Data Summarization

Data Summarization

- Basic statistical summarization
  - `mean(x)`: takes the mean of x
  - `sd(x)`: takes the standard deviation of x
  - `median(x)`: takes the median of x
  - `quantile(x)`: displays sample quantiles of x. Default is min, IQR, max
  - `range(x)`: displays the range. Same as `c(min(x), max(x))`
  - `sum(x)`: sum of x
  - `max(x)`: maximum value in x
  - `min(x)`: minimum value in x
  - **all have a `na.rm` for missing data**
- Transformations
  - `log` - log (base `e`) transformation
  - `log10` - log base 10 transform
  - `sqrt` - square root

Statistical summarization

The vector getting summarized goes inside the parentheses:

```
x <- c(1, 5, 7, 4, 2, 8)
mean(x)
```

```
[1] 4.5
```

```
range(x)
```

```
[1] 1 8
```

```
sum(x)
```

```
[1] 27
```

Statistical summarization

Note that many of these functions have additional inputs regarding missing data, typically requiring the `na.rm` argument ("remove NAs").

```
x <- c(1, 5, 7, 4, 2, 8, NA)
mean(x)
```

```
[1] NA
```

```
mean(x, na.rm = TRUE)
```

```
[1] 4.5
```

```
quantile(x)
```

```
Error in quantile.default(x): missing values and NaN's not allowed if 'na.rm' is FALSE
```

```
quantile(x, na.rm = TRUE)
```

```
  0%  25%  50%  75% 100% 
 1.0  2.5  4.5  6.5  8.0 
```

Statistical summarization

We will talk more about data types later, but you can only do summarization on numeric or logical types. Not characters or factors.

```
x <- c(1, 5, 7, 4, 2, 8)
sum(x)
```

```
[1] 27
```

```
y <- c(TRUE, FALSE, FALSE, TRUE) # FALSE == 0 and TRUE == 1
sum(y)
```

```
[1] 2
```

```
z <- c("TRUE", "FALSE", "FALSE", "TRUE")
sum(z)
```

```
Error in sum(z): invalid 'type' (character) of argument
```

```
mean(z)
```

```
Warning in mean.default(z): argument is not numeric or logical: returning NA
```

Some examples

We can use the `jhu_cars` to explore different ways of summarizing data. The `head` command displays the first rows of an object:

```
library(jhur)
head(jhu_cars)
```

```
                car  mpg cyl disp  hp drat    wt  qsec vs am gear carb
1         Mazda RX4 21.0   6  160 110 3.90 2.620 16.46  0  1    4    4
2     Mazda RX4 Wag 21.0   6  160 110 3.90 2.875 17.02  0  1    4    4
3        Datsun 710 22.8   4  108  93 3.85 2.320 18.61  1  1    4    1
4    Hornet 4 Drive 21.4   6  258 110 3.08 3.215 19.44  1  0    3    1
5 Hornet Sportabout 18.7   8  360 175 3.15 3.440 17.02  0  0    3    2
6           Valiant 18.1   6  225 105 2.76 3.460 20.22  1  0    3    1
```

Statistical summarization

Note - the `$` references/selects columns from a `data.frame`/`tibble`:

```
mean(jhu_cars$hp)
```

```
[1] 146.6875
```

```
quantile(jhu_cars$hp)
```

```
   0%   25%   50%   75%  100% 
 52.0  96.5 123.0 180.0 335.0 
```

Statistical summarization

The “tidy” way:

```
jhu_cars %>% pull(hp) %>% mean() # alt: pull(jhu_cars, hp) %>% mean()
```

```
[1] 146.6875
```

```
jhu_cars %>% pull(hp) %>% quantile()
```

```
   0%   25%   50%   75%  100% 
 52.0  96.5 123.0 180.0 335.0 
```

Statistical summarization

```
jhu_cars %>% pull(wt) %>% median()
```

```
[1] 3.325
```

```
jhu_cars %>% pull(wt) %>% quantile(probs = 0.6)
```

```
 60% 
3.44 
```

Data Summarization on data frames

- Basic statistical summarization
  - `rowMeans(x)`: takes the means of each row of x
  - `colMeans(x)`: takes the means of each column of x
  - `rowSums(x)`: takes the sum of each row of x
  - `colSums(x)`: takes the sum of each column of x
  - `summary(x)`: for data frames, displays the quantile information

TB Incidence

Let's read in a `tibble` of values from TB incidence.

