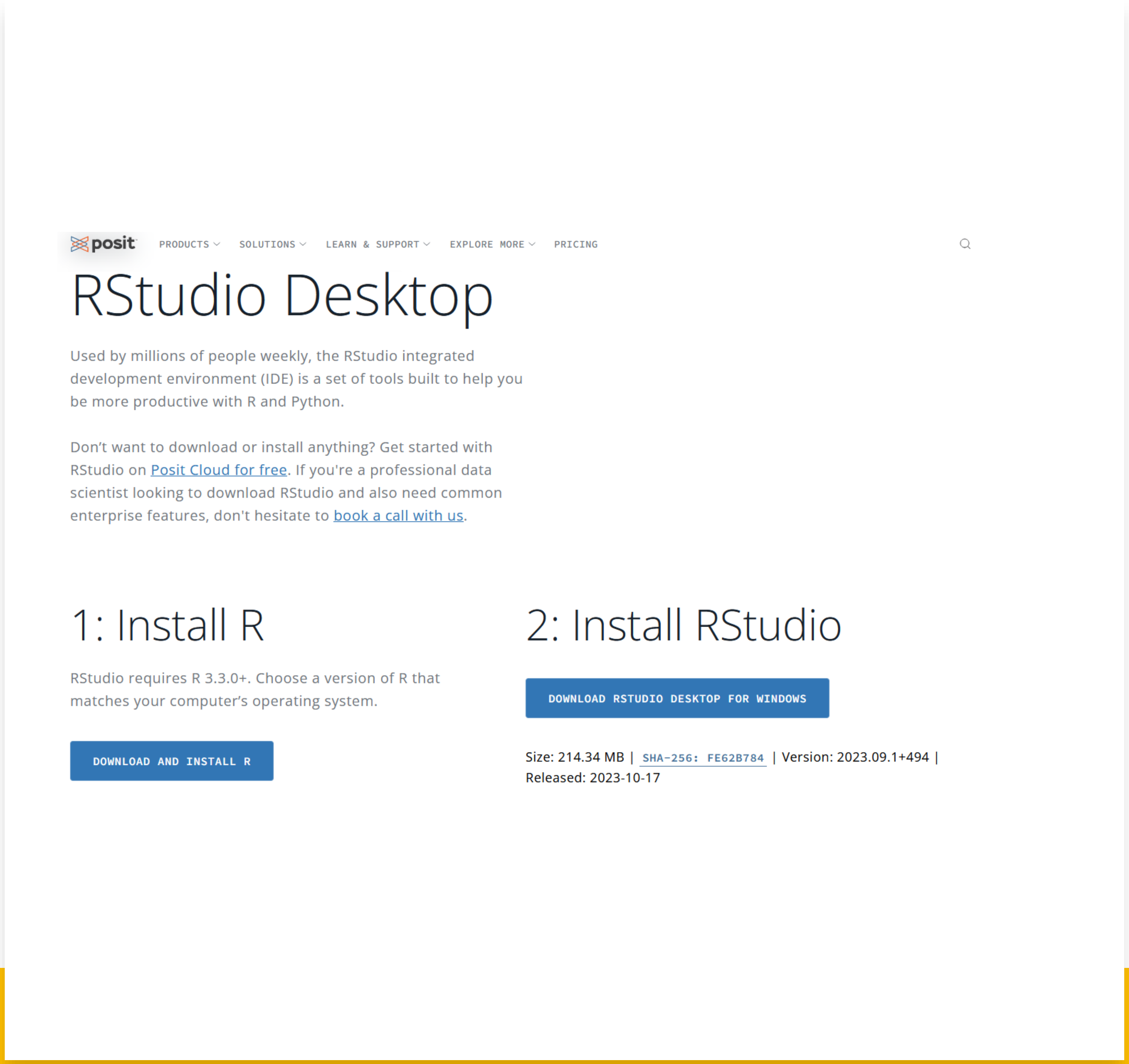
Introduction

- Using R is a long road
- I am hoping this can provide a easy introduction into using it as a data pipeline
- You can join AIR-RUG (Association for Institutional Research R-User Group) for support in all your work

Installation

- There are two ways you can use R and R Studio
- Install R https://cran.r-project.org/
- Install R Studio from POSIT https://posit.co/download/rstudio-desktop/
- Note the last link will have links to both R and R Studio



RStudio Desktop

Used by millions of people weekly, the RStudio integrated development environment (IDE) is a set of tools built to help you be more productive with R and Python.

Don't want to download or install anything? Get started with RStudio on Posit Cloud for free. If you're a professional data scientist looking to download RStudio and also need common enterprise features, don't hesitate to book a call with us.

1: Install R

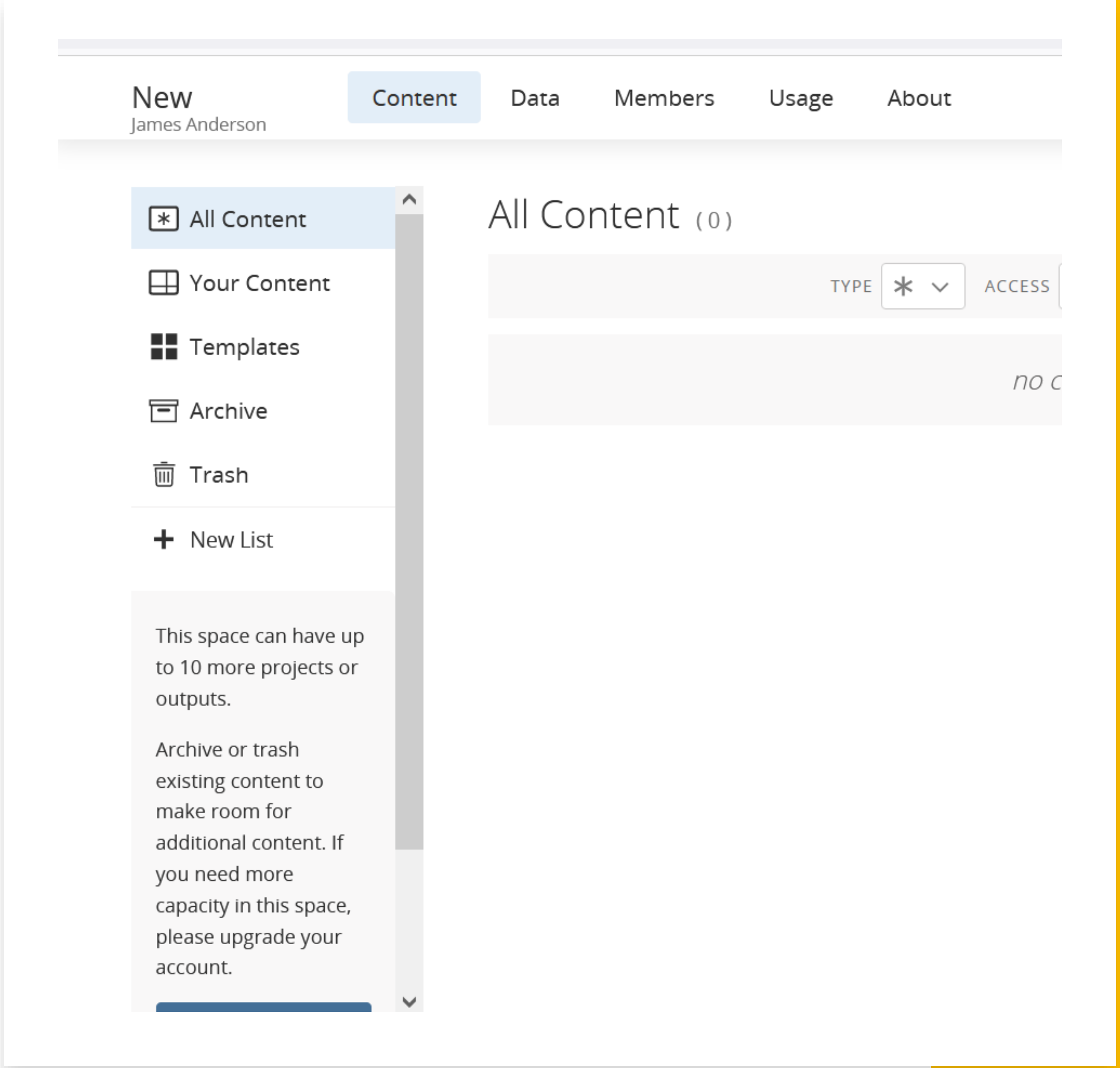
RStudio requires R 3.3.0+. Choose a version of R that matches your computer's operating system.

2: Install RStudio

Size: 214.34 MB | SHA-256: FE62B784 | Version: 2023.09.1+494 | Released: 2023-10-17

Online

- The second way is to work with it online
- Posit has an online option that is free
- Access at https://posit.cloud/
- You have a 25 projects and 25 hours free
- If you don't have Rstudio installed already please use that for today!

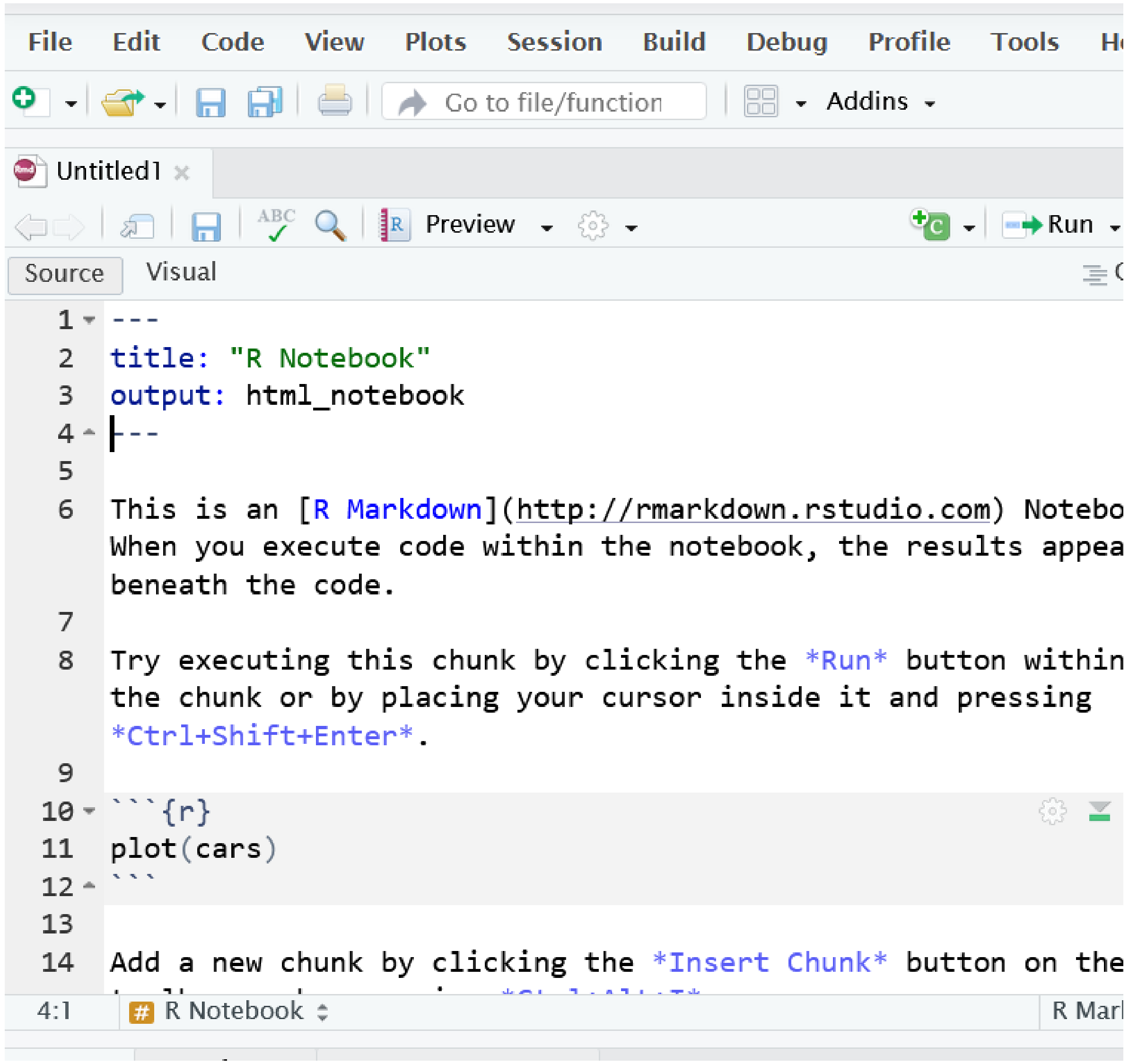


This space can have up to 10 more projects or outputs.

Archive or trash existing content to make room for additional content. If you need more capacity in this space, please upgrade your account.

