Basic Statistics in R

ECO 6416

2022-08-28

Contents

Here are all the packages needed to get started.

```
library(gt)
library(tidyverse)
library(gtsummary)
library(plotly)
library(readxl)
library(plotly)
library(corrplot)





sessionInfo()
```

```
## R version 4.2.1 (2022-06-23 ucrt)
## Platform: x86_64-w64-mingw32/x64 (64-bit)
## Running under: Windows 10 x64 (build 19044)
##
## Matrix products: default
##
## locale:
## [1] LC_COLLATE=English_United States.utf8
## [2] LC_CTYPE=English_United States.utf8
## [3] LC_MONETARY=English_United States.utf8
## [4] LC_NUMERIC=C
## [5] LC_TIME=English_United States.utf8
##
## attached base packages:
## [1] stats     graphics  grDevices utils     datasets  methods   base
##
## other attached packages:
##  [1] corrplot_0.92   readxl_1.4.1    plotly_4.10.0   gtsummary_1.6.1
##  [5] forcats_0.5.2   stringr_1.4.0   dplyr_1.0.9     purrr_0.3.4
##  [9] readr_2.1.2     tidyr_1.2.0     tibble_3.1.7    ggplot2_3.3.6
## [13] tidyverse_1.3.2 gt_0.7.0
##
## loaded via a namespace (and not attached):
##  [1] lubridate_1.8.0     assertthat_0.2.1    digest_0.6.29
##  [4] utf8_1.2.2          R6_2.5.1            cellranger_1.1.0
##  [7] backports_1.4.1     reprex_2.0.2        evaluate_0.15
## [10] httr_1.4.3          pillar_1.7.0        rlang_1.0.3
## [13] lazyeval_0.2.2      googlesheets4_1.0.1 rstudioapi_0.14
## [16] data.table_1.14.2   rmarkdown_2.14      googledrive_2.0.0
## [19] htmlwidgets_1.5.4   munsell_0.5.0       broom_1.0.0
## [22] compiler_4.2.1      modelr_0.1.9        xfun_0.31
## [25] pkgconfig_2.0.3     htmltools_0.5.2     tidyselect_1.1.2
## [28] viridisLite_0.4.0   fansi_1.0.3         crayon_1.5.1
## [31] tzdb_0.3.0          dbplyr_2.2.1        withr_2.5.0
## [34] grid_4.2.1          jsonlite_1.8.0      gtable_0.3.0
## [37] lifecycle_1.0.1     DBI_1.1.3           magrittr_2.0.3
## [40] scales_1.2.0        cli_3.3.0           stringi_1.7.8
## [43] broom.helpers_1.8.0 fs_1.5.2            xml2_1.3.3
## [46] ellipsis_0.3.2      generics_0.1.3      vctrs_0.4.1
## [49] tools_4.2.1         glue_1.6.2          hms_1.1.1
## [52] fastmap_1.1.0       yaml_2.3.5          colorspace_2.0-3
## [55] gargle_1.2.0        rvest_1.0.3         knitr_1.39
## [58] haven_2.5.1
```

1 Univariate Analysis

In univariate analysis, we look at a single variable and describe 3 different things:

- Center
- Shape
- Spread

1.1 Center

Helps explain where the middle of the data is. This can be measured in 3 main ways.

1.1.1 Mean

```r
grades <- c(78,79,80,81,82)

mean(grades)
```

```
## [1] 80
```

1.1.2 Median

```r
median(grades)
```

```
## [1] 80
```

1.1.3 Mode

There isn't an easy way of doing this, so I created a function instead.

```r
getModes <- function(x) {
  ux <- unique(x)
  tab <- tabulate(match(x, ux))
  ux[tab == max(tab)]
}


getModes(grades)
```