If you have the `jhur` package installed successfully:

```
tb <- jhur::read_tb()
```

If not, download the `xlsx` file from this link and read it in using `read_csv()`:
http://jhudatascience.org/intro_to_r/data/tb_incidence.xlsx

TB Incidence

Check out the data:

```
head(tb)
```

```
# A tibble: 6 × 19
  `TB incidence, all f…` `1990` `1991` `1992` `1993` `1994` `1995` `1996` `1997`
  <chr>                  <dbl>  <dbl>  <dbl>  <dbl>  <dbl>  <dbl>  <dbl>  <dbl>
1 Afghanistan              168    168    168    168    168    168    168    168
2 Albania                   25     24     25     26     26     27     27     28
3 Algeria                   38     38     39     40     41     42     43     44
4 American Samoa            21      7      2      9      9     11      0     12
5 Andorra                   36     34     32     30     29     27     26     26
6 Angola                   205    209    214    218    222    226    231    236
# … with 10 more variables: `1998` <dbl>, `1999` <dbl>, `2000` <dbl>,
#   `2001` <dbl>, `2002` <dbl>, `2003` <dbl>, `2004` <dbl>, `2005` <dbl>,
#   `2006` <dbl>, `2007` <dbl>
```

```
colnames(tb)
```

```
 [1] "TB incidence, all forms (per 100 000 population per year)"
 [2] "1990"
 [3] "1991"
 [4] "1992"
```

Indicator of TB

Before we go further, let's rename the first column to be the country measured using the `rename` function in `dplyr`.

In this case, we have to use the backticks (`) because there are spaces and funky characters in the name:

```
library(dplyr)
tb <- tb %>% rename(country = `TB incidence, all forms (per 100 000 population per year)`)
```

`colnames` will show us the column names and show that country is renamed:

```
colnames(tb)
```

```
 [1] "country" "1990"    "1991"    "1992"    "1993"    "1994"    "1995"
 [8] "1996"    "1997"    "1998"    "1999"    "2000"    "2001"    "2002"
[15] "2003"    "2004"    "2005"    "2006"    "2007"
```

Summarize the data: `dplyr summarize` function

`dplyr::summarize` will allow you to summarize data. Format is `new = SUMMARY`.

```
# General format - Not the code!
{data object to update} <- {data to use} %>%
                           summarize({summary column name} = {operator(source column)})
```

```
tb %>% summarize(mean_2006 = mean(`2006`, na.rm = TRUE))
```

```
# A tibble: 1 × 1
  mean_2006
      <dbl>
1      135.
```

Summarize the data: `dplyr summarize` function

`summarize` can do multiple operations at once. Just separate by a comma.

```
tb %>%
  summarize(mean_2006 = mean(`2006`, na.rm = TRUE),
            median_2007 = median(`2007`, na.rm = TRUE),
            median(`2004`, na.rm = TRUE))
```

```
# A tibble: 1 × 3
  mean_2006 median_2007 `median(\`2004\`, na.rm = TRUE)`
      <dbl>       <dbl>                            <dbl>
1      135.          53                               56
```

Notice how when we forget to provide a new name, output is still provided, but the column name is messy.

Iterative summaries: `dplyr summarize` and `across` functions

Use the `across` function with `summarize` to summarize across multiple columns of your data.

```
tb %>%
  summarize(across( c(`1990`, `1991`, `1992`, `1993`), ~ sum(.x, na.rm = TRUE)))
```

```
# A tibble: 1 × 4
  `1990` `1991` `1992` `1993`
   <dbl>  <dbl>  <dbl>  <dbl>
1  21855  22288  22421  22836
```

```
tb %>%
  summarize(across( starts_with("2"), ~ range(.x, na.rm = TRUE)))
```

```
# A tibble: 2 × 8
  `2000` `2001` `2002` `2003` `2004` `2005` `2006` `2007`
   <dbl>  <dbl>  <dbl>  <dbl>  <dbl>  <dbl>  <dbl>  <dbl>
1      0      0      3      0      0      0      0      0
2    801    916    994   1075   1127   1141   1169   1198
```

Row means

`colMeans` and `rowMeans` require **all numeric data**.

Let's see what the mean is across each row (country):

```
tb_2 <- column_to_rownames(tb, "country") # opposite of rownames_to_column() !
head(tb_2, 2)
```

```
            1990 1991 1992 1993 1994 1995 1996 1997 1998 1999 2000 2001 2002
Afghanistan  168  168  168  168  168  168  168  168  168  168  168  168  168
Albania       25   24   25   26   26   27   27   28   28   27   25   23   23
            2003 2004 2005 2006 2007
Afghanistan  168  168  168  168  168
Albania       22   21   20   18   17
```

```
rowMeans(tb_2, na.rm = TRUE)
```

```
                Afghanistan                     Albania
                 168.000000                   24.000000
                    Algeria              American Samoa
                   46.388889                    7.611111
                    Andorra                      Angola
                   24.944444                  243.888889
                    Anguilla          Antigua and Barbuda
```

Row means

`colMeans` gives you very similar output to functions we've seen previously in this lecture (`summarize` and `across`).

```
colMeans(tb_2, na.rm = TRUE)
```

```
    1990     1991     1992     1993     1994     1995     1996     1997
105.5797 107.6715 108.3140 110.3188 111.9662 114.1981 115.3527 118.8792
    1998     1999     2000     2001     2002     2003     2004     2005
121.5169 125.0435 127.8454 130.7488 136.1739 136.1932 136.9662 135.6683
    2006     2007
134.6106 133.3865
```

```
tb_2 %>%
  summarize(across( colnames(tb_2), ~ mean(.x, na.rm = TRUE)))
```

```
      1990     1991    1992     1993     1994     1995     1996     1997
1 105.5797 107.6715 108.314 110.3188 111.9662 114.1981 115.3527 118.8792
      1998     1999     2000     2001     2002     2003     2004     2005
1 121.5169 125.0435 127.8454 130.7488 136.1739 136.1932 136.9662 135.6683
      2006     2007
1 134.6106 133.3865
```

