Lab 1 - Introduction to R

ECOL 620 - Applications in Landscape Ecology

19 January, 2023

Contents

This lab assignment is meant as an introduction to the spatial capabilities of R. Here, we focus on reading comma separated value (csv) files, reading shapefiles, and plotting spatial data using ggplot2. Download Laboratory #1 materials and store the files locally on your computer.

RMarkdown

We will be using RMarkdown files to complete your lab assignments. RMarkdown lets you seamlessly interweave R code and a natural language (i.e. English) to create reproducible documents. This document that you are reading now was created using RMarkdown. As you keep reading this document, you will see how well RMarkdown can integrate text and code. You can create .html, .pdf, .doc/.docx, .ppt, and many more types of files using RMarkdown.

Every time you open a new Rmarkdown file (.Rmd), you will see an example .Rmd file. Notice how the code is inside "chunks" surrounded by 3 backticks (‘) at the top and 3 backticks at the bottom. This tells the markdown file that you are including computer code. Also, notice that right after the first 3 backticks, there is the letter r surrounded by curly brackets ({r}). This tells the markdown file that the computer code inside the chunk was written in the language R.

Also, notice that within the curly brackets, you can tell the .Rmd how you want it to display the code. For example, this: `{r echo=FALSE}` would hide the code in your final version of the document. `results = "hide"` will hide the console output from being included in the final version of your document. `message = FALSE` will hide any messages that R gives in the output from being included in the final version of your document. `message = FALSE` is especially useful if you want to show the code you used to load packages because there are usually many messages from this that you wouldn't want to include in the final report. There are many more options for chunk evaluations, but these are the three most common that you might need. To find more information, google Rmarkdown chunk options

You can write in English outside of the code chunks, and it's almost just like writing in MS Word. RStudio even has spell check that you can use. In order to *italicize* words or phrases, you need to surround the word

with one asterisks (*) at the beginning and end of the word/phrase like this (*word*). If you want to **bold**, you need to surround the word/phrase with two asterisks (**) on each side, like this (**word**).

Another thing that can be a little weird to get used to, is that in order to start a new line of text, you can't just hit enter like you would in Word. You must hit enter twice. The two spaces tells the file to start a new line of text.

Use the knit option at the top of your editor to have Rmarkdown make your file (i.e., .html, .pdf, .doc, etc.). You can make .html or .doc files right away without downloading any more software. If you want to make .pdf's you will need to download more software, and I don't expect you to turn any lab assignments in the .pdf format. In fact, I want all lab assignments turned in using the .html file format.

I often knit my .Rmd files many times to make sure that it's coming out the way I want it to.

Load Packages

We will use the following packages for the examples below

```
library(tidyverse)
library(sf)      # Functions to work with shapefiles and KML files
library(cowplot)
library(maps) # maps::map_data()
library(mapproj) # is required for `coord_map()`
```

Super basic R primer

We will start be moving through some of the functions of the R coding environment. In many ways, R at its most basic level, can simply act like a calculator. Let's have a look.

```
1+1
first_stored_value = 4.5+4.5
(20*first_stored_value)/2
sqrt(first_stored_value)
first_stored_value^3
```

Basic calculations

Building a vector A vector is the simplest type of data structure in R. Simply put, a vector is a sequence of data elements of the same basic type. Members of a vector are called components. Here is a vector containing six numeric values.

```
first_vector = c(1,2,4,5,7,19)
class(first_vector)
```

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

We can also make a vector by filling the structure with an ordered sequence.

```
seq(from=1, to=10, by=.1)
```

```
##  [1]  1.0  1.1  1.2  1.3  1.4  1.5  1.6  1.7  1.8  1.9  2.0  2.1  2.2  2.3  2.4
## [16]  2.5  2.6  2.7  2.8  2.9  3.0  3.1  3.2  3.3  3.4  3.5  3.6  3.7  3.8  3.9
## [31]  4.0  4.1  4.2  4.3  4.4  4.5  4.6  4.7  4.8  4.9  5.0  5.1  5.2  5.3  5.4
## [46]  5.5  5.6  5.7  5.8  5.9  6.0  6.1  6.2  6.3  6.4  6.5  6.6  6.7  6.8  6.9
## [61]  7.0  7.1  7.2  7.3  7.4  7.5  7.6  7.7  7.8  7.9  8.0  8.1  8.2  8.3  8.4
## [76]  8.5  8.6  8.7  8.8  8.9  9.0  9.1  9.2  9.3  9.4  9.5  9.6  9.7  9.8  9.9
## [91] 10.0
```

The `ls` function return a vector of character strings giving the names of the objects in the specified environment.

```
ls()
```

```
## [1] "encoding"           "first_stored_value" "first_vector"
## [4] "inputFile"          "out_dir"
```

Let's make three vectors with the aim of uniting them into the columns of a spreadsheet (i.e., data frame)

```
plot_id = (1:20)
species_richness = rpois(20, lambda = 10)
plant_mass = rnorm(mean=20, sd=3, n=20)
plot_group = rep(LETTERS[seq(from = 1, to = 5)],times=4)
```

Building a data frame A data frame is the most common way of storing data in R and, generally, is the data structure most often used for data analyses. Under the hood, a data frame is a list of equal-length vectors. Each element of the list can be thought of as a column and the length of each element of the list is the number of rows. As a result, data frames can store different classes of objects in each column (i.e. numeric, character, factor). In essence, the easiest way to think of a data frame is as an Excel worksheet that contains columns of different types of data but are all of equal length rows.

```
plot_data = cbind.data.frame(plot_id, plot_group, plant_mass, species_richness)
```

Indexing a data frame The elements of a data frame can be extracted by their name, either as an index, or by using the `$` operator.

```
plot_data$plot_id
```

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

```
plot_data$plot_group
```

```
##  [1] "A" "B" "C" "D" "E" "A" "B" "C" "D" "E" "A" "B" "C" "D" "E" "A" "B" "C" "D"
## [20] "E"
```

Like vectors, values of a data frame can be accessed through indexing. There are different ways to do this, but it is generally easiest to use two numbers in a double index. The first number is for the row number(s) and the second number is for the column number(s).

```
plot_data[1,] #1st row
```

```
##   plot_id plot_group plant_mass species_richness
## 1       1          A   18.62374                9
```

```
plot_data[,1] # 1st column
```

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

```
plot_data[1,4] #1st row, 4th column
```

```
## [1] 9
```

Subsetting and summarizing a data frame The `%>%` operator is used by the `dplyr` package. We can subset a portion of our data frame to the rows where plot_id equals "4". To do this, we used the `filter` function.

```
plot_data %>% filter(plot_id==4)
```

```
##   plot_id plot_group plant_mass species_richness
## 1       4          D   21.84275                8
```

Subset the data frame to the rows where plot_group equals "A".

```
plot_data %>% filter(plot_group=="A")
```

```
##   plot_id plot_group plant_mass species_richness
## 1       1          A   18.62374                9
## 2       6          A   17.86254               12
## 3      11          A   23.26431                7
## 4      16          A   24.20126               10
```

We often will want to check the class of our data. If you are encountering an error, I often first check the class of my data to make sure it's what I think it should be, e.g., character, numeric, factor, etc.

```
class(plot_data$plot_id)
```

```
## [1] "integer"
```

```
class(plot_data$plant_mass)
```

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

```
class(plot_data$plot_group)
```

```
## [1] "character"
```

The `str` function will reveal the dimensions of the data frame and the class of each column

```
str(plot_data)
```

```
## 'data.frame':    20 obs. of  4 variables:
##  $ plot_id         : int  1 2 3 4 5 6 7 8 9 10 ...
##  $ plot_group      : chr  "A" "B" "C" "D" ...
##  $ plant_mass      : num  18.6 16.3 15.4 21.8 19.3 ...
##  $ species_richness: int  9 9 11 8 14 12 6 12 7 14 ...
```

