

Tutorial

Alexandre Piche

2017-03-06

Contents

McGill wins \$84 million grant for neuroscience	1
To Do Before the Tutorial	1
Download R	1
Download R studio	1
Opening RStudio for the First Time	2
New Project	2
Rmarkdown	2
Required Packages	2
Tutorial	2
Use R for Basic Operations	2
Basic Data Structure	3
Data Cleaning	4
Exploring the Data	6
Real World Problems	15
Ressources	24
Datacamp	24
R for Data Science	24
Advanced R	24

McGill wins \$84 million grant for neuroscience

Great opportunities for undergraduate students with a background in neuro and knowledge of statistics and statistical software!

<https://www.mcgill.ca/newsroom/channels/news/mcgill-wins-84-million-grant-neuroscience-262441>

To Do Before the Tutorial

Download R

<https://cran.r-project.org/>

Download R studio

<https://www.rstudio.com/products/rstudio/download/>

Opening RStudio for the First Time

On the right hand side there is the console. It is where we are going to communicate with R by submitting our instructions.

On the left hand side you have the Environment and the History in the top panel. The Environment lists all the variables that you currently have in your work space (i.e. that you can call in the console). History registers all the operations you have sent to R. You can browse it to see your previous commands in the console.

New Project

I encourage creating a new project for the course as File -> New Project... -> New Directory -> Empty Project -> Directory Name -> R_Tutorial and browse to choose the folder where you want the project to be. It will be easier to manage your project and dataset.

Rmarkdown

You may export your code and graphs easily to MS word or pdf following these instructions.

File -> New File -> R markdown -> select the output format to Word -> Ok -> Knit

Note that what is written in MS Word is not connected to the console, and vice-versa. You will need to write the command in both places if you want to keep their workspaces the same.

To insert code in your file click on “Insert” at the top of the top left panel.

Required Packages

We need to install certain packages for today’s tutorial. It can be done this way:

```
# install.packages(c("tidyr", "reshape2"))
```

Tutorial

Use R for Basic Operations

Typing in the console we can use R as a basic calculator.

```
1+2
```

```
## [1] 3
```

```
4/2
```

```
## [1] 2
```

```
5*(1+2)
```

```
## [1] 15
```

```
5^2
```

```
## [1] 25
```

We can also assign value to variable,

```
my_variable <- 7
```

and access it by calling the variable name in the console.

```
my_variable
```

```
## [1] 7
```

Basic Data Structure

Vectors

We can create vectors using the “c” command, where “c” stands for combine.

```
x <- c(6,7,8,9,10)
```

```
x
```

```
## [1] 6 7 8 9 10
```

```
x[1]
```

```
## [1] 6
```

```
x[c(2,4)]
```

```
## [1] 7 9
```

```
x*2
```

```
## [1] 12 14 16 18 20
```

```
x <- 1:7
```

```
x
```

```
## [1] 1 2 3 4 5 6 7
```

Vectors in R are more general than their mathematical equivalent. They can hold different type of data e.g. string or level.

```
y <- c("hello", "world")
```

```
y
```

```
## [1] "hello" "world"
```

However, all elements will be coerced to the same type, they are coerced to string in the following case.

```
y <- c(1, "hello")
```

```
y
```

```
## [1] "1" "hello"
```

Dataframe

Usually, we will have to load our own datasets in R using the “Import Dataset” command in the Environment panel, more on that later. For simplicity, we will now experiment on a built-in dataset in R studio named “mtcars”. Using the command “head(mtcars)” let us access the first 6 records of the dataset and their names.

```
dim(mtcars)
```

```
## [1] 32 11
```

```
head(mtcars)
```

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

We can use vectors to assess different rows and columns.

```
mtcars[1:4, c(1,4)]
```

```
##           mpg  hp
## Mazda RX4      21.0 110
## Mazda RX4 Wag  21.0 110
## Datsun 710     22.8  93
## Hornet 4 Drive 21.4 110
```