`summary` Function

Using `summary` can give you rough snapshots of each column, but you would likely use `mean`, `min`, `max`, and `quantile` when necessary (and number of NAs):

```
summary(tb)
```

```
  country                1990            1991            1992      
Length:208         Min.   :  0.0   Min.   :  4.0   Min.   :  2.0  
Class :character   1st Qu.: 27.5   1st Qu.: 27.0   1st Qu.: 27.0  
Mode  :character   Median : 60.0   Median : 58.0   Median : 56.0  
                   Mean   :105.6   Mean   :107.7   Mean   :108.3  
                   3rd Qu.:165.0   3rd Qu.:171.0   3rd Qu.:171.5  
                   Max.   :585.0   Max.   :594.0   Max.   :606.0  
                   NA's   :1       NA's   :1       NA's   :1      
     1993            1994          1995            1996            1997      
Min.   :  4.0   Min.   :  0   Min.   :  3.0   Min.   :  0.0   Min.   :  0.0  
1st Qu.: 27.5   1st Qu.: 26   1st Qu.: 26.5   1st Qu.: 25.5   1st Qu.: 24.5  
Median : 56.0   Median : 57   Median : 58.0   Median : 60.0   Median : 64.0  
Mean   :110.3   Mean   :112   Mean   :114.2   Mean   :115.4   Mean   :118.9  
3rd Qu.:171.0   3rd Qu.:174   3rd Qu.:177.5   3rd Qu.:179.0   3rd Qu.:181.0  
Max.   :618.0   Max.   :630   Max.   :642.0   Max.   :655.0   Max.   :668.0  
NA's   :1       NA's   :1     NA's   :1       NA's   :1       NA's   :1      
     1998            1999            2000            2001      
Min.   :  0.0   Min.   :  0.0   Min.   :  0.0   Min.   :  0.0  
```

Lab Part 1

Website

Youth Tobacco Survey

Here we will be using the Youth Tobacco Survey data:
http://jhudatascience.org/intro_to_r/data/Youth_Tobacco_Survey_YTS_Data.csv

```
yts <- jhur::read_yts()
head(yts)
```

```
# A tibble: 6 × 31
   YEAR LocationAbbr LocationDesc TopicType     TopicDesc MeasureDesc DataSource
  <dbl> <chr>        <chr>        <chr>         <chr>     <chr>       <chr>
1  2015 AZ           Arizona      Tobacco Use … Cessatio… Percent of… YTS
2  2015 AZ           Arizona      Tobacco Use … Cessatio… Percent of… YTS
3  2015 AZ           Arizona      Tobacco Use … Cessatio… Percent of… YTS
4  2015 AZ           Arizona      Tobacco Use … Cessatio… Quit Attem… YTS
5  2015 AZ           Arizona      Tobacco Use … Cessatio… Quit Attem… YTS
6  2015 AZ           Arizona      Tobacco Use … Cessatio… Quit Attem… YTS
# … with 24 more variables: Response <chr>, Data_Value_Unit <chr>,
#   Data_Value_Type <chr>, Data_Value <dbl>, Data_Value_Footnote_Symbol <chr>,
#   Data_Value_Footnote <chr>, Data_Value_Std_Err <dbl>,
#   Low_Confidence_Limit <dbl>, High_Confidence_Limit <dbl>, Sample_Size <dbl>,
#   Gender <chr>, Race <chr>, Age <chr>, Education <chr>, GeoLocation <chr>,
#   TopicTypeId <chr>, TopicId <chr>, MeasureId <chr>, StratificationID1 <chr>,
#   StratificationID2 <chr>, StratificationID3 <chr>, …
```

Length and unique

`unique(x)` will return the unique elements of `x`

```
locations <- yts %>% pull(LocationDesc)
unique(locations) %>% head()
```

```
[1] "Arizona"     "Connecticut" "Georgia"     "Hawaii"      "Illinois"
[6] "Louisiana"
```

`length` will tell you the length of a vector. Combined with `unique`, tells you the number of unique elements:

```
length(unique(locations))
```

```
[1] 50
```

`table` and `dplyr`: `count`

`table(x)` will return a frequency table of unique elements of `x`

```
table(locations)
```

```
locations
```

Alabama	Arizona	Arkansas
378	240	210
California	Colorado	Connecticut
96	48	384
Delaware	District of Columbia	Florida
312	48	96
Georgia	Guam	Hawaii
282	48	270
Idaho	Illinois	Indiana
48	282	264
Iowa	Kansas	Kentucky
276	186	255
Louisiana	Maine	Maryland
240	48	96
Massachusetts	Michigan	Minnesota
48	138	141
Mississippi	Missouri	National (States and DC)
567	294	26

