Demo

Cheng Ren and revised based on Evan Muzzall's markdown file

2/5/2020

This is an .HTML file generated using Rmarkdown (.Rmd). Click these hyperlinks and those throughout this document to learn more.

Be sure to install both R and RStudio for these exercises:
Click here to download R(Current Version 3.6.2)
Click here to download RStudio

learning objectives:

- Navigating RStudio
- Data types and type coercion
- Data structures (vectors, factors, and data frames)
- Changing column names in a data frame
- The `$` operator
- Load the GSS data subset .CSV file
- Summary/Crosstable
- Graph data
- Resources
- Appendix

Navigating RStudio

RStudio is a dynamic graphical user interface (GUI) that wraps around the R language. You will notice that it is separated into various window panes. RStudio makes it easy to save your code in a **script.**

To change the color of your background, click **Tools => Global Options => Appearance => Editor theme.** You will probably also want to "soft wrap" your code so that when you type it "wraps" to a new line - you can do this by clicking **Tools => Global Options => Code => check box "Soft wrap R source files.**

To open a new script file, click **File => New File => R Script.** By default this will appear in the upper-left pane. Here you can write code to save variables, import datasets, write comments to yourself, and summarize, visualize, and test data. By default, your output will appear in the lower-left pane called the **console.**

Variables you define are automatically saved into your **global environment** - this is physical space tasked with storing the variables you define. This is located in the upper-right pane.

The lower-right pane contains miscellaneous output. Here you will find your file structure, help pages, plots, and a package installation interface.

Saving variables

To save a variable in RStudio, you need three things:
1) unique name - if your variable name is the same as a previously defined one, it will overwrite it!
2) the assignment operator <-
3) a definition/statements to be evaluated - the code you want to run

The syntax looks like this: `unique_name <- definition`, and you read this like "the object unique_name is defined as 'definition' ".

Run a line of code by pressing **command + return (Mac)** or **Ctrl + return (Windows).** Try it!

```
my_name <- "Type your name here!" # this is character/text/string data

X <- 5 # R defaults to numeric type, even for whole numbers!
```

What happened here? Notice that in the global environment (the upper right hand pane), two variables are now defined: "name" and "X".

Data types and type coercion

R can store many different data types.
The most common ones are:
1) **numeric:** decimals and fractions
2) **integer:** positive and negative whole numbers, including zero
3) **character:** aka text or string data
4) **logical:** `TRUE` or `FALSE`
5) **factor:** categorical groupings of integer or character data

You can investigate data types using the `class()` function.

```
class(my_name)
```

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

```
class(X)
```

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

Data type coercion

Notice that "X" is defined as numeric type even though it is an integer! We can change ("coerce") its type using the `as.integer()` function:

```
X_char <- as.character(X)
class(X_char)
```

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

Data structures

Vectors

Vectors are central to R and are defined as organized groupings of the same type of data. This means that their position matters!

So far, we have only been saving one piece of data into our variables. We can use the `c()` funciton to save more than one piece of data, as long as it is all of the same type! Let's try it:

```
fruit <- c("Apple", "Apple", "Apple", "Apple",
           "Orange", "Orange", "Orange",
           "Strawberry", "Strawberry", "Strawberry")
fruit
```

```
##  [1] "Apple"      "Apple"      "Apple"      "Apple"      "Orange"
##  [6] "Orange"     "Orange"     "Strawberry" "Strawberry" "Strawberry"
```

```
class(fruit)
```

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

Note that we never refer to a vector merely as a vector! We always refer to it by its type: character vector, integer vector, logical vector, and so on.

See these examples for generating sequences of numbers.

Factors

Factors are categorical groupings of data that can be used to make comparisons between other data.

For example, what if a variable named "price" contains the cost of each of these fruits and we want to investigate differences in cost between fruits? Let's store this information in a character vector:

```
price <- c(0.79, 0.79, 0.79, 0.79, # price of apples
           0.99, 0.99, 0.99, # oranges
           0.59, 0.59, 0.59) # strawberries
price
```

```
##  [1] 0.79 0.79 0.79 0.79 0.99 0.99 0.99 0.59 0.59 0.59
```

```
class(price)
```

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

Unfortunately, it is difficult to make comparisons of the prices if we keep the "fruit" vector as character type. However, if we coerce it to factor type, R knows that it should group the like fruits by name.

Similar to type coercion above, we can convert "fruit" from character to factor type using the `as.factor()` function:

```
fruit_fac <- as.factor(fruit)
fruit_fac
```

```
##  [1] Apple      Apple      Apple      Apple      Orange     Orange
##  [7] Orange     Strawberry Strawberry Strawberry
## Levels: Apple Orange Strawberry
```

```
class(fruit_fac)
```

```
## [1] "factor"
```

What changed? We can then view the factor levels via the `levels()` function:

```
levels(fruit_fac)
```

```
## [1] "Apple"      "Orange"     "Strawberry"
```

Get more help with factors by clicking the link!

Data frames

A major strength of R is that it can store data in a "data frame". This allows you to combine vectors of equal length into a dataset that can be operated upon. The best way to think of this is to think of a basic spreadsheet!

```
fruit_prices <- data.frame(fruit, price)
fruit_prices
```

```
##         fruit price
## 1       Apple  0.79
## 2       Apple  0.79
## 3       Apple  0.79
## 4       Apple  0.79
## 5      Orange  0.99
## 6      Orange  0.99
## 7      Orange  0.99
## 8  Strawberry  0.59
## 9  Strawberry  0.59
## 10 Strawberry  0.59
```

```
class(fruit_prices)
```

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

Load the GSS data subset from file with `read.csv()`

Remember to set your working directory first by clicking **Session => Set Working Directory => Choose Directory** and choose the "csv file" folder.

```r
#To check Whether you are in the Working directory
setwd('/Users/cheng/Downloads')
getwd()
```

```
## [1] "C:/Users/cheng/Downloads"
```

```r
#if you have readly in the working directory, you can just read the file
gss <- read.csv("gss_sub.csv", header = T)
?read.csv()
```

```
## starting httpd help server ... done
```

```r
# Or, you can just type the absolute file path into `read.csv()`
gss <- read.csv("/Users/cheng/Downloads/gss_sub.csv", header = T)
#Please be aware it is "/" not "\", if you copy from the file, the address is "\" like"\OneDrive\Deskto
```