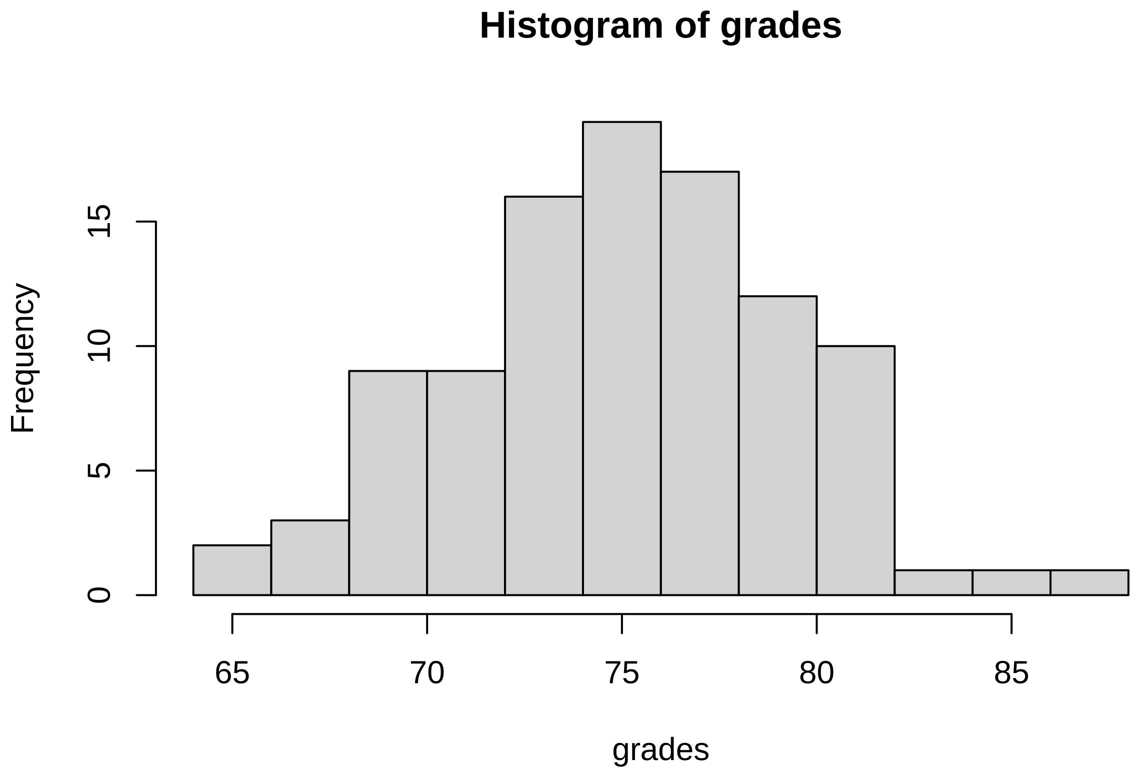
```
## [1] 78 79 80 81 82
```

Since there is no value that occurs most frequently, they all show. Mode is rarely used.

1.2 Shape

For this, I am going to randomly generate 100 exam scores.

```r
grades <- rnorm(100,mean = 75, sd = 5 )
hist(grades)
```



You can mess with the `breaks =` arguement to get different numbers of bins.

1.3 Spread

Let's look at a different dataset. Starwars character heights.

```
var(starwars$height)
```

```
## [1] NA
```

What happened? We have nulls in our data, so we cannot calculate the variance until we tell the system to ignore null values

```
var(starwars$height, na.rm = TRUE)
```

```
## [1] 1208.983
```

```
sd(starwars$height, na.rm = TRUE)
```

```
## [1] 34.77043
```

```
IQR(starwars$height, na.rm = TRUE)
```

```
## [1] 24
```

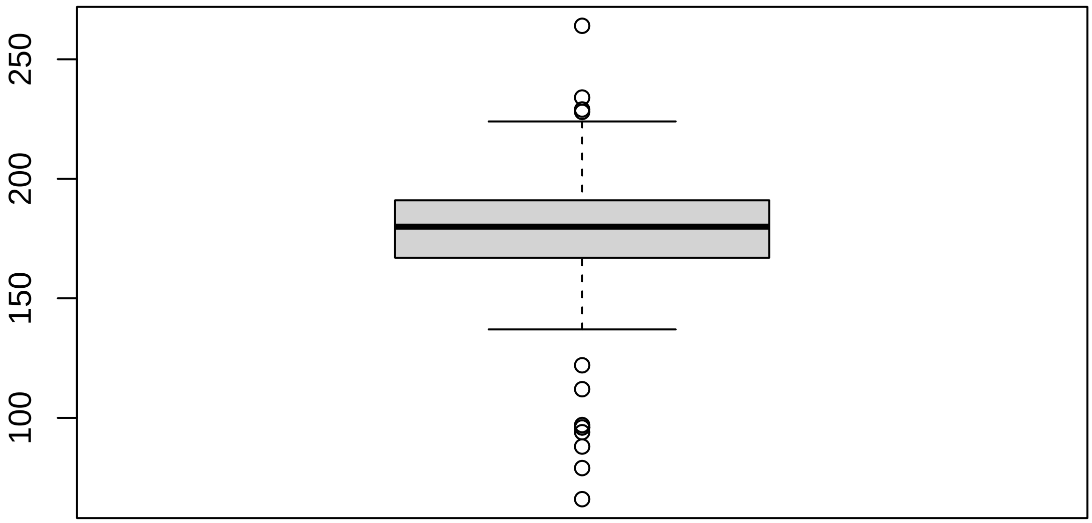
```
range(starwars$height, na.rm = TRUE)
```

```
## [1]  66 264
```

1.3.1 Visualizing Spread

You can visualize the spread as well this with a boxplot.

