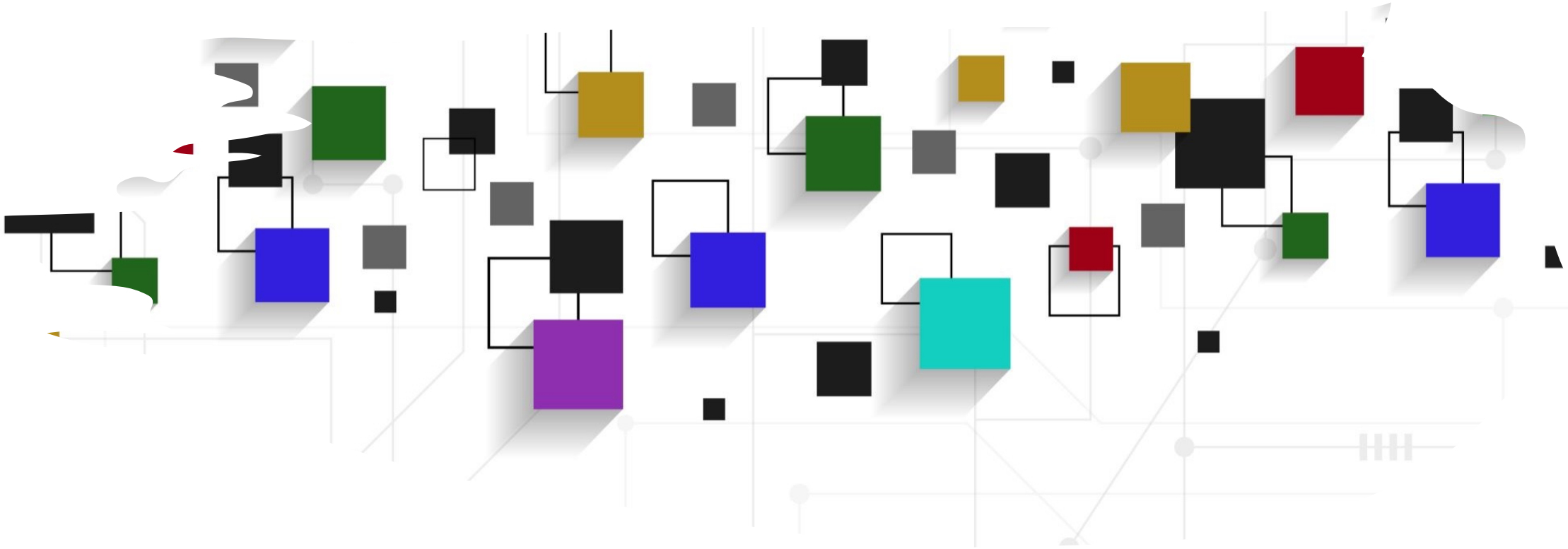


CogLab: Visualize Data

WEEK 8

recap: Oct 12, 2023

- what we covered:
  - reviewing data analysis plan
  - R 101
- your to-do's were:
  - *prep*: Visualization Basics primer from posit
  - *prep*: work on project milestone #5
  - *prep*: work on formative milestone #1 resubmission

today's agenda

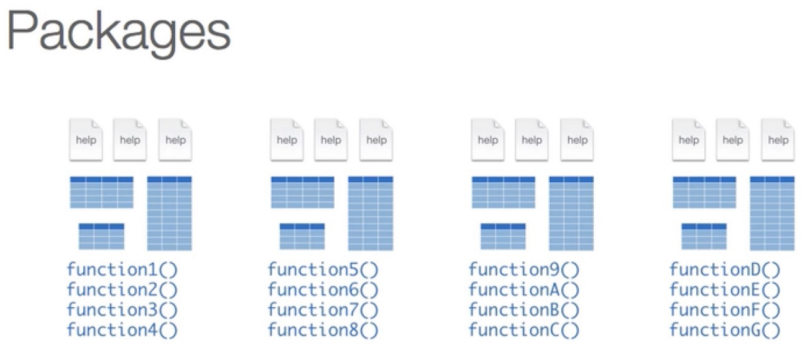
- packages in R
- graphing data

open your RStudio project

- open the project and your .Rmd file

packages in R

- packages contain functions
- before we can use them, we need to install packages
  - installation only happens once
- 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)



first-R-notebook.Rmd

Preview on Save | Preview

Source | Visual

```
1  ---
2  title: "R Notebook"
3  output: html_notebook
4  editor_options:
5    chunk_output_type: inline
6  ---
7
8
```