Inspect the data frame

```r
str(gss)
```

```
## 'data.frame':    62466 obs. of  39 variables:
##  $ YEAR    : int  1972 1972 1972 1972 1972 1972 1972 1972 1972 1972 ...
##  $ AGE     : Factor w/ 72 levels "18","19","20",..: 6 53 31 10 44 9 11 10 4 13 ...
##  $ SEX     : Factor w/ 2 levels "FEMALE","MALE": 1 2 1 1 1 2 2 2 1 1 ...
##  $ RACE    : Factor w/ 3 levels "BLACK","OTHER",..: 3 3 3 3 3 3 3 3 1 1 ...
##  $ ETHNIC  : Factor w/ 42 levels "AFRICA","AMERICAN INDIAN",..: NA 18 19 4 40 10 18 NA NA NA ...
##  $ HEALTH  : Factor w/ 4 levels "EXCELLENT","FAIR",..: 3 2 1 3 3 3 1 3 1 2 ...
##  $ EDUC    : int  16 10 12 17 12 14 13 16 12 12 ...
##  $ DEGREE  : Factor w/ 5 levels "BACHELOR","GRADUATE",..: 1 5 3 1 3 3 3 1 3 3 ...
##  $ PAEDUC  : int  10 8 8 16 8 18 16 16 12 10 ...
##  $ MAEDUC  : int  NA 8 8 12 8 19 12 14 12 7 ...
##  $ WRKSTAT : Factor w/ 8 levels "KEEPING HOUSE",..: 7 3 8 7 1 7 7 7 8 7 ...
##  $ DWELLING: Factor w/ 10 levels "2 UNITS-ONE ABOVE",..: NA NA NA NA NA NA NA NA NA NA ...
##  $ MARITAL : Factor w/ 5 levels "DIVORCED","MARRIED",..: 3 2 2 2 2 3 1 3 3 2 ...
##  $ CHILDS  : int  0 5 4 0 2 0 2 0 2 4 ...
##  $ FAMILY16: Factor w/ 9 levels "FATHER","FATHER & STPMOTHER",..: 1 4 7 7 7 7 7 7 4 7 ...
##  $ BORN    : Factor w/ 2 levels "NO","YES": NA NA NA NA NA NA NA NA NA NA ...
##  $ LOCKEDUP: Factor w/ 2 levels "No","Yes": NA NA NA NA NA NA NA NA NA NA ...
##  $ CONVICTD: Factor w/ 2 levels "No","Yes": NA NA NA NA NA NA NA NA NA NA ...
##  $ PARTYID : Factor w/ 8 levels "IND,NEAR DEM",..: 1 4 3 4 7 1 1 1 7 7 ...
##  $ RELIG   : Factor w/ 13 levels "BUDDHISM","CATHOLIC",..: 6 2 13 11 13 13 2 6 13 13 ...
##  $ HAPPY   : Factor w/ 3 levels "NOT TOO HAPPY",..: 1 1 2 1 2 2 1 1 2 2 ...
##  $ SIBS    : int  3 4 5 5 2 1 7 1 2 7 ...
##  $ HOMPOP  : int  1 2 4 2 2 1 1 1 3 7 ...
##  $ CONINC  : num  25926 33333 33333 41667 69444 ...
##  $ CONRINC : num  NA NA NA NA NA NA NA NA NA NA ...
##  $ EARNRS  : int  1 0 2 2 1 1 1 1 1 2 ...
##  $ EMPYEARS: num  NA NA NA NA NA NA NA NA NA NA ...
##  $ TOTALNUM: Factor w/ 7 levels "1-9","1,000-1,999",..: NA NA NA NA NA NA NA NA NA NA ...
##  $ HRSRELAX: int  NA NA NA NA NA NA NA NA NA NA ...
##  $ TVHOURS : int  NA NA NA NA NA NA NA NA NA NA ...
```

```
##  $ RADIOHRS: int  NA NA NA NA NA NA NA NA NA NA ...
##  $ EMAILHR : int  NA NA NA NA NA NA NA NA NA NA ...
##  $ WWWHR   : int  NA NA NA NA NA NA NA NA NA NA ...
##  $ NUMFREND: Factor w/ 26 levels "1","10","12",..: NA NA NA NA NA NA NA NA NA NA ...
##  $ WEALTH  : Factor w/ 15 levels "$1 million to $2 million",..: NA NA NA NA NA NA NA NA NA NA ...
##  $ POLVIEWS: Factor w/ 7 levels "CONSERVATIVE",..: NA NA NA NA NA NA NA NA NA NA ...
##  $ DRAFT   : Factor w/ 2 levels "DRAFT","VOLUNTEERS": NA NA NA NA NA NA NA NA NA NA ...
##  $ CHLDIDEL: Factor w/ 9 levels "0","1","2","3",..: 3 4 3 3 3 4 5 3 3 5 ...
##  $ WTSSALL : num  0.445 0.889 0.889 0.889 0.889 ...
```

```
dim(gss)
```

```
## [1] 62466    39
```

```
head(gss)
```

```
##   YEAR AGE    SEX  RACE          ETHNIC    HEALTH EDUC         DEGREE
## 1 1972  23 FEMALE WHITE            <NA>      GOOD   16       BACHELOR
## 2 1972  70   MALE WHITE         IRELAND      FAIR   10 LT HIGH SCHOOL
## 3 1972  48 FEMALE WHITE           ITALY EXCELLENT   12    HIGH SCHOOL
## 4 1972  27 FEMALE WHITE          ARABIC      GOOD   17       BACHELOR
## 5 1972  61 FEMALE WHITE     SWITZERLAND      GOOD   12    HIGH SCHOOL
## 6 1972  26   MALE WHITE ENGLAND & WALES      GOOD   14    HIGH SCHOOL
##   PAEDUC MAEDUC          WRKSTAT DWELLING       MARITAL CHILDS
## 1     10     NA WORKING FULLTIME     <NA> NEVER MARRIED      0
## 2      8      8          RETIRED     <NA>       MARRIED      5
## 3      8      8 WORKING PARTTIME     <NA>       MARRIED      4
## 4     16     12 WORKING FULLTIME     <NA>       MARRIED      0
## 5      8      8    KEEPING HOUSE     <NA>       MARRIED      2
## 6     18     19 WORKING FULLTIME     <NA> NEVER MARRIED      0
##            FAMILY16 BORN LOCKEDUP CONVICTD          PARTYID      RELIG
## 1            FATHER <NA>     <NA>     <NA>     IND,NEAR DEM     JEWISH
## 2 M AND F RELATIVES <NA>     <NA>     <NA> NOT STR DEMOCRAT   CATHOLIC
## 3   MOTHER & FATHER <NA>     <NA>     <NA>      INDEPENDENT PROTESTANT
## 4   MOTHER & FATHER <NA>     <NA>     <NA> NOT STR DEMOCRAT      OTHER
## 5   MOTHER & FATHER <NA>     <NA>     <NA>  STRONG DEMOCRAT PROTESTANT
## 6   MOTHER & FATHER <NA>     <NA>     <NA>     IND,NEAR DEM PROTESTANT
##           HAPPY SIBS HOMPOP CONINC CONRINC EARNRS EMPYEARS TOTALNUM
## 1 NOT TOO HAPPY    3      1  25926      NA      1       NA     <NA>
## 2 NOT TOO HAPPY    4      2  33333      NA      0       NA     <NA>
## 3  PRETTY HAPPY    5      4  33333      NA      2       NA     <NA>
## 4 NOT TOO HAPPY    5      2  41667      NA      2       NA     <NA>
## 5  PRETTY HAPPY    2      2  69444      NA      1       NA     <NA>
## 6  PRETTY HAPPY    1      1  60185      NA      1       NA     <NA>
##   HRSRELAX TVHOURS RADIOHRS EMAILHR WWWHR NUMFREND WEALTH POLVIEWS DRAFT
## 1       NA      NA       NA      NA    NA     <NA>   <NA>     <NA>  <NA>
## 2       NA      NA       NA      NA    NA     <NA>   <NA>     <NA>  <NA>
## 3       NA      NA       NA      NA    NA     <NA>   <NA>     <NA>  <NA>
## 4       NA      NA       NA      NA    NA     <NA>   <NA>     <NA>  <NA>
## 5       NA      NA       NA      NA    NA     <NA>   <NA>     <NA>  <NA>
## 6       NA      NA       NA      NA    NA     <NA>   <NA>     <NA>  <NA>
##   CHLDIDEL WTSSALL
## 1        2  0.4446
```

```
## 2        3  0.8893
## 3        2  0.8893
## 4        2  0.8893
## 5        2  0.8893
## 6        3  0.4446
```