Let say that we are interested in the median miles per gallon (mpg) in this dataset. We can use the command “median(mtcars\$mpg)”, where “mtcars” is the dataset and “mpg” is the column of interest that we accessed with the dollar sign operator “\$”.

```
mtcars$mpg
```

```
## [1] 21.0 21.0 22.8 21.4 18.7 18.1 14.3 24.4 22.8 19.2 17.8 16.4 17.3 15.2
## [15] 10.4 10.4 14.7 32.4 30.4 33.9 21.5 15.5 15.2 13.3 19.2 27.3 26.0 30.4
## [29] 15.8 19.7 15.0 21.4
```

```
median(mtcars$mpg)
```

```
## [1] 19.2
```

```
mean(mtcars$mpg)
```

```
## [1] 20.09062
```

```
var(mtcars$mpg)
```

```
## [1] 36.3241
```

To have more information on a function, we can use the question mark before it e.g. “?mean”.

Data Cleaning

Most of the analysis consist of data cleaning. Here is an example of some manipulation that you might be asked to do.

#	Attribute Description
1.	mpg - Miles/(US) gallon
2.	cyl - Number of cylinders
3.	disp - Displacement (cu.in.)
4.	hp - Gross horsepower
5.	drat - Rear axle ratio
6.	wt - Weight (1000 lbs)
7.	qsec - 1/4 mile time
8.	vs - V/S

#	Attribute Description
9.	am - Transmission (0 = automatic, 1 = manual)
10.	gear - Number of forward gears
11.	carb - Number of carburetors

```
str(mtcars)
```

```
## 'data.frame': 32 obs. of 11 variables:
## $ mpg : num 21 21 22.8 21.4 18.7 18.1 14.3 24.4 22.8 19.2 ...
## $ cyl : num 6 6 4 6 8 6 8 4 4 6 ...
## $ disp: num 160 160 108 258 360 ...
## $ hp : num 110 110 93 110 175 105 245 62 95 123 ...
## $ drat: num 3.9 3.9 3.85 3.08 3.15 2.76 3.21 3.69 3.92 3.92 ...
## $ wt : num 2.62 2.88 2.32 3.21 3.44 ...
## $ qsec: num 16.5 17 18.6 19.4 17 ...
## $ vs : num 0 0 1 1 0 1 0 1 1 1 ...
## $ am : num 1 1 1 0 0 0 0 0 0 0 ...
## $ gear: num 4 4 4 3 3 3 3 4 4 4 ...
## $ carb: num 4 4 1 1 2 1 4 2 2 4 ...
```

Should the number of cylinders and the weight be both considered as numeric variable? Weight is a continuous variable while the number of cylinders is categorical e.g. 4,6 or 8 cylinders.

```
mtcars$cyl <- factor(mtcars$cyl)
```

```
mtcars$vs <- factor(mtcars$vs)
```

```
mtcars$am <- factor(mtcars$am)
```

```
mtcars$gear <- factor(mtcars$gear)
```

```
mtcars$carb <- factor(mtcars$carb)
```

```
str(mtcars)
```

```
## 'data.frame': 32 obs. of 11 variables:
## $ mpg : num 21 21 22.8 21.4 18.7 18.1 14.3 24.4 22.8 19.2 ...
## $ cyl : Factor w/ 3 levels "4","6","8": 2 2 1 2 3 2 3 1 1 2 ...
## $ disp: num 160 160 108 258 360 ...
## $ hp : num 110 110 93 110 175 105 245 62 95 123 ...
## $ drat: num 3.9 3.9 3.85 3.08 3.15 2.76 3.21 3.69 3.92 3.92 ...
## $ wt : num 2.62 2.88 2.32 3.21 3.44 ...
## $ qsec: num 16.5 17 18.6 19.4 17 ...
## $ vs : Factor w/ 2 levels "0","1": 1 1 2 2 1 2 1 2 2 2 ...
## $ am : Factor w/ 2 levels "0","1": 2 2 2 1 1 1 1 1 1 1 ...
## $ gear: Factor w/ 3 levels "3","4","5": 2 2 2 1 1 1 1 2 2 2 ...
## $ carb: Factor w/ 6 levels "1","2","3","4",...: 4 4 1 1 2 1 4 2 2 4 ...
```