We can us the pipe features to group our data (`group_by`), then take a summary (`summarise`) of our data frame (e.g., sample size, mean, min, median, etc.)

```
plot_data %>% group_by(plot_group) %>%
  summarise(n=n(),
            mean_species_richness=mean(species_richness),
            mean_plant_mass=mean(plant_mass))
```

```
## # A tibble: 5 x 4
##   plot_group     n mean_species_richness mean_plant_mass
##   <chr>      <int>                 <dbl>           <dbl>
## 1 A              4                   9.5            21.0
## 2 B              4                  10              19.8
## 3 C              4                  11.2            17.3
## 4 D              4                   9.5            21.9
## 5 E              4                  11              19.5
```

Examining what is in the environment We can determine what is in our environment using the `ls` function (also referred to as the *list* function).

```
ls()
```

```
##  [1] "encoding"           "first_stored_value" "first_vector"
##  [4] "inputFile"          "out_dir"            "plant_mass"
##  [7] "plot_data"          "plot_group"         "plot_id"
## [10] "species_richness"
```

Remove an item in our environment

```
remove("plot_data")
```

Now we cab clear our environment completely

```
remove(list = ls())
```

---

ggplot fun

ggplot2 is a system for declaratively creating graphics, based on The Grammar of Graphics. You provide the data, tell ggplot2 how to map variables to aesthetics, what graphical primitives to use, and it takes care of the details. You always start with ggplot() and then you supply a dataset and aesthetic mapping (with `aes()`). You then add on layers (like `geom_point()` or `geom_histogram()`), scales (like `scale_colour_brewer()`), faceting specifications (like `facet_wrap()`) and coordinate systems (like `coord_sf()`).

We will use a pre-existing dataset in R to explore plotting.

```
iris = iris
str(iris)
```

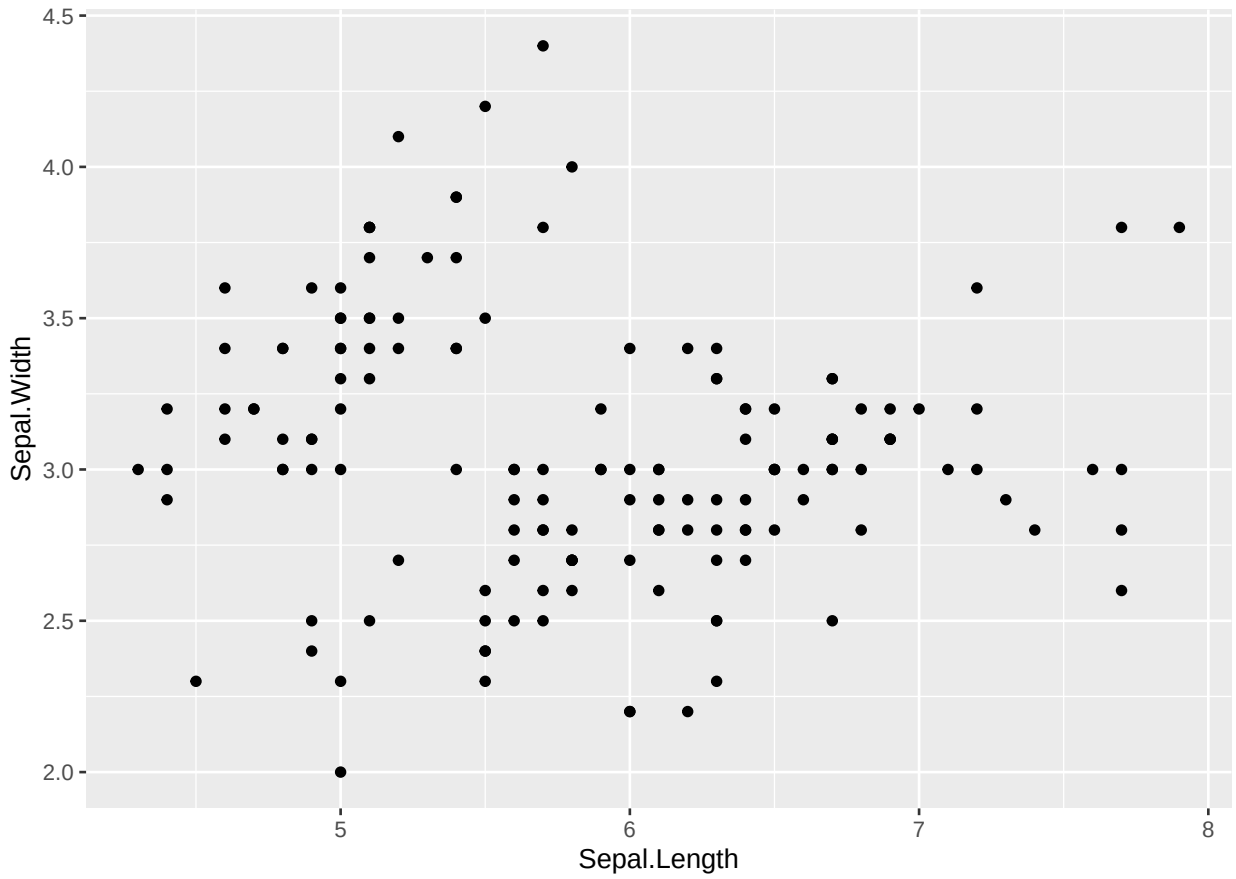
```
## 'data.frame':    150 obs. of  5 variables:
##  $ Sepal.Length: num  5.1 4.9 4.7 4.6 5 5.4 4.6 5 4.4 4.9 ...
##  $ Sepal.Width : num  3.5 3 3.2 3.1 3.6 3.9 3.4 3.4 2.9 3.1 ...
##  $ Petal.Length: num  1.4 1.4 1.3 1.5 1.4 1.7 1.4 1.5 1.4 1.5 ...
##  $ Petal.Width : num  0.2 0.2 0.2 0.2 0.2 0.4 0.3 0.2 0.2 0.1 ...
##  $ Species     : Factor w/ 3 levels "setosa","versicolor",..: 1 1 1 1 1 1 1 1 1 1 ...
```

We will first create a blank ggplot canvas. However, we will keep building on this canvas, adding new and more detailed layers.

```
ggplot()
```