Identifying missing data (NA)

Missing data are especially problematic in all data science endeavors as identifying missing data is a fundamental step in data preparation. In R, you can identify missing data with the `is.na()` function. Let's just look at the "HEALTH" column. R recognizes missing data only as `NA`.

`TRUE` means data are missing; `FALSE` means data are present.

```
is.na(gss$HEALTH)
```

If we want to see sum how many cells have missing data within a vector, we can wrap this in the `sum()` or `table()` function:

```
sum(is.na(gss$HEALTH)) # the HEALTH column has 16,445 cells with missing data
```

```
## [1] 16445
```

```
table(is.na(gss$HEALTH))# what is the difference?
```

```
##
## FALSE  TRUE
## 46021 16445
```

If we want see the missing situation of each column

```
colSums(is.na(gss))
```

```
##     YEAR      AGE      SEX     RACE   ETHNIC   HEALTH     EDUC   DEGREE
##        0      221        0        0    13310    16445      174      173
##   PAEDUC   MAEDUC  WRKSTAT DWELLING  MARITAL   CHILDS FAMILY16     BORN
##    18316    10685       19    12902       25      197     1546     9262
## LOCKEDUP CONVICTD  PARTYID    RELIG    HAPPY     SIBS   HOMPOP   CONINC
##    60718    60718      385      269     4756     1691        6     6324
##  CONRINC   EARNRS EMPYEARS TOTALNUM HRSRELAX  TVHOURS RADIOHRS  EMAILHR
##    25942      558    61563    61897    56620    25059    57962    53371
##    WWWHR NUMFREND   WEALTH POLVIEWS    DRAFT CHLDIDEL  WTSSALL
##    51347    61626    60311     9385    56339    23858        0
```

```
round(colSums(is.na(gss))/nrow(gss)*100,2)
```

```
##     YEAR      AGE      SEX     RACE   ETHNIC   HEALTH     EDUC   DEGREE
##     0.00     0.35     0.00     0.00    21.31    26.33     0.28     0.28
##   PAEDUC   MAEDUC  WRKSTAT DWELLING  MARITAL   CHILDS FAMILY16     BORN
##    29.32    17.11     0.03    20.65     0.04     0.32     2.47    14.83
```

```
## LOCKEDUP CONVICTD  PARTYID    RELIG    HAPPY     SIBS   HOMPOP   CONINC
##    97.20    97.20     0.62     0.43     7.61     2.71     0.01    10.12
##  CONRINC   EARNRS EMPYEARS TOTALNUM HRSRELAX  TVHOURS RADIOHRS  EMAILHR
##    41.53     0.89    98.55    99.09    90.64    40.12    92.79    85.44
##    WWWHR NUMFREND   WEALTH POLVIEWS    DRAFT CHLDIDEL  WTSSALL
##    82.20    98.66    96.55    15.02    90.19    38.19     0.00
```

Summarizing your data

Summarizing data are a fundamental step in the analytical process, often even before visualization. You want to quickly produce some descriptive statistics since they can inform what you should consider choosing for your visualizations.

The `summary()` and `table()` functions are great places to start:

```
table(gss$HEALTH) # this will compute frequencies for factor variables
```

```
##
## EXCELLENT      FAIR      GOOD      POOR
##     13827      8768     20788      2638
```

```
summary(gss$CONINC) # this will produce six number summaries for numeric variables
```

```
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.    NA's
##     363   18241   35602   44839   59569  180386    6324
```

```
aggregate(gss$CONINC, by = list(gss$HEALTH),summary)
```

```
##     Group.1    x.Min. x.1st Qu.  x.Median    x.Mean x.3rd Qu.    x.Max.
## 1 EXCELLENT    363.00  25926.00  44643.00  53937.72  72223.00 180386.00
## 2      FAIR    363.00  12071.00  24258.00  32934.38  43612.00 180386.00
## 3      GOOD    363.00  20062.00  37461.00  45620.58  59895.00 180386.00
## 4      POOR    363.00   7693.00  14761.00  23105.93  29574.00 180386.00
##      x.NA's
## 1   1193.00
## 2   1021.00
## 3   1999.00
## 4    365.00
```

```
aggregate(gss$CONINC, by = list(gss$HEALTH),mean)
```

```
##     Group.1  x
## 1 EXCELLENT NA
## 2      FAIR NA
## 3      GOOD NA
## 4      POOR NA
```

```
#Please think for a while why it returns NAs
aggregate(gss$CONINC, by = list(gss$HEALTH),mean,na.rm=TRUE)
```

```
##     Group.1        x
## 1 EXCELLENT 53937.72
## 2      FAIR 32934.38
## 3      GOOD 45620.58
## 4      POOR 23105.93
```