RMD file structure

- A RMD file is a way to create a document that will turn into PDF, Word, or HTML document with code in it.
- It allows you to use R, python, html, or other code in a document
- You can use LaTex writeup and YAML headers to create the document
- Lets go and look at one now!



~~~
---
title: "R Notebook"
output: html_notebook
---

This is an [R Markdown](http://rmarkdown.rstudio.com) Notebo
When you execute code within the notebook, the results appea
beneath the code.

Try executing this chunk by clicking the *Run* button within
the chunk or by placing your cursor inside it and pressing
*Ctrl+Shift+Enter*.

```{r}
plot(cars)
```

Add a new chunk by clicking the *Insert Chunk* button on the
~~~

Packages

One of the most versatile part of R is the use of packages.

What are packages?

- I have some code that I made that other people have found useful
- I write it up and create a "package" of functions to use
- Other people can download and use the package

Some Important Packages

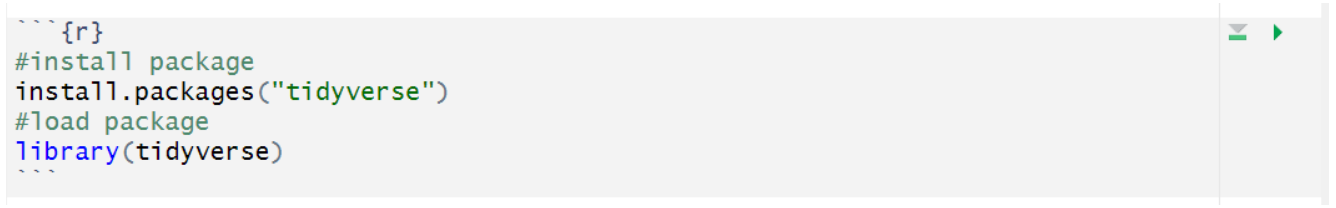
Tidyverse: an ecosystem of packages by Hadley Wickman

kableExtra: a handy table creation package that I use daily by Hao Zhu

readxl: Importing excel files from Jenny Bryan

scales: Used mostly for percentages, also by Hadley Wickman

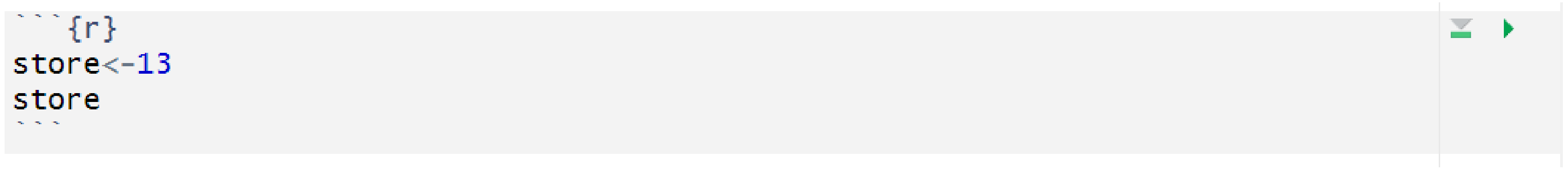
Installation and loading of Packages



~~~
```{r}
#install package
install.packages("tidyverse")
#load package
library(tidyverse)
```
~~~

- You use the command install.packages to install an individual package.
- A package only has to be installed once. Once it is installed, you still have to load it into the environment in order to use it.

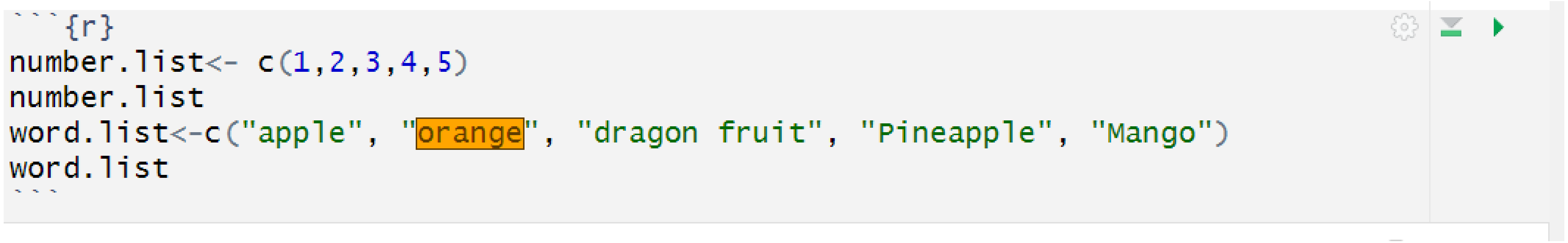
Storing Objects



```{r}
store<-13
store
```

- We can make a code section by clicking on insert new code section or ctrl+alt+i
- The <- symbol is used to store objects for later use without having to retype them
- The name of the object cannot start with a number

Lists



~~~
```{r}
number.list<- c(1,2,3,4,5)
number.list
word.list<-c("apple", "orange", "dragon fruit", "Pineapple", "Mango")
word.list
```
~~~

- R uses the letter c to start lists (The c means combine)
- You can use numbers or words in the list
- Words use quotations around each individual entry

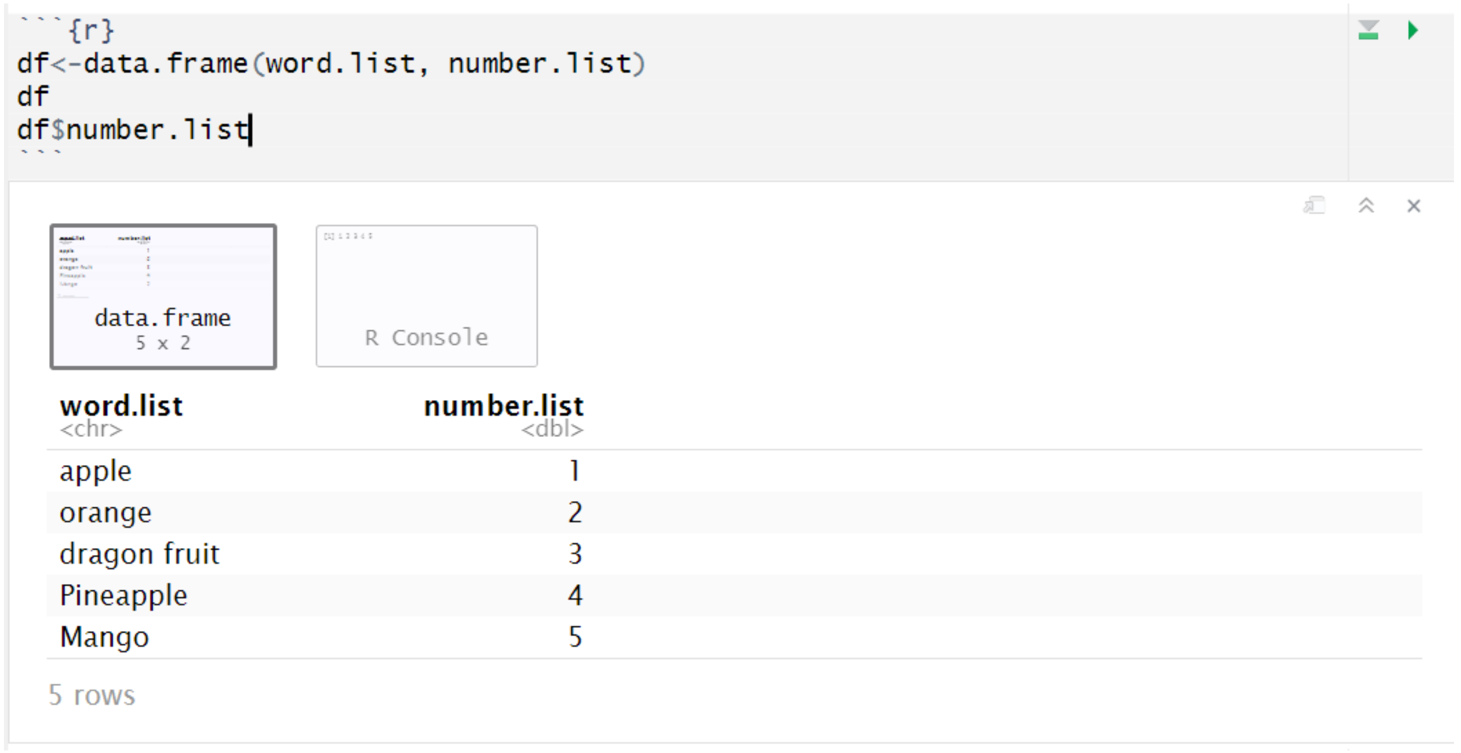
Dataframes

A data frame is a combination of rows and columns

- You can combine lists into a data frame
- You can import data as a data frame
- You can merge data using left join, right join, and inner joins (not covered today)

Combining lists using data.frame

- data.frame allows you to take lists of similar data and put them in as individual columns of data in a data frame.
- From here, I can call individual columns like so:



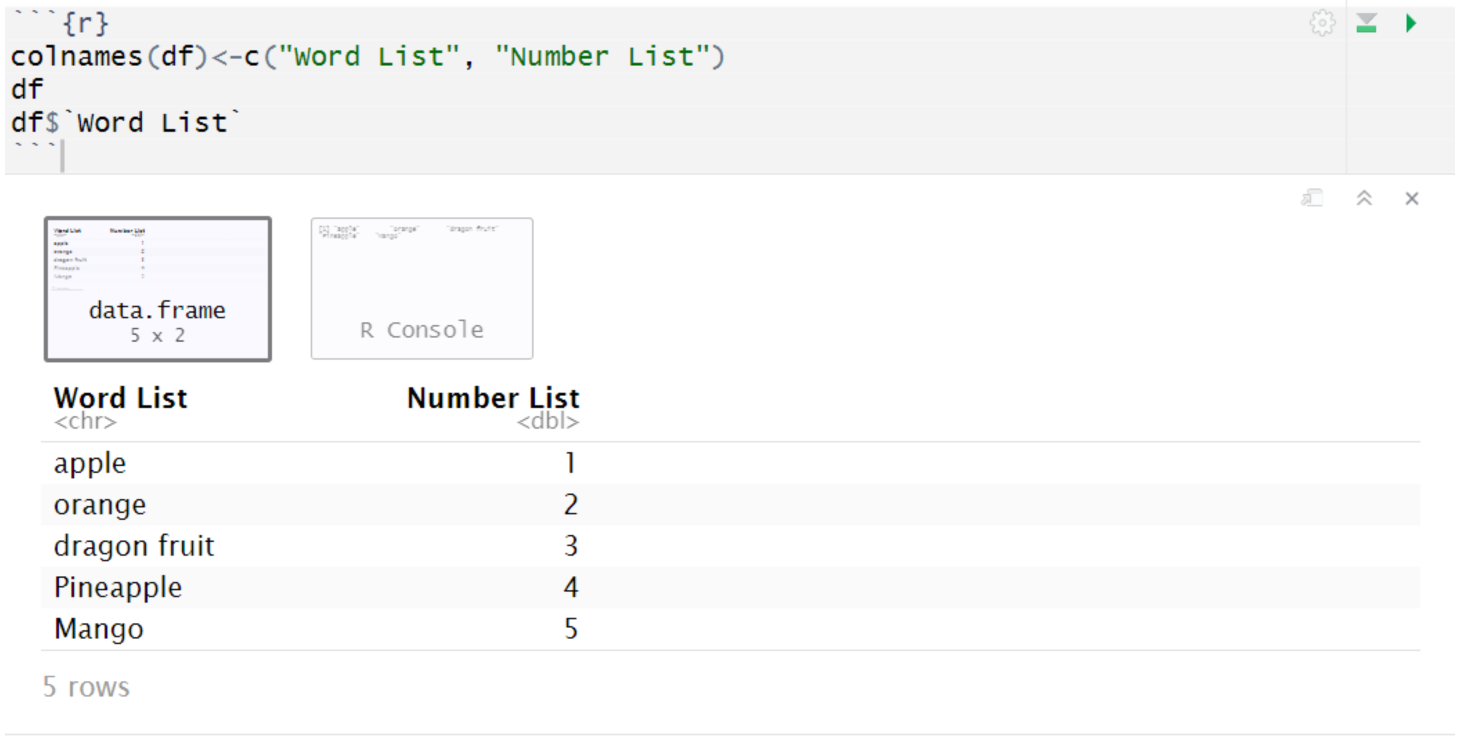
~~~
```{r}
df<-data.frame(word.list, number.list)
df
df$number.list
```
~~~

word.list <chr>	number.list <dbl>
apple	1
orange	2
dragon fruit	3
Pineapple	4
Mango	5

