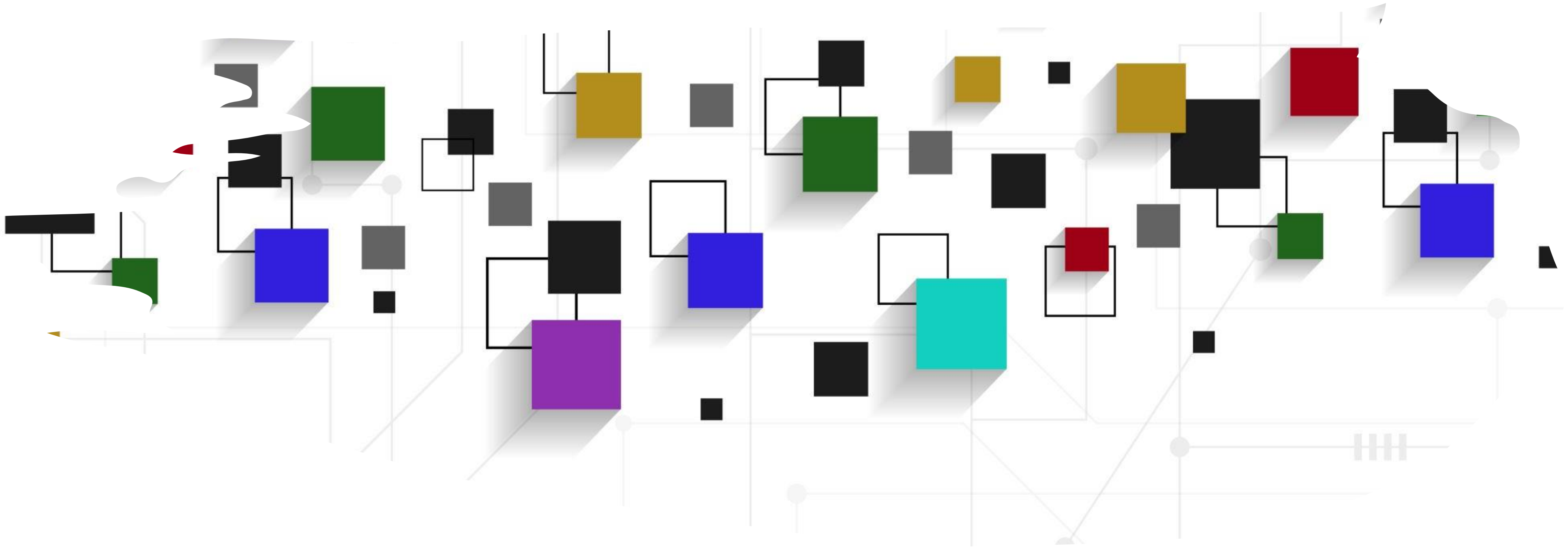
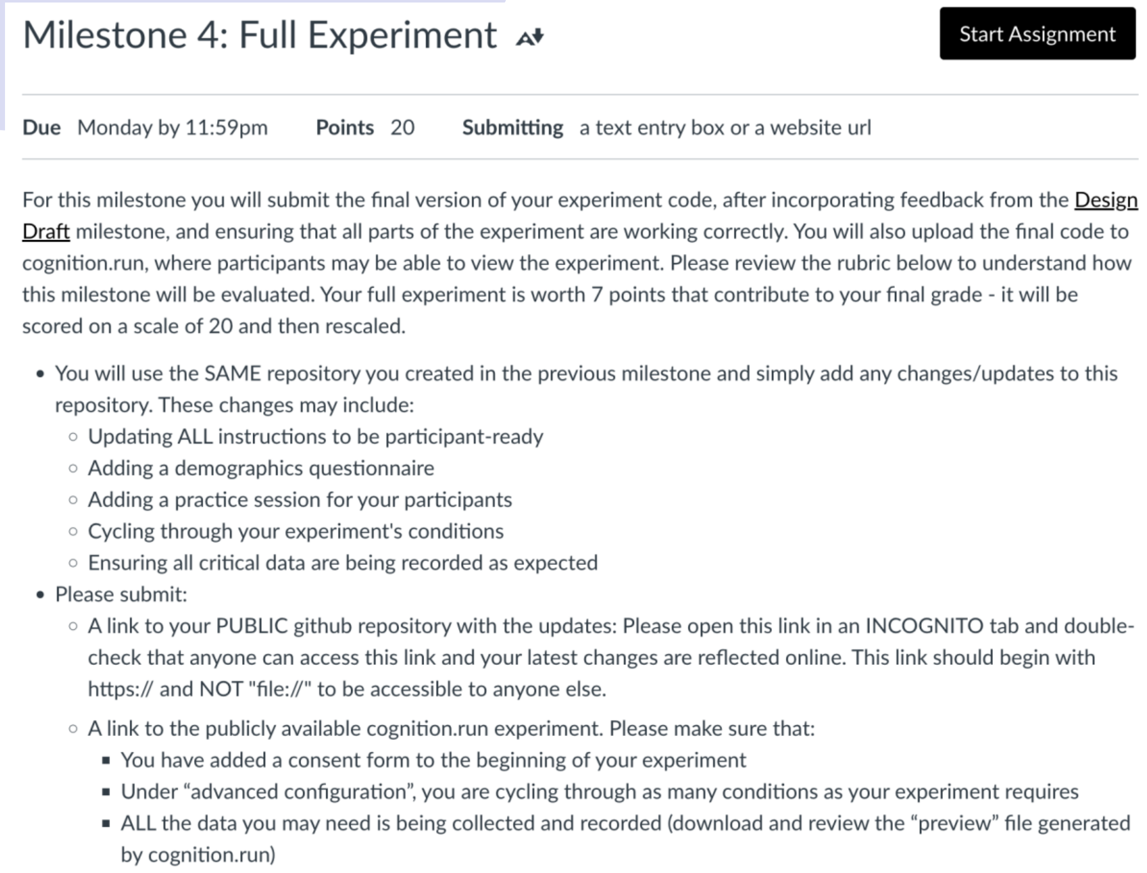


CogLab: Import and Plot Data

WEEK 7

recap

- what we covered:
    - reviewing data analysis plan
    - R 101
- your to-do's were:
    - *prep*: work on project milestone #5