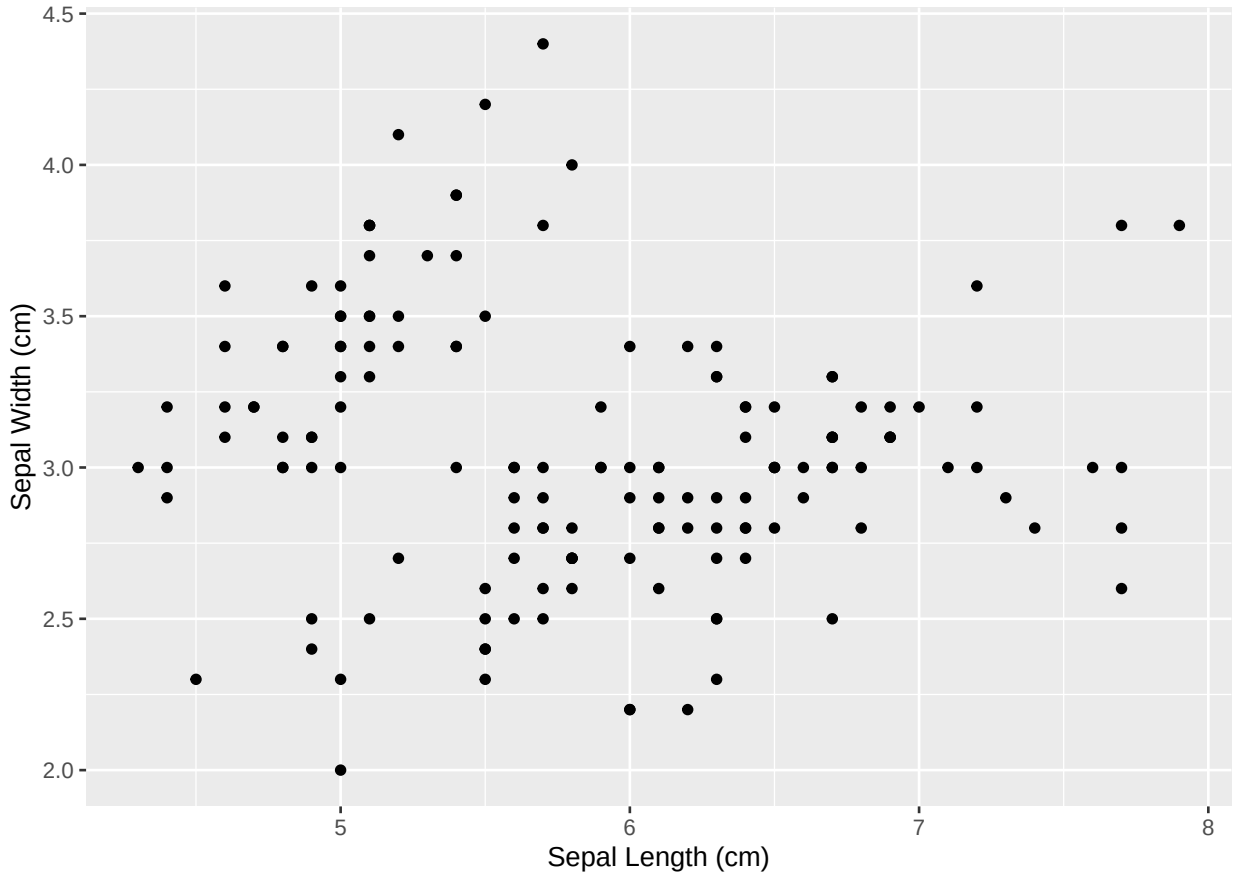
Now, let's make our first scatter plot using the `geom_point` function.

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



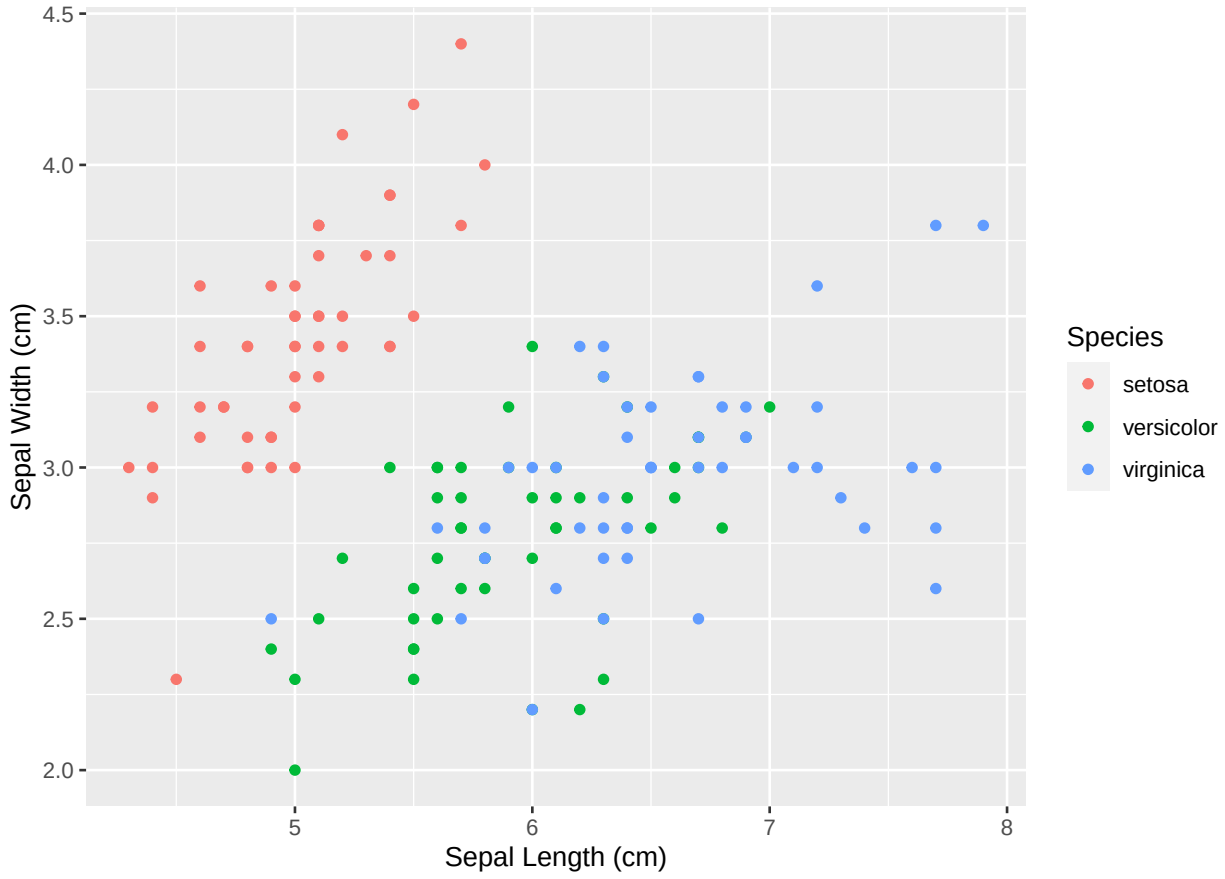
Let's add axis labels.

```
ggplot(data=iris)+
  geom_point(mapping=aes(x=Sepal.Length, y=Sepal.Width))+
  labs(x="Sepal Length (cm)", y="Sepal Width (cm)")
```



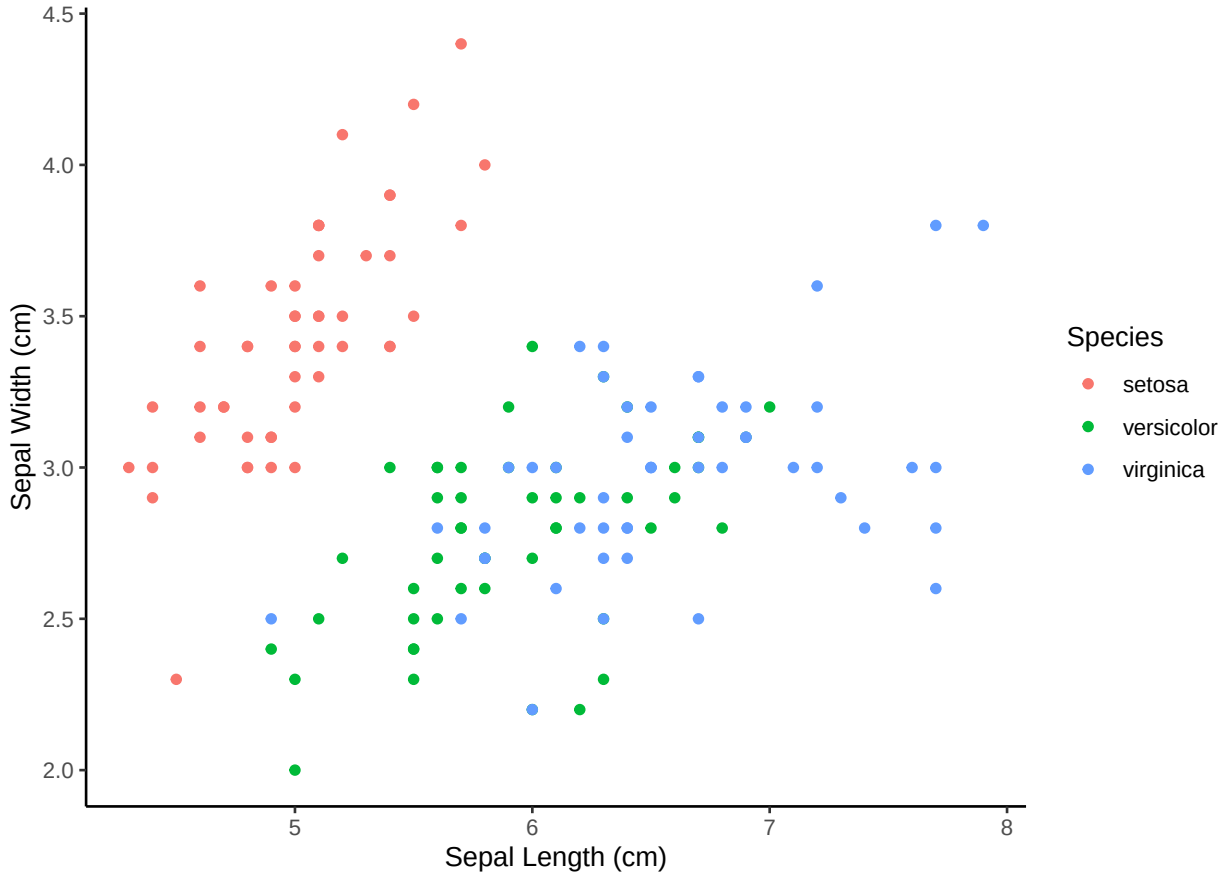
Let's add color to the points, and give different colors by *species*.