5 rows

Adding column and row names

- Also, since this will most likely be turned into a table, we don't like those columns.- We can rename the columns using the following command.- When we do so, the way that the column pulls up changes (notice it's in single quotes now)



~~~
```{r}
colnames(df)<-c("Word List", "Number List")
df
df$`Word List`
```
~~~

Word List <chr>	Number List <dbl>
apple	1
orange	2
dragon fruit	3
Pineapple	4
Mango	5

5 rows

Importing Data

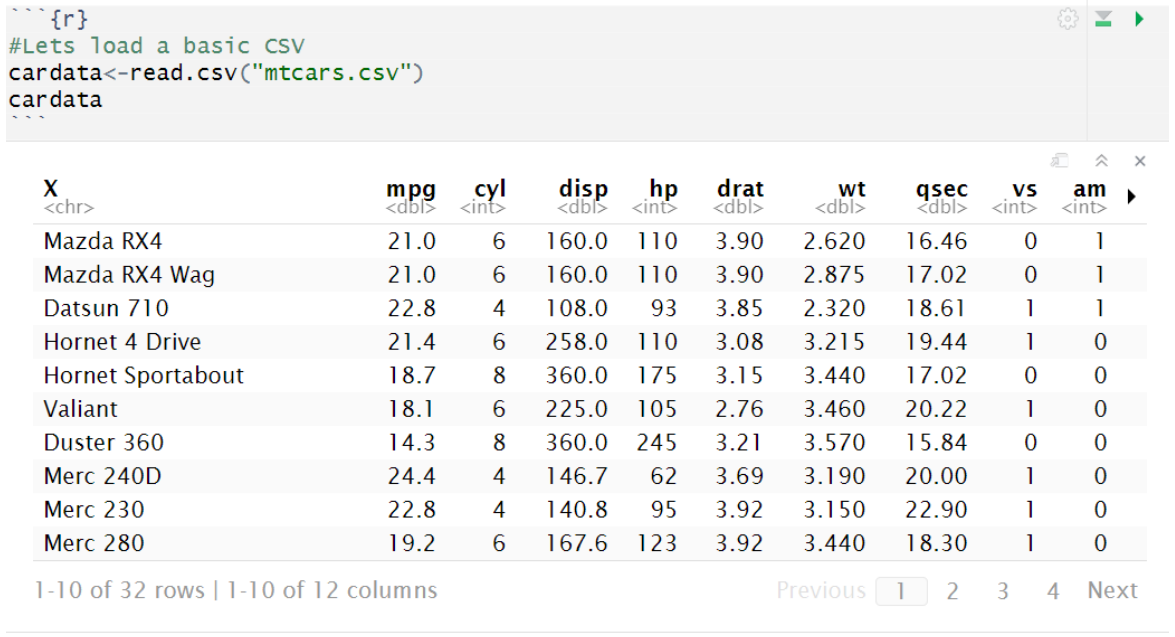
We can use a couple different ways to import data

R can load nearly all types of data including

- CSV
- Excel
- SAS save files
- Table Files
- Text Files
- Many more (I've even loaded GPS data files once)

The read.csv file loads csv files

Loading a CSV file



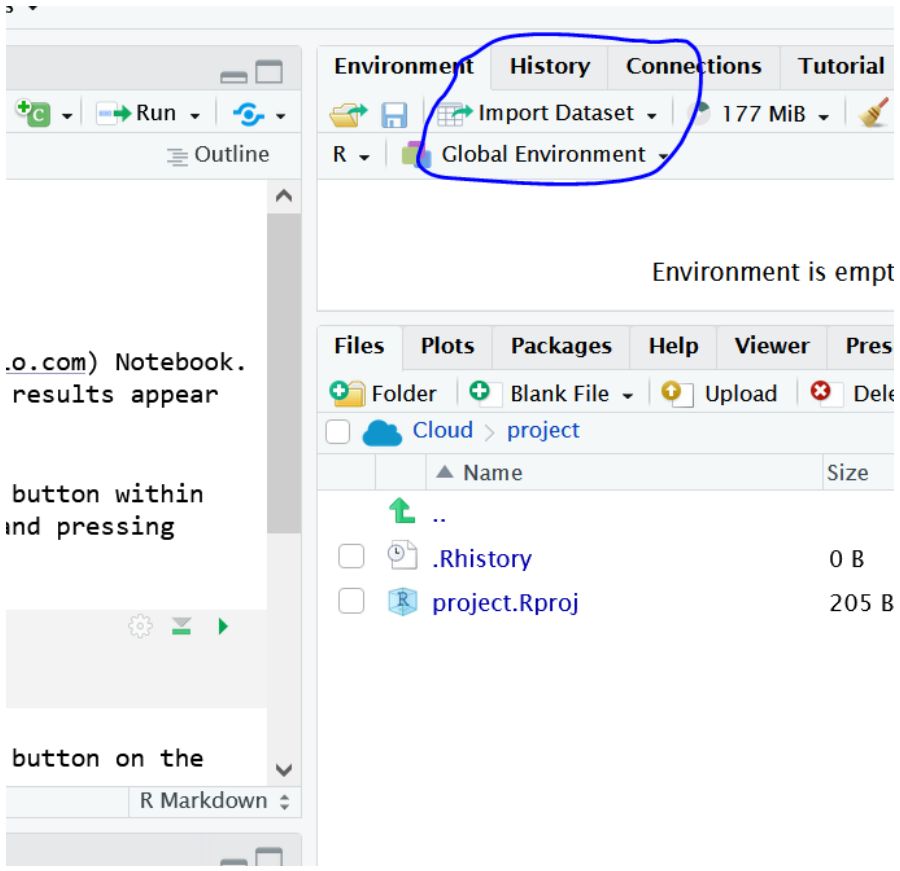
~~~r
```{r}
#Lets load a basic CSV
cardata<-read.csv("mtcars.csv")
cardata
~~~
```

X <chr>	mpg <dbl>	cyl <int>	disp <dbl>	hp <int>	drat <dbl>	wt <dbl>	qsec <dbl>	vs <int>	am <int>
Mazda RX4	21.0	6	160.0	110	3.90	2.620	16.46	0	1
Mazda RX4 Wag	21.0	6	160.0	110	3.90	2.875	17.02	0	1
Datsun 710	22.8	4	108.0	93	3.85	2.320	18.61	1	1
Hornet 4 Drive	21.4	6	258.0	110	3.08	3.215	19.44	1	0
Hornet Sportabout	18.7	8	360.0	175	3.15	3.440	17.02	0	0
Valiant	18.1	6	225.0	105	2.76	3.460	20.22	1	0
Duster 360	14.3	8	360.0	245	3.21	3.570	15.84	0	0
Merc 240D	24.4	4	146.7	62	3.69	3.190	20.00	1	0
Merc 230	22.8	4	140.8	95	3.92	3.150	22.90	1	0
Merc 280	19.2	6	167.6	123	3.92	3.440	18.30	1	0

1-10 of 32 rows | 1-10 of 12 columns
```

## Using a GUI to make life easier!

We don't always have things where they are easy, so we can make things easy for you to load!

The top right of your Rstudio GUI has an area for Importing Data that we can use to make life easier!

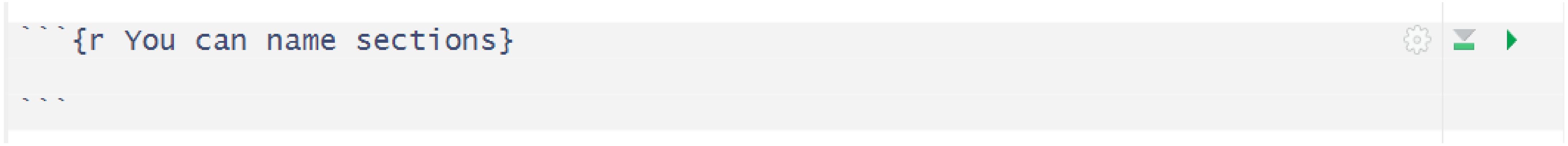
- Make sure Environment Tab is selected
- Click on Import Dataset
- Select Type of Data you want to import (avoid base to make it more user friendly)
- If you are using POSIT online you need to upload your file now
- Click Browse to search for file
- Change the name in the bottom left to a descriptive name
- Copy the first two lines of code in the bottom right
- Paste the code into a code section



Environment is empt

```
.o.com) Notebook.
results appear

button within
nd pressing

button on the
```

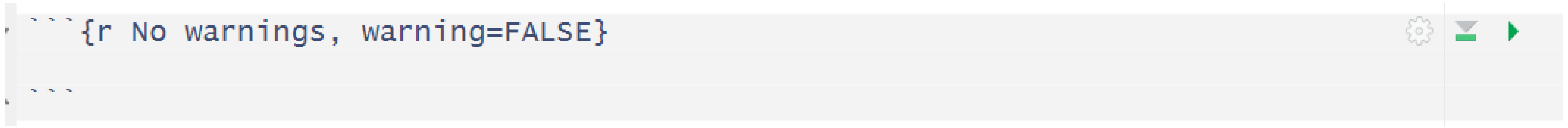
Name	Size
..	
.Rhistory	0 B
project.Rproj	205 B



~~~
```{r You can name sections}

```
~~~

Naming Code Sections

- When you are working, things break randomly and you have to fix them
- We like to know where they break, especially if you are using code sections
- Use descriptive names by clicking on the wheel in the code section or by typing next to the r in the code section

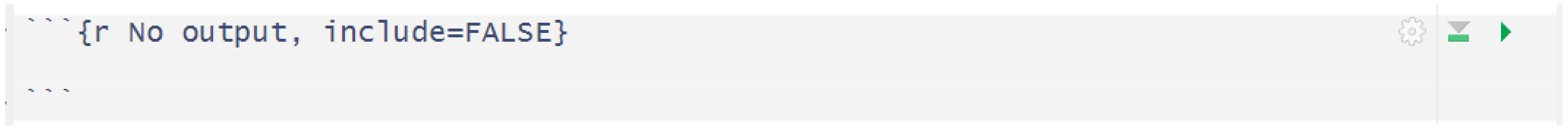


~~~
```{r No warnings, warning=FALSE}

```
~~~

Suppressing code, output, and messages

We don't always want to show code, you usually don't want messages, and sometimes don't even want to run the code.

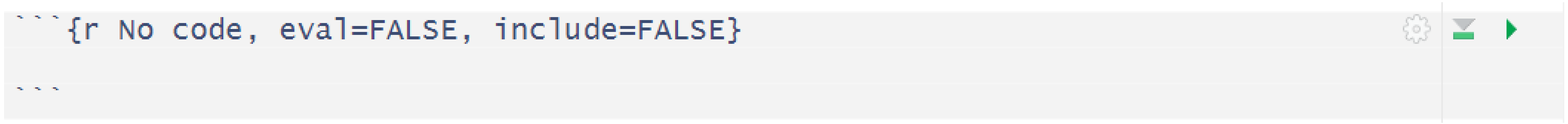
- We can suppress warnings like this


~~~
```{r No output, include=FALSE}

```
~~~

Suppressing code, output, and messages

We don't always want to show code, you usually don't want messages, and sometimes don't even want to run the code.

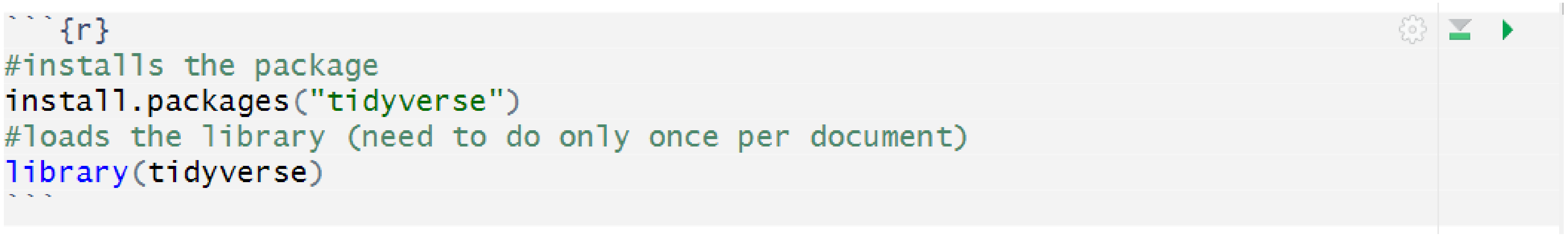
- We can also suppress output like this


~~~
```{r No code, eval=FALSE, include=FALSE}

```
~~~

Suppressing code, output, and messages

We don't always want to show code, you usually don't want messages, and sometimes don't even want to run the code.

- We can also suppress code like this