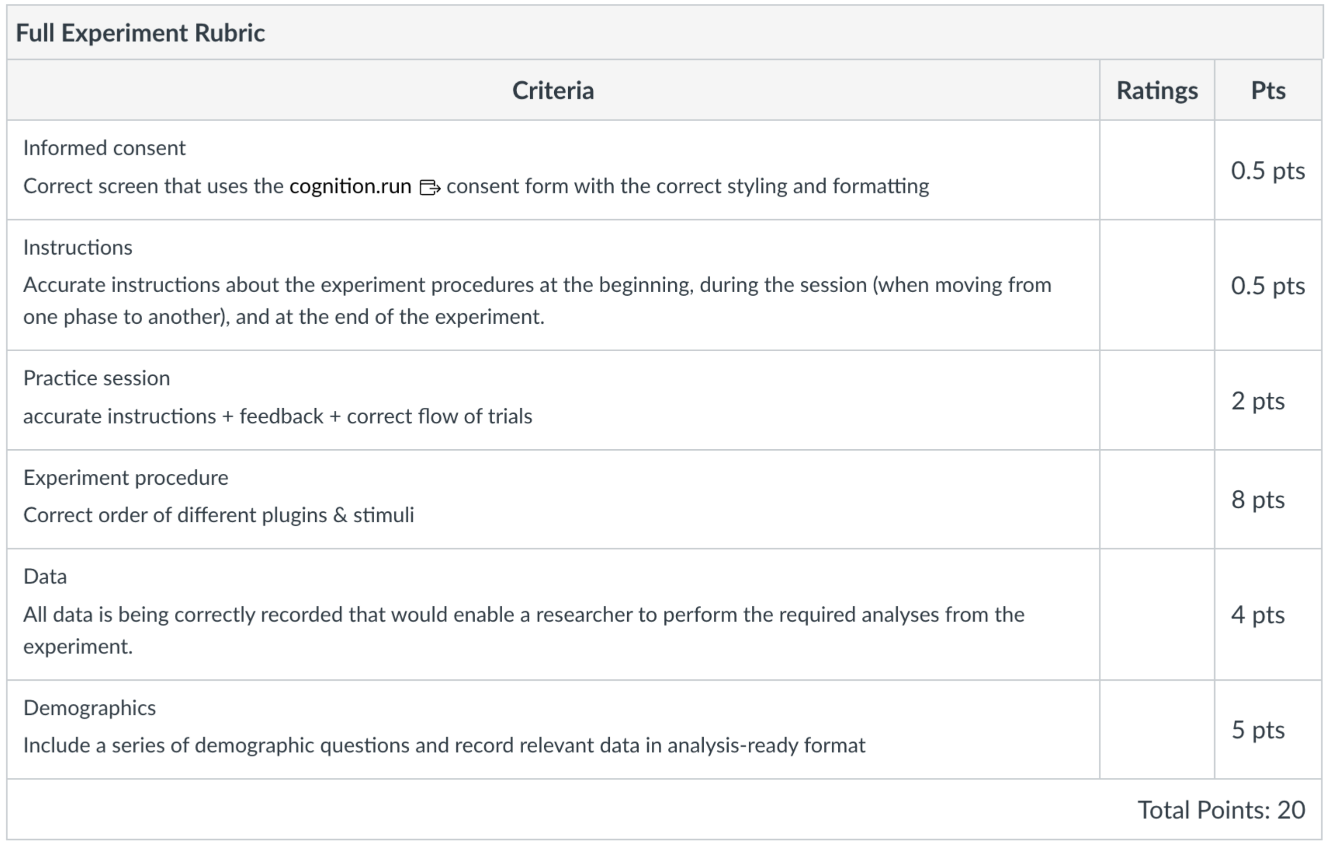


Milestone 4: Full Experiment

Due Monday by 11:59pm **Points** 20 **Submitting** a text entry box or a website url

For this milestone you will submit the final version of your experiment code, after incorporating feedback from the Design Draft milestone, and ensuring that all parts of the experiment are working correctly. You will also upload the final code to cognition.run, where participants may be able to view the experiment. Please review the rubric below to understand how this milestone will be evaluated. Your full experiment is worth 7 points that contribute to your final grade - it will be scored on a scale of 20 and then rescaled.

- You will use the SAME repository you created in the previous milestone and simply add any changes/updates to this repository. These changes may include:
  - Updating ALL instructions to be participant-ready
  - Adding a demographics questionnaire
  - Adding a practice session for your participants
  - Cycling through your experiment's conditions
  - Ensuring all critical data are being recorded as expected
- Please submit:
  - A link to your PUBLIC github repository with the updates: Please open this link in an INCOGNITO tab and double-check that anyone can access this link and your latest changes are reflected online. This link should begin with https:// and NOT "file://" to be accessible to anyone else.
  - A link to the publicly available cognition.run experiment. Please make sure that:
    - You have added a consent form to the beginning of your experiment
    - Under "advanced configuration", you are cycling through as many conditions as your experiment requires
    - ALL the data you may need is being collected and recorded (download and review the "preview" file generated by cognition.run)



Full Experiment Rubric

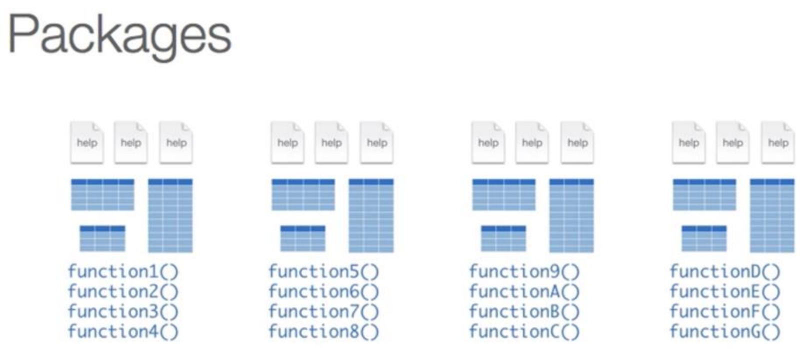
Criteria	Ratings	Pts
Informed consent Correct screen that uses the cognition.run consent form with the correct styling and formatting		0.5 pts
Instructions Accurate instructions about the experiment procedures at the beginning, during the session (when moving from one phase to another), and at the end of the experiment.		0.5 pts
Practice session accurate instructions + feedback + correct flow of trials		2 pts
Experiment procedure Correct order of different plugins & stimuli		8 pts
Data All data is being correctly recorded that would enable a researcher to perform the required analyses from the experiment.		4 pts
Demographics Include a series of demographic questions and record relevant data in analysis-ready format		5 pts
		Total Points: 20

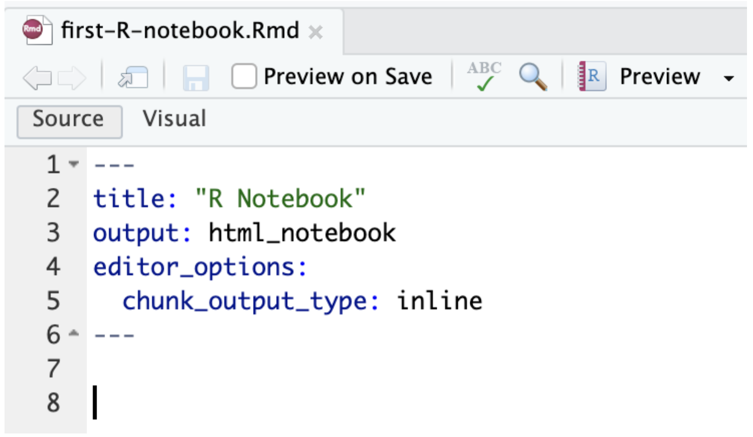
today’s agenda

- packages in R
- importing data
- graphing data

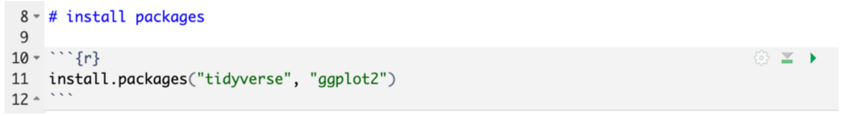
packages in R

- packages contain functions
- before we can use them, we need to install packages
  - installation only happens once on a computer
- inside your notebook, delete everything after the last ---
- create a new heading # install packages
- create a new code chunk using command + option + I
  - or use the green C icon
- install tidyverse
- run the chunk:
  - command + shift + enter OR green play button
- comment (#) the install line after installation is complete (we will not need to do this again)

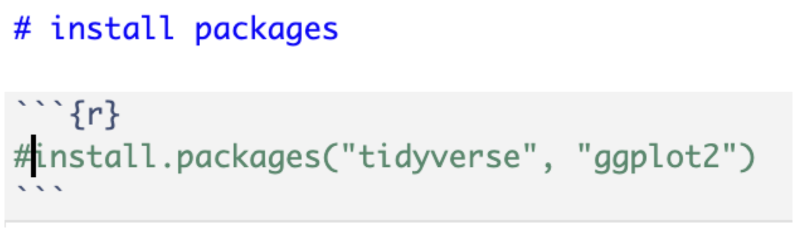




```
---
title: "R Notebook"
output: html_notebook
editor_options:
  chunk_output_type: inline
---

```



~~~
# install packages

```{r}
install.packages("tidyverse", "ggplot2")
```
~~~



~~~
# install packages