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



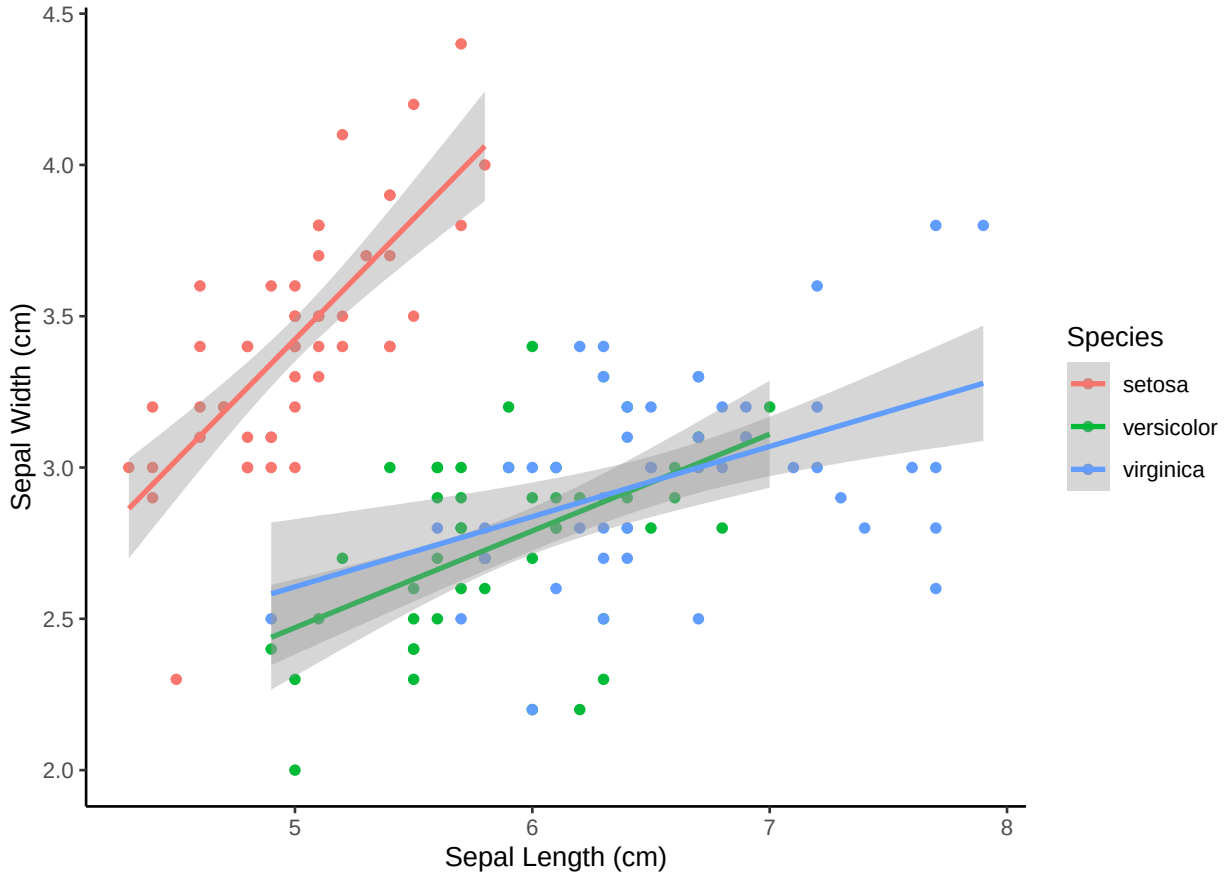
We can create a more professional plot using the `theme_classic()` aesthetic function.

```
ggplot(data=iris)+
  geom_point(mapping=aes(x=Sepal.Length, y=Sepal.Width, color=Species))+
  labs(x="Sepal Length (cm)", y="Sepal Width (cm)")+
  theme_classic()
```



Lastly, we can add a lines of best fit from linear regression models done by species using the `geom_smooth()`function.

```
ggplot(data=iris)+
  geom_point(mapping=aes(x=Sepal.Length, y=Sepal.Width, color=Species))+
  labs(x="Sepal Length (cm)", y="Sepal Width (cm)")+
  geom_smooth(aes(x=Sepal.Length, y=Sepal.Width,group=Species, colour=Species), formula ='y ~ x',method=
  theme_classic()
```



US Maps

Within R, there are some spatial datasets we can manipulate and plot.

```
states = as.data.frame(state.x77)
states$region = tolower(rownames(states))
```

ggplot has some map data too.

```
states_map = map_data("state")
class(states_map)
```

```
## [1] "data.frame"
```

Let's look to see which states are represented?

```
unique(states_map$region)
```

```
##  [1] "alabama"              "arizona"              "arkansas"
##  [4] "california"           "colorado"             "connecticut"
```