~~~
```{r}
#installs the package
install.packages("tidyverse")
#loads the library (need to do only once per document)
library(tidyverse)
```
~~~

Welcome to the TidyVerse!

Installation of tidyverse

What is tidyverse?

Tidyverse is a collection of packages created by Hadly Wickman. It allows for:

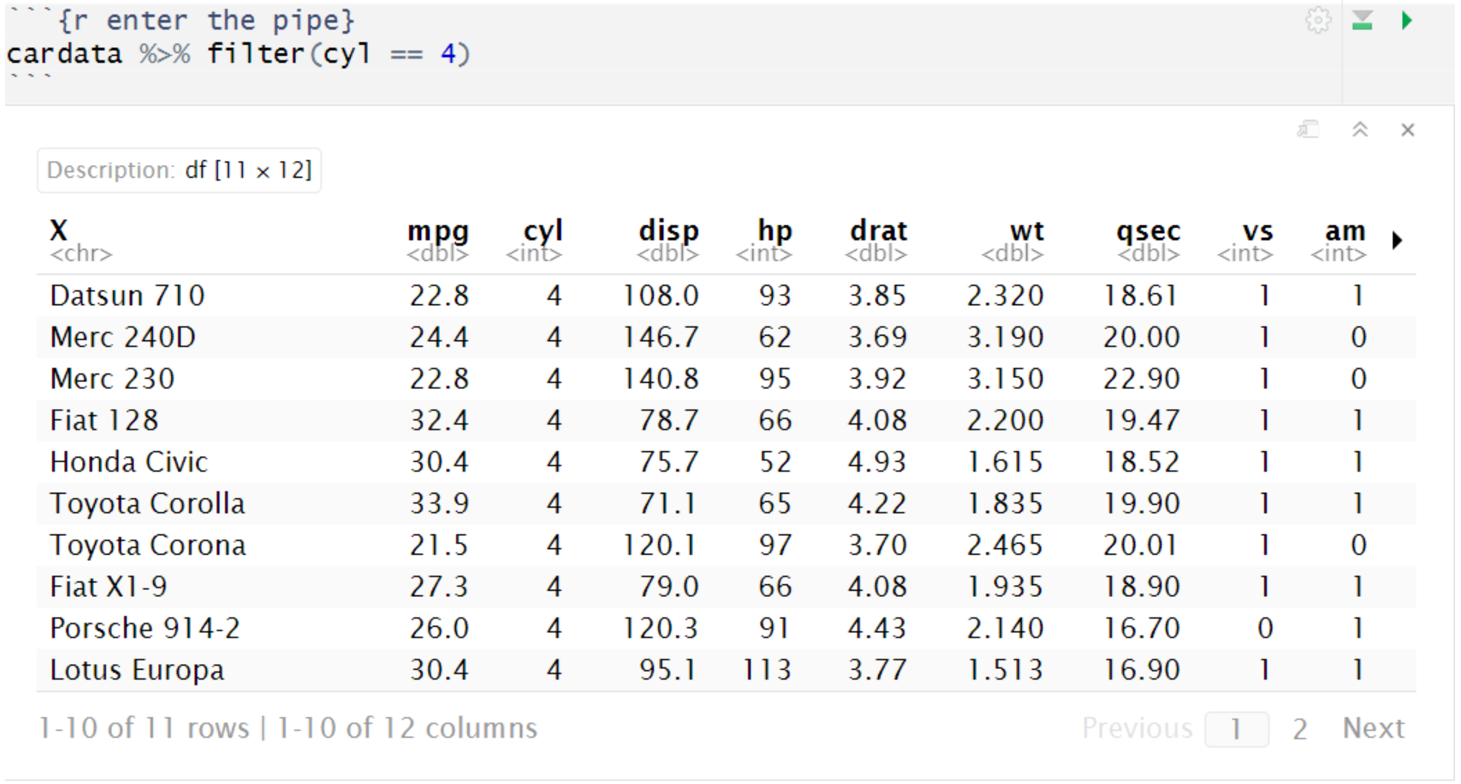
- Data wrangling from the dplyr package for filtering, transforming, and summarizing data
- Data Visualization from ggplot2 package for flexible and working visualizations
- All basic commands, such as pipes, interwork in the system allowing for ease of use

dplyr, the pipe, and summarise

- A lot of what we do in Institutional Research is to take data and summarize it.
- The dplyr package uses pipes and summarise to do a lot of what we do easily

Lets use the pipe

- I can use the pipe to take the output of one of the things I'm doing and use it in the next.
- If I want to only filter out, for instance, cars with 4 cylinders I can use filter



~~~
```{r enter the pipe}
cardata %>% filter(cyl == 4)
```
~~~

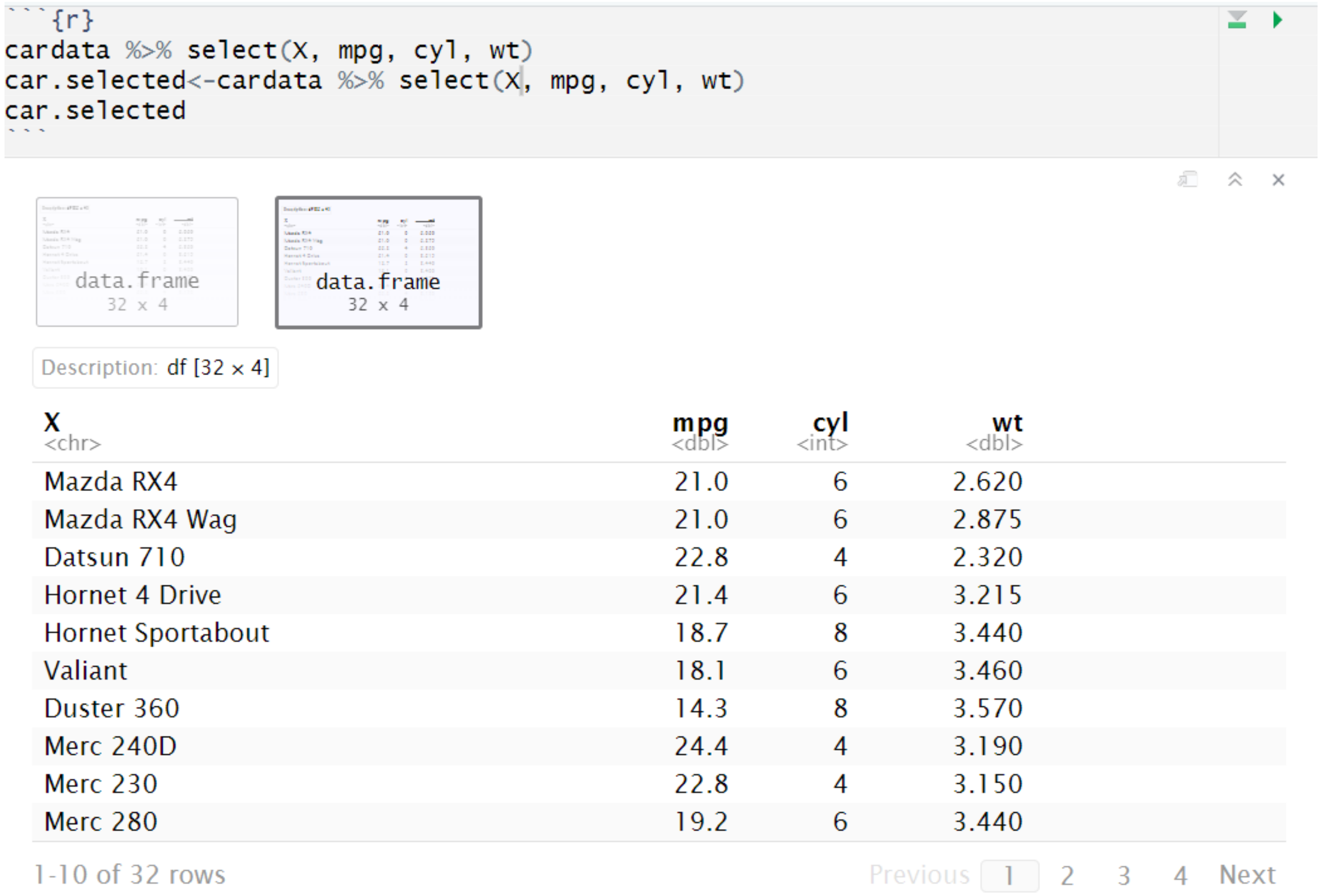
Description: df [11 x 12]

X <chr>	mpg <dbl>	cyl <int>	disp <dbl>	hp <int>	drat <dbl>	wt <dbl>	qsec <dbl>	vs <int>	am <int>
Datsun 710	22.8	4	108.0	93	3.85	2.320	18.61	1	1
Merc 240D	24.4	4	146.7	62	3.69	3.190	20.00	1	0
Merc 230	22.8	4	140.8	95	3.92	3.150	22.90	1	0
Fiat 128	32.4	4	78.7	66	4.08	2.200	19.47	1	1
Honda Civic	30.4	4	75.7	52	4.93	1.615	18.52	1	1
Toyota Corolla	33.9	4	71.1	65	4.22	1.835	19.90	1	1
Toyota Corona	21.5	4	120.1	97	3.70	2.465	20.01	1	0
Fiat X1-9	27.3	4	79.0	66	4.08	1.935	18.90	1	1
Porsche 914-2	26.0	4	120.3	91	4.43	2.140	16.70	0	1
Lotus Europa	30.4	4	95.1	113	3.77	1.513	16.90	1	1

1-10 of 11 rows | 1-10 of 12 columns

Selecting Columns

If I only want to select certain things, I can do that as well using select



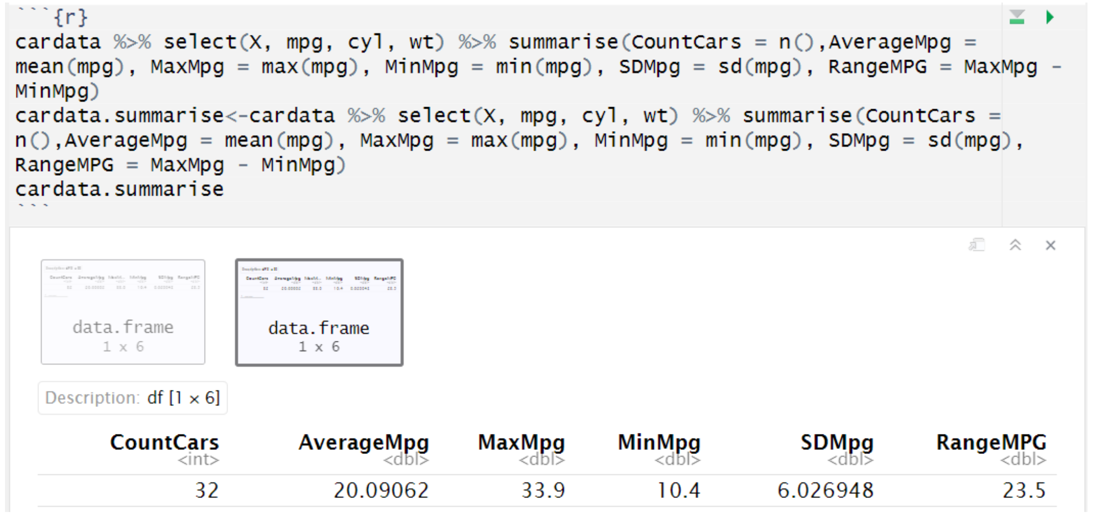
~~~
```{r}
cardata %>% select(X, mpg, cyl, wt)
car.selected<-cardata %>% select(X, mpg, cyl, wt)
car.selected
```
~~~

Description: df [32 × 4]

X <chr>	mpg <dbl>	cyl <int>	wt <dbl>
Mazda RX4	21.0	6	2.620
Mazda RX4 Wag	21.0	6	2.875
Datsun 710	22.8	4	2.320
Hornet 4 Drive	21.4	6	3.215
Hornet Sportabout	18.7	8	3.440
Valiant	18.1	6	3.460
Duster 360	14.3	8	3.570
Merc 240D	24.4	4	3.190
Merc 230	22.8	4	3.150
Merc 280	19.2	6	3.440

1-10 of 32 rows

Summarizing