`table` and `dplyr`: `count`

Use `count` directly on a data.frame and column without needing to use `pull`.

```
yts %>% count(LocationDesc)
```

```
# A tibble: 50 × 2
   LocationDesc            n
   <chr>               <int>
 1 Alabama               378
 2 Arizona               240
 3 Arkansas              210
 4 California             96
 5 Colorado               48
 6 Connecticut           384
 7 Delaware              312
 8 District of Columbia   48
 9 Florida                96
10 Georgia               282
# … with 40 more rows
```

`table` and `dplyr`: `count`

Multiple columns listed further subdivides the count.

```
yts %>% count(LocationDesc, TopicDesc)
```

```
# A tibble: 146 × 3
   LocationDesc TopicDesc                          n
   <chr>        <chr>                          <int>
 1 Alabama      Cessation (Youth)                 90
 2 Alabama      Cigarette Use (Youth)            144
 3 Alabama      Smokeless Tobacco Use (Youth)    144
 4 Arizona      Cessation (Youth)                 60
 5 Arizona      Cigarette Use (Youth)             99
 6 Arizona      Smokeless Tobacco Use (Youth)     81
 7 Arkansas     Cessation (Youth)                 42
 8 Arkansas     Cigarette Use (Youth)             78
 9 Arkansas     Smokeless Tobacco Use (Youth)     90
10 California   Cessation (Youth)                 24
# … with 136 more rows
```

Grouping

Perform Operations By Groups: dplyr

`group_by` allows you group the data set by grouping variables:

```
#
yts
```

```
# A tibble: 9,794 × 31
    YEAR LocationAbbr LocationDesc TopicType    TopicDesc MeasureDesc DataSource
   <dbl> <chr>        <chr>        <chr>        <chr>     <chr>       <chr>
 1  2015 AZ           Arizona      Tobacco Use… Cessatio… Percent of… YTS
 2  2015 AZ           Arizona      Tobacco Use… Cessatio… Percent of… YTS
 3  2015 AZ           Arizona      Tobacco Use… Cessatio… Percent of… YTS
 4  2015 AZ           Arizona      Tobacco Use… Cessatio… Quit Attem… YTS
 5  2015 AZ           Arizona      Tobacco Use… Cessatio… Quit Attem… YTS
 6  2015 AZ           Arizona      Tobacco Use… Cessatio… Quit Attem… YTS
 7  2015 AZ           Arizona      Tobacco Use… Cigarett… Smoking St… YTS
 8  2015 AZ           Arizona      Tobacco Use… Cigarett… Smoking St… YTS
 9  2015 AZ           Arizona      Tobacco Use… Cigarett… Smoking St… YTS
10  2015 AZ           Arizona      Tobacco Use… Cigarett… Smoking St… YTS
# … with 9,784 more rows, and 24 more variables: Response <chr>,
#   Data_Value_Unit <chr>, Data_Value_Type <chr>, Data_Value <dbl>,
#   Data_Value_Footnote_Symbol <chr>, Data_Value_Footnote <chr>,
#   Data_Value_Std_Err <dbl>, Low_Confidence_Limit <dbl>,
#   High_Confidence_Limit <dbl>, Sample_Size <dbl>, Gender <chr>, Race <chr>,
```

Perform Operations By Groups: dplyr

`group_by` allows you group the data set by grouping variables:

```
yts <- yts %>% group_by(Response)
yts
```

```
# A tibble: 9,794 × 31
# Groups:   Response [4]
    YEAR LocationAbbr LocationDesc TopicType    TopicDesc MeasureDesc DataSource
   <dbl> <chr>        <chr>        <chr>        <chr>     <chr>       <chr>
 1  2015 AZ           Arizona      Tobacco Use… Cessatio… Percent of… YTS
 2  2015 AZ           Arizona      Tobacco Use… Cessatio… Percent of… YTS
 3  2015 AZ           Arizona      Tobacco Use… Cessatio… Percent of… YTS
 4  2015 AZ           Arizona      Tobacco Use… Cessatio… Quit Attem… YTS
 5  2015 AZ           Arizona      Tobacco Use… Cessatio… Quit Attem… YTS
 6  2015 AZ           Arizona      Tobacco Use… Cessatio… Quit Attem… YTS
 7  2015 AZ           Arizona      Tobacco Use… Cigarett… Smoking St… YTS
 8  2015 AZ           Arizona      Tobacco Use… Cigarett… Smoking St… YTS
 9  2015 AZ           Arizona      Tobacco Use… Cigarett… Smoking St… YTS
10  2015 AZ           Arizona      Tobacco Use… Cigarett… Smoking St… YTS
# … with 9,784 more rows, and 24 more variables: Response <chr>,
#   Data_Value_Unit <chr>, Data_Value_Type <chr>, Data_Value <dbl>,
#   Data_Value_Footnote_Symbol <chr>, Data_Value_Footnote <chr>,
#   Data_Value_Std_Err <dbl>, Low_Confidence_Limit <dbl>,
#   High_Confidence_Limit <dbl>, Sample_Size <dbl>, Gender <chr>, Race <chr>,
```