```{r}
#install.packages("tidyverse", "ggplot2")
```
~~~

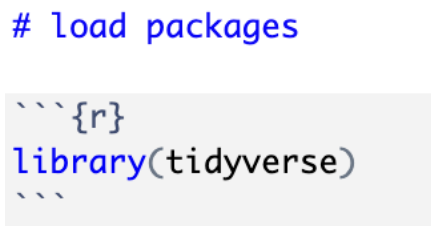
tidyverse

- a package of **packages**, which contain their own **functions**
- extremely flexible ways of manipulating and plotting data

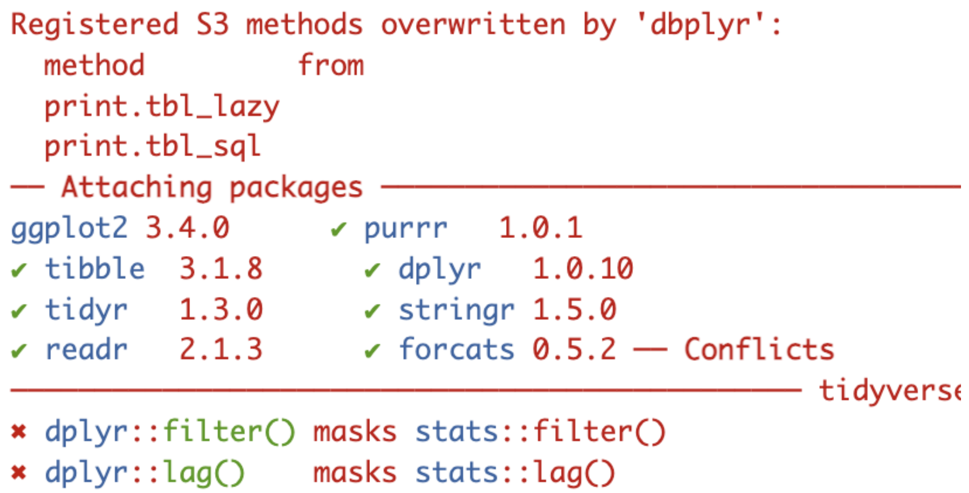
dplyr
readr
purrr
tidyverse
ggplot2
tidyr
TIBBLE

loading packages in R

- we must always load the packages when we want to run code that uses functions from those packages
- create a new heading
  # load packages
- create a new code chunk
- load tidyverse
- run the chunk
- save your notebook

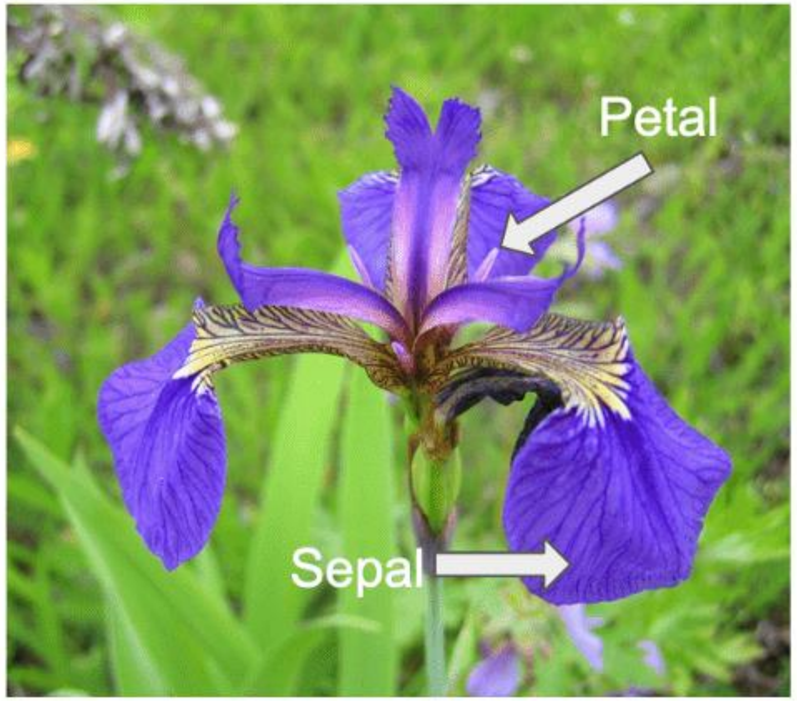


~~~
# load packages

```{r}
library(tidyverse)
```
~~~

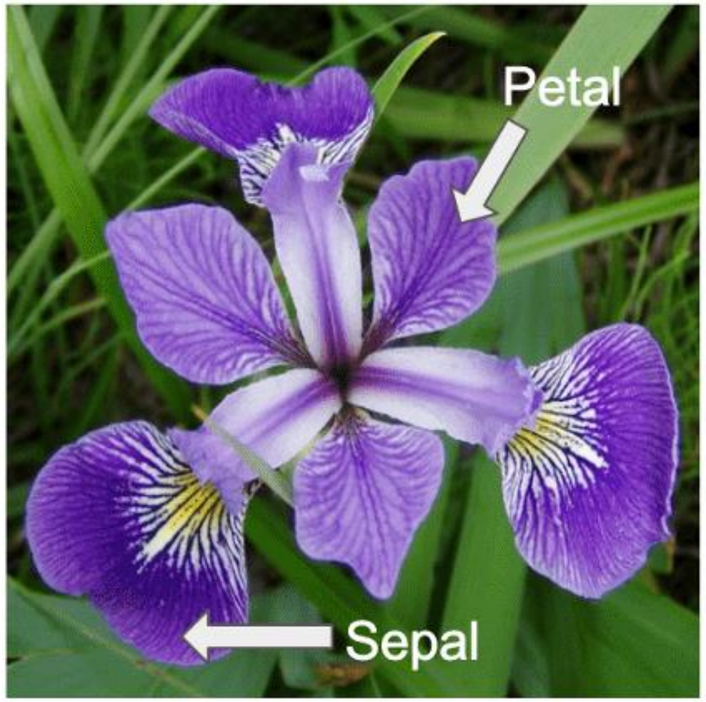


```
Registered S3 methods overwritten by 'dbplyr':
  method         from
  print.tbl_lazy
  print.tbl_sql
── Attaching packages ───────────────────────────────────
ggplot2 3.4.0      ✔ purrr   1.0.1
✔ tibble  3.1.8      ✔ dplyr   1.0.10
✔ tidyr   1.3.0      ✔ stringr 1.5.0
✔ readr   2.1.3      ✔ forcats 0.5.2 ── Conflicts
──────────────────────────────────────────────── tidyverse
✖ dplyr::filter() masks stats::filter()
✖ dplyr::lag()    masks stats::lag()
```