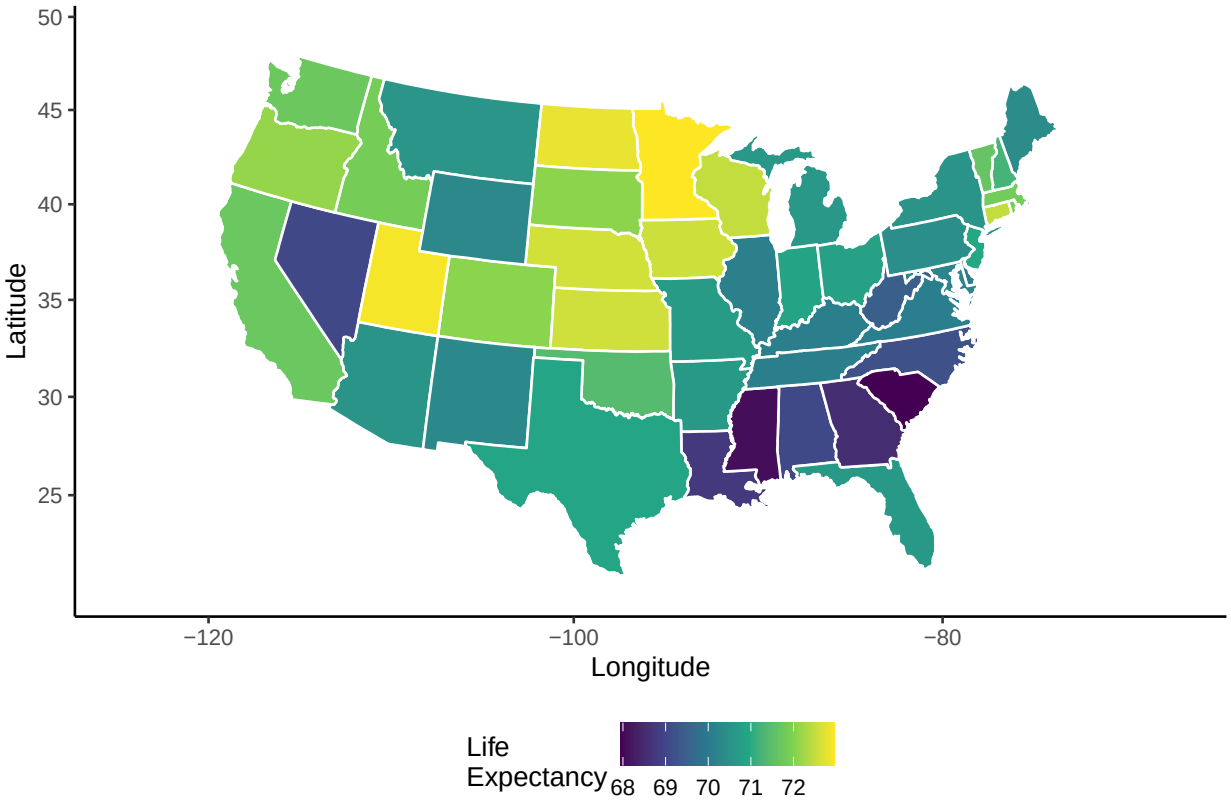
```
##  [7] "delaware"            "district of columbia" "florida"
## [10] "georgia"             "idaho"                "illinois"
## [13] "indiana"             "iowa"                 "kansas"
## [16] "kentucky"            "louisiana"            "maine"
## [19] "maryland"            "massachusetts"        "michigan"
## [22] "minnesota"           "mississippi"          "missouri"
## [25] "montana"             "nebraska"             "nevada"
## [28] "new hampshire"       "new jersey"           "new mexico"
## [31] "new york"            "north carolina"       "north dakota"
## [34] "ohio"                "oklahoma"             "oregon"
## [37] "pennsylvania"        "rhode island"         "south carolina"
## [40] "south dakota"        "tennessee"            "texas"
## [43] "utah"                "vermont"              "virginia"
## [46] "washington"          "west virginia"        "wisconsin"
## [49] "wyoming"
```

Let's merge the two datasets. This merges the *states* data with the spatial information for each state.

```
fact_join = left_join(states_map, states, by = "region")
```

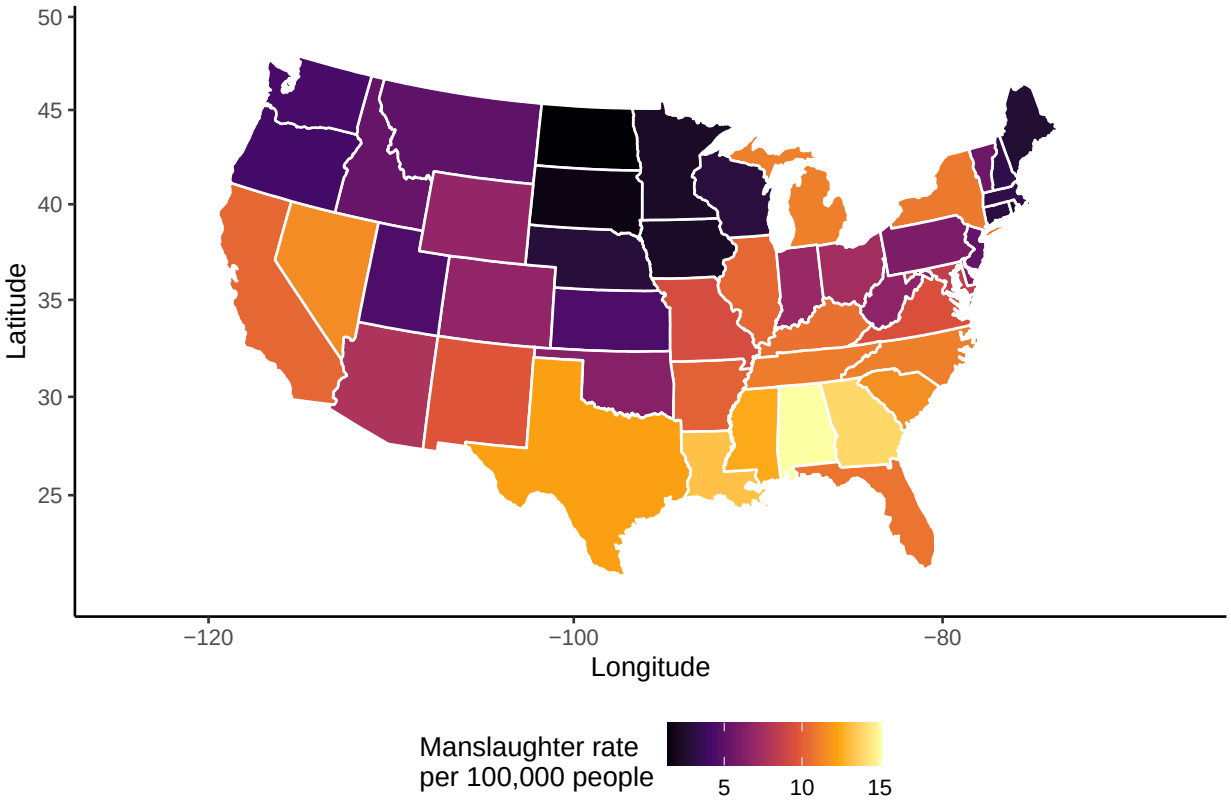
Let's map *Life Expectancy in Years (1969–71)* by state.

```
life_expectancy = ggplot(data=fact_join, mapping=aes(x=long, y=lat, group = group))+
  geom_polygon(aes(fill = `Life Exp`), colour = "white")+
  scale_fill_viridis_c(option = "D")+
  theme_classic()+
  coord_map("bonne", lat0 = 40)+
  labs(y = "Latitude", x = "Longitude", fill="Life\nExpectancy")+
  theme(legend.position = "bottom")
print(life_expectancy)
```



Now, let's map *Murder and non-negligent manslaughter rate per 100,000 population (1976)* by state.