~~~
```{r}
cardata %>% select(X, mpg, cyl, wt) %>% summarise(CountCars = n(),AverageMpg =
mean(mpg), MaxMpg = max(mpg), MinMpg = min(mpg), SDMpg = sd(mpg), RangeMPG = MaxMpg -
MinMpg)
cardata.summarise<-cardata %>% select(X, mpg, cyl, wt) %>% summarise(CountCars =
n(),AverageMpg = mean(mpg), MaxMpg = max(mpg), MinMpg = min(mpg), SDMpg = sd(mpg),
RangeMPG = MaxMpg - MinMpg)
cardata.summarise
```
~~~

data.frame
1 x 6

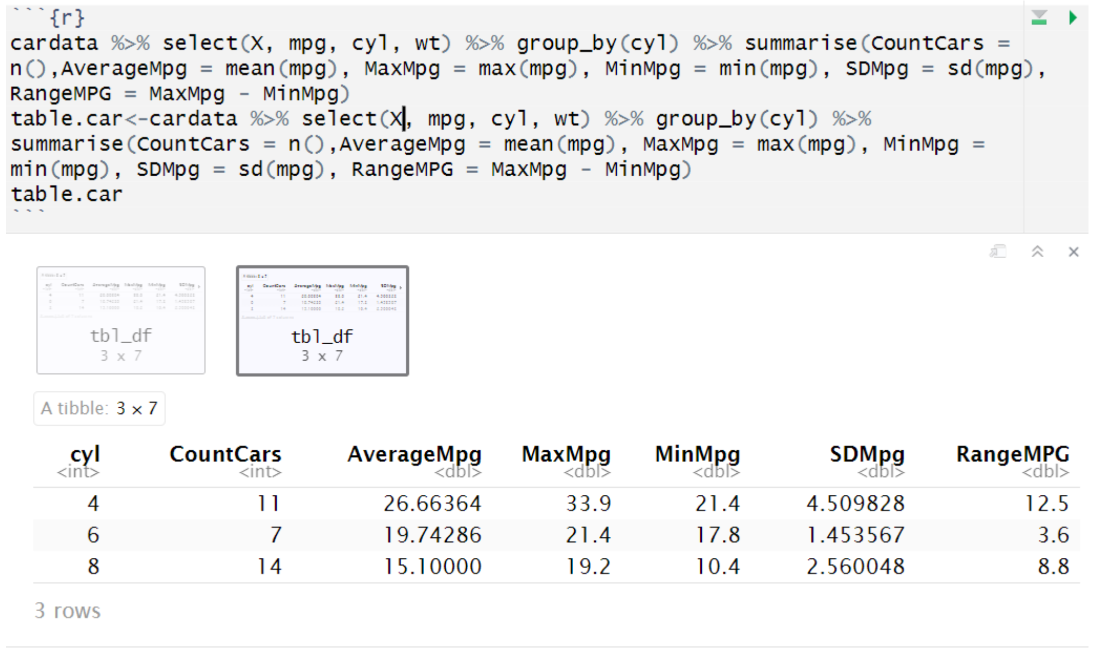
data.frame
1 x 6

Description: df [1 × 6]

CountCars <int>	AverageMpg <dbl>	MaxMpg <dbl>	MinMpg <dbl>	SDMpg <dbl>	RangeMPG <dbl>
32	20.09062	33.9	10.4	6.026948	23.5

- I can use summarise in order to do basic statistical commands.
- It can create counts, means, max, min, standard deviation, and call upon different functions.

Grouping



~~~
```{r}
cardata %>% select(X, mpg, cyl, wt) %>% group_by(cyl) %>% summarise(CountCars =
n(),AverageMpg = mean(mpg), MaxMpg = max(mpg), MinMpg = min(mpg), SDMpg = sd(mpg),
RangeMPG = MaxMpg - MinMpg)
table.car<-cardata %>% select(X, mpg, cyl, wt) %>% group_by(cyl) %>%
summarise(CountCars = n(),AverageMpg = mean(mpg), MaxMpg = max(mpg), MinMpg =
min(mpg), SDMpg = sd(mpg), RangeMPG = MaxMpg - MinMpg)
table.car
```
~~~

tbl_df
3 x 7

tbl_df
3 x 7

A tibble: 3 x 7

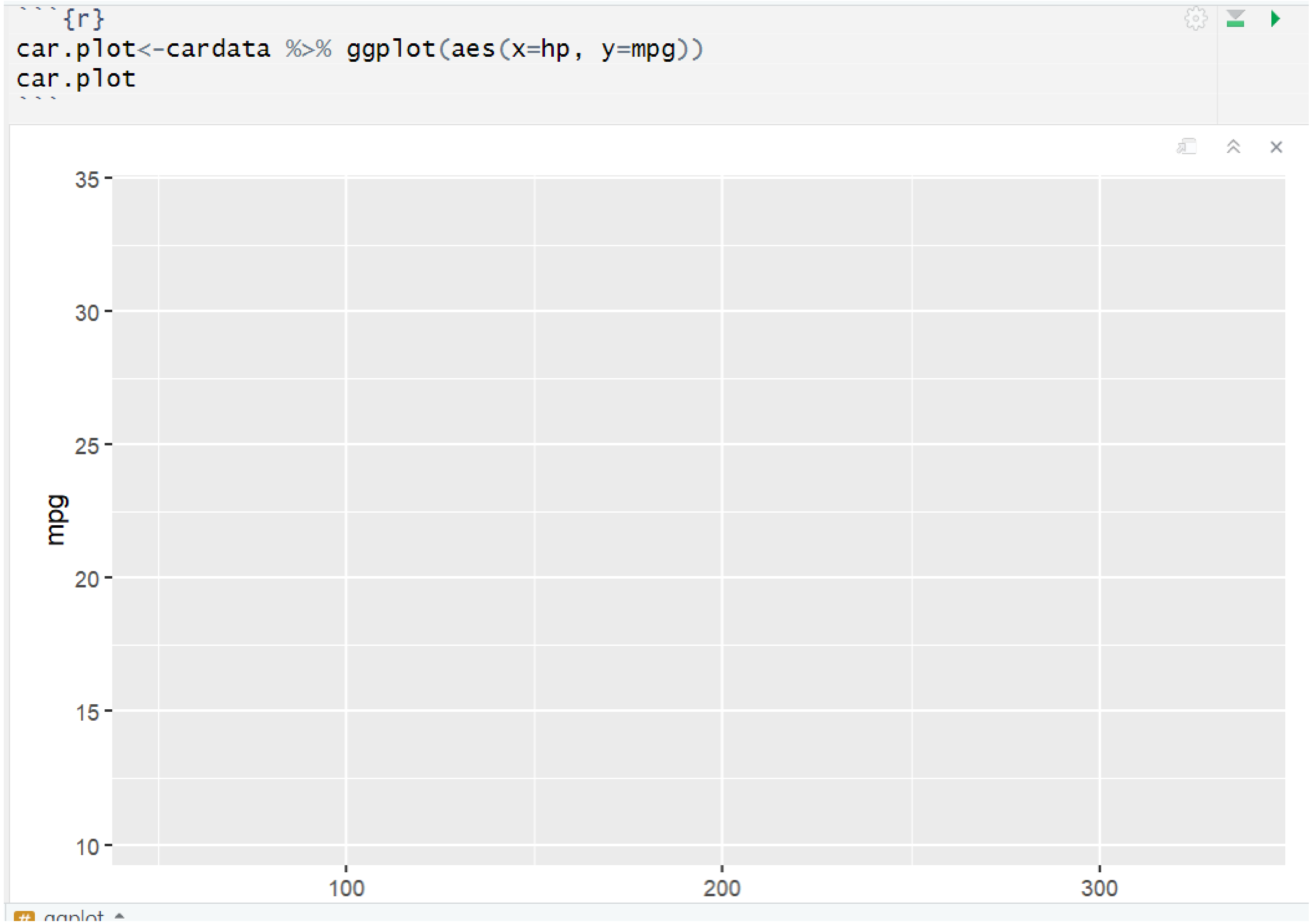
cyl <int>	CountCars <int>	AverageMpg <dbl>	MaxMpg <dbl>	MinMpg <dbl>	SDMpg <dbl>	RangeMPG <dbl>
4	11	26.66364	33.9	21.4	4.509828	12.5
6	7	19.74286	21.4	17.8	1.453567	3.6
8	14	15.10000	19.2	10.4	2.560048	8.8

3 rows

- The real power comes when you start using group_by in order to create wider tables.
- Lets group the cars based on cyl

Graphing

- The package ggplot2 is a massive tool that has a wide variety of information in it.
- You can learn to plot many different things in the program
  - Simple x by y plotting
  - Adding extra dimensions through color and fill
  - Create a wide variety of types of plots like line, smooth, histograms, and more!