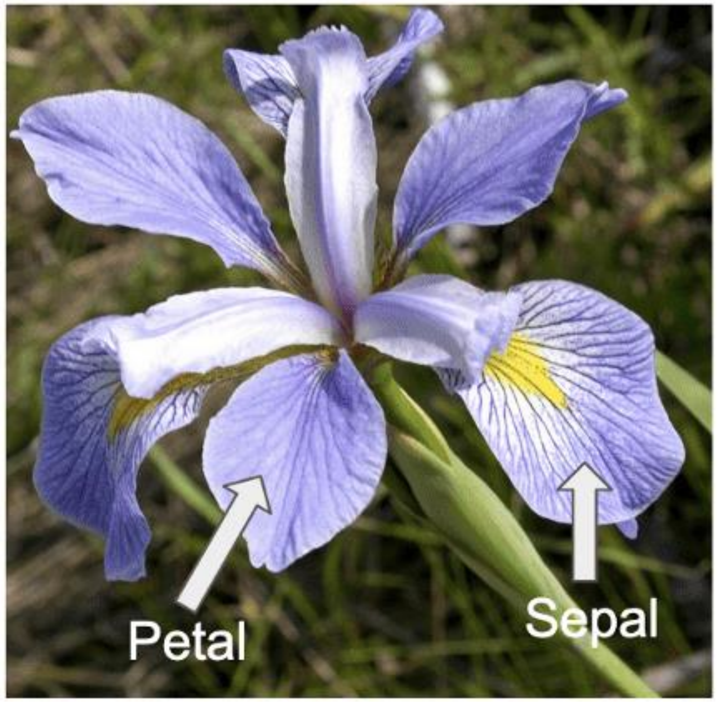
Iris setosa



Iris versicolor

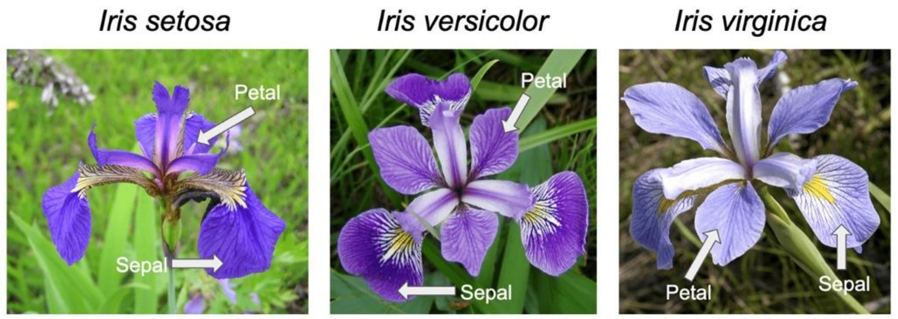


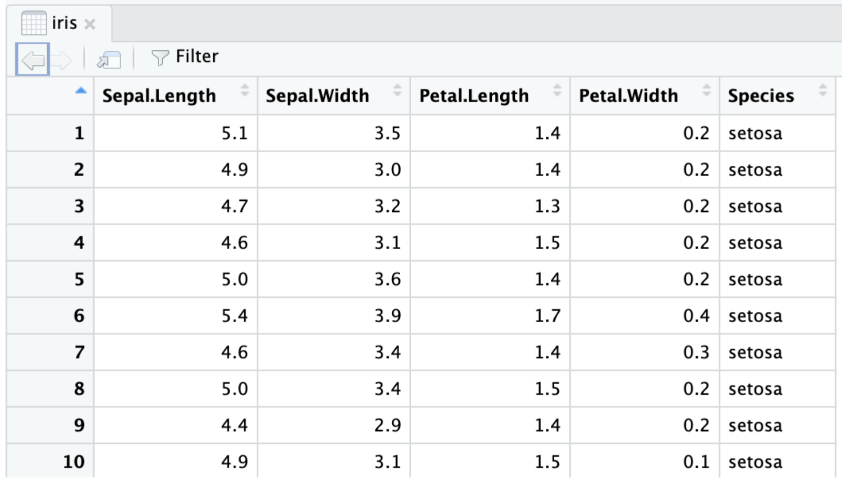
Iris virginica



examining a dataframe

- view the iris dataset
- how many rows and columns?
- what are the column names?
- research question: how are the species different from each other?
- first we plot, then we analyze





	Sepal.Length	Sepal.Width	Petal.Length	Petal.Width	Species
1	5.1	3.5	1.4	0.2	setosa
2	4.9	3.0	1.4	0.2	setosa
3	4.7	3.2	1.3	0.2	setosa
4	4.6	3.1	1.5	0.2	setosa
5	5.0	3.6	1.4	0.2	setosa
6	5.4	3.9	1.7	0.4	setosa
7	4.6	3.4	1.4	0.3	setosa
8	5.0	3.4	1.5	0.2	setosa
9	4.4	2.9	1.4	0.2	setosa
10	4.9	3.1	1.5	0.1	setosa

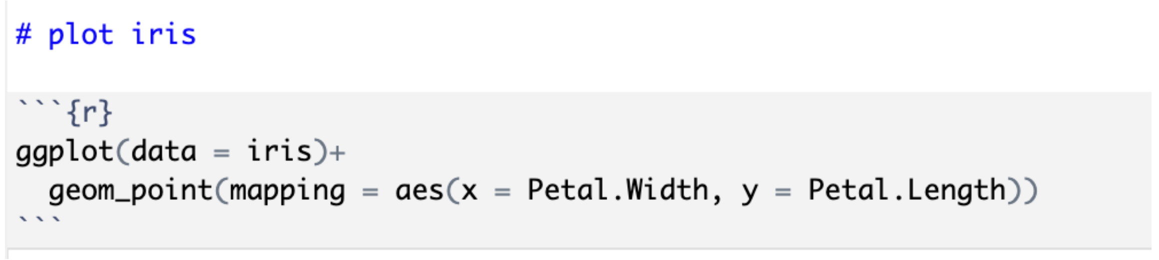
```
> nrow(iris)
[1] 150
> ncol(iris)
[1] 5
> colnames(iris)
[1] "Sepal.Length" "Sepal.Width"  "Petal.Length" "Petal.Width"  "Species"
```

the grammar of graphics

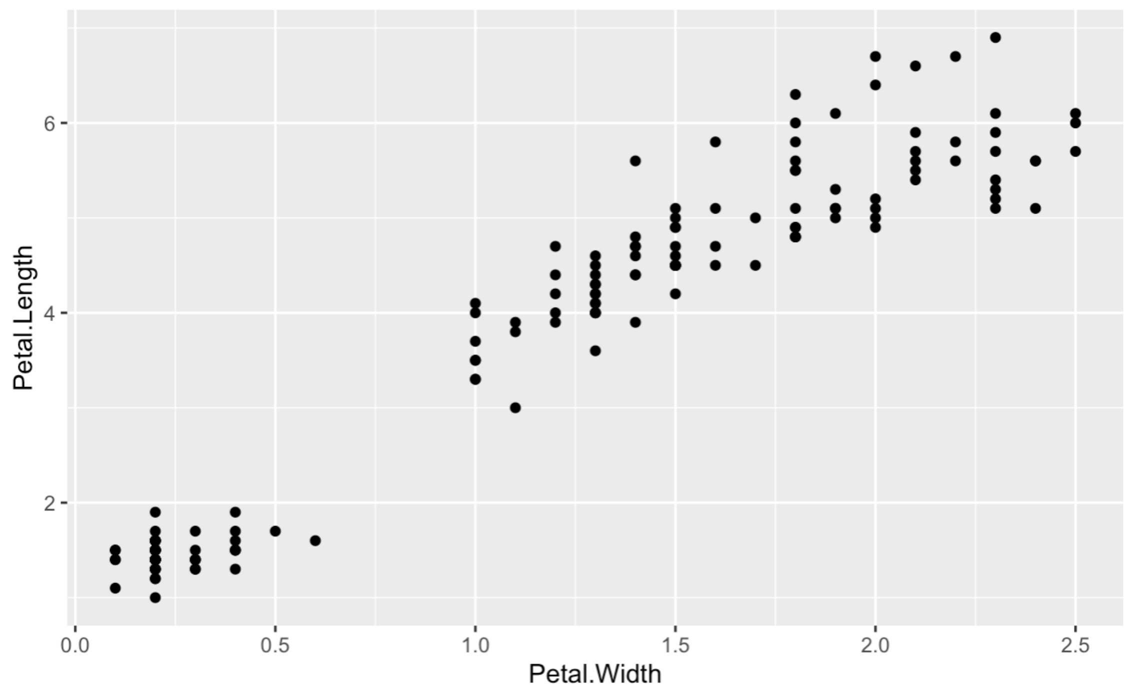
- converting data to a visual display
- ggplot() is a function from the ggplot2 package, which is included in the tidyverse
- three key steps to use ggplot():
  - select a dataset
  - pick a geometric object
  - specify aesthetics

ggplot basic structure

- plot the petal width (x) and length (y) from the iris dataset
- ggplot() specifies the dataset and creates an empty graph
- geom_point() adds a layer of points to this empty graph
- mapping specifies how to map the data to the points