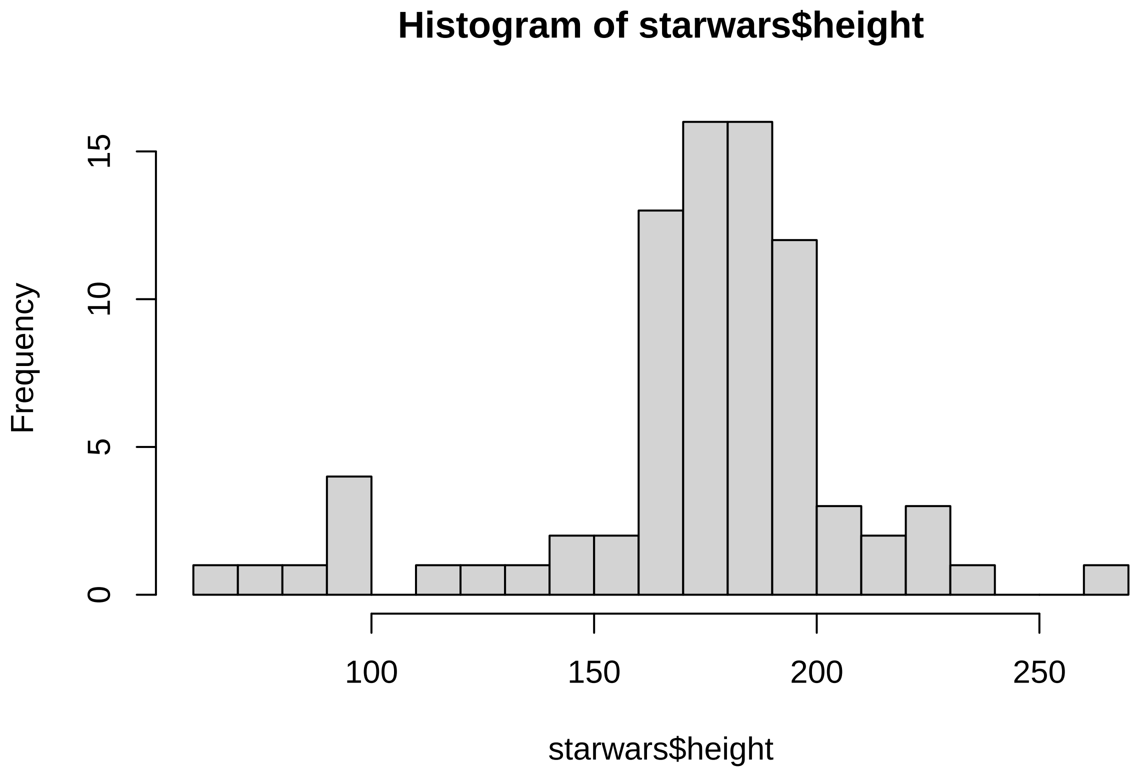
```
boxplot(starwars$height)
```



You can see that a lot of them fall outside of the fences.

You could also visualize this with a histogram

```
hist(starwars$height, breaks = "fd")
```



1.4 Easier Way

To get most of the items of interest, you can simply use the `summary()` function. The standard deviation nor the variance is displayed, so you would still need those.

```
summary(mtcars)
```

```
##       mpg             cyl             disp             hp       
##  Min.   :10.40   Min.   :4.000   Min.   : 71.1   Min.   : 52.0  
##  1st Qu.:15.43   1st Qu.:4.000   1st Qu.:120.8   1st Qu.: 96.5  
##  Median :19.20   Median :6.000   Median :196.3   Median :123.0  
##  Mean   :20.09   Mean   :6.188   Mean   :230.7   Mean   :146.7  
##  3rd Qu.:22.80   3rd Qu.:8.000   3rd Qu.:326.0   3rd Qu.:180.0  
##  Max.   :33.90   Max.   :8.000   Max.   :472.0   Max.   :335.0  
##       drat             wt             qsec             vs        
##  Min.   :2.760   Min.   :1.513   Min.   :14.50   Min.   :0.0000  
##  1st Qu.:3.080   1st Qu.:2.581   1st Qu.:16.89   1st Qu.:0.0000  
##  Median :3.695   Median :3.325   Median :17.71   Median :0.0000  
##  Mean   :3.597   Mean   :3.217   Mean   :17.85   Mean   :0.4375  
##  3rd Qu.:3.920   3rd Qu.:3.610   3rd Qu.:18.90   3rd Qu.:1.0000  
##  Max.   :4.930   Max.   :5.424   Max.   :22.90   Max.   :1.0000  
##        am              gear            carb      
##  Min.   :0.0000   Min.   :3.000   Min.   :1.000  
##  1st Qu.:0.0000   1st Qu.:3.000   1st Qu.:2.000  
##  Median :0.0000   Median :4.000   Median :2.000  
##  Mean   :0.4062   Mean   :3.688   Mean   :2.812  
##  3rd Qu.:1.0000   3rd Qu.:4.000   3rd Qu.:4.000  
##  Max.   :1.0000   Max.   :5.000   Max.   :8.000
```

2 Bivariate Analysis

When looking at 2 variables, we use correlation, scatterplots, and time series graphs.

2.1 Correlation

```
cor(mtcars$mpg, mtcars$hp)
```

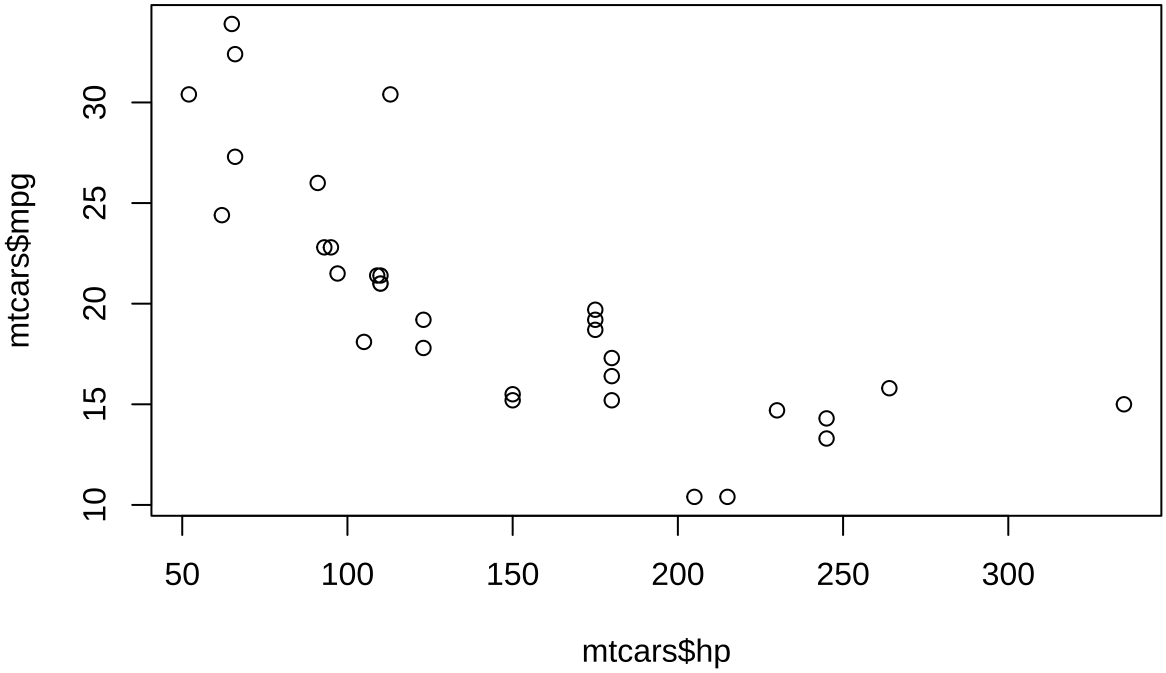
```
## [1] -0.7761684
```

These are strongly negatively correlated. We might say, horsepower has a inverse relationship with mpg, or mpg has a negative relationship with horsepower.