Cross Table supoprt file

```
m<-table(gss$FAMILY16,gss$DEGREE)
m
```

```
##
##                       BACHELOR GRADUATE HIGH SCHOOL JUNIOR COLLEGE
##   FATHER                   117       54         736             69
##   FATHER & STPMOTHER       140       62         603             72
##   FEMALE RELATIVE           52       25         479             27
##   M AND F RELATIVES         90       50         696             67
##   MALE RELATIVE             12        5          89              8
##   MOTHER                   883      357        4196            466
##   MOTHER & FATHER         7046     3618       22020           2375
##   MOTHER & STPFATHER       325      108        1722            194
##   OTHER                    105       58         681             86
##
##                       LT HIGH SCHOOL
##   FATHER                         479
##   FATHER & STPMOTHER             284
##   FEMALE RELATIVE                349
##   M AND F RELATIVES              473
##   MALE RELATIVE                   88
##   MOTHER                        1932
##   MOTHER & FATHER               8155
##   MOTHER & STPFATHER             698
##   OTHER                          608
```

```
round(prop.table(m,1),4)*100
```

```
##
##                       BACHELOR GRADUATE HIGH SCHOOL JUNIOR COLLEGE
##   FATHER                  8.04     3.71       50.58           4.74
##   FATHER & STPMOTHER     12.06     5.34       51.94           6.20
##   FEMALE RELATIVE         5.58     2.68       51.39           2.90
##   M AND F RELATIVES       6.54     3.63       50.58           4.87
##   MALE RELATIVE           5.94     2.48       44.06           3.96
##   MOTHER                 11.27     4.56       53.56           5.95
##   MOTHER & FATHER        16.30     8.37       50.96           5.50
##   MOTHER & STPFATHER     10.67     3.54       56.51           6.37
##   OTHER                   6.83     3.77       44.28           5.59
##
##                       LT HIGH SCHOOL
##   FATHER                         32.92
##   FATHER & STPMOTHER             24.46
##   FEMALE RELATIVE                37.45
##   M AND F RELATIVES              34.38
```

```
##   MALE RELATIVE               43.56
##   MOTHER                      24.66
##   MOTHER & FATHER             18.87
##   MOTHER & STPFATHER          22.91
##   OTHER                       39.53
```

```
round(prop.table(m,2),4)*100
```

```
##
##                       BACHELOR GRADUATE HIGH SCHOOL JUNIOR COLLEGE
##   FATHER                  1.33     1.25        2.36           2.05
##   FATHER & STPMOTHER      1.60     1.43        1.93           2.14
##   FEMALE RELATIVE         0.59     0.58        1.53           0.80
##   M AND F RELATIVES       1.03     1.15        2.23           1.99
##   MALE RELATIVE           0.14     0.12        0.29           0.24
##   MOTHER                 10.07     8.23       13.44          13.85
##   MOTHER & FATHER        80.34    83.42       70.53          70.60
##   MOTHER & STPFATHER      3.71     2.49        5.52           5.77
##   OTHER                   1.20     1.34        2.18           2.56
##
##                       LT HIGH SCHOOL
##   FATHER                        3.67
##   FATHER & STPMOTHER            2.17
##   FEMALE RELATIVE               2.67
##   M AND F RELATIVES             3.62
##   MALE RELATIVE                 0.67
##   MOTHER                       14.79
##   MOTHER & FATHER              62.41
##   MOTHER & STPFATHER            5.34
##   OTHER                         4.65
```

Correcltion is a statistical relationship,it commonly refers to the degree to which a pair of variables are linearly related. The range of correcltion is [-1,1] and negative means negative relationship and vice versa.

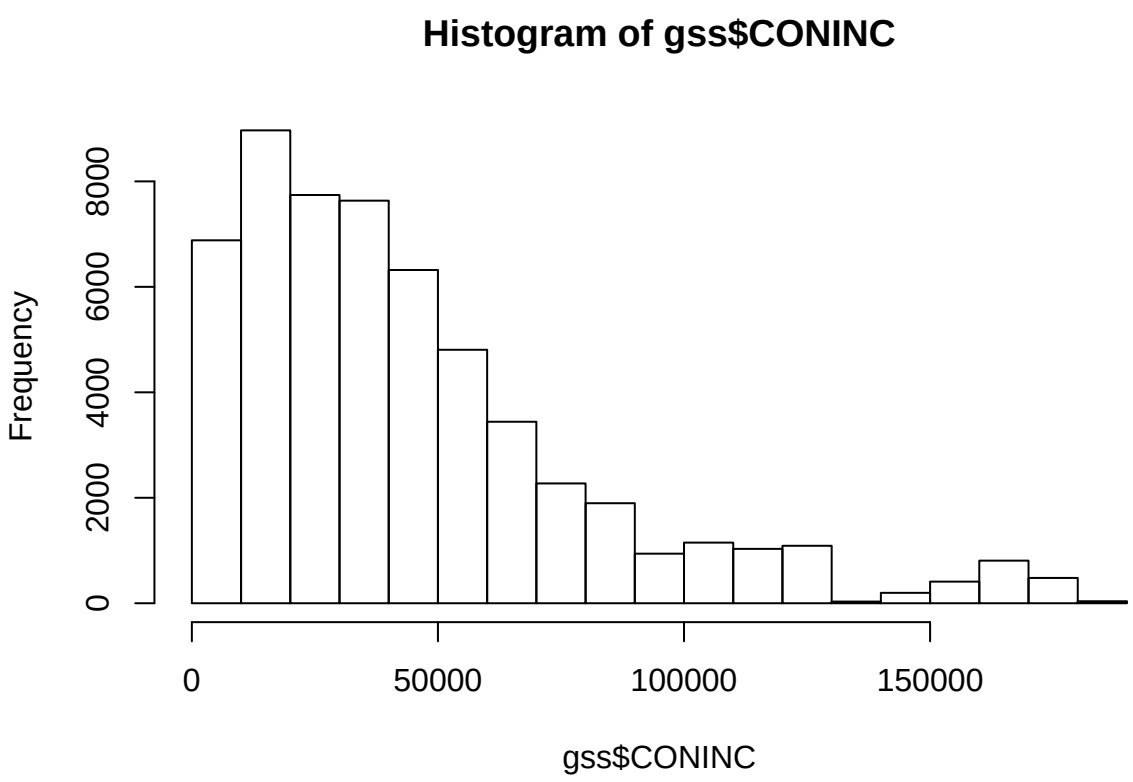
```
cor(gss$EDUC,gss$PAEDUC,use = "complete.obs")
```

```
## [1] 0.4892568
```

ggplot2 GSS visualizations (layers, data, aes, and geom_)

While base plotting functions `hist()`, `plot()`, and `boxplot()` (not covered in this lesson) are useful for fast visualizations, they get a bit clunky in more complex operations. However, check out the help pages to learn more.

```
?hist
?plot
?boxplot
hist(gss$CONINC)
```



What this means is that you create a base layer that contains information about your data, definitions for your coordinate system, and how colors and point shapes should be mapped to variables in your dataset. Then, each time you add something new (a title, new axis label, theme, etc.) it gets its own layer and is superimposed on top of the base layer and any previous layers.

See this awesome ggplot2 tutorial for clear, in-depth walkthroughs!

The package is called "ggplot2", but the actual function to create a plot is `ggplot()` and requires three things:
1) data
2) "aes"thetics (this is where you define the coordinate system, define point colors and shapes, etc)
3) "geom_"s (this is how you choose your data to be represented - points, lines, bars, ribbons, etc.)

We will walk through a histogram, boxplot, and scatterplot.