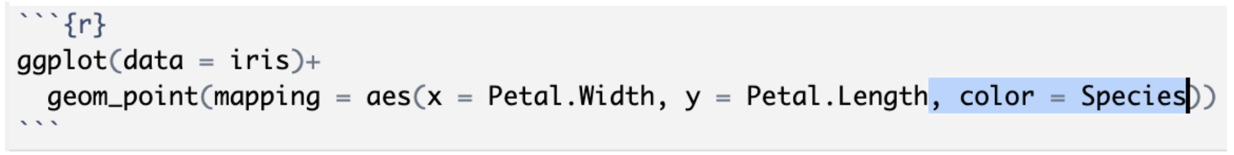


~~~
# plot iris

```{r}
ggplot(data = iris)+
  geom_point(mapping = aes(x = Petal.Width, y = Petal.Length))
```
~~~

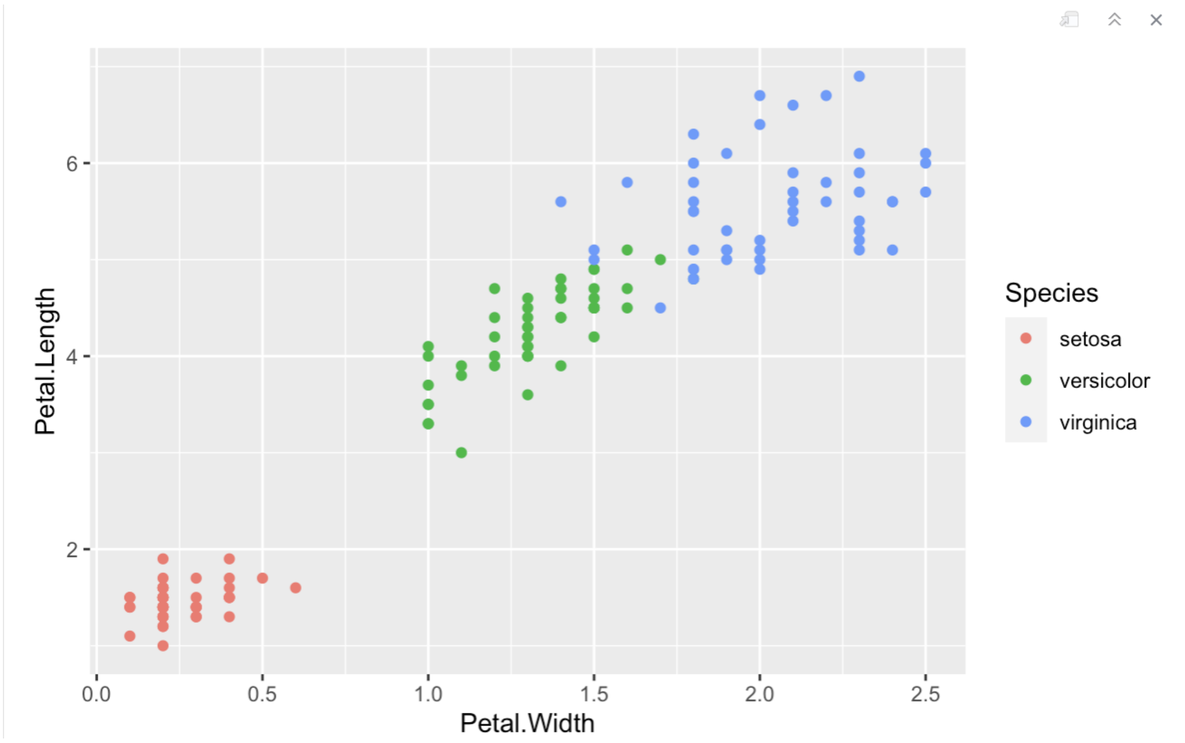


aesthetics

- aesthetics define visual properties of objects in the plot
- allow us to include a third variable in a 2-d plot using properties like size, shape, color, etc.
- map the color of the points to species

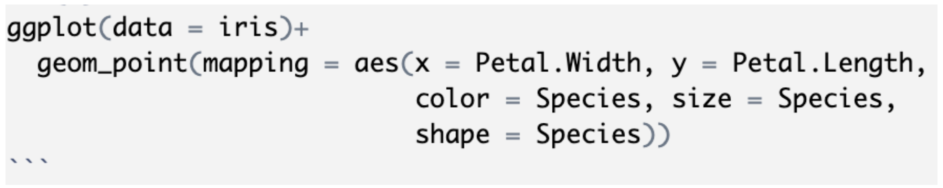


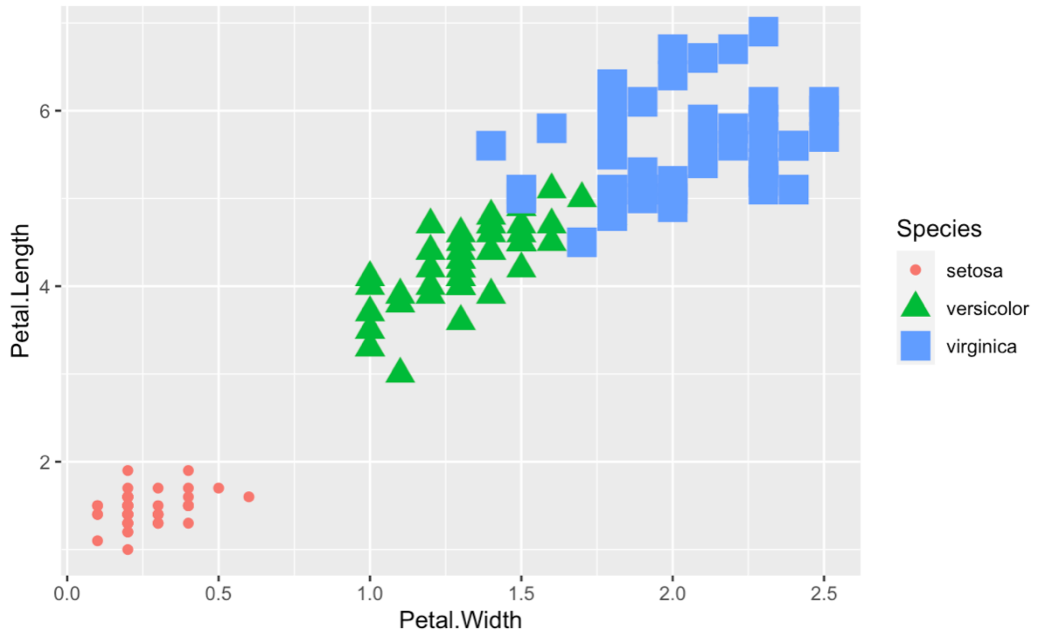
~~~
```{r}
ggplot(data = iris)+
  geom_point(mapping = aes(x = Petal.Width, y = Petal.Length, color = Species))
```
~~~



other aesthetics

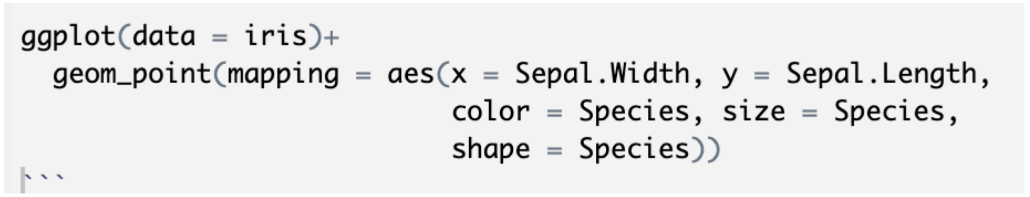
- map size and shape to the species as well
- is petal information helpful is distinguishing the species?