Summarize the grouped data

It's grouped! Grouping doesn't change the data in any way, but how **functions operate on it**. Now we can summarize `Data_Value` (percent of respondents) by group:

```
yts %>% summarize(avg_percent = mean(Data_Value, na.rm = TRUE))
```

```
# A tibble: 4 × 2
  Response avg_percent
  <chr>          <dbl>
1 Current         9.68
2 Ever           26.1
3 Frequent        3.48
4 <NA>           53.5
```

Using the `pipe` to connect these

Pipe `yts` into `group_by`, then pipe that into `summarize`:

```
yts %>%
  group_by(Response) %>%
  summarize(avg_percent = mean(Data_Value, na.rm = TRUE),
            max_percent = max(Data_Value, na.rm = TRUE))
```

```
# A tibble: 4 × 3
  Response avg_percent max_percent
  <chr>          <dbl>       <dbl>
1 Current         9.68        40.6
2 Ever           26.1         98
3 Frequent        3.48        23.9
4 <NA>           53.5         81.9
```

Ungroup the data

The `ungroup` function will allow you to clear the groups from the data. You can also overwrite the first `group_by` with a new one.

```
yts = ungroup(yts)
yts
```

```
# A tibble: 9,794 × 31
    YEAR LocationAbbr LocationDesc TopicType    TopicDesc MeasureDesc DataSource
   <dbl> <chr>        <chr>        <chr>        <chr>     <chr>       <chr>     
 1  2015 AZ           Arizona      Tobacco Use… Cessatio… Percent of… YTS       
 2  2015 AZ           Arizona      Tobacco Use… Cessatio… Percent of… YTS       
 3  2015 AZ           Arizona      Tobacco Use… Cessatio… Percent of… YTS       
 4  2015 AZ           Arizona      Tobacco Use… Cessatio… Quit Attem… YTS       
 5  2015 AZ           Arizona      Tobacco Use… Cessatio… Quit Attem… YTS       
 6  2015 AZ           Arizona      Tobacco Use… Cessatio… Quit Attem… YTS       
 7  2015 AZ           Arizona      Tobacco Use… Cigarett… Smoking St… YTS       
 8  2015 AZ           Arizona      Tobacco Use… Cigarett… Smoking St… YTS       
 9  2015 AZ           Arizona      Tobacco Use… Cigarett… Smoking St… YTS       
10  2015 AZ           Arizona      Tobacco Use… Cigarett… Smoking St… YTS       
# … with 9,784 more rows, and 24 more variables: Response <chr>,
#   Data_Value_Unit <chr>, Data_Value_Type <chr>, Data_Value <dbl>,
#   Data_Value_Footnote_Symbol <chr>, Data_Value_Footnote <chr>,
#   Data_Value_Std_Err <dbl>, Low_Confidence_Limit <dbl>,
```

`group_by` with `mutate` - just add data

We can also use `mutate` to calculate the mean value for each year and add it as a column:

```
yts %>%
  group_by(YEAR) %>%
  mutate(year_avg = mean(Data_Value, na.rm = TRUE)) %>%
  select(LocationDesc, Data_Value, year_avg)
```

```
# A tibble: 9,794 × 4
# Groups:   YEAR [17]
    YEAR LocationDesc Data_Value year_avg
   <dbl> <chr>             <dbl>    <dbl>
 1  2015 Arizona              NA     15.2
 2  2015 Arizona              NA     15.2
 3  2015 Arizona              NA     15.2
 4  2015 Arizona              NA     15.2
 5  2015 Arizona              NA     15.2
 6  2015 Arizona              NA     15.2
 7  2015 Arizona             3.2     15.2
 8  2015 Arizona             3.2     15.2
 9  2015 Arizona             3.1     15.2
10  2015 Arizona            12.5     15.2
# … with 9,784 more rows
```

Counting

There are other functions, such as `n()` count the number of observations.

```
yts %>%
  group_by(YEAR) %>%
  summarize(n = n(),
            mean = mean(Data_Value, na.rm = TRUE))
```

```
# A tibble: 17 × 3
    YEAR     n  mean
   <dbl> <int> <dbl>
 1  1999   372  26.1
 2  2000  1224  26.7
 3  2001   426  23.4
 4  2002  1016  25.2
 5  2003   498  21.3
 6  2004   611  20.7
 7  2005   636  21.8
 8  2006   518  21.8
 9  2007   516  20.0
10  2008   483  18.2
11  2009   686  18.3
12  2010   447  17.8
13  2011   521  17.8
14  2012   244  15.5
```