2.2 Scatterplots

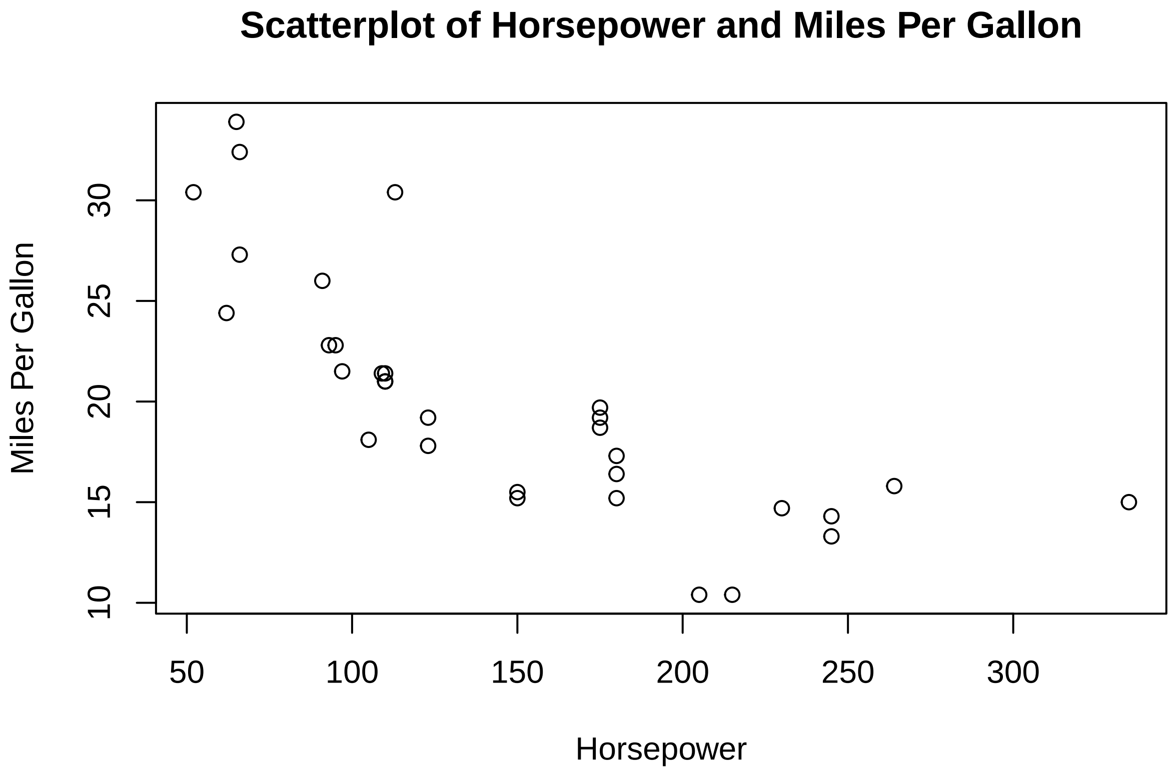
We can also visualize this relationship with a scatterplot.

```
plot(mtcars$hp, mtcars$mpg)
```



we can also make this look nicer with some labels

```
plot(mtcars$hp, mtcars$mpg,
     main = "Scatterplot of Horsepower and Miles Per Gallon",
     xlab = "Horsepower",
     ylab = "Miles Per Gallon")
```