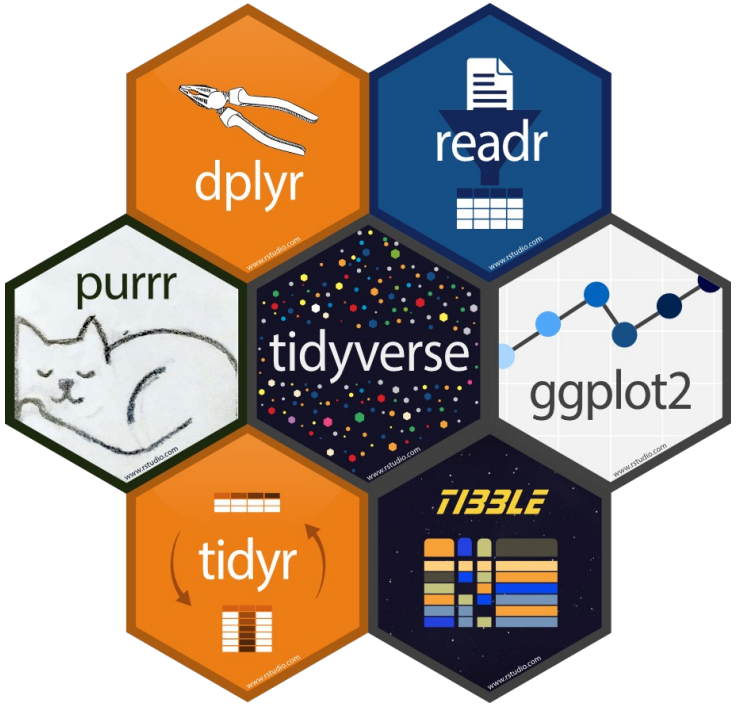
```
8  # install packages
9
10 ```{r}
11 install.packages("tidyverse", "ggplot2")
12 ```
```

install packages

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

tidyverse

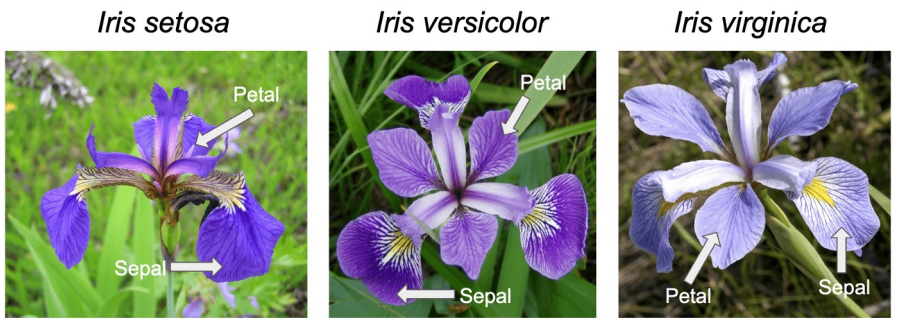
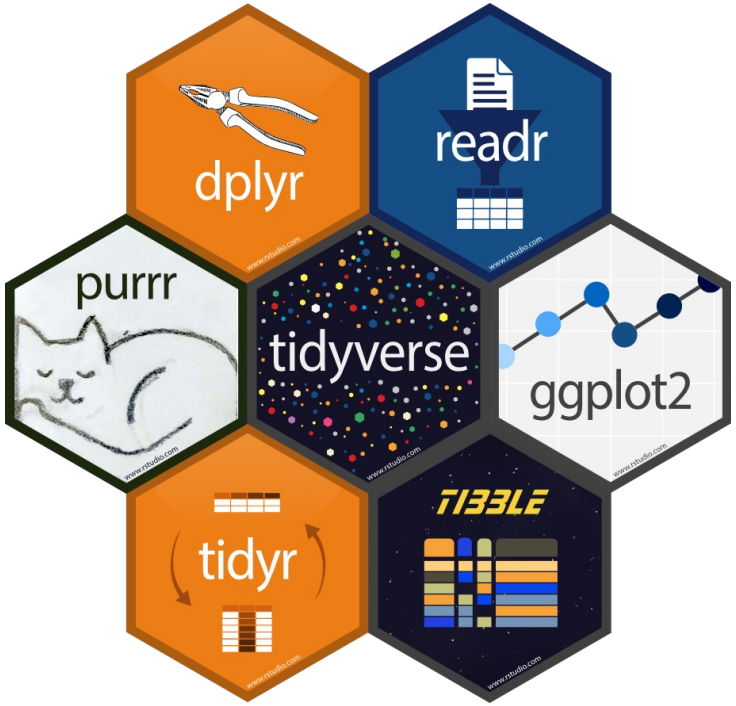
- a package of **packages**, which contain their own **functions**
- extremely flexible ways of manipulating and plotting data