Lab Part 2

Website

Preview: plotting

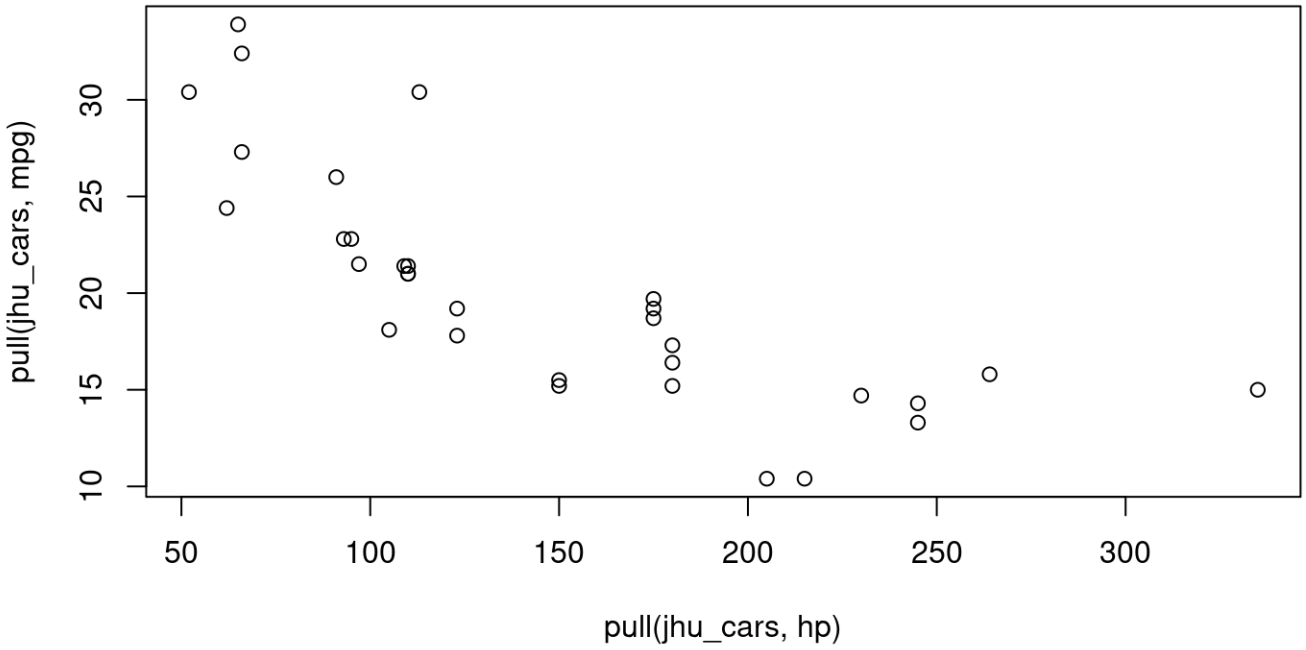
Basic Plots

Plotting is an important component of exploratory data analysis. These are some rough one-line plots that you can use in realtime while exploring your data. We will go over formatting and making plots look nicer in additional lectures.

- Basic summarization plots:
  - `plot(x,y)`: scatterplot of x and y
  - `boxplot(y~x)`: boxplot of y against levels of x
  - `hist(x)`: histogram of x
  - `plot(density(x))`: kernel density plot of x

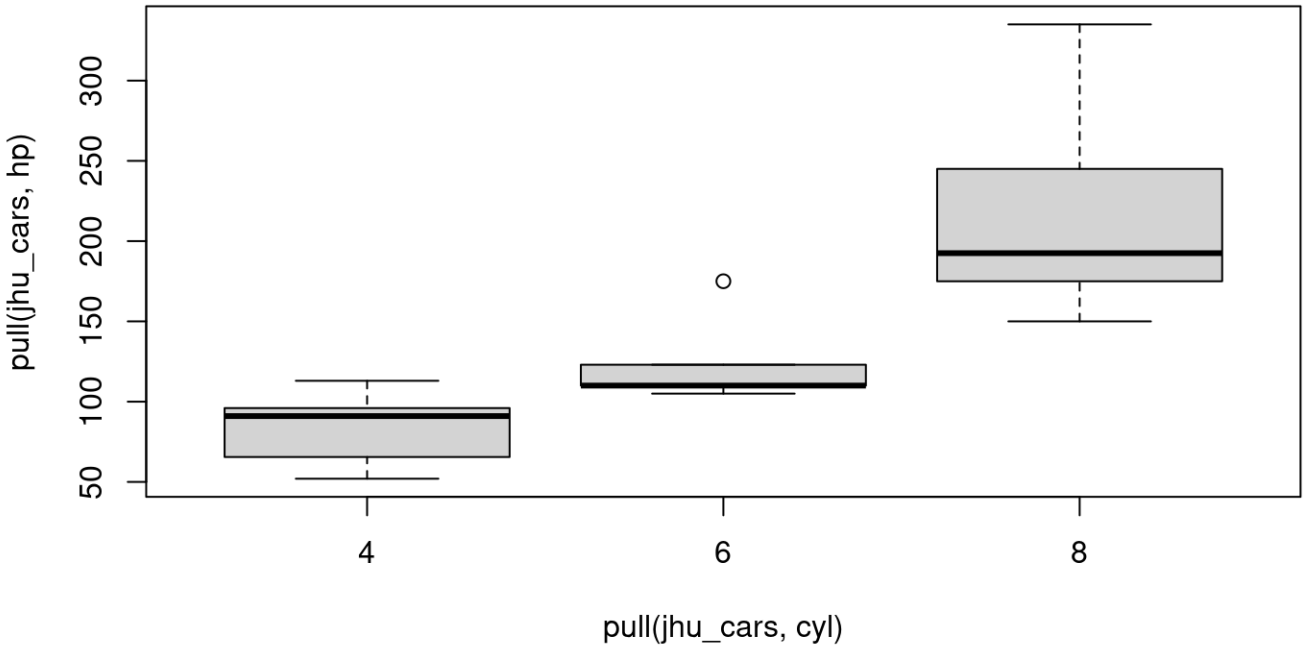
Scatterplot

```
plot( pull(jhu_cars,hp), pull(jhu_cars,mpg) ) # alt: plot(jhu_cars$hp, jhu_cars$mpg)
```