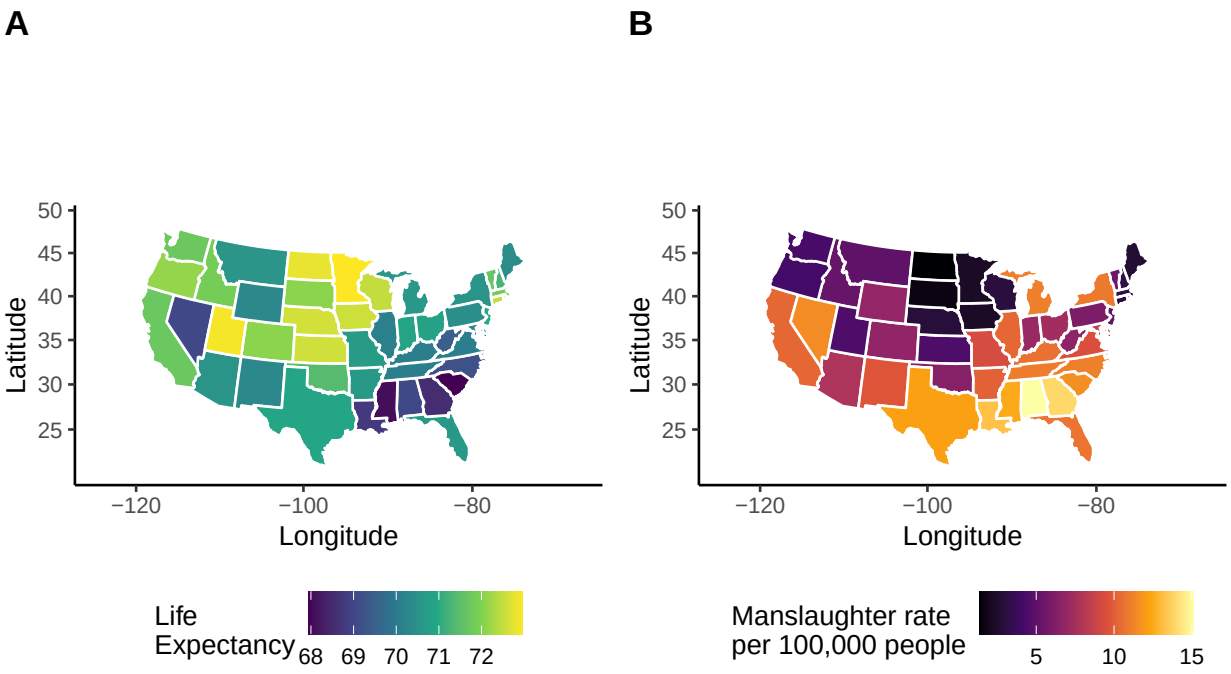
```
murder = ggplot(fact_join, aes(long, lat, group = group))+
  geom_polygon(aes(fill = Murder), color = "white")+
  scale_fill_viridis_c(option = "B")+
  theme_classic()+
  coord_map("bonne", lat0 = 40)+
  labs(y = "Latitude", x = "Longitude", fill="Manslaughter rate\nper 100,000 people") +
  theme(legend.position = "bottom")
print(murder)
```



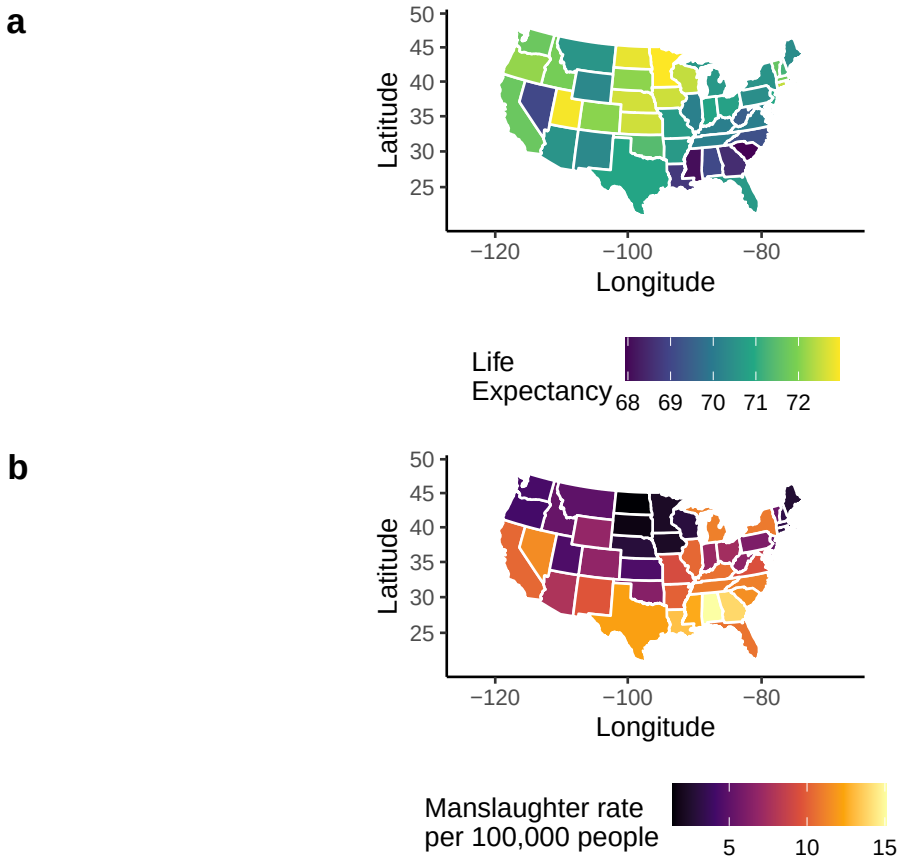
We can used the `plot_grid` function to make aggregate plots.

We can make the panel plot either horizontal or vertical by changing the `nrow` attribute.

```
plot_grid(life_expectancy, murder, labels = "AUTO", nrow=1)
```




```
plot_grid(life_expectancy, murder, labels = "auto", ncol=1)
```



Bring datasets into the R environment.

Good data management

1. No spaces within object names (same for names in data files, use . or _ instead).
2. Don't name an object similar/same as a function (e.g., don't name your data "data.frame").
3. Can't start object names with a number.

Import data file Many ways data can be imported Many types of files can be imported (shape files, text files, csv)

`read.table` reads any table, can specify which format `read.csv` fields are separated by a comma `readxl` reads Micrsoft Excel files

Let's read in a file containing US college and university geographic information.

```
us_uni_csv = read.csv("../data/universities.csv")
```

From the file we just read in, named `us_uni_csv`, let's subset the data frame to just *Colorado State University.*

```
csu = us_uni_csv %>% filter(NAME=="Colorado State University")
print(csu)
```

```
##   IPEDSID                      NAME                      LADDR LADDR2
## 1  126818 Colorado State University 102 Administration Building     NA
##          LCITY LSTATE  LZIP LZIP4 COUNTRY      PHONE STFIPS COFIPS      LAT
## 1 Fort Collins     CO 80523   100      US 9704911101      8   8069 40.57476
##         LON SECTOR LEVEL INST_TYPE NAICS_CODE
## 1 -105.0808      1     1         1     611310
##                                    NAICS_DESC HI_OFFER DEG_GRANT LOCALE
## 1 Colleges, Universities, and Professional Schools       11         1     12
##   STATUS CLOSE_DATE MERGE_ID ALIAS SIZE_SET INST_SIZE PT_ENROLL FT_ENROLL
## 1      A         -2       -2  <NA>       15         5      5585     22211
##   TOT_ENROLL HOUSING DORM_CAP  FTE                          NCES_URL
## 1      27796       1     5240 8473 http://nces.ed.gov/GLOBALLOCATOR/
##          SCHOOL_URL SHELTER_ID LAST_UPDAT
## 1 www.colostate.edu         NA    1/15/10
```