```{r}
car.plot<-cardata %>% ggplot(aes(x=hp, y=mpg))
car.plot
```

ggplot

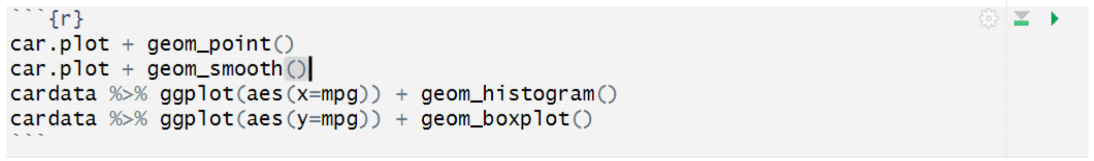
Lets start with the pipe and into ggplot!

Nothing shows up because we didn't tell it to do anything!

Types of Plots

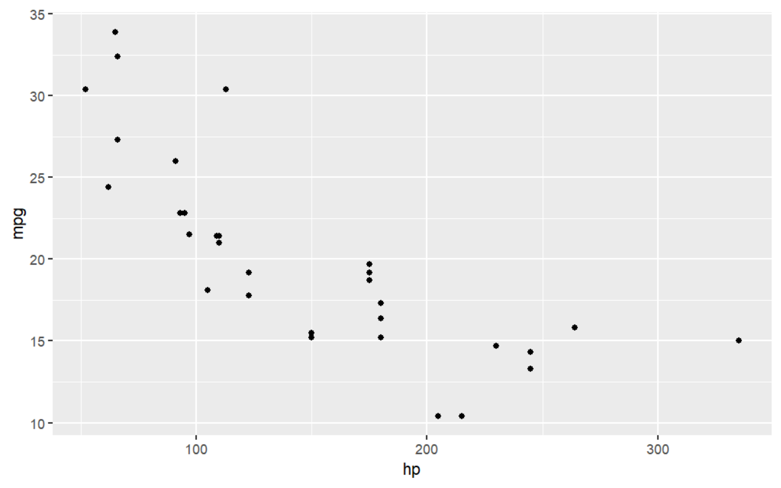
We have a few different types of basic plots

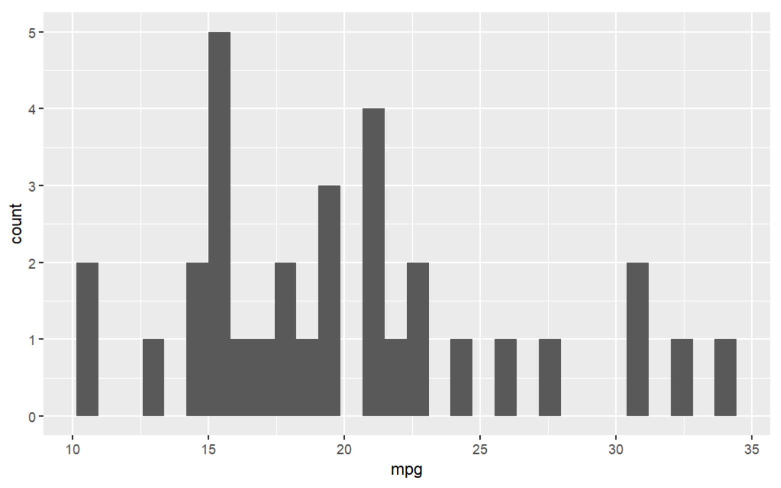
- geom_point (or geom_jitter) adds a scatterplot
- geom_smooth adds a smooth line
- geom_histogram adds a histogram
- geom_boxplot adds a boxplot

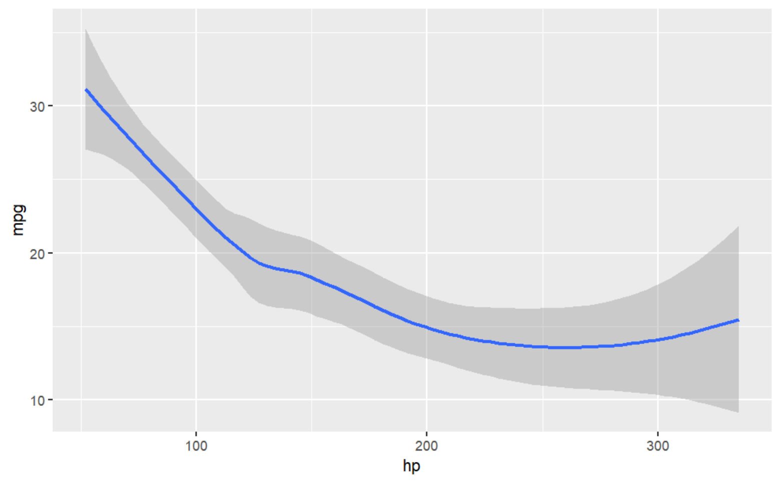


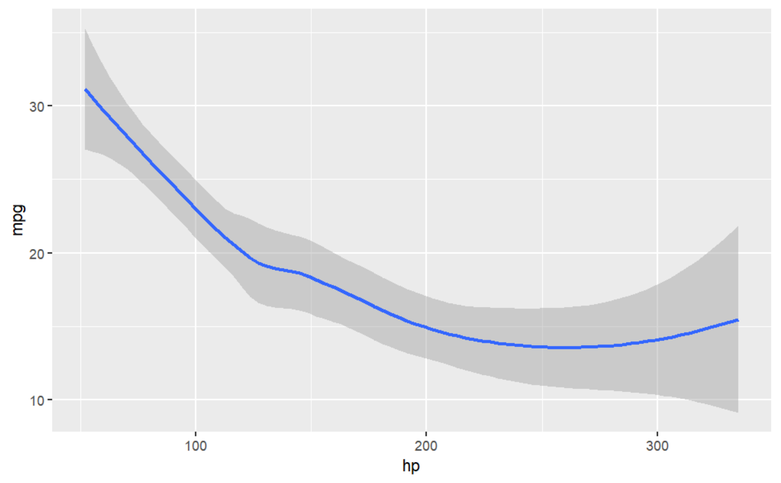
~~~
```{r}
car.plot + geom_point()
car.plot + geom_smooth()
cardata %>% ggplot(aes(x=mpg)) + geom_histogram()
cardata %>% ggplot(aes(y=mpg)) + geom_boxplot()
```
~~~

Types of Plots





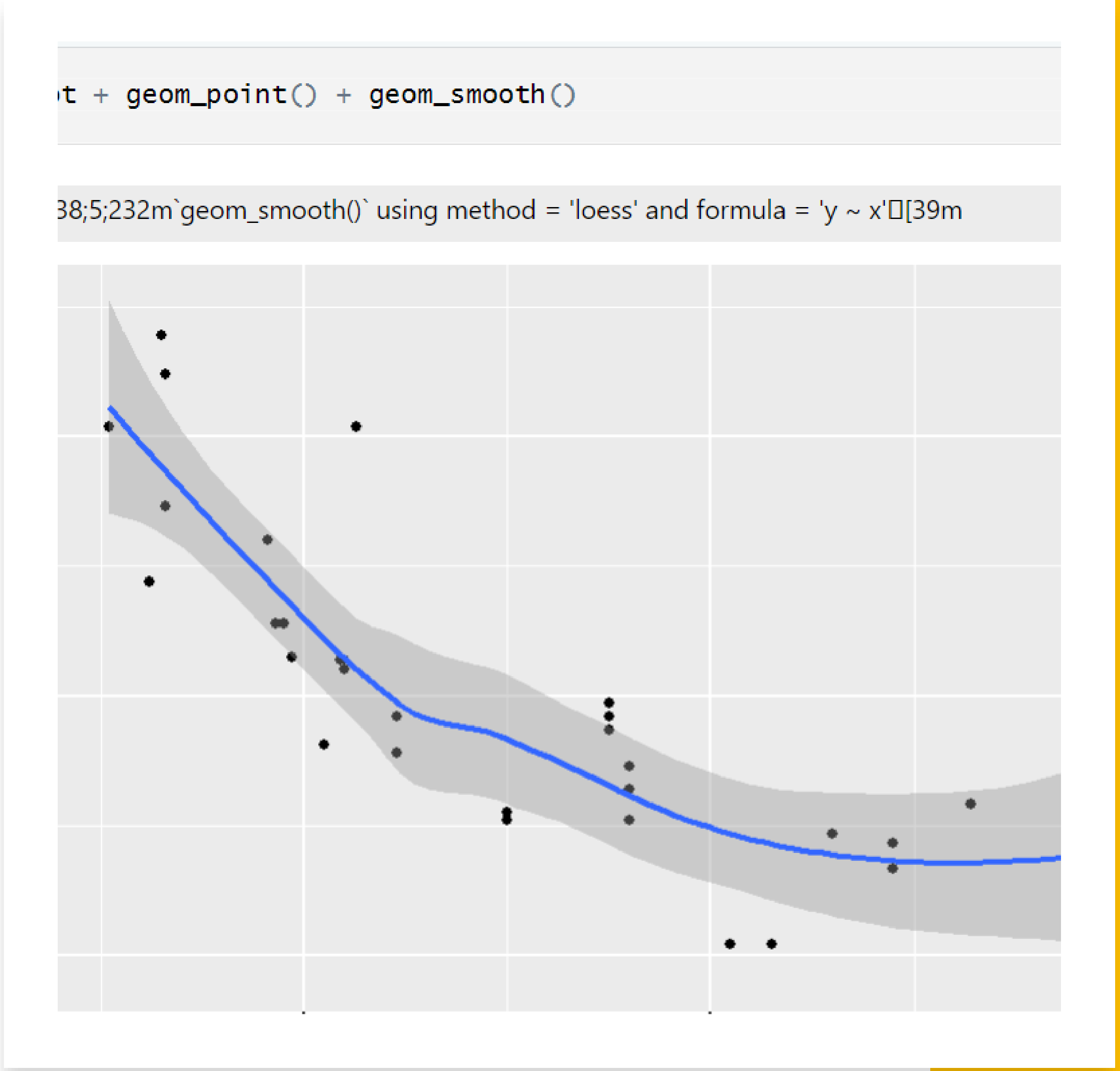




Multiple elements in the same graph

We can do multiple elements in the same graph as well.

Just use a + and add more

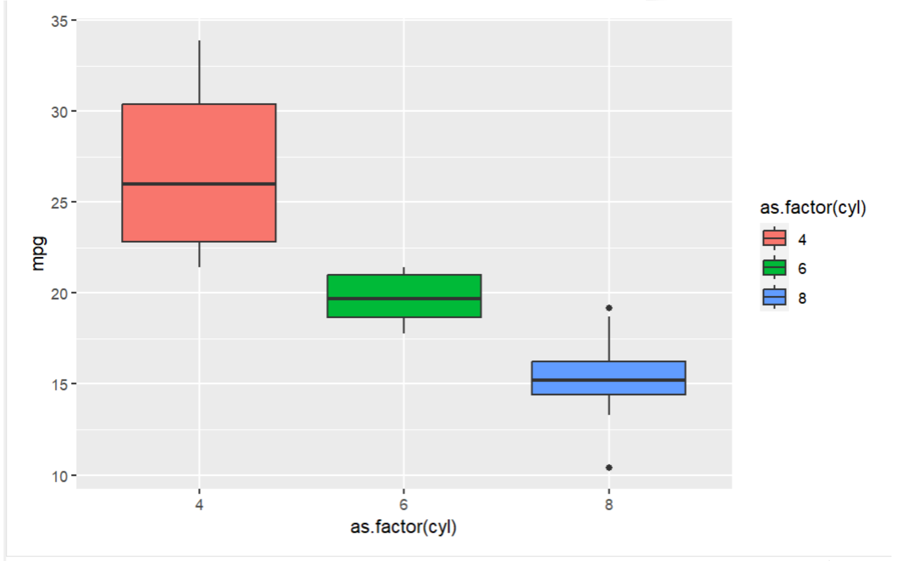


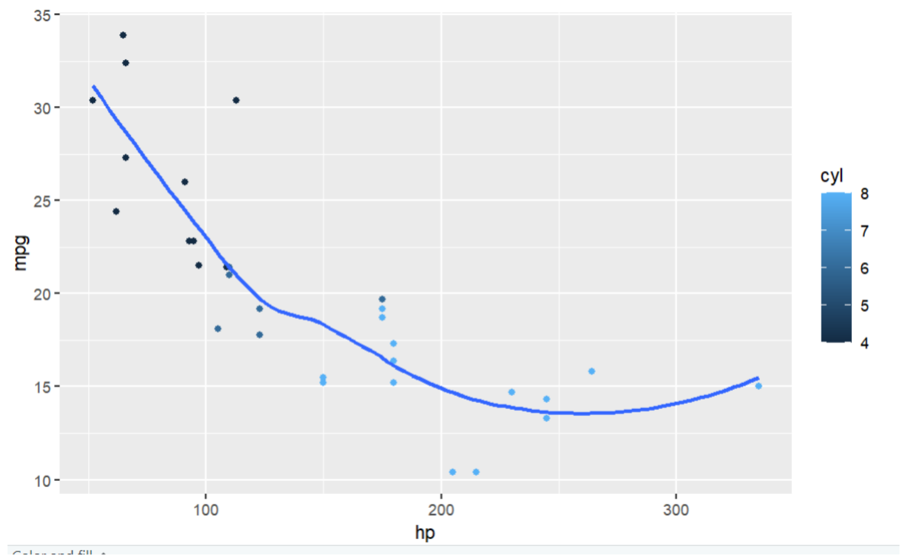
```
t + geom_point() + geom_smooth()
```

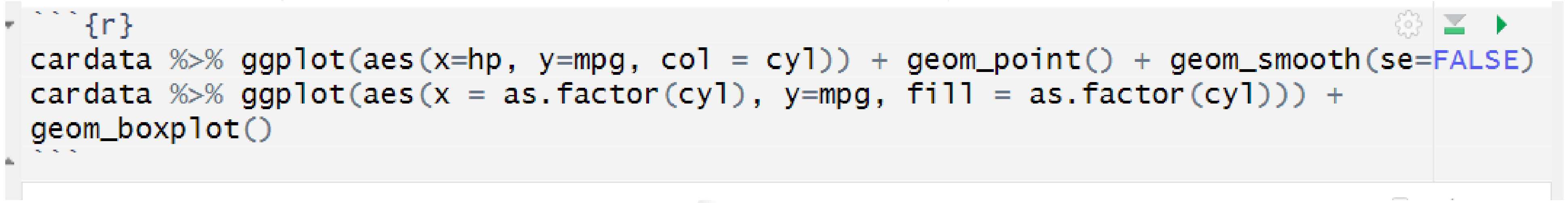
```
38;5;232m`geom_smooth()` using method = 'loess' and formula = 'y ~ x'[39m
```

Color and fill

We can use col and fill in aes to add a third dimension



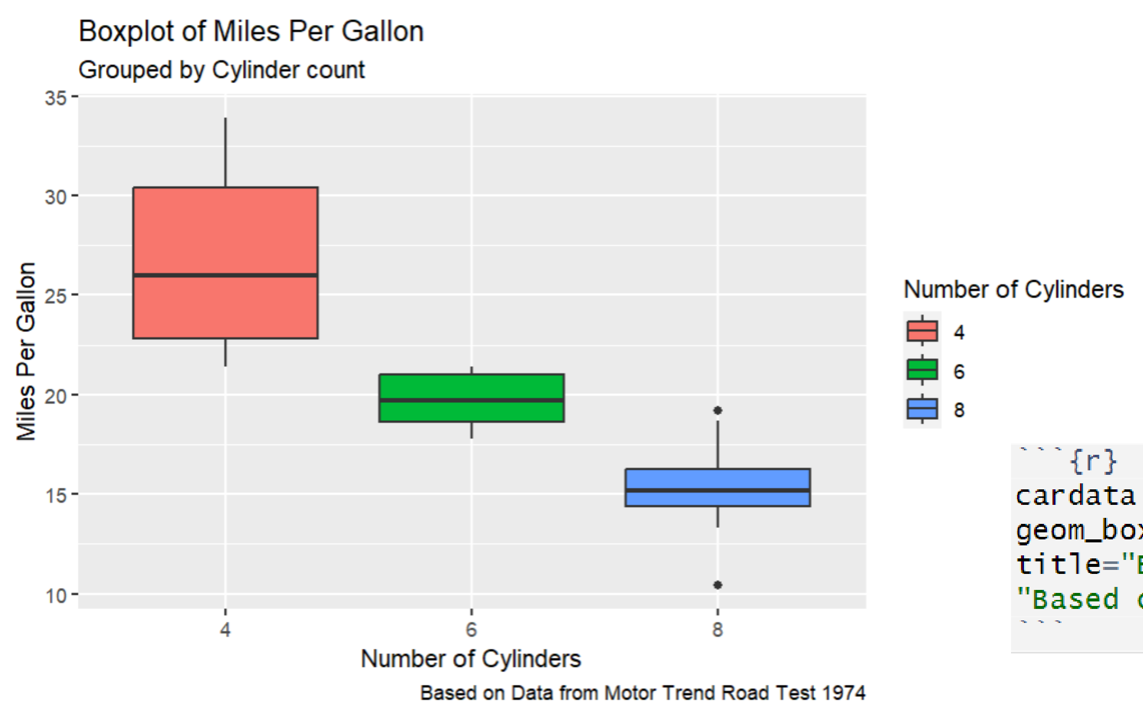


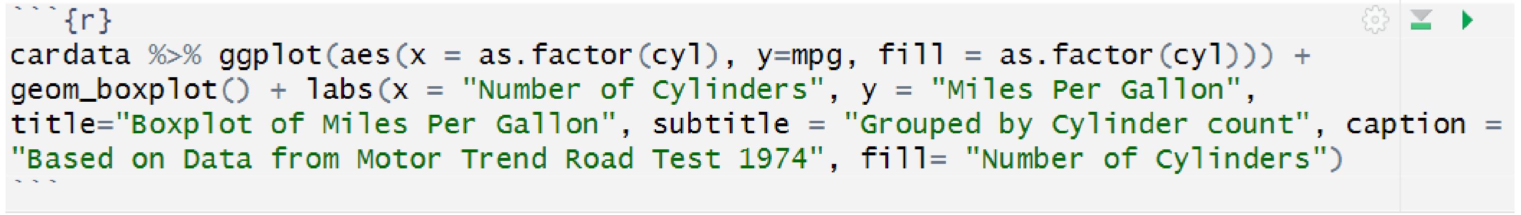