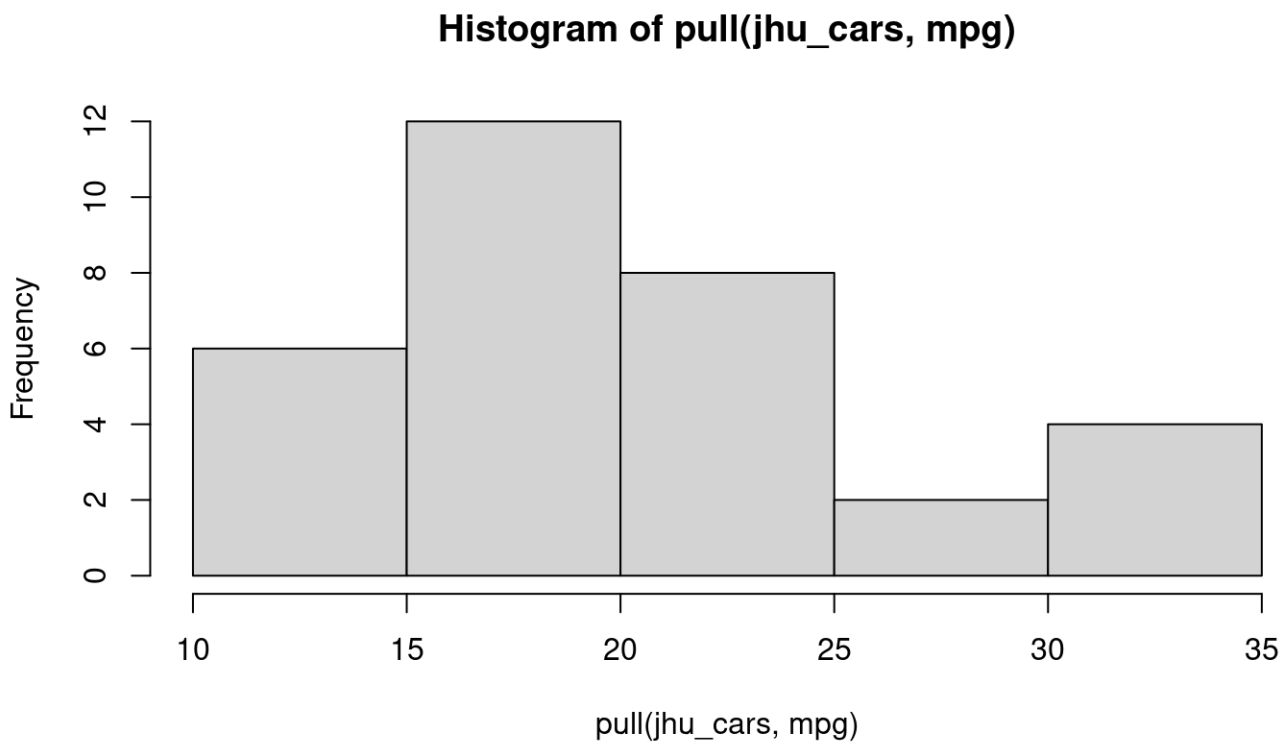
Boxplot

```
boxplot( pull(jhu_cars,hp) ~ pull(jhu_cars,cyl) )
```