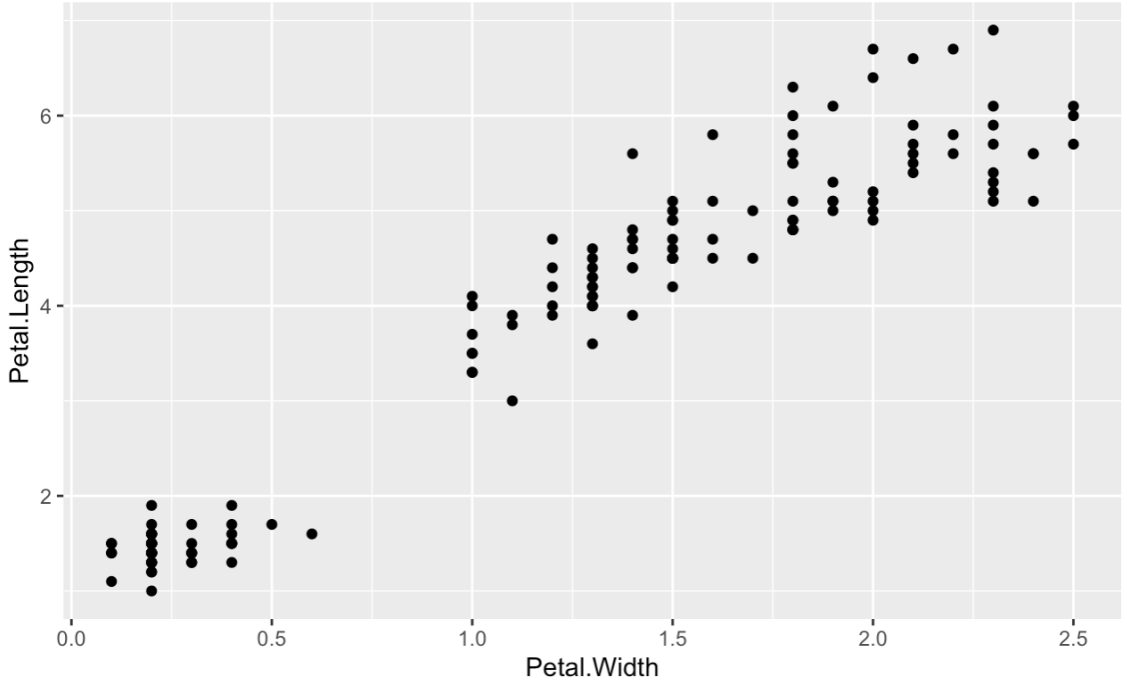
loading packages in R

- 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()
```

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



iris

Filter

	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

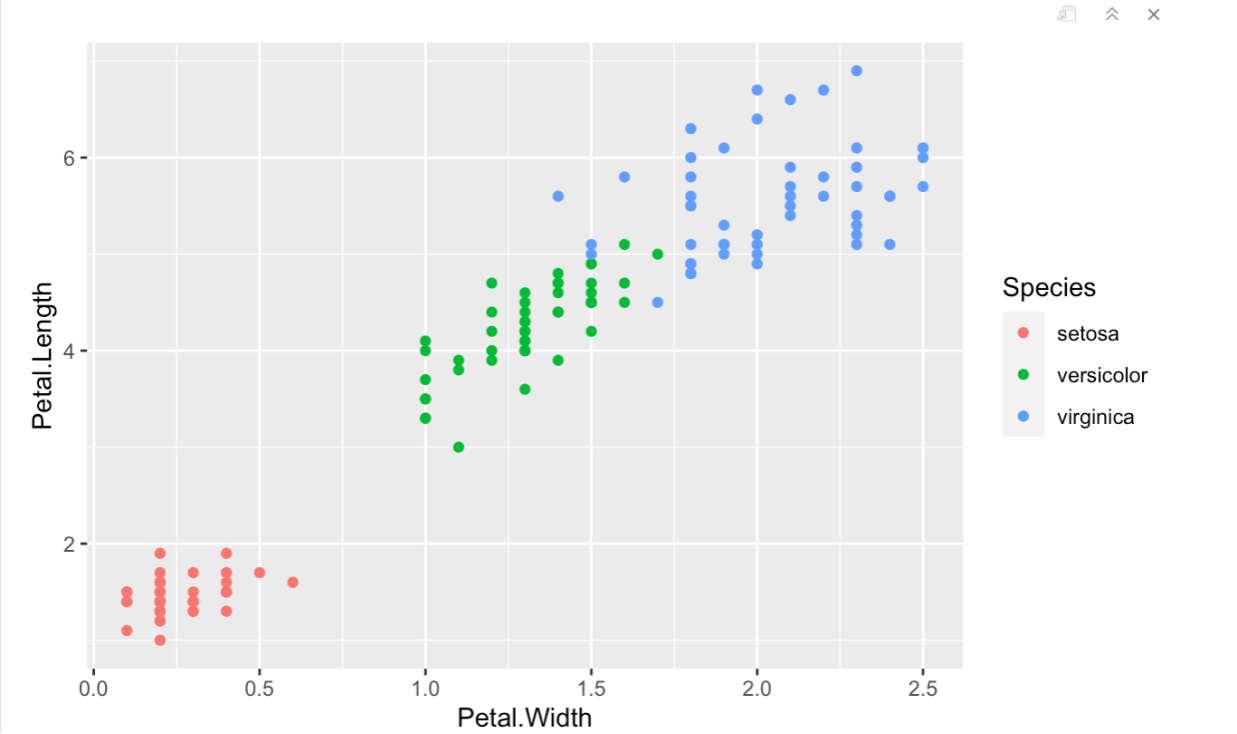
```{r}