It might be annoying to have to remember that 0 stands for automatic and 1 manual transmission. Let us fix it.

```
levels(mtcars$am) <- c("automatic", "manual")
str(mtcars)
```

```
## 'data.frame': 32 obs. of 11 variables:
## $ mpg : num 21 21 22.8 21.4 18.7 18.1 14.3 24.4 22.8 19.2 ...
```

```
## $ cyl : Factor w/ 3 levels "4","6","8": 2 2 1 2 3 2 3 1 1 2 ...
## $ disp: num 160 160 108 258 360 ...
## $ hp : num 110 110 93 110 175 105 245 62 95 123 ...
## $ drat: num 3.9 3.9 3.85 3.08 3.15 2.76 3.21 3.69 3.92 3.92 ...
## $ wt : num 2.62 2.88 2.32 3.21 3.44 ...
## $ qsec: num 16.5 17 18.6 19.4 17 ...
## $ vs : Factor w/ 2 levels "0","1": 1 1 2 2 1 2 1 2 2 2 ...
## $ am : Factor w/ 2 levels "automatic","manual": 2 2 2 1 1 1 1 1 1 1 ...
## $ gear: Factor w/ 3 levels "3","4","5": 2 2 2 1 1 1 1 2 2 2 ...
## $ carb: Factor w/ 6 levels "1","2","3","4",...: 4 4 1 1 2 1 4 2 2 4 ...
```