Now, let's subset to the state of Colorado.

```
colorado_universities = us_uni_csv %>% filter(LSTATE=="CO")
head(colorado_universities)
```

```
##   IPEDSID                                NAME                   LADDR LADDR2
## 1  126915  Delta Montrose Technical College           1765 US Hwy 50     NA
## 2  127219           Glenwood Beauty Academy 51241 Hwy 6 and 24 Ste 1     NA
## 3  128391 Western State College of Colorado              600 N Adams     NA
## 4  127185                Fort Lewis College           1000 Rim Drive     NA
## 5  126164    The Salon Professional Academy        2938 North Ave #B     NA
## 6  128188 Intellitec College-Grand Junction           772 Horizon Dr     NA
##              LCITY LSTATE  LZIP LZIP4 COUNTRY      PHONE STFIPS COFIPS      LAT
## 1            Delta     CO 81416    NA      US 9708747671      8   8029 38.70252
## 2 Glenwood Springs     CO 81601    NA      US 9709450485      8   8045 39.57547
## 3         Gunnison     CO 81231    NA      US 9709430120      8   8051 38.54742
## 4          Durango     CO 81301  3999      US 9702477010      8   8067 37.27543
## 5   Grand Junction     CO 81504    NA      US 9702451110      8   8077 39.07820
## 6   Grand Junction     CO 81506    NA      US 9702458101      8   8077 39.11635
##         LON SECTOR LEVEL INST_TYPE NAICS_CODE
## 1 -108.0311      7     3         1     611519
## 2 -107.4467      6     2         3     611511
## 3 -106.9197      1     1         1     611310
## 4 -107.8670      1     1         1     611310
## 5 -108.5076      6     2         3     611519
## 6 -108.5306      6     2         3     611210
##                                    NAICS_DESC HI_OFFER DEG_GRANT LOCALE
## 1                Other Technical and Trade Schools        0         2     42
## 2                   Cosmetology and Barber Schools        0         2     33
## 3 Colleges, Universities, and Professional Schools       30         1     33
## 4 Colleges, Universities, and Professional Schools       30         1     32
## 5                Other Technical and Trade Schools        0         2     13
## 6                                  Junior Colleges       40         1     13
##   STATUS CLOSE_DATE MERGE_ID ALIAS SIZE_SET INST_SIZE PT_ENROLL FT_ENROLL
```

```
## 1      A         -2       -2  <NA>       -3         1      1500       115
## 2      A         -2       -2  <NA>       -3         1        18        34
## 3      A         -2       -2  <NA>       10         2       273      1850
## 4      A         -2       -2   FLC       13         2       336      3410
## 5      A         -2       -2  <NA>       -3         1         0        74
## 6      A         -2       -2  <NA>        1         1         0       770
##   TOT_ENROLL HOUSING DORM_CAP FTE                          NCES_URL
## 1       1615       2        0  49 http://nces.ed.gov/GLOBALLOCATOR/
## 2         52       2        0   9 http://nces.ed.gov/GLOBALLOCATOR/
## 3       2123       1     1130 301 http://nces.ed.gov/GLOBALLOCATOR/
## 4       3746       1     1391 598 http://nces.ed.gov/GLOBALLOCATOR/
## 5         74       2        0  11 http://nces.ed.gov/GLOBALLOCATOR/
## 6        770       2        0  84 http://nces.ed.gov/GLOBALLOCATOR/
##                           SCHOOL_URL SHELTER_ID LAST_UPDAT
## 1                        www.dmtc.edu         NA     9/1/10
## 2                               <NA>         NA    1/15/10
## 3                     www.western.edu         NA    1/15/10
## 4                   www.fortlewis.edu         NA     9/1/10
## 5 www.thesalonprofessionalacademy.net         NA     9/1/10
## 6           www.intelliteccollege.com         NA     9/1/10
```

How many institutions reside in Colorado?

```
nrow(colorado_universities)
```

```
## [1] 112
```

What else is in this data frame? We see that data frame includes information on the Total Enrollment, e.g.,`$TOT_ENROLL`, and many other pieces of information.

```
str(colorado_universities)
```

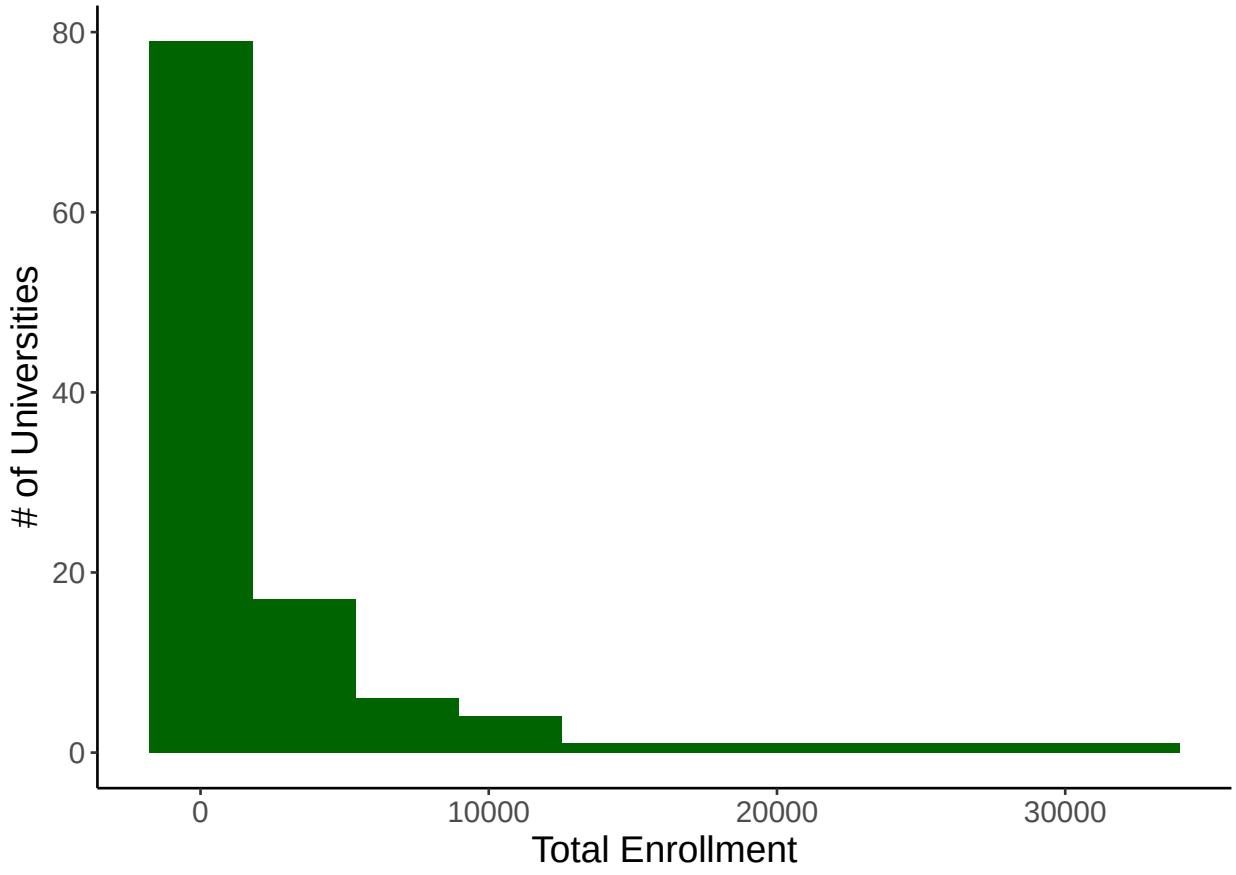
```
## 'data.frame':    112 obs. of  38 variables:
##  $ IPEDSID   : int  126915 127219 128391 127185 126164 128188 127556 128036 126748 126711 ...
##  $ NAME      : chr  "Delta Montrose Technical College" "Glenwood Beauty Academy" "Western State Colle
##  $ LADDR     : chr  "1765 US Hwy 50" "51241 Hwy 6 and 24 Ste 1" "600 N Adams" "1000 Rim Drive" ...
##  $ LADDR2    : logi  NA NA NA NA NA NA ...
##  $ LCITY     : chr  "Delta" "Glenwood Springs" "Gunnison" "Durango" ...
##  $ LSTATE    : chr  "CO" "CO" "CO" "CO" ...
##  $ LZIP      : int  81416 81601 81231 81301 81504 81506 81501 81328 81648 81601 ...
##  $ LZIP4     : int  NA NA NA 3999 NA NA 3122 9196 3598 233 ...
##  $ COUNTRY   : chr  "US" "US" "US" "US" ...
##  $ PHONE     : chr  "9708747671" "9709450485" "9709430120" "9702477010" ...
##  $ STFIPS    : int  8 8 8 8 8 8 8 8 8 8 ...
##  $ COFIPS    : int  8029 8045 8051 8067 8077 8077 8077 8083 8103 8045 ...
##  $ LAT       : num  38.7 39.6 38.5 37.3 39.1 ...
##  $ LON       : num  -108 -107 -107 -108 -109 ...
##  $ SECTOR    : int  7 6 1 1 6 6 1 7 4 4 ...
##  $ LEVEL     : int  3 2 1 1 2 2 1 3 2 2 ...
##  $ INST_TYPE : int  1 3 1 1 3 3 1 1 1 1 ...
##  $ NAICS_CODE: int  611519 611511 611310 611310 611519 611210 611310 611519 611210 611210 ...
##  $ NAICS_DESC: chr  "Other Technical and Trade Schools" "Cosmetology and Barber Schools" "Colleges, U
```

```
##  $ HI_OFFER  : int  0 0 30 30 0 40 20 0 40 40 ...
##  $ DEG_GRANT : int  2 2 1 1 2 1 1 2 1 1 ...
##  $ LOCALE    : int  42 33 33 32 13 13 13 42 43 33 ...
##  $ STATUS    : chr  "A" "A" "A" "A" ...
##  $ CLOSE_DATE: chr  "-2" "-2" "-2" "-2" ...
##  $ MERGE_ID  : int  -2 -2 -2 -2 -2 -2 -2 -2 -2 -2 ...
##  $ ALIAS     : chr  NA NA NA "FLC" ...
##  $ SIZE_SET  : int  -3 -3 10 13 -3 1 12 -3 2 3 ...
##  $ INST_SIZE : int  1 1 2 2 1 1 3 1 2 3 ...
##  $ PT_ENROLL : int  1500 18 273 336 0 0 1772 233 771 3730 ...
##  $ FT_ENROLL : int  115 34 1850 3410 74 770 4384 156 494 1449 ...
##  $ TOT_ENROLL: int  1615 52 2123 3746 74 770 6156 389 1265 5179 ...
##  $ HOUSING   : int  2 2 1 1 2 2 1 2 1 1 ...
##  $ DORM_CAP  : int  0 0 1130 1391 0 0 1290 0 320 620 ...
##  $ FTE       : int  49 9 301 598 11 84 642 69 144 3588 ...
##  $ NCES_URL  : chr  "http://nces.ed.gov/GLOBALLOCATOR/" "http://nces.ed.gov/GLOBALLOCATOR/" "http://
##  $ SCHOOL_URL: chr  "www.dmtc.edu" NA "www.western.edu" "www.fortlewis.edu" ...
##  $ SHELTER_ID: int  NA NA NA NA NA NA NA NA NA NA ...
##  $ LAST_UPDAT: chr  "9/1/10" "1/15/10" "1/15/10" "9/1/10" ...
```