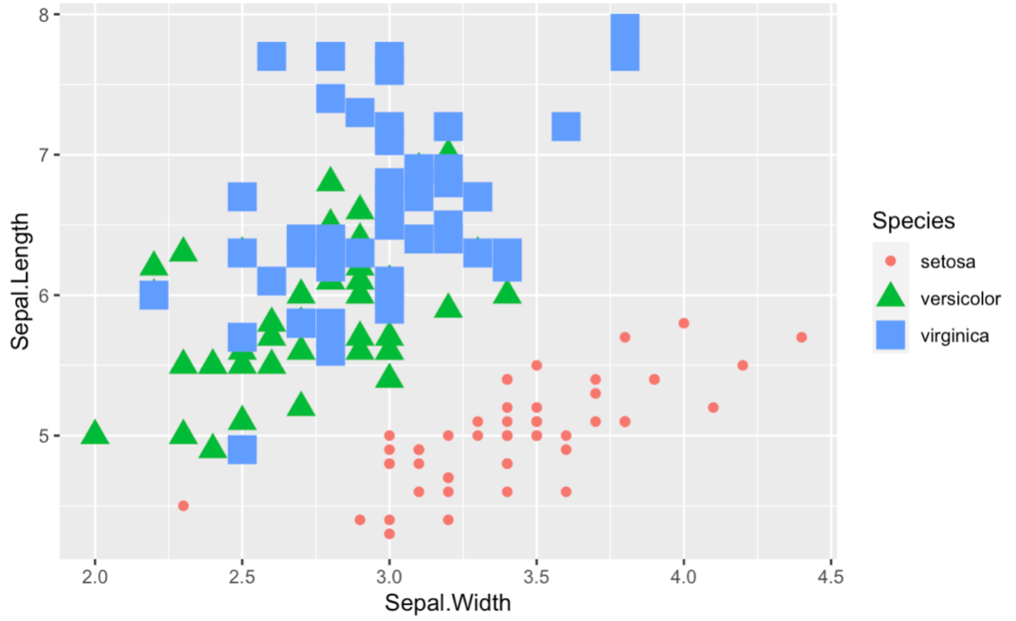


```
ggplot(data = iris)+
  geom_point(mapping = aes(x = Petal.Width, y = Petal.Length,
                           color = Species, size = Species,
                           shape = Species))
```
```


```

exercise

- plot sepal width and length for the three species
- is sepal information helpful is distinguishing the species?

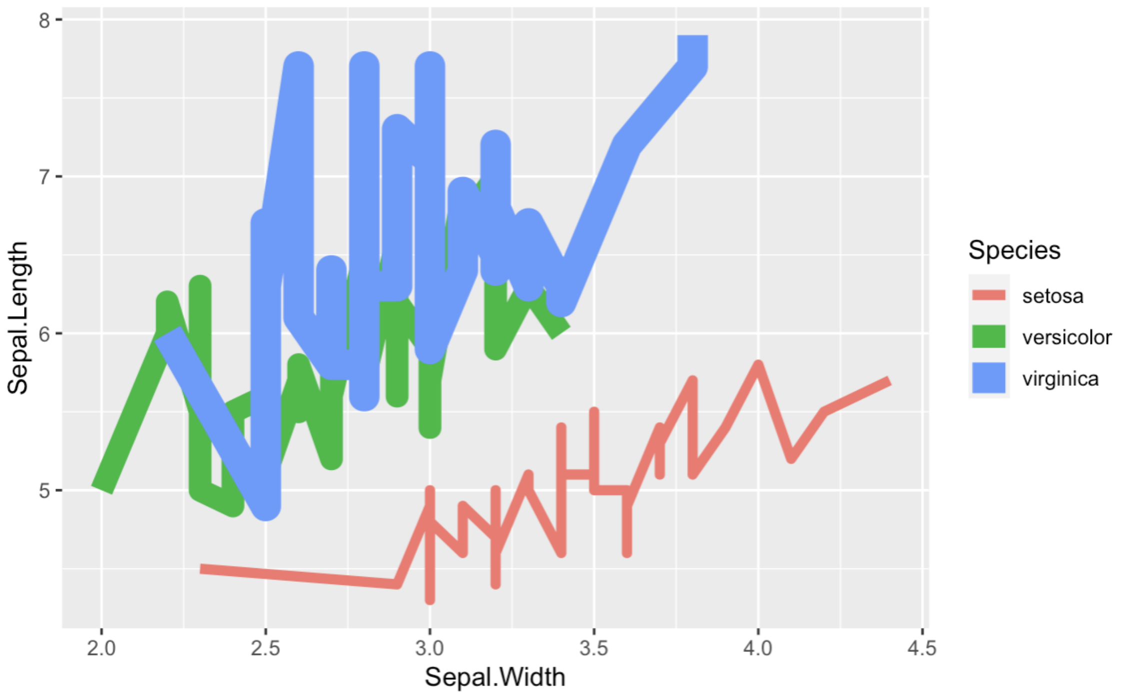


```
ggplot(data = iris)+
  geom_point(mapping = aes(x = Sepal.Width, y = Sepal.Length,
                           color = Species, size = Species,
                           shape = Species))
```
```


```

geometric objects

- geoms specify the type of visual object that will be used to display the data
  - scatterplots: geom_point()
  - histograms: geom_histogram ()
  - density plots: geom_density()
  - bar plots: geom_bar() / geom_col()
  - line plots: geom_line()
  - curves: geom_smooth()
  - boxplots: geom_boxplot()
- change geom_point() to geom_line() in the sepal plot

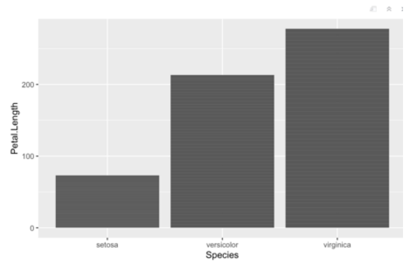
```
ggplot(data = iris)+
  geom_line(mapping = aes(x = Sepal.Width, y = Sepal.Length,
                          color = Species, size = Species,
                          shape = Species))
```



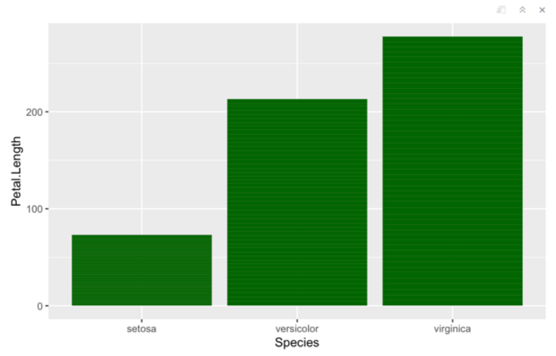
displaying aggregate information

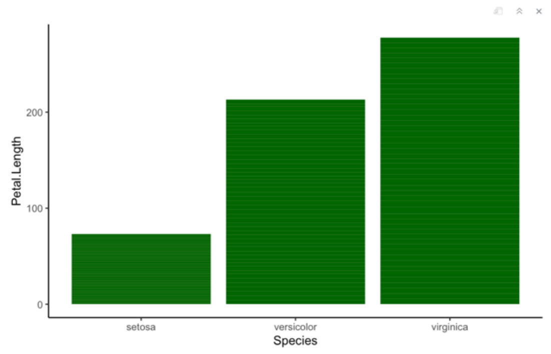
- if you simply wanted to display the aggregate information, bar/column plots are more suitable
- use geom_col() to display the petal lengths as a function of species
- fill the plot with any color you want (colors in R)
- add a theme to the plot (ggplot2 themes)

```
ggplot(data = iris)+
  geom_col(mapping = aes(x = Species, y = Petal.Length))
```



```
ggplot(data = iris)+
  geom_col(mapping = aes(x = Species, y = Petal.Length), fill = "darkgreen")
```