Exploring the Data

Graphing with ggplot2

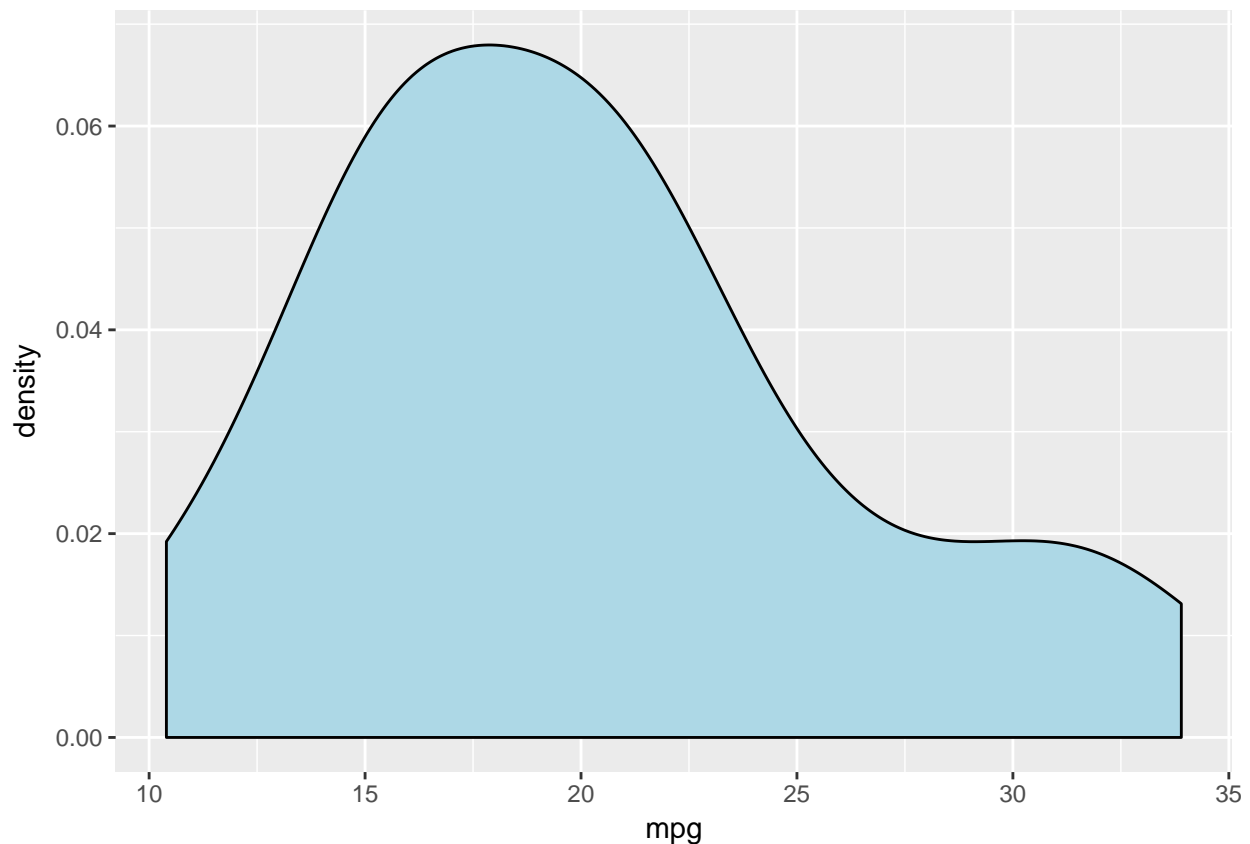
Exploring the Variability in a Dataset

Suppose being interested in the distribution of mpg. The first argument is the dataset name, the second is “aes” (standing for aesthetic) which that the dependent and independent variables.

We then add a layer to the plot using the “+” sign e.g. the density:

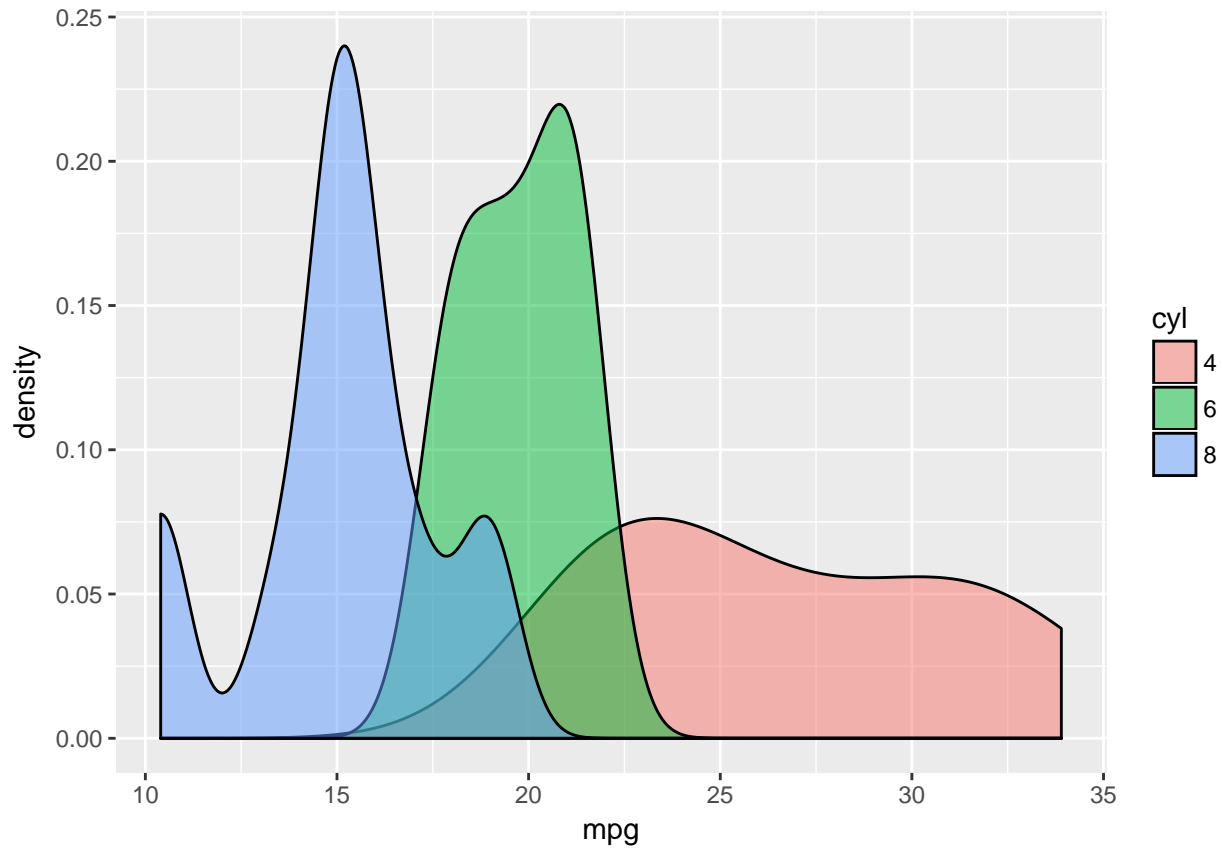
```
# To use the ggplot2 library we have to call it with the library command
# One could think of the command "install.packages" as buying shoes and "library" as putting them on.
library(ggplot2)

ggplot(mtcars, aes(mpg)) + geom_density(fill="lightblue")
```