ggplot(data = iris)+
  geom_point(mapping = aes(x = Petal.Width, y = Petal.Length))
```
```



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

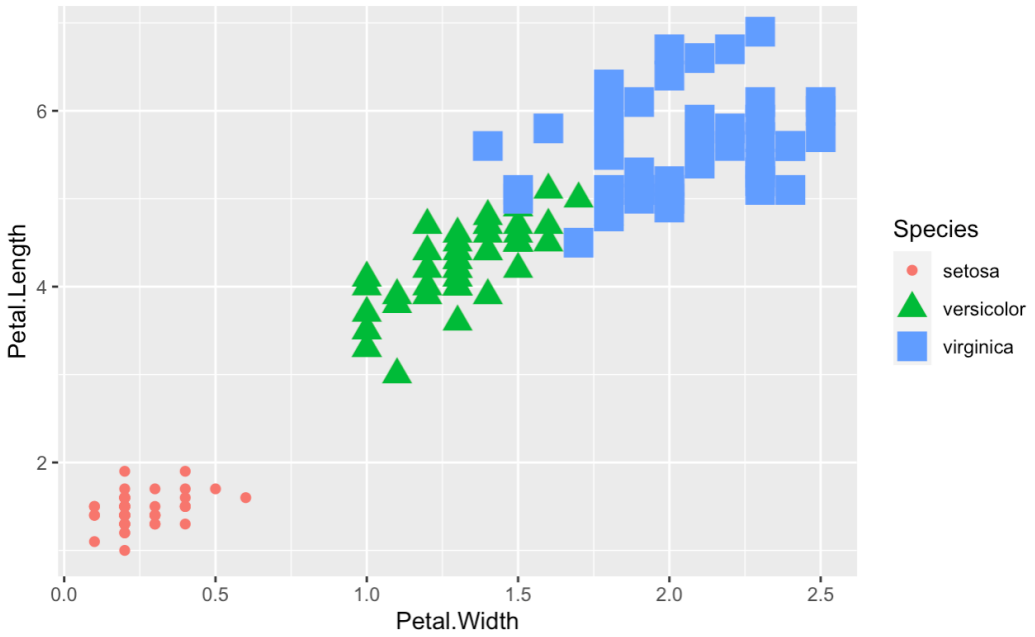
```{r}
ggplot(data = iris)+
  geom_point(mapping = aes(x = Petal.Width, y = Petal.Length, color = Species))
```