You can explore some simple statistics on total enrollment and the other metrics if you follow this link: https://www.sciencebase.gov/catalog/item/4f4e4acee4b07f02db67fb39.

Let's make a plot of the distribution of the *Total Enrollment* across Colorado institutions of higher education.

```
ggplot(data=colorado_universities)+
  geom_histogram(mapping=aes(TOT_ENROLL), bins = 10, fill="darkgreen")+
  theme_classic()+
  labs(y = "# of Universities", x = "Total Enrollment")+
  theme(text = element_text(size=15))
```



We can explore the `range` of enrollment and the determine the `mean`.

```
range(colorado_universities$TOT_ENROLL)
```

```
## [1]     0 32191
```

```
mean(colorado_universities$TOT_ENROLL)
```

```
## [1] 2892.384
```

Let's say we need to share our subset data frame with a collaborator. Currently, our Colorado-specific dataset only lives within R. We need to output the data frame. A common format is a *.csv*, which stands for comma separated values.

```
write.csv(colorado_universities, "colorado_universities.csv", row.names=F)

#if you're working with large datasets, it's good to use fwrite and fread from the data.table package
```

Let's make a shapefile of the locations using the `sf` package commands

```
colorado_universities_shp = st_as_sf(colorado_universities, coords = c("LON", "LAT"), crs = "+proj=long
```

Like the previous example, our collaborator needs a shapefile of the Colorado institutions. Let's fill that request by outputting a shapefile using the `st_write` function.

```
st_write(colorado_universities_shp, "../data/collegesuniversities.shp", driver = "ESRI Shapefile", dele
```

```
## Deleting layer ‘collegesuniversities’ using driver ‘ESRI Shapefile’
## Writing layer ‘collegesuniversities’ to data source
##   ‘../data/collegesuniversities.shp’ using driver ‘ESRI Shapefile’
## Writing 112 features with 36 fields and geometry type Point.
```

Now, let's read the shapefile of the Colorado institutions that you just created.

```
colorado_universities_shp = st_read("../data/collegesuniversities.shp")
```

```
## Reading layer ‘collegesuniversities’ from data source
##   ‘C:\Data\ECOL620\ecol620_Lab1\data\collegesuniversities.shp’
##   using driver ‘ESRI Shapefile’
## Simple feature collection with 112 features and 36 fields
## Geometry type: POINT
## Dimension:     XY
## Bounding box:  xmin: -108.7903 ymin: 37.17294 xmax: -102.6156 ymax: 40.63688
## Geodetic CRS:  GCS_unknown
```

```
colorado_universities_shp = st_transform(colorado_universities_shp, "epsg:4326") #this command allows y
```

Read in a Colorado county shapefile

```
co_counties = st_read("../data/colorado_county_boundaries.shp") #this might take a couple of seconds to
```

```
## Reading layer ‘colorado_county_boundaries’ from data source
##   ‘C:\Data\ECOL620\ecol620_Lab1\data\colorado_county_boundaries.shp’
##   using driver ‘ESRI Shapefile’
## Simple feature collection with 64 features and 9 fields
## Geometry type: MULTIPOLYGON
## Dimension:     XY
## Bounding box:  xmin: -109.0603 ymin: 36.99242 xmax: -102.0409 ymax: 41.00344
## Geodetic CRS:  NAD83
```

```
co_counties = st_transform(co_counties, "epsg:4326") #this will transform the coordinates to WGS84
```

Let's plot just the schools with enrollment over 1000 students. First, we will use the `subset` function. Because we are now working with a shapefile, or class `sp`, the `dplyr::filter` function will not work. It needs to be a dataframe. However, to plot these data using `ggplot`, the shapefile will need to converted to a dataframe.

```
colorado_universities_shp = subset(colorado_universities_shp, TOT_ENROLL>1000)
```

Let's determine the range and save the values. We will use these stored values below for plotting

```
min_enroll=min(colorado_universities_shp$TOT_ENROLL)
max_enroll=max(colorado_universities_shp$TOT_ENROLL)
```

Use ggplot to map a state map

```
CO_MAP_UNI=ggplot() +
  geom_sf(data = co_counties,  fill = NA, color ="black", lwd=.1) +
  geom_sf(data = colorado_universities_shp, aes(size=TOT_ENROLL, colour=TOT_ENROLL), alpha=.9) +
  theme_bw()+
  theme(panel.grid.minor=element_blank(),panel.grid.major=element_blank())+
  theme( panel.border=element_blank())+
  scale_colour_viridis_c(limits=c(min_enroll, max_enroll), breaks=seq(5000, 30000, by=5000), name = "To
  guides(color= guide_legend(), size=guide_legend())+
  scale_size_continuous(limits=c(min_enroll, max_enroll), breaks=seq(5000, 30000, by=5000),name = "Tota
  labs(y = "Latitude", x = "Longitude")
CO_MAP_UNI
```