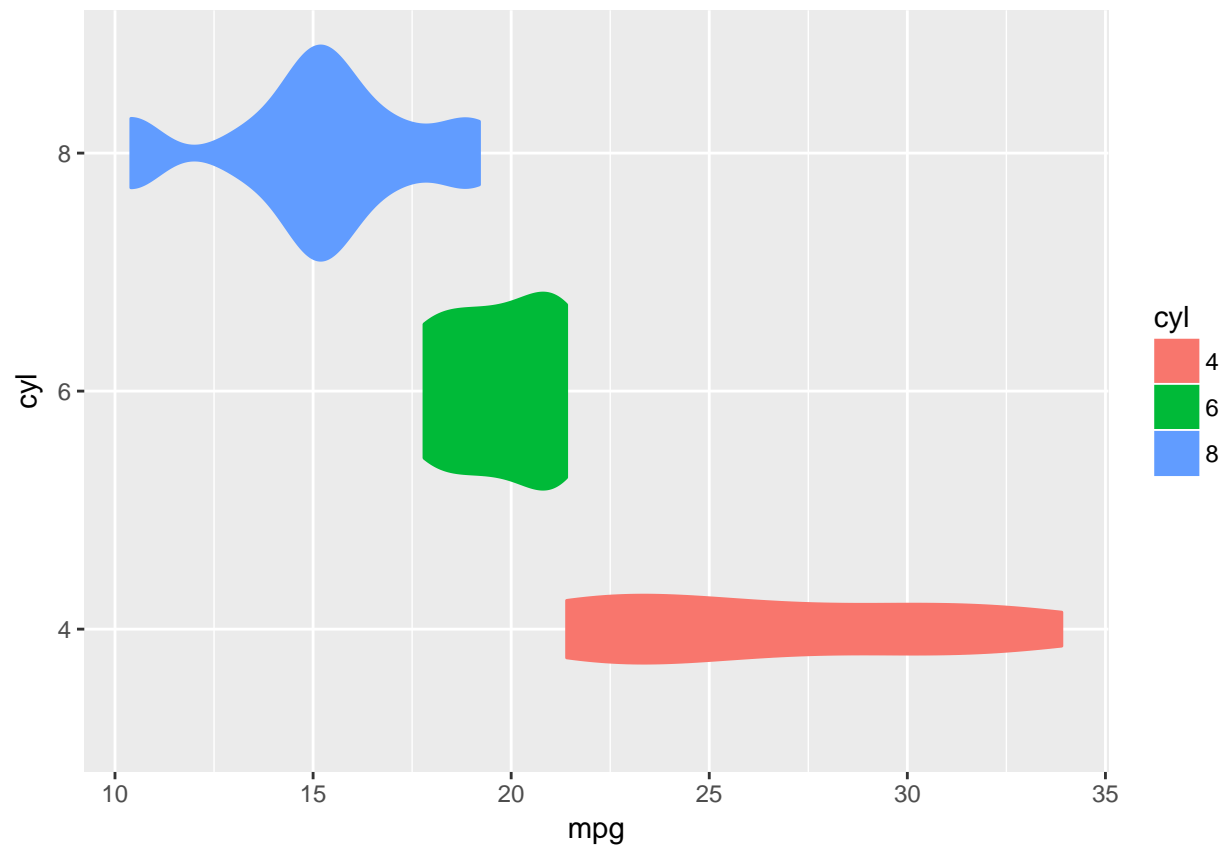
The number of cylinders in a car might impact its consumption. Let us stratify the densities by cylinders. Note the transparency of the plot using the parameter `alpha`.