other aesthetics

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

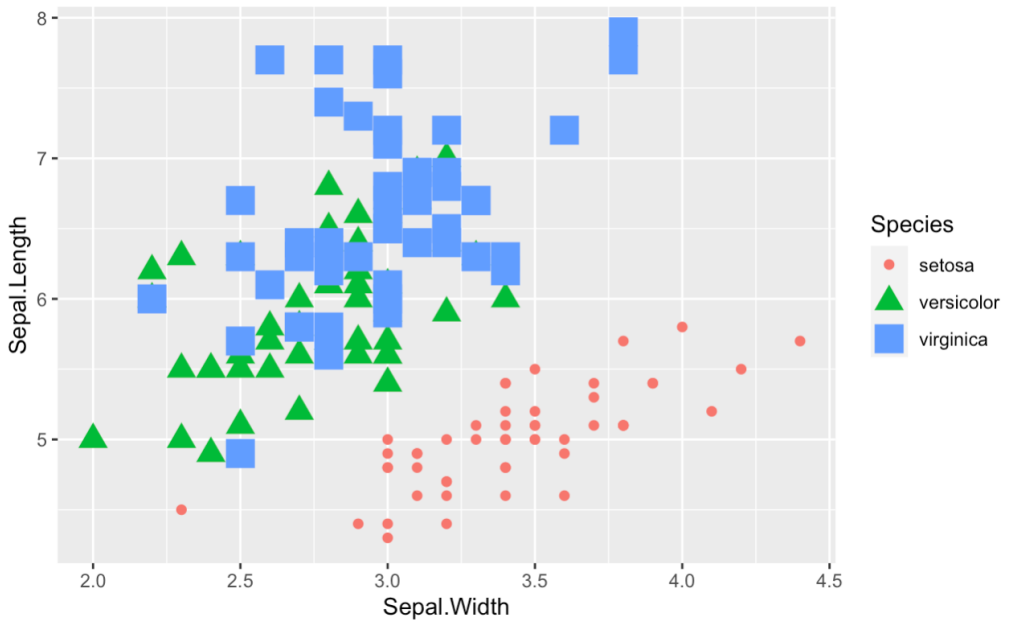
```
ggplot(data = iris)+
  geom_point(mapping = aes(x = Petal.Width, y = Petal.Length,
                           color = Species, size = Species,
                           shape = Species))
```



exercise

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

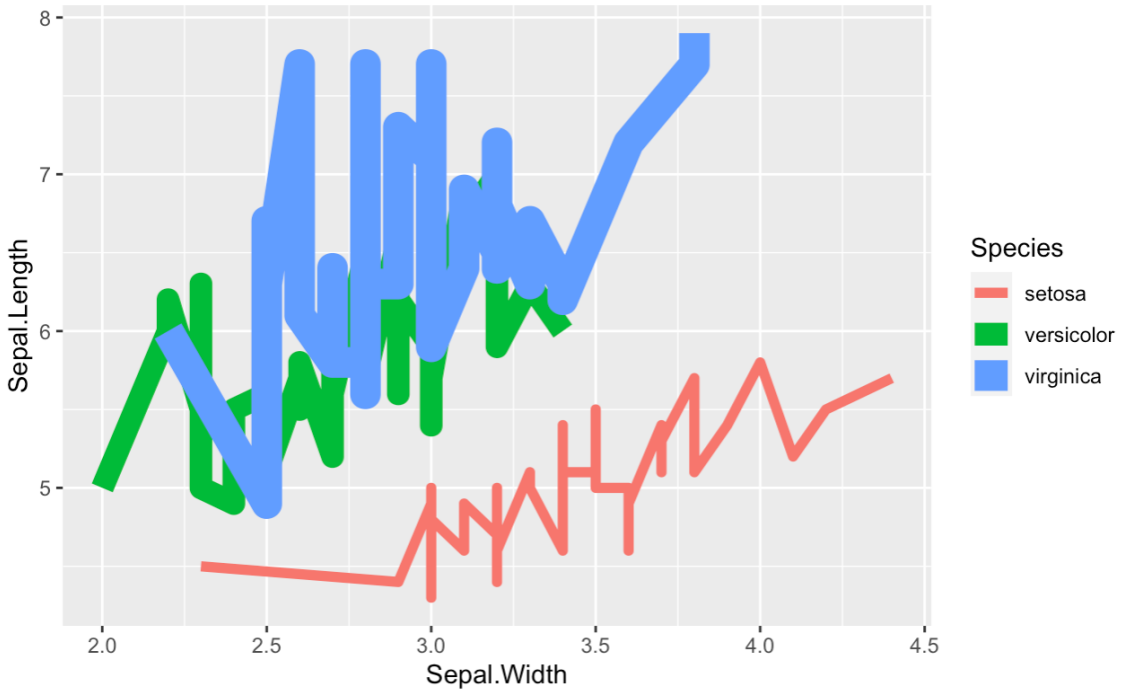
```
ggplot(data = iris)+
  geom_point(mapping = aes(x = Sepal.Width, y = Sepal.Length,
                           color = Species, size = Species,
                           shape = Species))
```