~~~
```{r}
cardata %>% ggplot(aes(x=hp, y=mpg, col = cyl)) + geom_point() + geom_smooth(se=FALSE)
cardata %>% ggplot(aes(x = as.factor(cyl), y=mpg, fill = as.factor(cyl))) +
geom_boxplot()
```
~~~

Making them look better

We can label the title, subtitle, x, y and legend using labs



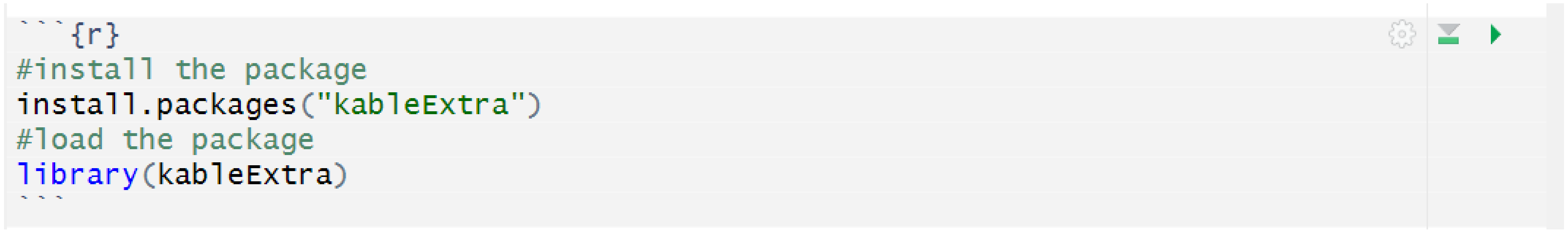


~~~
```{r}
cardata %>% ggplot(aes(x = as.factor(cyl), y=mpg, fill = as.factor(cyl))) +
geom_boxplot() + labs(x = "Number of Cylinders", y = "Miles Per Gallon",
title="Boxplot of Miles Per Gallon", subtitle = "Grouped by Cylinder count", caption =
"Based on Data from Motor Trend Road Test 1974", fill= "Number of Cylinders")
```
~~~

Making Tables with kableExtra

The Packages

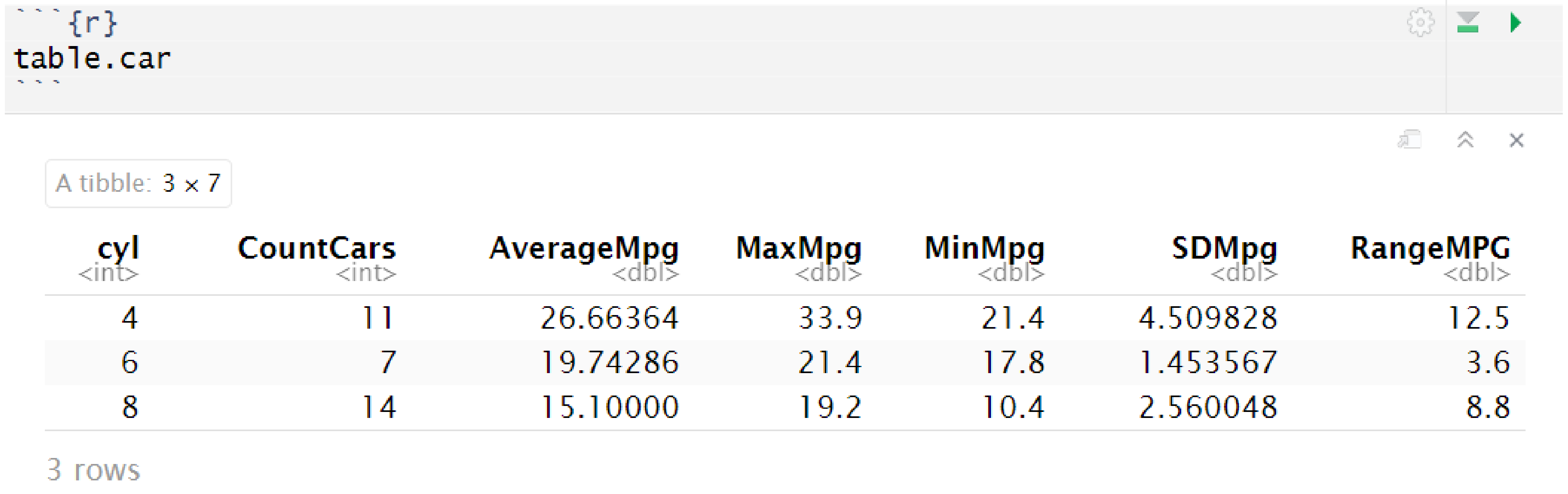
- The kableExtra package is one that lets you create publication ready tables for use in pdf and HTML documents.
- We can use the tables we created in tidyverse in order to make tables.
- The hard part of this is that you can only see them when they are rendered (we will revisit this in a little bit)



~~~
```{r}
#install the package
install.packages("kableExtra")
#load the package
library(kableExtra)
```
~~~

The table we will use

We made the table of MPG based on cylinders before. Lets use that now.



```{r}
table.car
```

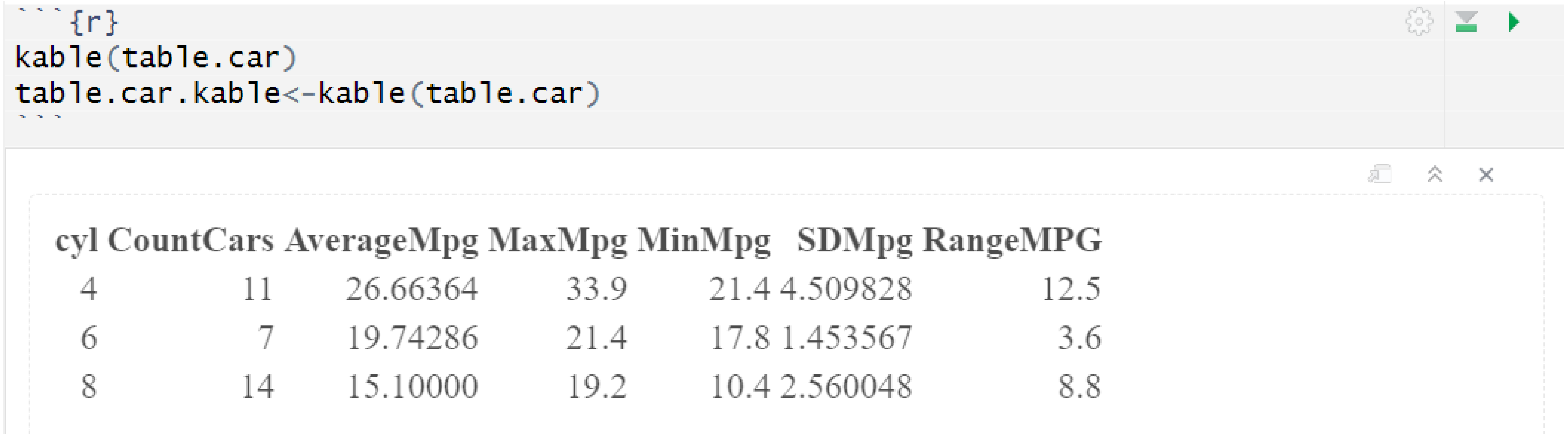
A tibble: 3 x 7

cyl <int>	CountCars <int>	AverageMpg <dbl>	MaxMpg <dbl>	MinMpg <dbl>	SDMpg <dbl>	RangeMPG <dbl>
4	11	26.66364	33.9	21.4	4.509828	12.5
6	7	19.74286	21.4	17.8	1.453567	3.6
8	14	15.10000	19.2	10.4	2.560048	8.8

3 rows

kable function

A basic table is made using kable. We can see the basic output.

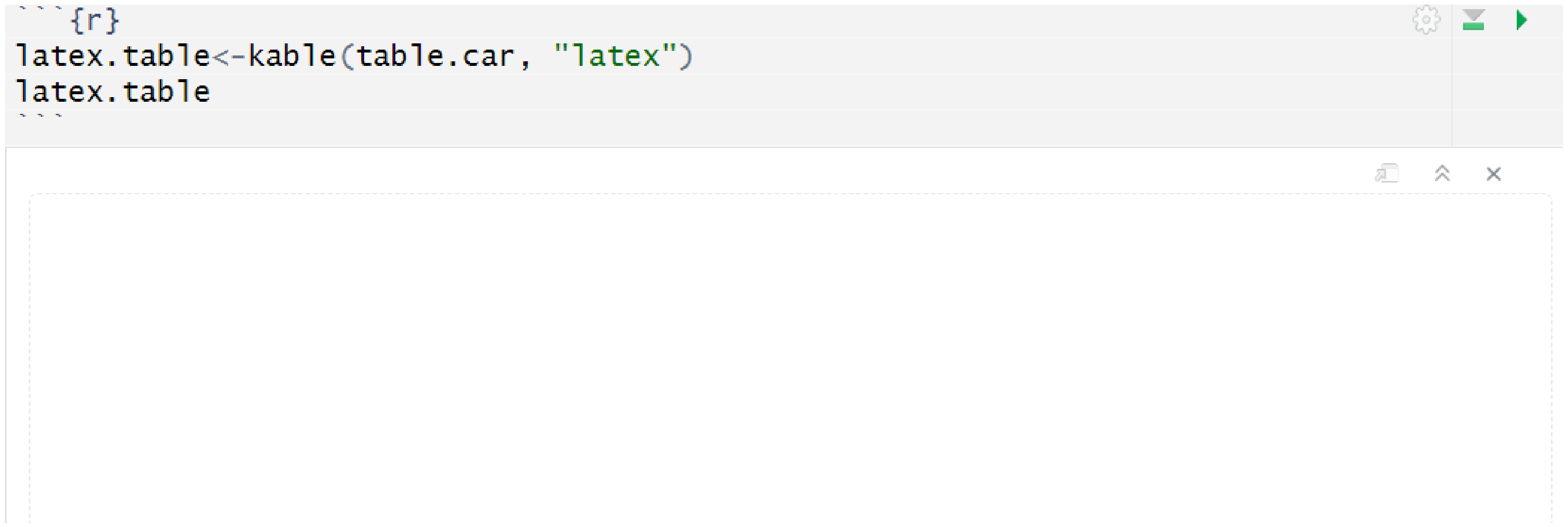


~~~
```{r}
kable(table.car)
table.car.kable<-kable(table.car)
```
~~~

```
cyl CountCars AverageMpg MaxMpg MinMpg    SDMpg RangeMPG
  4        11   26.66364   33.9   21.4 4.509828     12.5
  6         7   19.74286   21.4   17.8 1.453567      3.6
  8        14   15.10000   19.2   10.4 2.560048      8.8
```

kable function

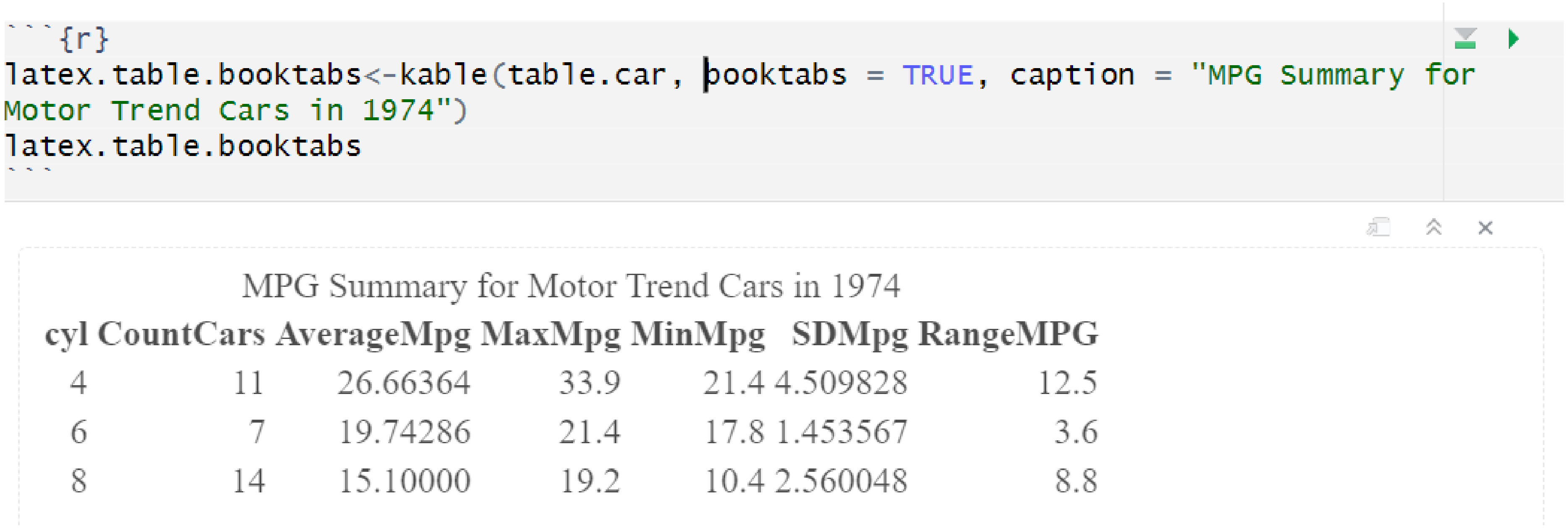
- If we want to make it into a pdf, we use latex, if we are making an html, we use html.
- Note, once you use LaTex you will not be able to see it in the output only at the end!!