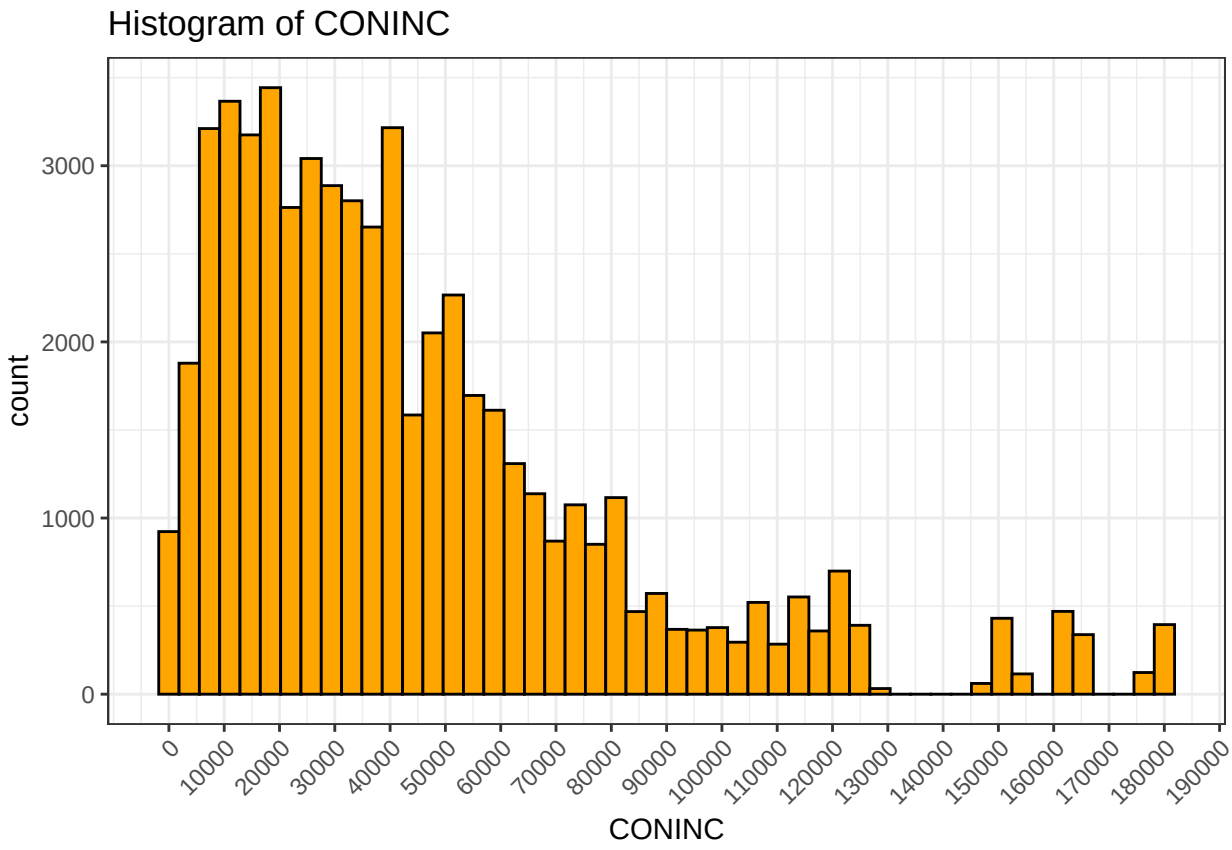
Histogram - look at the distribution of the CONINC variable

```
library(ggplot2)
```

```
## Warning: package 'ggplot2' was built under R version 3.6.1
```

```
ggplot(data = gss, aes(x = CONINC)) +
  geom_histogram(fill = "orange", color = "black", bins = 50) +
  ggtitle("Histogram of CONINC") +
  scale_x_continuous(breaks = pretty(gss$CONINC, n = 20)) +
  theme_bw() +
  theme(axis.text.x = element_text(angle = 45, hjust = 1))
```

```
## Warning: Removed 6324 rows containing non-finite values (stat_bin).
```



Saving your figures - in the "Plots" viewer (lower right pane), click Export, then either:
* Save as image,
* Save as PDF, or * Copy to Clipboard

Specify your options and then save it! Where do you think it is saved to? (hint: your working directory!)

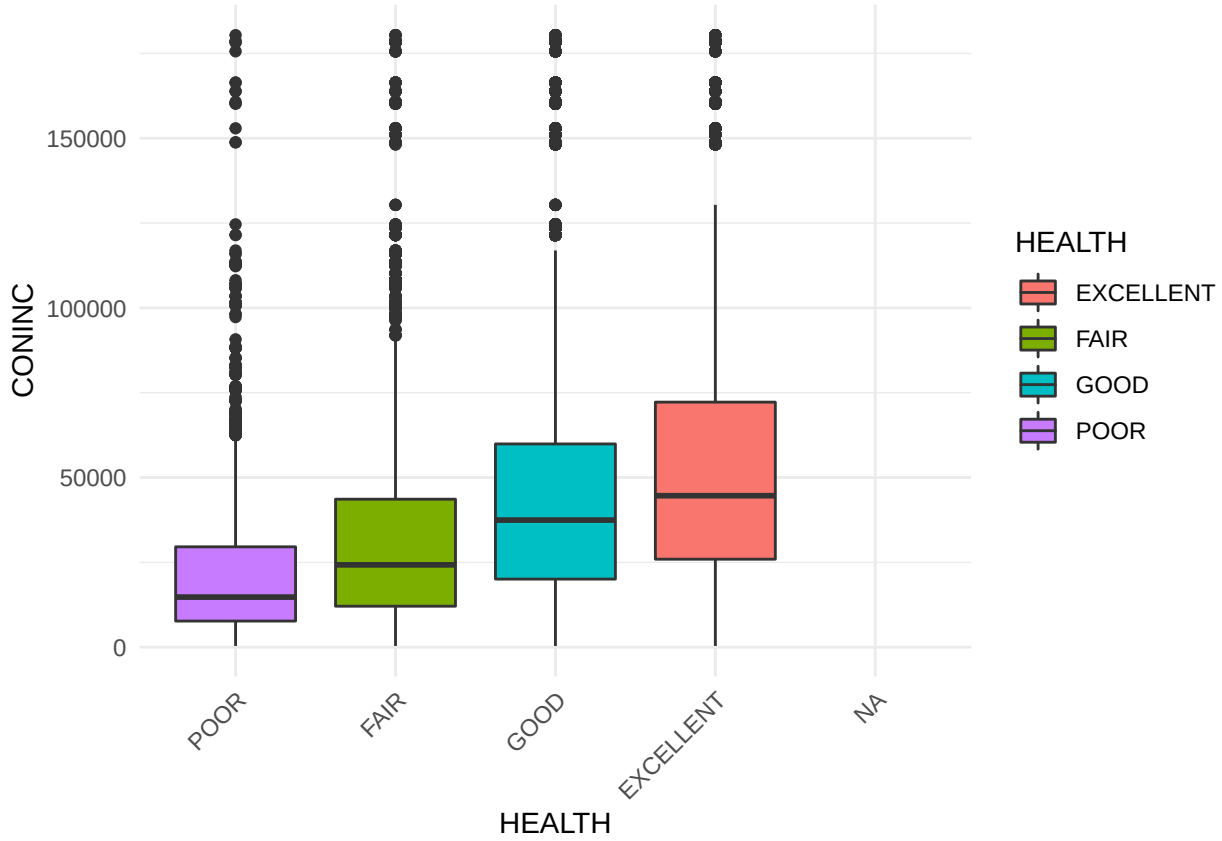
Boxplots - again, let's look at the distribution of the CONINC variable but this time as as parsed by the different levels of HEALTH.