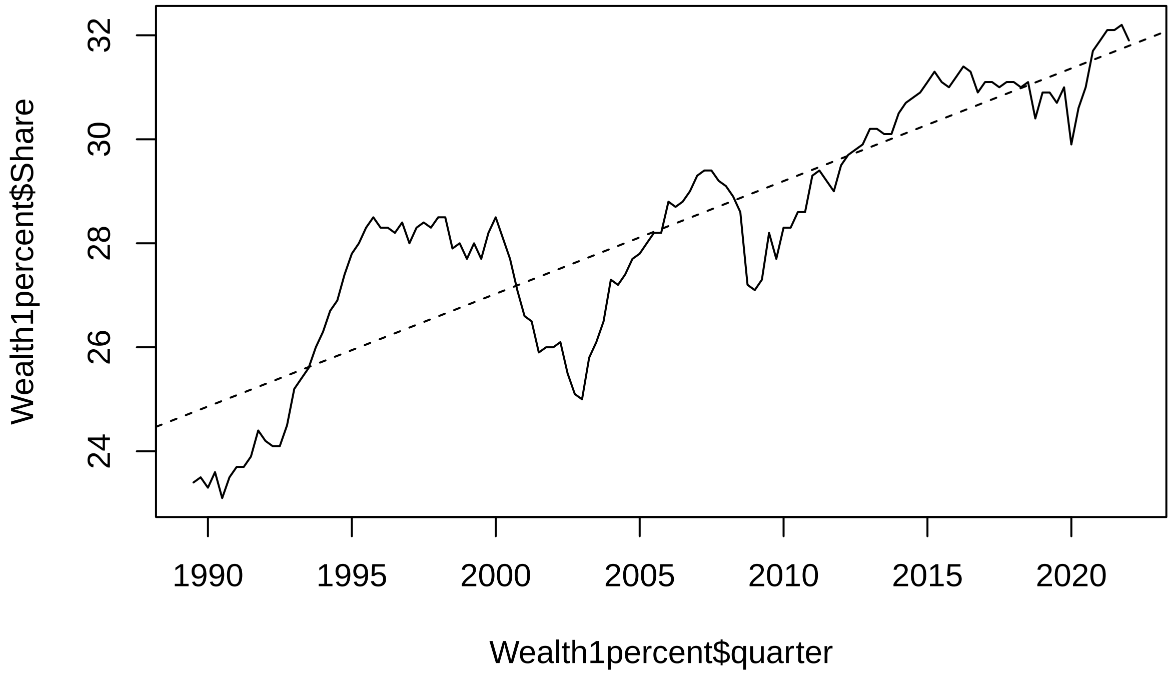
2.3 Time series graphs

Since the data we've seen thus far has been only cross section, let's bring in the data from the previous module. `Wealth1Percent.xlsx`

```
Wealth1percent <- read_excel("../Data/Wealth1percent.xlsx",
                             col_types = c("date", "numeric","numeric"))
```

To add the trend line, we need to do something that we haven't discussed yet. Don't worry, I'll explain it in the next module.

```
plot(Wealth1percent$quarter,Wealth1percent$Share, type = "l")
abline(lm(Share ~ quarter, data = Wealth1percent),lty = 2)
```



3 More Visual Displays!

Here are some more ways to display data

3.1 Data Tables

We can represent data in categorical fashion:

```
table(starwars$hair_color)
```

```
##
##        auburn  auburn, grey auburn, white         black         blond
##             1             1             1            13             3
##        blonde         brown   brown, grey          grey          none
##             1            18             1             1            37
##       unknown         white
##             1             4
```

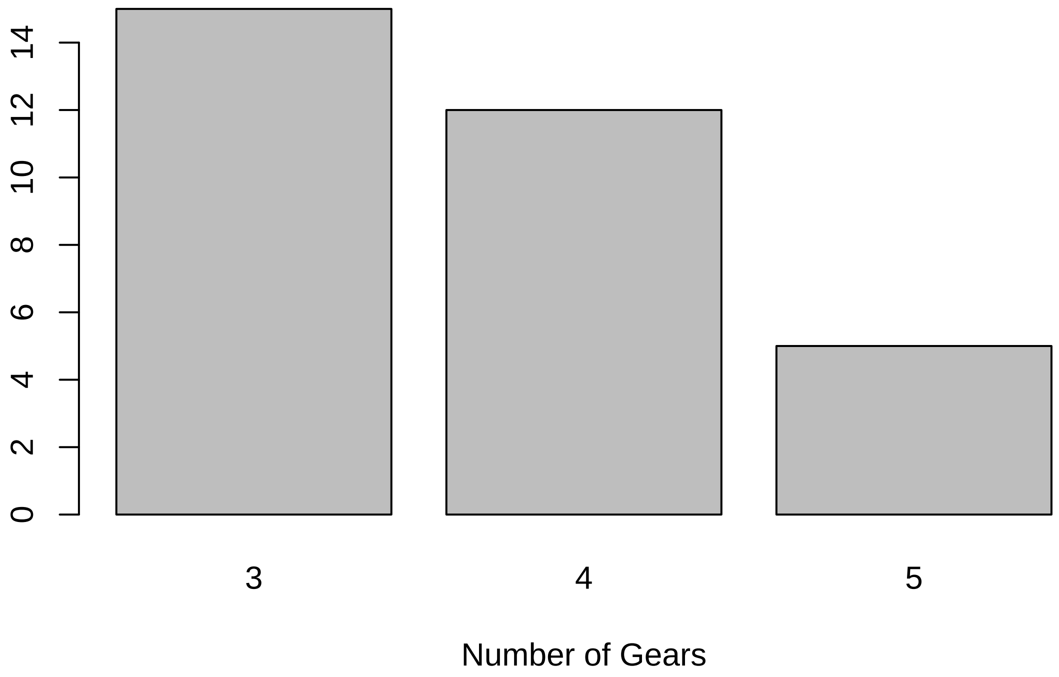
Or quantitative

```
bins <- seq(10,34,by = 2)

mpg <- cut(mtcars$mpg,bins)

table(mpg)
```

```
## mpg
## (10,12] (12,14] (14,16] (16,18] (18,20] (20,22] (22,24] (24,26] (26,28] (28,30]
##       2       1       7       3       5       5       2       2       1       0
## (30,32] (32,34]
##       2       2
```

3.2 Bar Charts

```
n <- table(mtcars$gear)

barplot(n,xlab="Number of Gears")
```



3.3 Stem and Leaf Plots

We can add `scale = 3` to make this line up properly

```
stem(mtcars$hp, scale = 3)
```

```
##
##   The decimal point is 1 digit(s) to the right of the |
##
##    5 | 2
##    6 | 2566
##    7 |
##    8 |
##    9 | 1357
##   10 | 59
##   11 | 0003
##   12 | 33
##   13 |
##   14 |
##   15 | 00
##   16 |
##   17 | 555
##   18 | 000
##   19 |
##   20 | 5
##   21 | 5
##   22 |
##   23 | 0
##   24 | 55
##   25 |
##   26 | 4
##   27 |
##   28 |
##   29 |
##   30 |
##   31 |
##   32 |
##   33 | 5
```

4 Fancier Output

Check out this fun stuff! Makes things look much cleaner.

