extra functionality in the kableExtra package. You need to download this and laod it if you want to use it.

Creating tables

```r
summat <- flights %>%
            # this groups ROWS based on their origin value
            group_by(origin) %>%
            # create means by group
            summarize(avg_dep_delay = mean(dep_delay, na.rm = T),
                      avg_arr_delay = mean(arr_delay, na.rm = T),
                      avg_air_time = mean(air_time, na.rm = T),
                      flights = n())
# output
kable(summat,
      align = "cccc")
```

Creating tables

origin	avg_dep_delay	avg_arr_delay	avg_air_time	flights
EWR	15.10795	9.107055	153.3000	120835
JFK	12.11216	5.551481	178.3490	111279
LGA	10.34688	5.783488	117.8258	104662

I don’t like that at all! Let’s make it pretty.

```r
summat <- flights %>%
            # this groups ROWS based on their origin value
            group_by(origin) %>%
            # create means by group, ROUNDING to two decimal places
            summarize(avg_dep_delay = round(mean(dep_delay, na.rm = T), 2),
                      avg_arr_delay = round(mean(arr_delay, na.rm = T), 2),
                      avg_air_time = round(mean(air_time, na.rm = T), 2),
                      flights = n())
# rename columns
colnames(summat) <- c("Origin", "Departure Delay", "Arrival Delay", "Flight Time", "Flights")
# output
kable(summat, caption = "Averages by origin (minutes)",
      align = "ccc", linesep = "",
      booktabs = TRUE) %>% # this is from kablextra. You don't have to use it, but I like it.
      kable_classic_2() # this is also from kablextra
```

I don't like that at all! Let's make it pretty.

Table 1: Averages by origin (minutes)

Origin	Departure Delay	Arrival Delay	Flight Time	Flights
EWR	15.11	9.11	153.30	120835
JFK	12.11	5.55	178.35	111279
LGA	10.35	5.78	117.83	104662

One more change!

```r
summat <- flights %>%
            # this groups ROWS based on their origin value
            group_by(origin) %>%
            # create means by group, ROUNDING to two decimal places
            summarize(avg_dep_delay = round(mean(dep_delay, na.rm = T), 2),
                      avg_arr_delay = round(mean(arr_delay, na.rm = T), 2),
                      avg_air_time = round(mean(air_time, na.rm = T), 2),
                      flights = n())
summat$flights <- format(summat$flights, big.mark = ",", scientific = FALSE)
# rename columns
colnames(summat) <- c("Origin", "Departure Delay", "Arrival Delay", "Flight Time", "Flights")
summat <- t(summat)
# output
kable(summat, caption = "Averages by origin (minutes)",
      align = "ccc", linesep = "",
      booktabs = TRUE) %>% # this is from kablextra. You don't have to use it, but I like it.
      row_spec(c(1, 4), hline_after = TRUE) %>% # this is also from kablextra
      kable_classic_2() # this is also from kablextra
```

One more change!

Table 2: Averages by origin (minutes)

Origin	EWR	JFK	LGA
Departure Delay	15.11	12.11	10.35
Arrival Delay	9.11	5.55	5.78
Flight Time	153.30	178.35	117.83
Flights	120,835	111,279	104,662

Enough for now

- That's enough on tables for now
- As you can see, there are lots of ways to customize tables
- Where this becomes really powerful is when you combine it with R code to create tables dynamically
  - I will teach you to use a package called fixest that helps automate some of this
  - If you change your specification, your tables will update AUTOMATICALLY!
  - Ever tried to manually change a table in Word? Never again.

Some tips

- When I write a paper in Markdown, I generally do not do all of my analysis in the Markdown document
- Instead, I do the analysis in another script and then save the resulting tables
- I then load these tables in the setup chunk of my Markdown document and use them in the document
    - For figures, it depends. For a simple summary figure, I might load the data in the Markdown document and create the figure there.

First assignment

- Assignment for next week (due one week from today):
  - Create a simple markdown document
  - You can find the assignment here (on the course GitHub page)
- Next week, you will turn in on e-KDIS:
  - R script (if there is one)
  - R Markdown script
  - pdf of the R Markdown document