Notice that we have included the `fill = HEALTH` argument inside `aes()`. This will color the boxes by the different levels of HEALTH (EXCELLENT, FAIR, GOOD, POOR). If this is too redundant, you can turn the legend off by deleting the two hashtags in the code below:

```
ggplot(data = gss, aes(x = HEALTH, y = CONINC, fill = HEALTH)) +
  geom_boxplot() +
  theme_minimal() +
  scale_x_discrete(limits = c("POOR", "FAIR", "GOOD", "EXCELLENT", "NA")) +
  theme(axis.text.x = element_text(angle = 45, hjust = 1)) +
  ylab("CONINC") # +
```

```
## Warning: Removed 16445 rows containing missing values (stat_boxplot).
```

```
## Warning: Removed 4578 rows containing non-finite values (stat_boxplot).
```

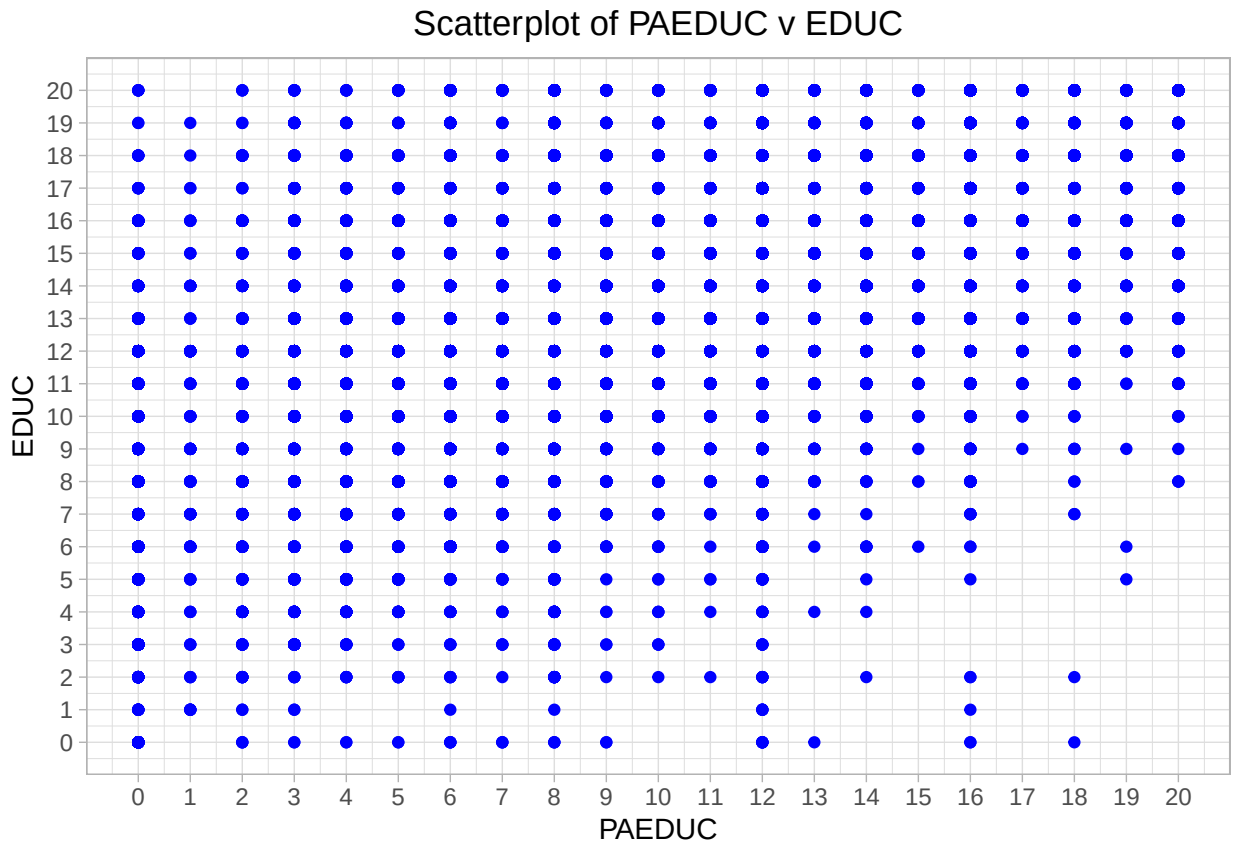



```
  # guides(fill = FALSE)
```

Scatterplots - are super fun in ggplot2. What do these different argument layers do? What might this scatterplot inform you about?

```
# Scatterplot 1 - father's education
ggplot(data = gss, aes(x = PAEDUC, y = EDUC)) +
  geom_point(color = "blue") +
  theme_light() +
  ggtitle("Scatterplot of PAEDUC v EDUC") +
  xlab("PAEDUC") +
  ylab("EDUC") +
  theme(plot.title = element_text(hjust = 0.5)) +
  scale_x_continuous(breaks = pretty(gss$PAEDUC, n = 20)) +
  scale_y_continuous(breaks = seq(0, 20, by = 1))
```