4.1 Table Output

Categorical

```
hair <- table(starwars$hair_color)%>%
  data.frame()

colnames(hair)[1] <- "Hair Color"

gt(hair)
```

Hair Color	Freq
auburn	1
auburn, grey	1
auburn, white	1
black	13
blond	3
blonde	1
brown	18
brown, grey	1
grey	1
none	37
unknown	1
white	4

Or quantitative

```
bins <- seq(10,34,by = 2)

mpg <- cut(mtcars$mpg,bins)

table(mpg)%>%
  data.frame() %>%
  gt()
```

4.2 Summary Statistics

Using our classic `mtcars` dataset.

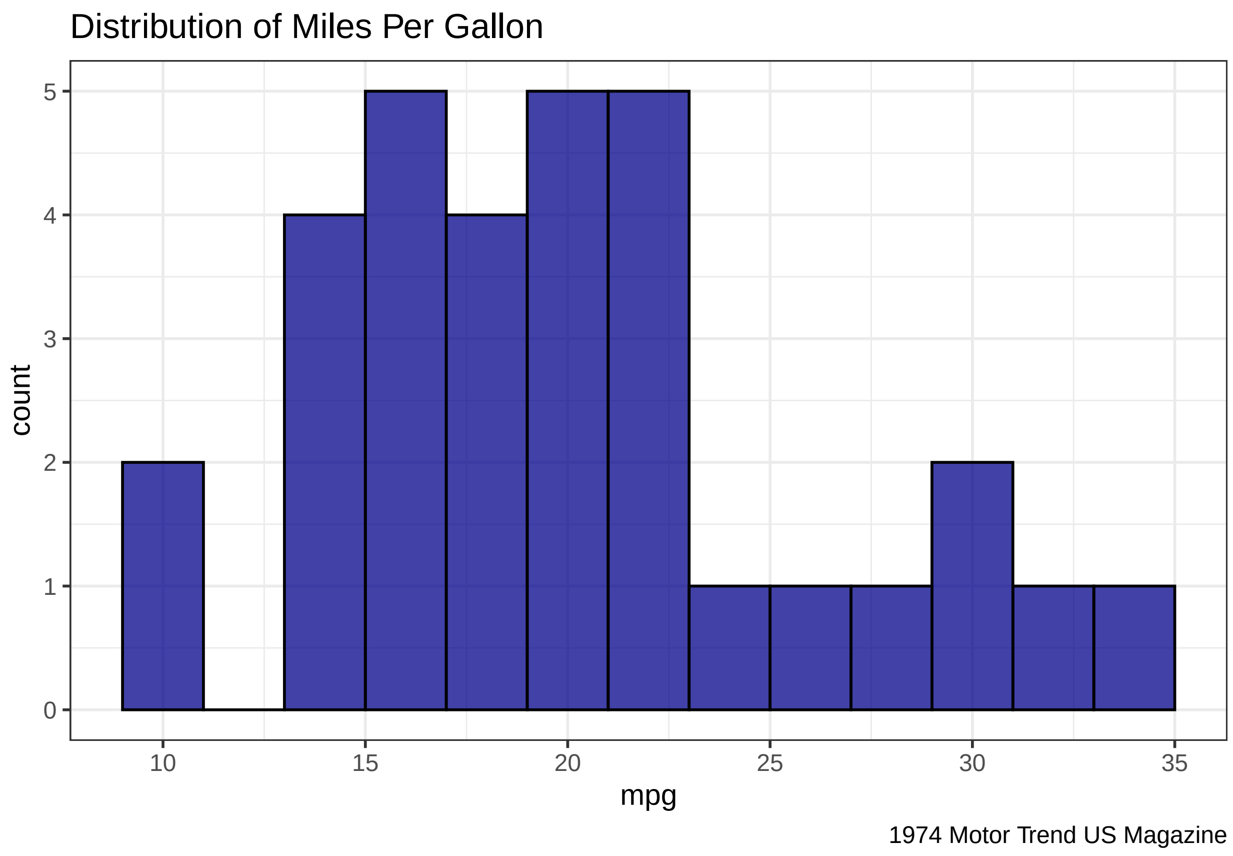
```
mtcars%>% select(mpg, cyl,hp) %>%
  tbl_summary(statistic = list(all_continuous() ~ c("{mean} ({sd})",
                                                    "{median} ({p25}, {p75})",
                                                    "{min}, {max}"),
                              all_categorical() ~ "{n} / {N} ({p}%)"),
              type = all_continuous() ~ "continuous2"
  )
```

Characteristic	**N = 32**
mpg	
Mean (SD)	20.1 (6.0)
Median (IQR)	19.2 (15.4, 22.8)
Range	10.4, 33.9
cyl	
4	11 / 32 (34%)
6	7 / 32 (22%)
8	14 / 32 (44%)
hp	
Mean (SD)	147 (69)
Median (IQR)	123 (96, 180)
Range	52, 335

4.3 Histograms

4.3.1 ggplot

```
ggplot(mtcars, aes(mpg))+
  geom_histogram(binwidth = 2,col = 'black', fill = 'darkblue', alpha = 0.75)+
  labs(title = 'Distribution of Miles Per Gallon', caption = "1974 Motor Trend US Magazine")+
  theme_bw()
```



4.3.2 plot_ly

Since this is an interactive graph, it will not show up in a .pdf file, but it is great to look at in an .html document.

```
plot_ly(x = ~mtcars$mpg, type = "histogram", alpha = 0.6) %>%
  layout(title = 'Distribution of Miles Per Gallon',
         xaxis = list(title = 'Miles Per Gallon'),
         yaxis = list(title = 'Count'))
```