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

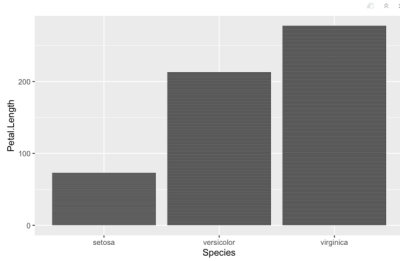
```
ggplot(data = iris)+
  geom_line(mapping = aes(x = Sepal.Width, y = Sepal.Length,
                          color = Species, size = Species,
                          shape = Species))
```



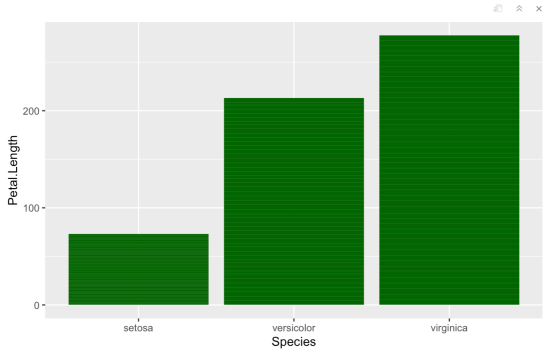
displaying aggregate information

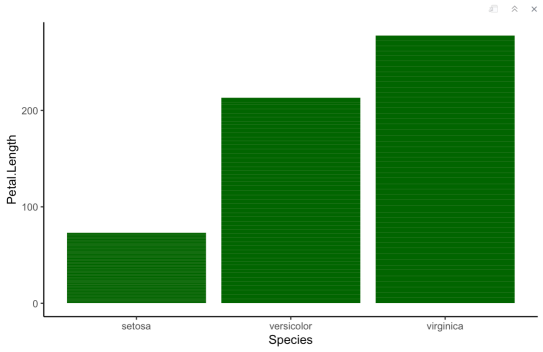
- if you simply wanted to display the aggregate information, bar/column plots are more suitable
- use geom_col() to display the petal lengths as a function of species
- fill the plot with any color you want (colors in R)
- add a theme to the plot (ggplot2 themes)

```
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")
```