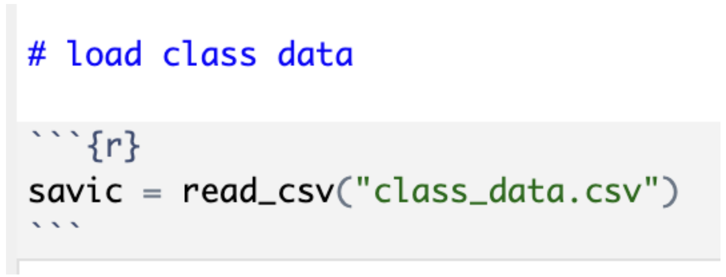




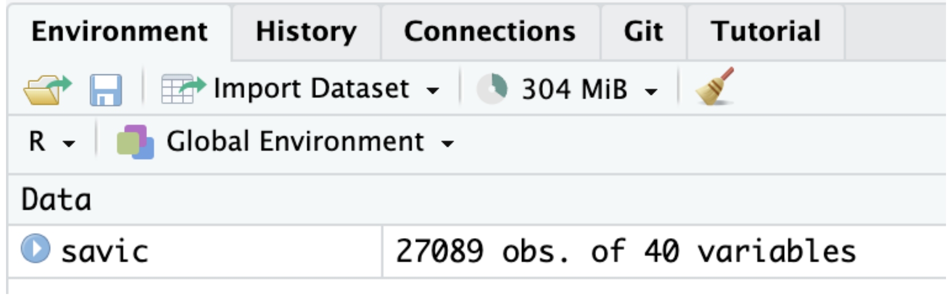
```
ggplot(data = iris)+
  geom_col(mapping = aes(x = Species, y = Petal.Length), fill = "darkgreen")+
  theme_classic()
```

class data from Savic et al. E1

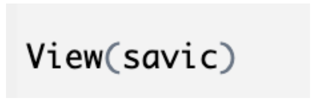
- create a new heading
  # load class data
- read in class data
- save and run the chunk
- view the data



~~~
# load class data

```{r}
savic = read_csv("class_data.csv")
```
~~~



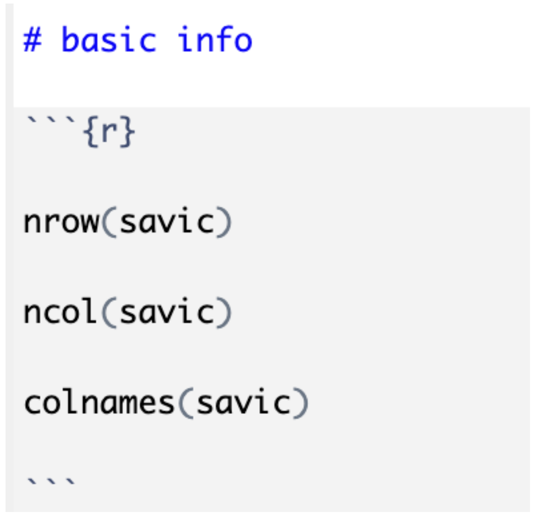
```
savic    27089 obs. of 40 variables
```



```
View(savic)
```

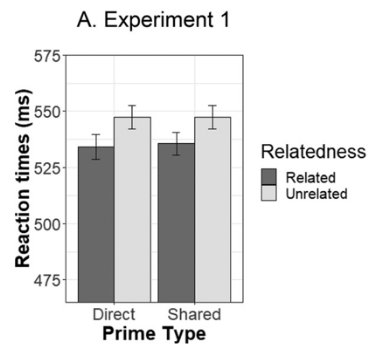
class data

- how many rows?
- how many columns?
- what are the column names?
- research question?



~~~
# basic info

```{r}

nrow(savic)

ncol(savic)

colnames(savic)

```
~~~

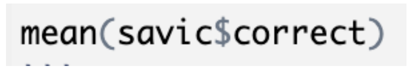
analysis preview



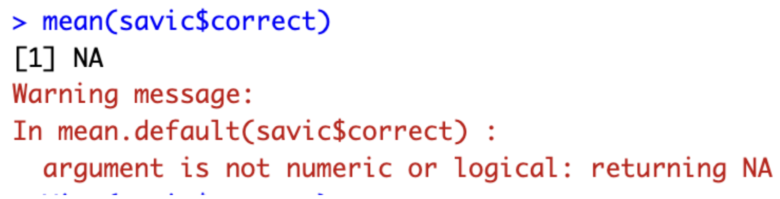
phase	measure	type	exclusion criteria
attention	accuracy	descriptive	< 0.75
priming	$RT_{related}$ vs. $RT_{unrelated}$ for direct and shared pairs	inferential (mixed effects model / ANOVA)	RT < 200 ms and RT > 1500 ms correct responses related/unrelated and direct/shared trials

basic descriptive information

- how would you find out the mean of the accuracy column?



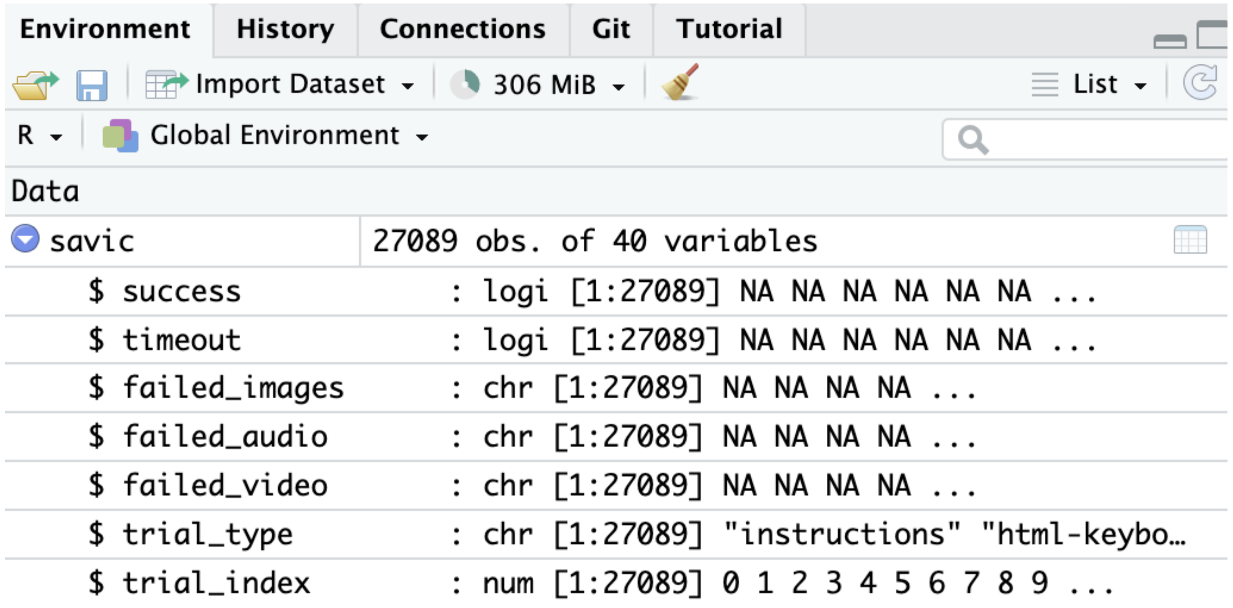
```
mean(savic$correct)
```



```
> mean(savic$correct)
[1] NA
Warning message:
In mean.default(savic$correct) :
  argument is not numeric or logical: returning NA
```

exploring data types

- look at the default data types assigned to these columns in our dataframe
    - correct
    - ID
    - rt
    - typeoftrial
    - relatedness
    - Type
- problem: fix datatypes