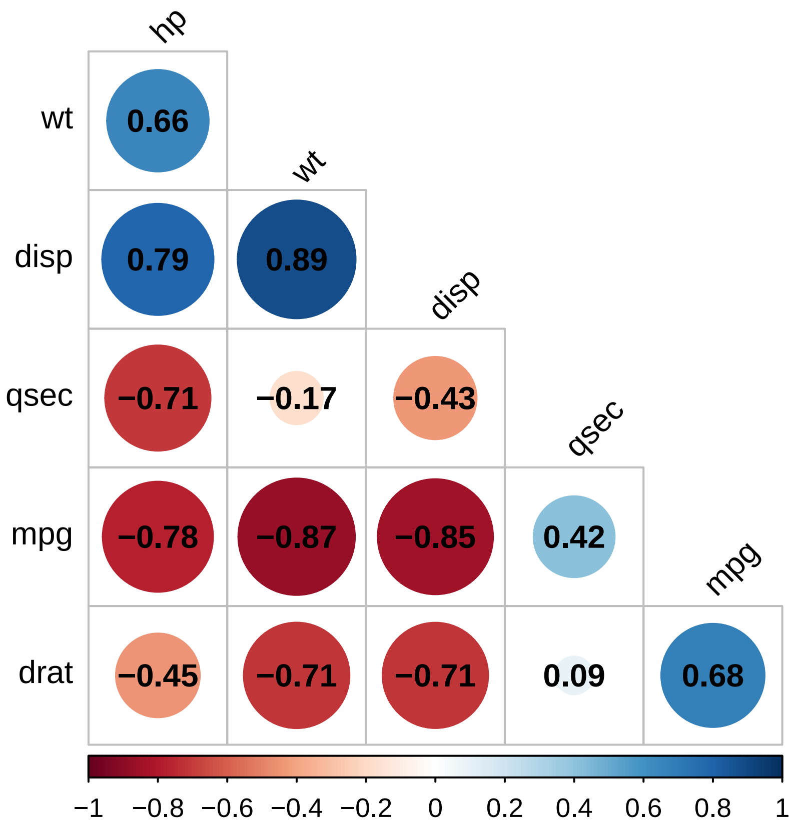
4.4 Boxplots

I think the plotly version of a boxplot is superior since it is interactive.

```
plot_ly(y = starwars$height, type = 'box', name = 'Height [cm]',text = starwars$name) %>%
  layout(title = 'Distribution of Star Wars Character Heights')
```

4.5 Correlation

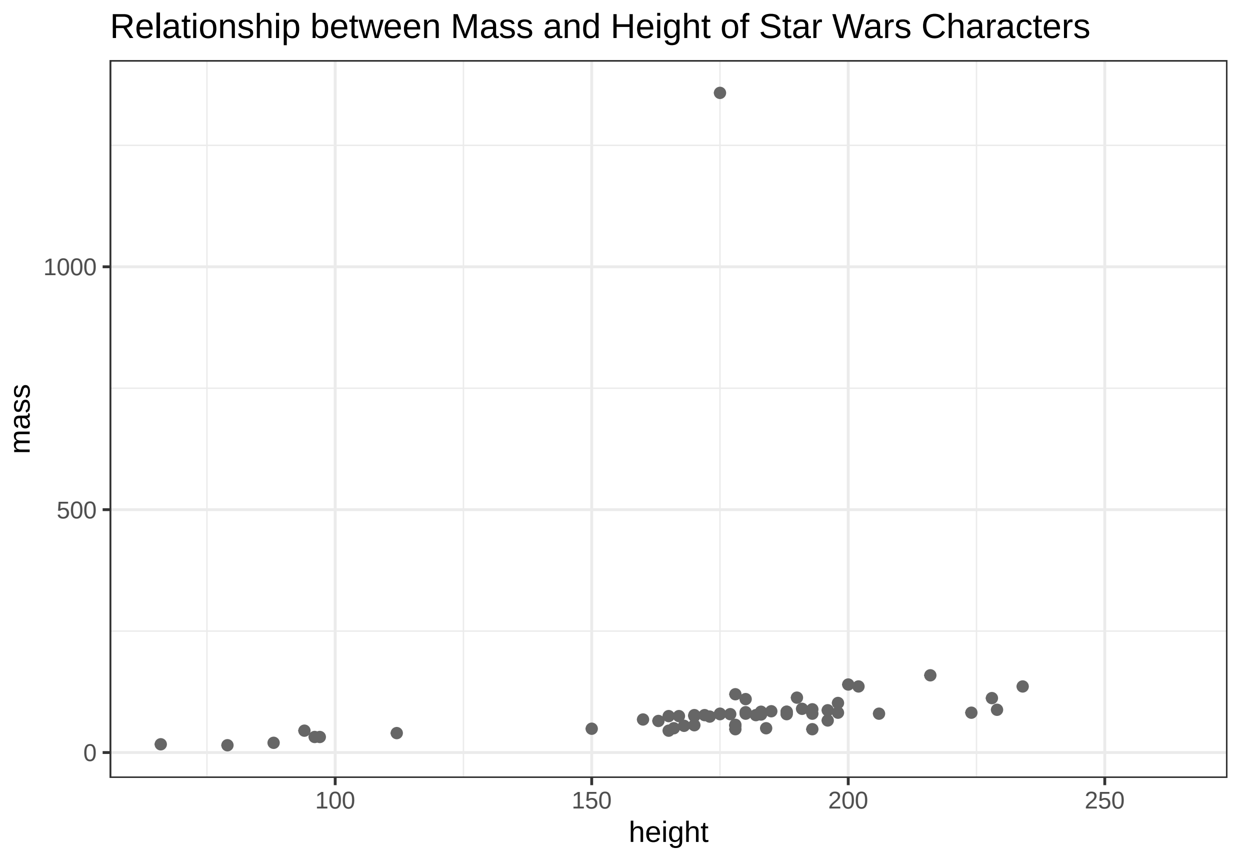
```
reduced <- mtcars %>%
  select(mpg, hp, wt,qsec,disp,drat)

corrplot(cor(reduced),
         type = "lower",
         order = "hclust",
         tl.col = "black",
         tl.srt = 45,
         addCoef.col = "black",
         diag = FALSE)
```