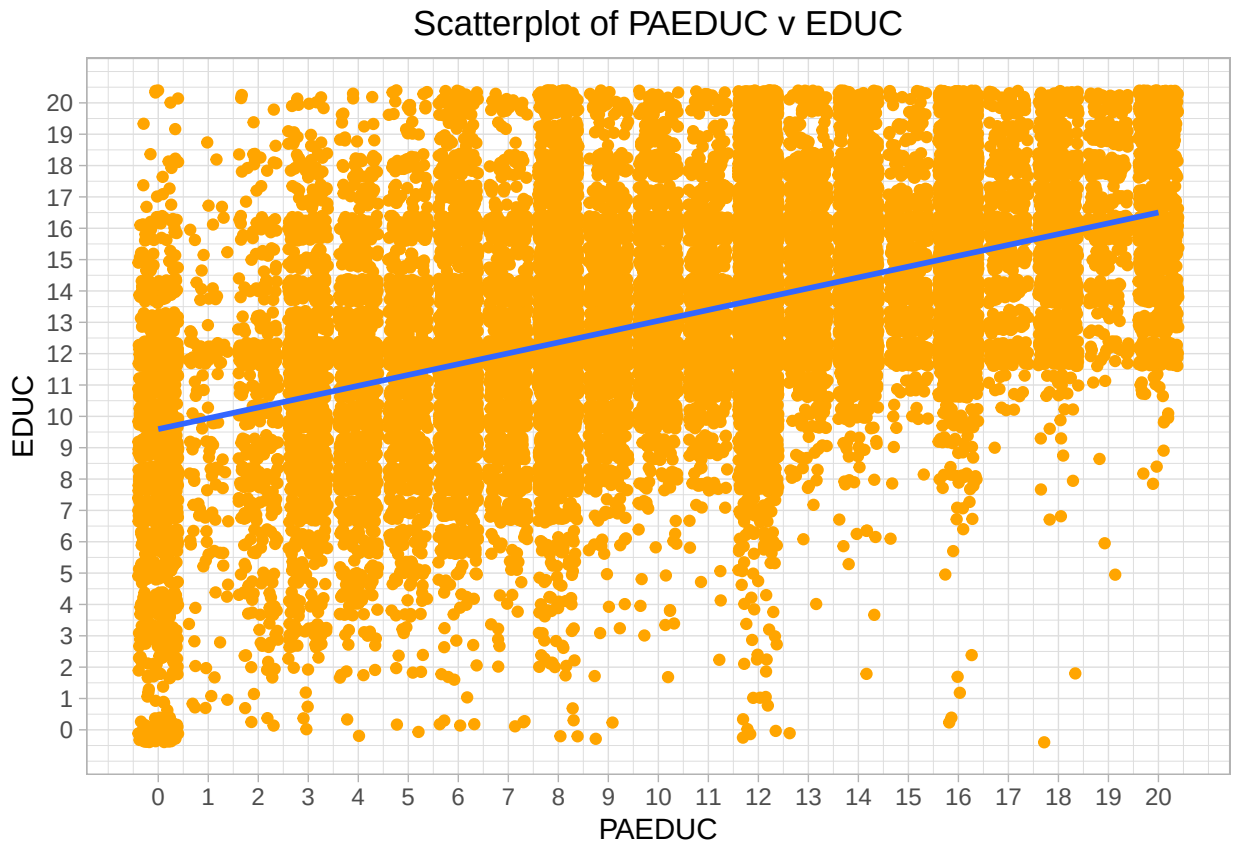
```
## Warning: Removed 18377 rows containing missing values (geom_point).
```




```
# Points work well, but we want geom_jitter() instead!
ggplot(data = gss, aes(x = PAEDUC, y = EDUC)) +
  geom_jitter(color = "orange") +
  geom_smooth(method=lm)+
  theme_light() +
  ggtitle("Scatterplot of PAEDUC v EDUC") +
  xlab("PAEDUC") +
  ylab("EDUC") +
  theme(plot.title = element_text(hjust = 0.5)) +
  scale_x_continuous(breaks = pretty(gss$PAEDUC, n = 20)) +
  scale_y_continuous(breaks = seq(0, 20, by = 1))
```

```
## Warning: Removed 18377 rows containing non-finite values (stat_smooth).
```

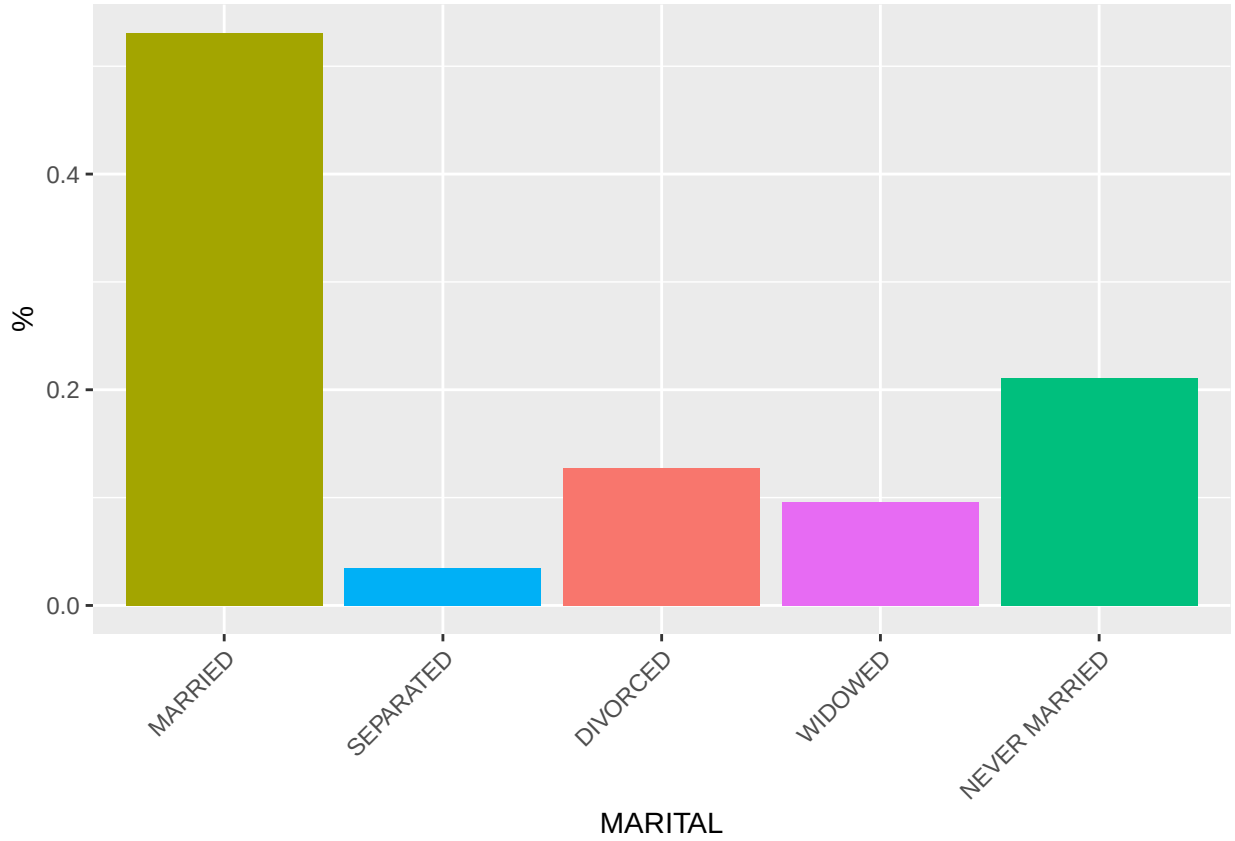
```
## Warning: Removed 18377 rows containing missing values (geom_point).
```



barplots - you can also use barplots to create histogram-like plots using proportions (%) instead of actual numbers