~~~
```{r}
latex.table<-kable(table.car, "latex")
latex.table
```
~~~

Making it look nice

- I use the booktabs = TRUE for a specific style of table
- Caption can be used in order to create a title to the table



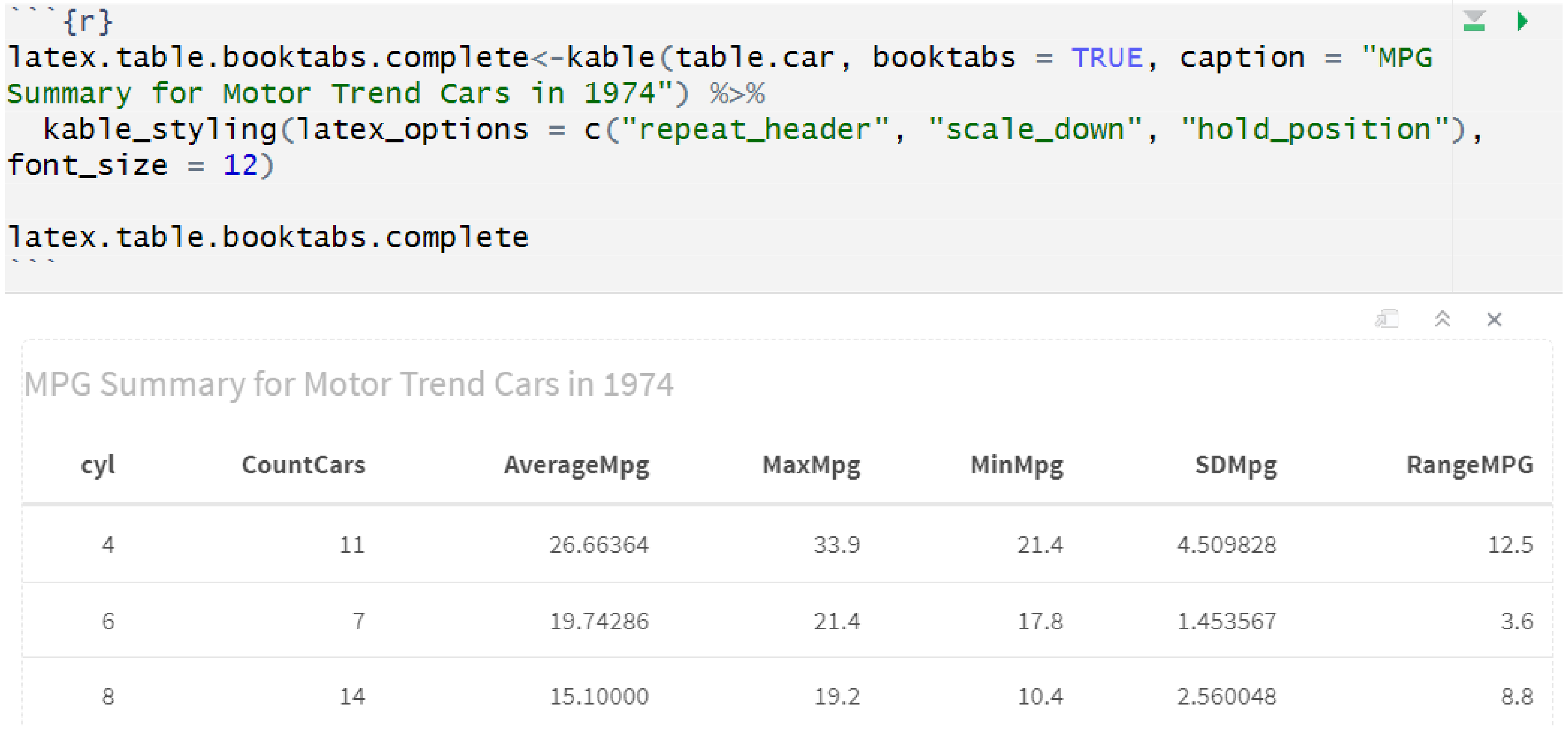
```{r}
latex.table.booktabs<-kable(table.car, booktabs = TRUE, caption = "MPG Summary for Motor Trend Cars in 1974")
latex.table.booktabs
```

MPG Summary for Motor Trend Cars in 1974

cyl	CountCars	AverageMpg	MaxMpg	MinMpg	SDMpg	RangeMPG
4	11	26.66364	33.9	21.4	4.509828	12.5
6	7	19.74286	21.4	17.8	1.453567	3.6
8	14	15.10000	19.2	10.4	2.560048	8.8

Many Different Options!

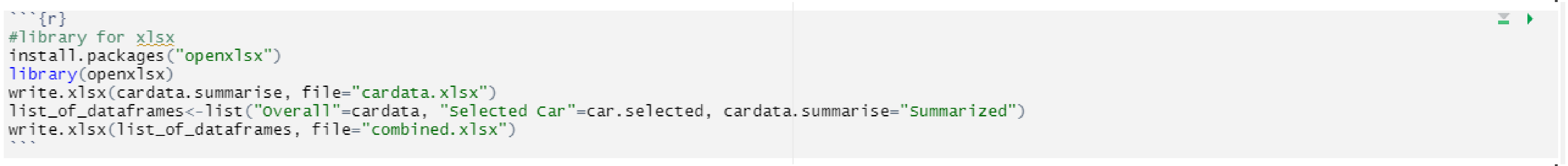
- I have played around with this code a lot to get the desired style.
- Here is an example of my code:
- You can hold position, scale text, repeat header if it scrolls over pages, and choose font size.
- These are all options and are not necessary



~~~
```{r}
latex.table.booktabs.complete<-kable(table.car, booktabs = TRUE, caption = "MPG
Summary for Motor Trend Cars in 1974") %>%
  kable_styling(latex_options = c("repeat_header", "scale_down", "hold_position"),
font_size = 12)

latex.table.booktabs.complete
```
~~~

MPG Summary for Motor Trend Cars in 1974

cyl	CountCars	AverageMpg	MaxMpg	MinMpg	SDMpg	RangeMPG
4	11	26.66364	33.9	21.4	4.509828	12.5
6	7	19.74286	21.4	17.8	1.453567	3.6
8	14	15.10000	19.2	10.4	2.560048	8.8



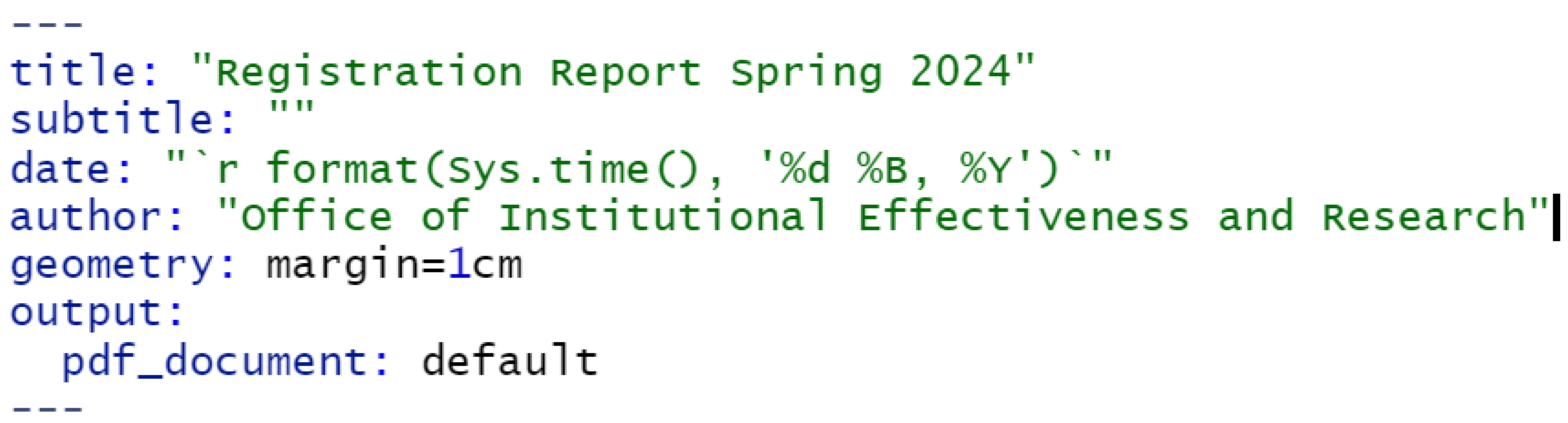
~~~
```{r}
#library for xlsx
install.packages("openxlsx")
library(openxlsx)
write.xlsx(cardata.summarise, file="cardata.xlsx")
list_of_dataframes<-list("Overall"=cardata, "Selected Car"=car.selected, cardata.summarise="Summarized")
write.xlsx(list_of_dataframes, file="combined.xlsx")
```
~~~

Exporting Data to CSV

- Even if you create a nice report, it's always nice to keep a copy of your data in a csv or excel file.
- R has a few packages that can help with that!
- Writing to xlsx file
- You can either write individual data frames or a list of data frames to multiple tabs in excel.

YAML Headers

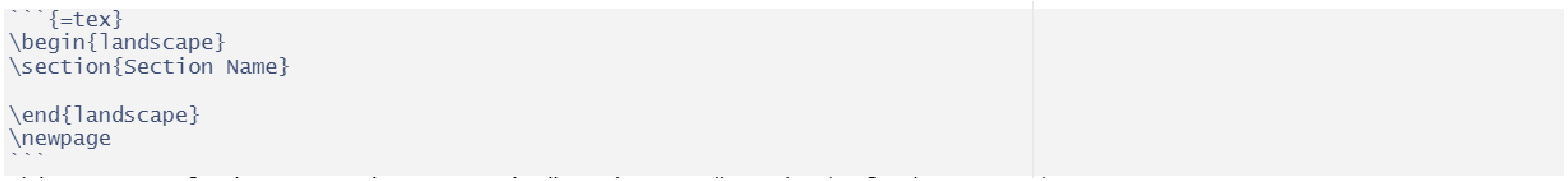
- YAML stands for YAML ain't Markdown Language
- Yes, it is recursive. No I didn't name it.
- It has basic information on how to render your document.
- You can add metadata, pictures, and options for the entire document in it.



```
---
title: "Registration Report Spring 2024"
subtitle: ""
date: "`r format(Sys.time(), '%d %B, %Y')`"
author: "Office of Institutional Effectiveness and Research"
geometry: margin=1cm
output:
  pdf_document: default
---
```

LaTex Commands and Knitting

- There are some basic commands in LaTex that I use for reports.
- The main one is newpage (\newpage)
- create a line break in the document you are knitting.
- \landscape is another code that you can use in order to change orientation of the document
- You use the following code for the sections. Note the 4 \`'s
- The last bit of LaTex I use is the section part
- This starts a landscape section, names it "Section Name", ends the landscape, and creates a new page.
- The code itself is in the middle.
- The knit button on top compiles everything into a document!



~~~
```{=tex}
\begin{landscape}
\section{Section Name}

\end{landscape}
\newpage
```
~~~

Questions and Quick Tips

- Word of caution before dealing with YAML or LaTex sections. Save a copy before you start editing
- Once you are done with everything you can go to the top and hit render.
- Your document will run all the code and you will get a report.
- It will have all the code that you have included in the report, and markup, and information or text you put in it.