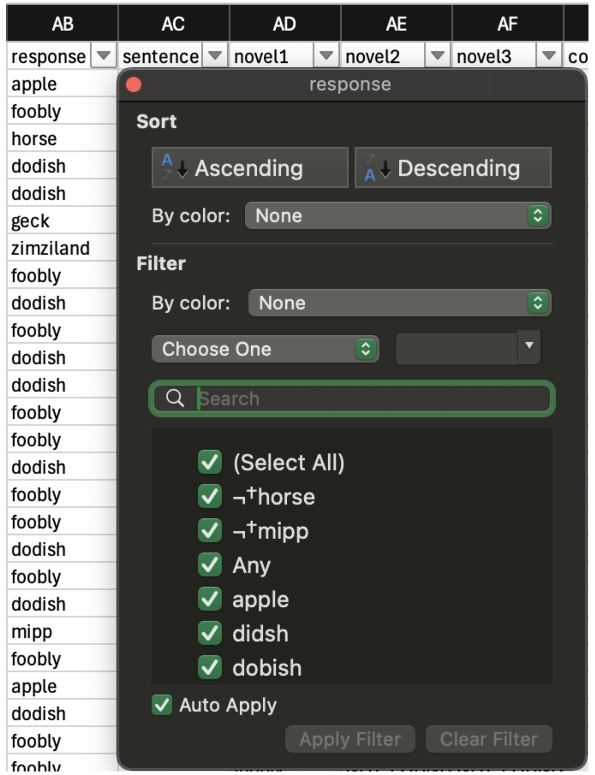


```
savic                        27089 obs. of 40 variables
    $ success          : logi [1:27089] NA NA NA NA NA NA ...
    $ timeout          : logi [1:27089] NA NA NA NA NA NA ...
    $ failed_images    : chr [1:27089] NA NA NA NA ...
    $ failed_audio     : chr [1:27089] NA NA NA NA ...
    $ failed_video     : chr [1:27089] NA NA NA NA ...
    $ trial_type       : chr [1:27089] "instructions" "html-keybo…
    $ trial_index      : num [1:27089] 0 1 2 3 4 5 6 7 8 9 ...
```

other issues?

(typical) plan for fixing responses

- upload class_data on drive
- open it up in google sheets
- create a “revised_response” and “revised_correct” column
- filter by incorrect attention responses
- fix typos in revised_response and accuracy in revised_correct



AB	AC	AD	AE	AF
response	sentence	novel1	novel2	novel3
apple				
foobly				
horse				
dodish				
dodish				
geck				
zimziland				
foobly				
dodish				
foobly				
dodish				
dodish				
foobly				
foobly				
dodish				
foobly				
foobly				
dodish				
foobly				
dodish				
mipp				
foobly				
apple				
dodish				
foobly				
foobly				

response

Sort

Ascending | Descending

By color: None

Filter

By color: None

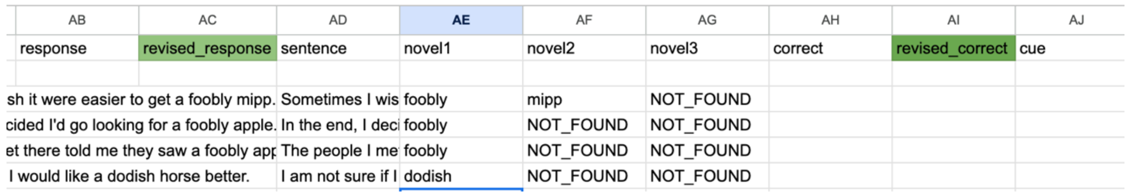
Choose One

Search

- ✓ (Select All)
- ✓ ¬†horse
- ✓ ¬†mipp
- ✓ Any
- ✓ apple
- ✓ didsh
- ✓ dobish

✓ Auto Apply

Apply Filter | Clear Filter



AB	AC	AD	AE	AF	AG	AH	AI	AJ
response	revised_response	sentence	novel1	novel2	novel3	correct	revised_correct	cue
sh it were easier to get a foobly mipp.		Sometimes I wis	foobly	mipp	NOT_FOUND			
cided I'd go looking for a foobly apple.		In the end, I deci	foobly	NOT_FOUND	NOT_FOUND			
et there told me they saw a foobly app		The people I me	foobly	NOT_FOUND	NOT_FOUND			
I would like a dodish horse better.		I am not sure if I	dodish	NOT_FOUND	NOT_FOUND			

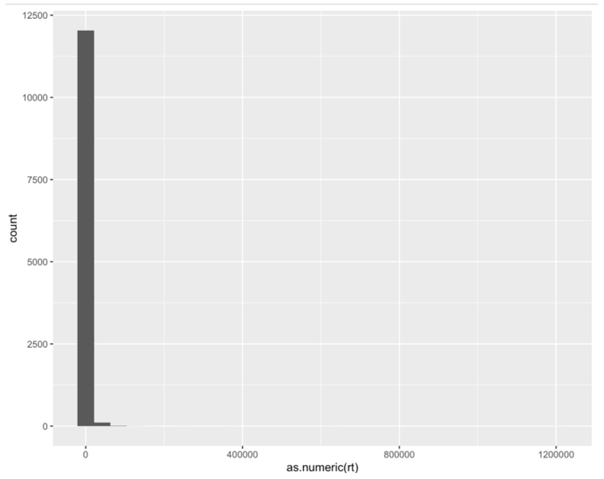
what kind of plots do we want?

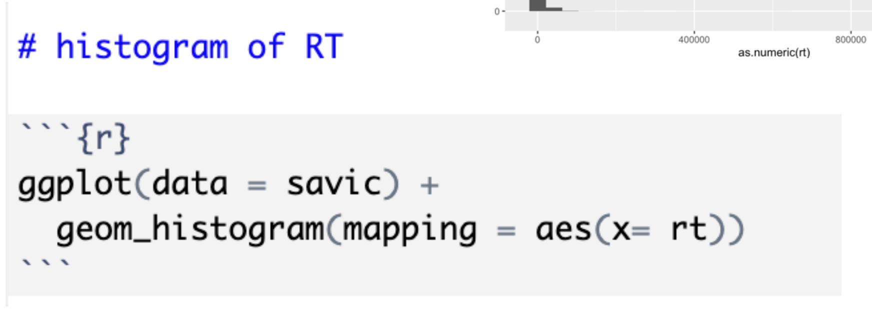
- think about each piece of data we will analyze
- which kind of plot will best capture those data?

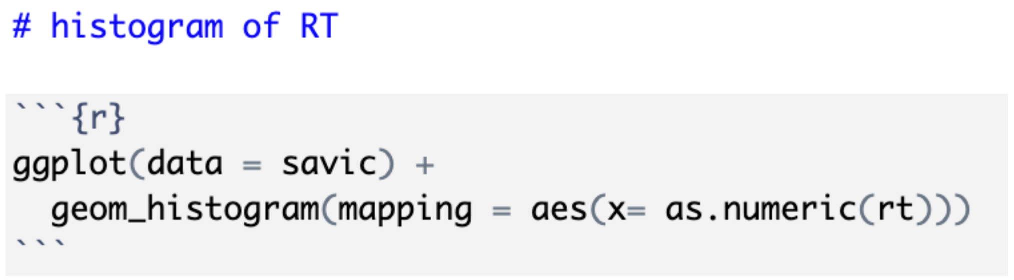
phase	measure	type	exclusion criteria
attention	accuracy	descriptive	< 0.75
priming	$RT_{related}$ vs. $RT_{unrelated}$ for direct and shared pairs	inferential (mixed effects model / ANOVA)	RT < 200 ms and RT > 1500 ms correct responses related/unrelated and direct/shared trials

histogram of reaction time

- create a new chunk titled
  # histogram of RT
- use ggplot to plot response time from the dataframe
- an error may be returned if R does not recognize RT as a number
- we can change the type of the column





~~~
# histogram of RT

```{r}
ggplot(data = savic) +
  geom_histogram(mapping = aes(x= rt))
~~~
```



```
# histogram of RT

```{r}
ggplot(data = savic) +
  geom_histogram(mapping = aes(x= as.numeric(rt)))
```
```
```

analyses checklist

- ❑ confirm/correct all datatypes
- ❑ figure out how to “filter” certain types of trials
- ❑ fix all typos in attention responses
- ❑ compute mean attention accuracy
- ❑ apply exclusions based on accuracy AND RTs
- ❑ create RT bar graph
- ❑ fit a statistical model
- ❑ report statistics

next class

- **before** class
  - *apply*: project milestone 4 (full experiment)
- **during** class
  - manipulating data