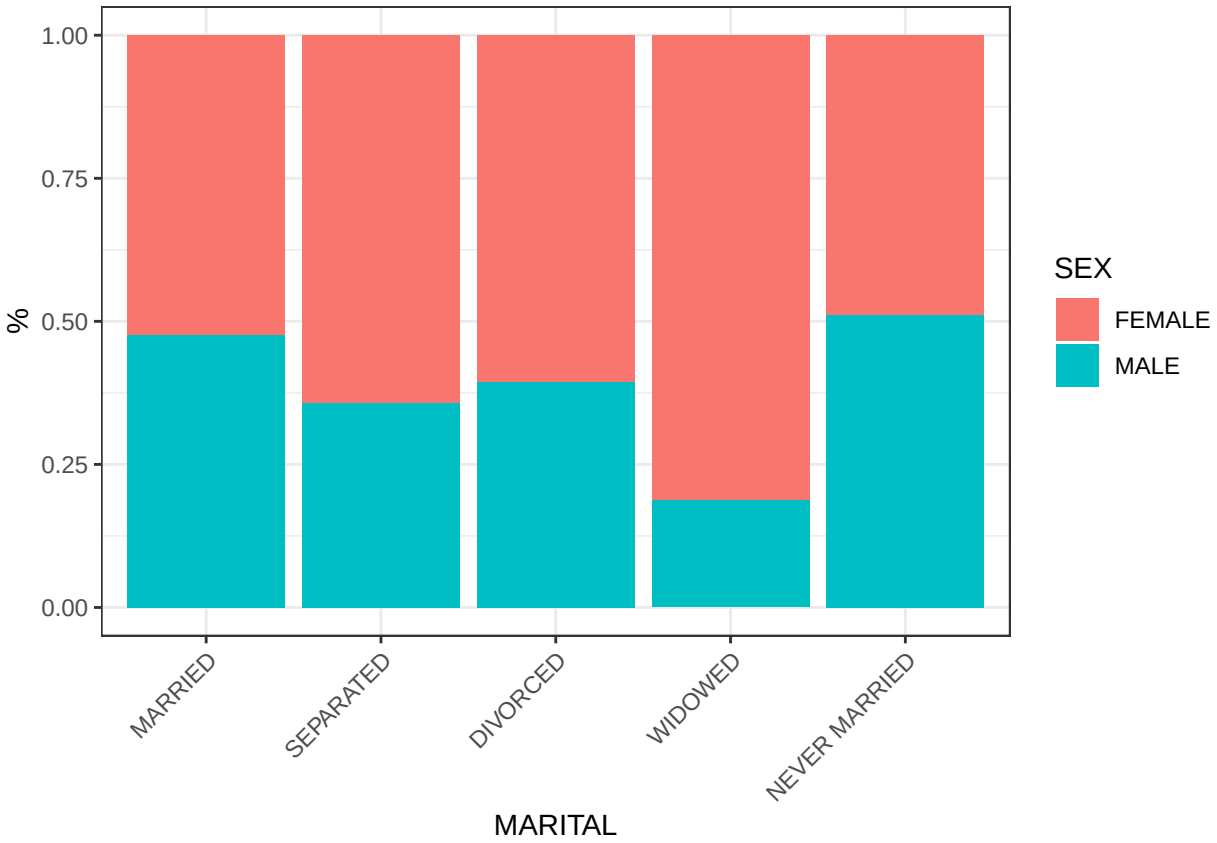
```
# by percentage (%)
ggplot(gss, aes(x = MARITAL, fill = MARITAL)) +
  geom_bar(aes(y = (..count..)/sum(..count..))) +
  scale_x_discrete(limits = c("MARRIED", "SEPARATED", "DIVORCED", "WIDOWED", "NEVER MARRIED")) +
  theme(axis.text.x = element_text(angle = 45, hjust = 1)) +
  ylab("%") +
  guides(fill = FALSE)
```

```
## Warning: Removed 25 rows containing non-finite values (stat_count).
```




```
# stacked percentage plot
ggplot(gss, aes(x = MARITAL)) +
  geom_bar(aes(fill = SEX), position = "fill") +
  theme_bw() +
  theme(axis.text.x = element_text(angle = 45, hjust = 1)) +
  ylab("%") +
  scale_x_discrete(limits = c("MARRIED", "SEPARATED", "DIVORCED", "WIDOWED", "NEVER MARRIED"))
```

```
## Warning: Removed 25 rows containing non-finite values (stat_count).
```



Appendix

The useful guideance of tidy census is https://walkerke.github.io/tidycensus/articles/basic-usage.html Before run this, you need to apply a cnesus API key. The website is https://www.census.gov/data/developers/guidance/api-user-guide.html

```
#install.packages("tidycensus")
library(tidycensus)
```

```
## Warning: package 'tidycensus' was built under R version 3.6.2
```

```
Y18 <- load_variables(2018, "acs5", cache = TRUE)
head(Y18)
```

```
## # A tibble: 6 x 3
##   name      label                          concept
##   <chr>     <chr>                          <chr>
## 1 B00001_0~ Estimate!!Total                UNWEIGHTED SAMPLE COUNT OF THE P~
## 2 B00002_0~ Estimate!!Total                UNWEIGHTED SAMPLE HOUSING UNITS
## 3 B01001_0~ Estimate!!Total                SEX BY AGE
## 4 B01001_0~ Estimate!!Total!!Male          SEX BY AGE
## 5 B01001_0~ Estimate!!Total!!Male!!Under~ SEX BY AGE
## 6 B01001_0~ Estimate!!Total!!Male!!5 to ~ SEX BY AGE
```

```r
ca <- get_acs(geography = "county",
              variables = c(medincome = "B19013_001"),
              state = "CA",
              year = 2018)
```

```
## Getting data from the 2014-2018 5-year ACS
```

```r
ca
```

```
## # A tibble: 58 x 5
##    GEOID NAME                            variable  estimate   moe
##    <chr> <chr>                           <chr>        <dbl> <dbl>
##  1 06001 Alameda County, California      medincome    92574  1023
##  2 06003 Alpine County, California       medincome    64688 12933
##  3 06005 Amador County, California       medincome    61198  3241
##  4 06007 Butte County, California        medincome    48443  1477
##  5 06009 Calaveras County, California    medincome    58151  3310
##  6 06011 Colusa County, California       medincome    56704  4501
##  7 06013 Contra Costa County, California medincome    93712   926
##  8 06015 Del Norte County, California    medincome    45258  5035
##  9 06017 El Dorado County, California    medincome    80582  2117
## 10 06019 Fresno County, California       medincome    51261   808
## # ... with 48 more rows
```