Histogram

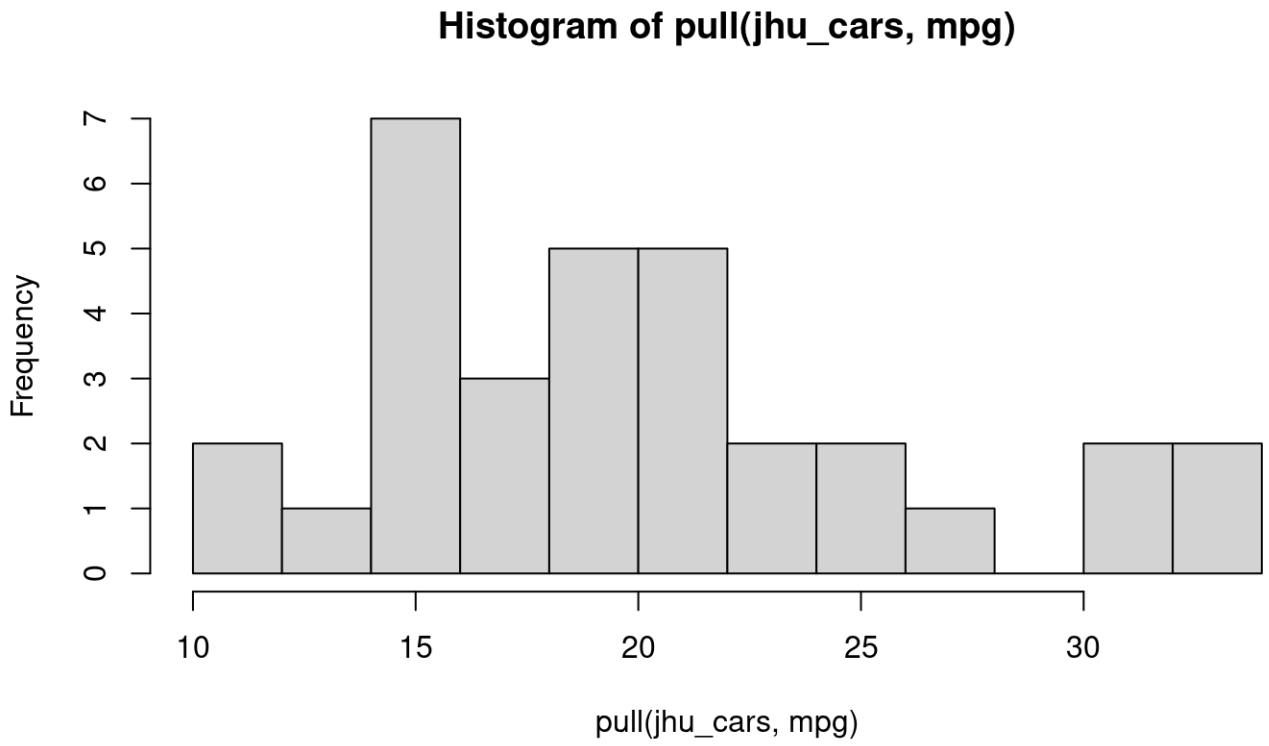
```
hist(pull(jhu_cars,mpg))
```



Histogram

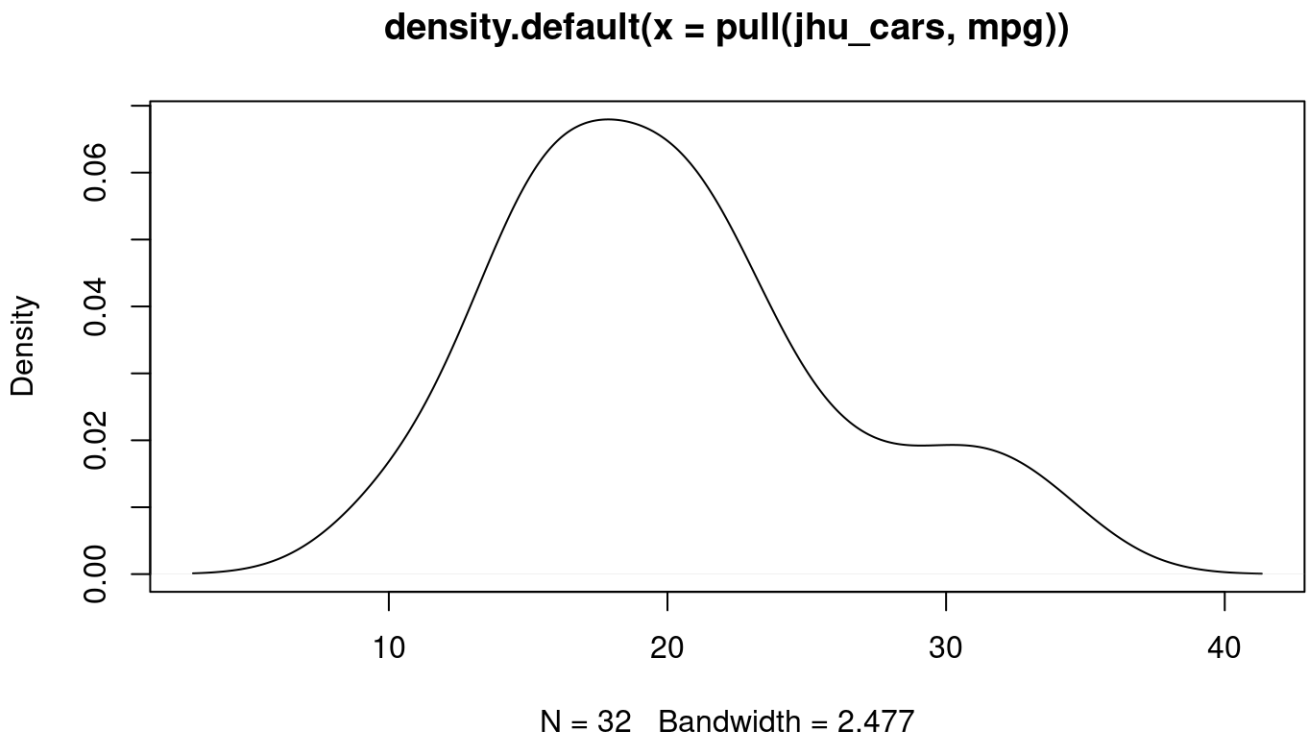
Use the `breaks =` argument to tweak the resolution:

```
hist(pull(jhu_cars,mpg), breaks = 10)
```



Density

```
plot(density(pull(jhu_cars,mpg)))
```



Lab Part 3

Website