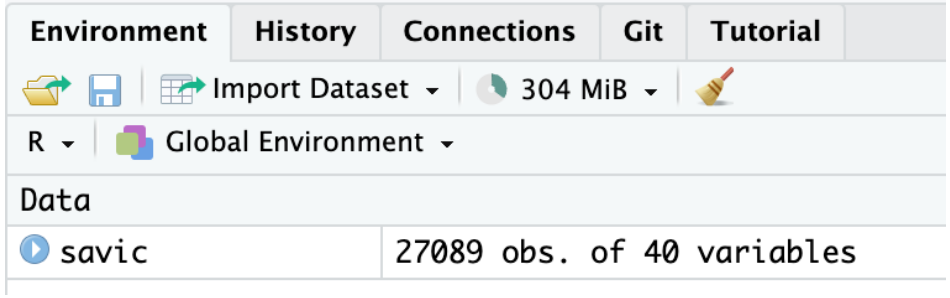


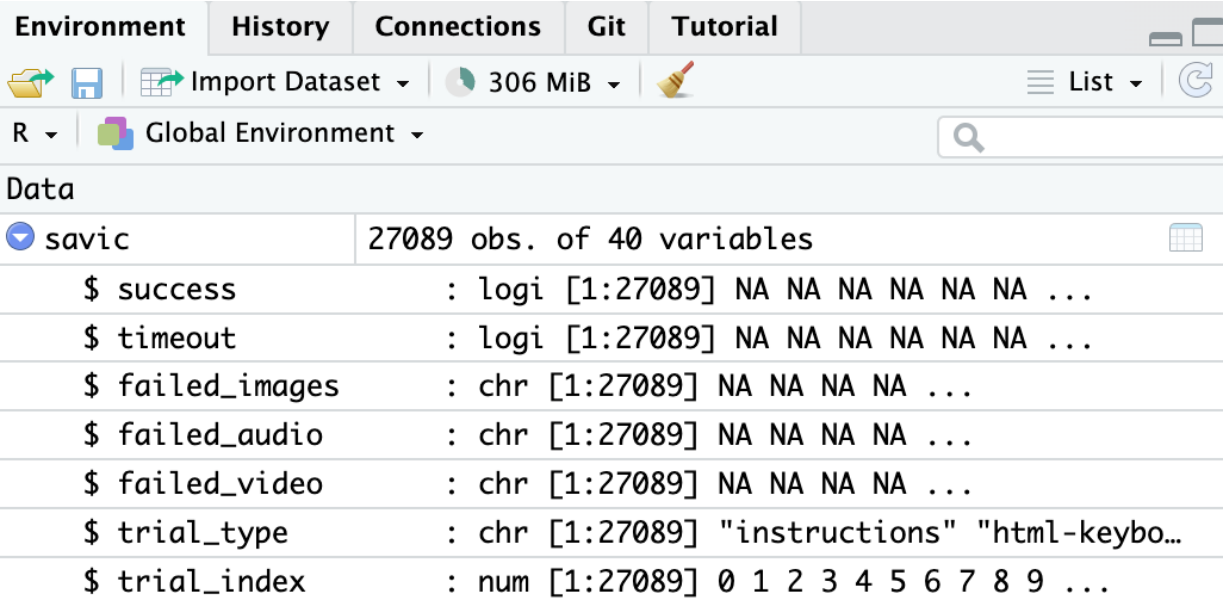
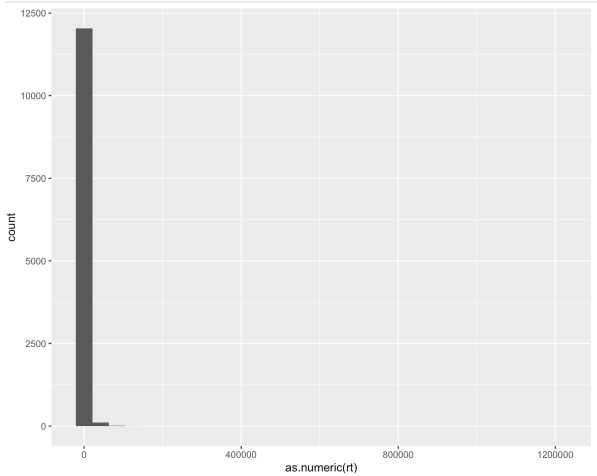

```
ggplot(data = iris)+
  geom_col(mapping = aes(x = Species, y = Petal.Length), fill = "darkgreen")+
  theme_classic()
```

class data

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

```
# load class data

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




```
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)

```
```

exploring data types

- look at the default data types assigned to these columns in our dataframe
- ID
- rt
- typeoftrial
- relatedness
- type



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

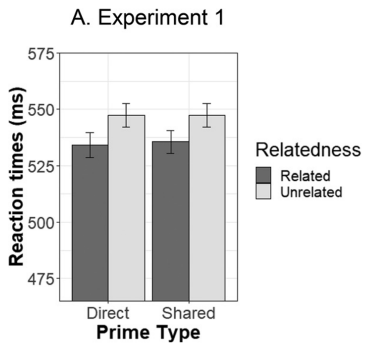
```{r}
ggplot(data = savic) +
  geom_histogram(mapping = aes(x= rt))
```
```

```
# histogram of RT

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

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?

analysis preview



phase	measure	type	exclusion criteria
attention	accuracy	descriptive	< 0.75
association	proportion of correct/congruent responses + direct/shared responding	descriptive	
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

HW: fixing accuracy

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 dec	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			

- go to revised_class_data on drive
- group task: fix the data!
  - Semantic Snakes: fix the attention check **responses + accuracy**
  - Berries: fix association responses, IDs 275998227- 276772242
  - Nellaphen: fix association responses, IDs 823472278 – 988749039
- complete before Tuesday (Oct 24)

next class

- **before** class
  - *prep:* Work with Data primer
  - *try:* HW, fix the data!
  - *apply:* Week 8 Quiz
  - *apply:* formative milestone # 1 resubmission
  - *apply:* project milestone 5 (full experiment)
- **during** class
  - manipulating data