4.6 Scatterplots

4.6.1 ggplot

```
ggplot(starwars,aes(height, mass))+
  geom_point(color = 'gray40')+
  theme_bw()+
  labs(title = "Relationship between Mass and Height of Star Wars Characters")
```



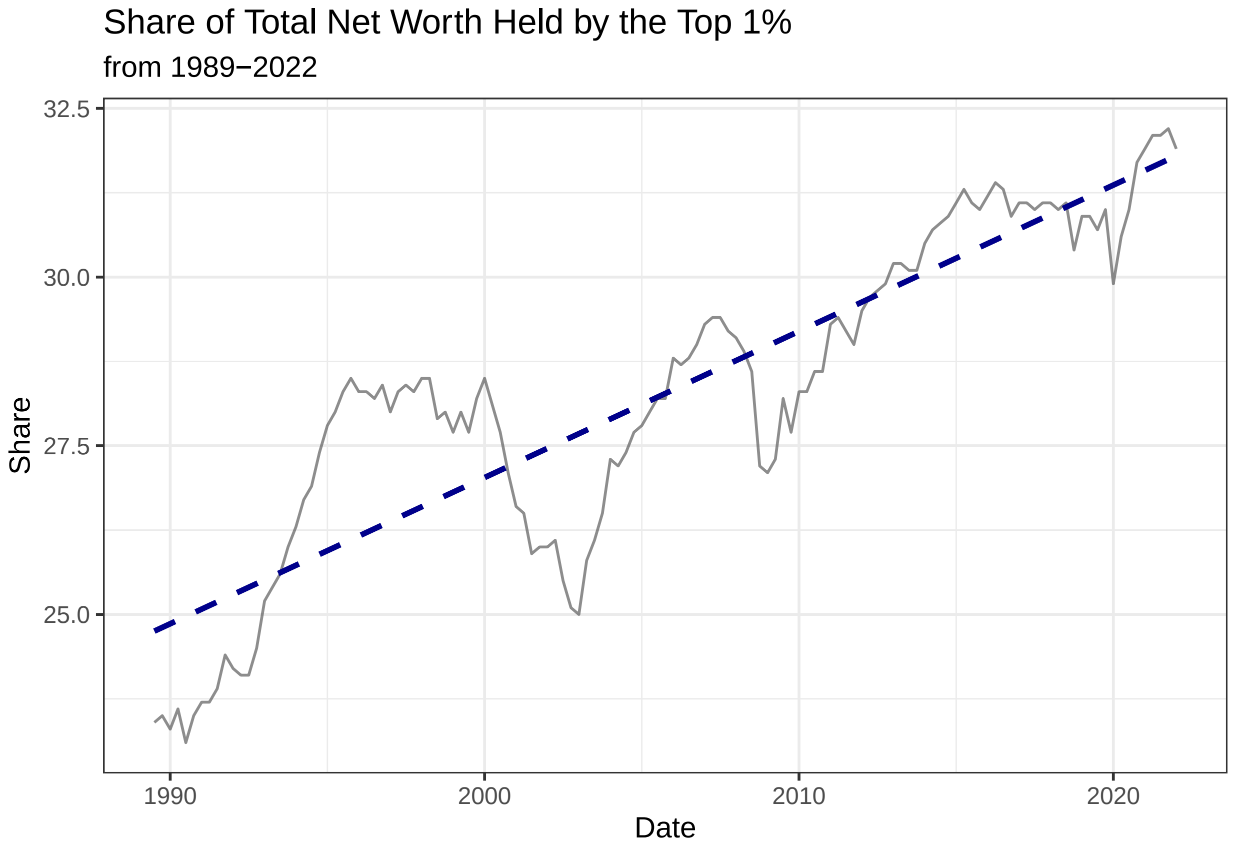
4.6.2 plotly

```
plot_ly(starwars, y = ~mass, x = ~height, type = 'scatter',text = ~name, mode = "markers")
```

4.7 Trends

4.7.1 ggplot

```
ggplot(Wealth1percent, aes(quarter, Share))+
  geom_line(color = 'gray40',alpha = 0.75)+
  geom_smooth(method = "lm", se = F, color = 'darkblue', linetype = 'dashed')+
  theme_bw()+
  labs(title = "Share of Total Net Worth Held by the Top 1%",
       subtitle = 'from 1989-2022',
       x = "Date",
       y = "Share")
```



4.7.2 plotly

To add the trend line, it gets quite tricky.

```
model <- lm(Share~quarter,Wealth1percent)
Trend <- predict(model,data = Wealth1percent$quarter)

plot_ly(Wealth1percent, x = ~quarter, y = ~Share, type = 'scatter', mode = 'lines', name = "Share") %>%
add_trace(y = ~Trend, name = 'Trend', mode = 'lines')
```