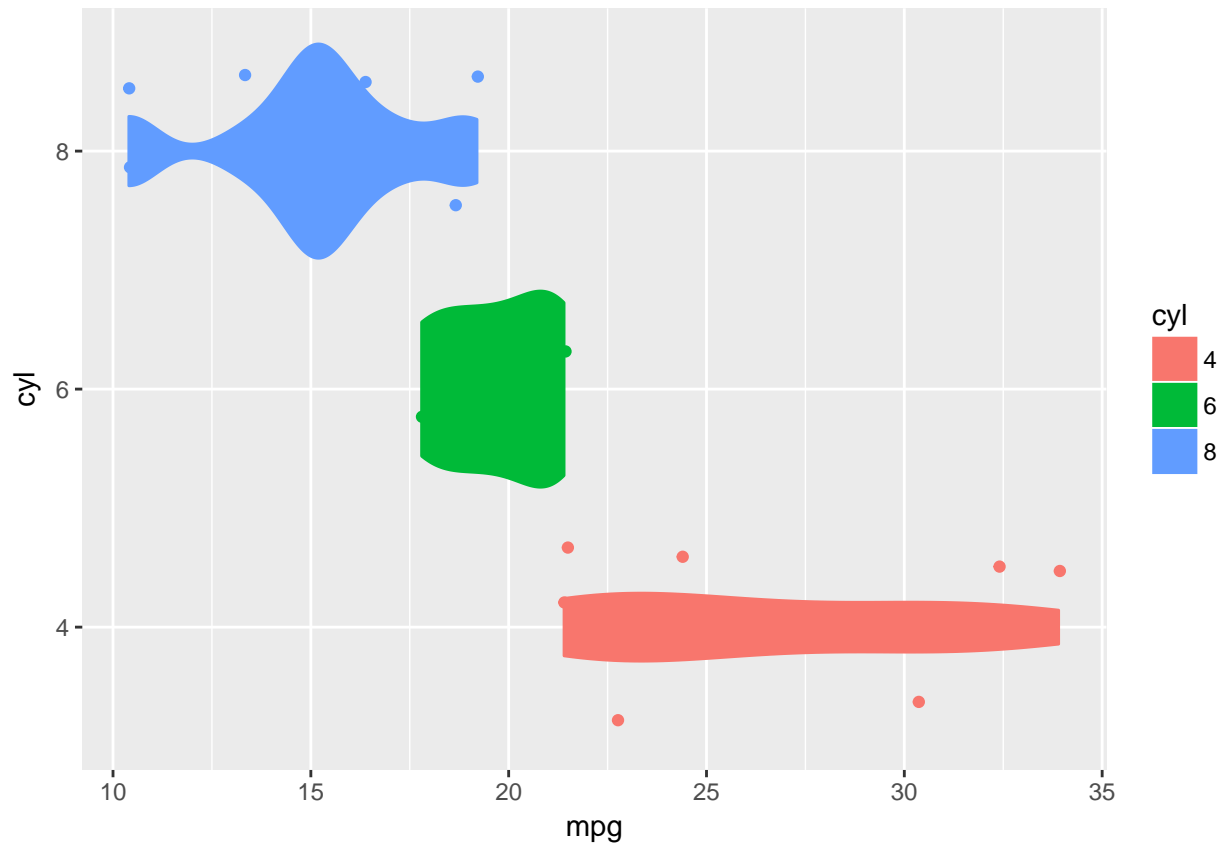
```
ggplot(mtcars, aes(mpg, fill=cyl)) + geom_density(alpha=.5)
```



It might be useful to be able to see every observation on the graph. We can do so by adding the layer: `geom_jitter`.

```
p <- ggplot(mtcars, aes(cyl, mpg, fill=cyl, color=cyl)) + geom_violin() + coord_flip()
p
```

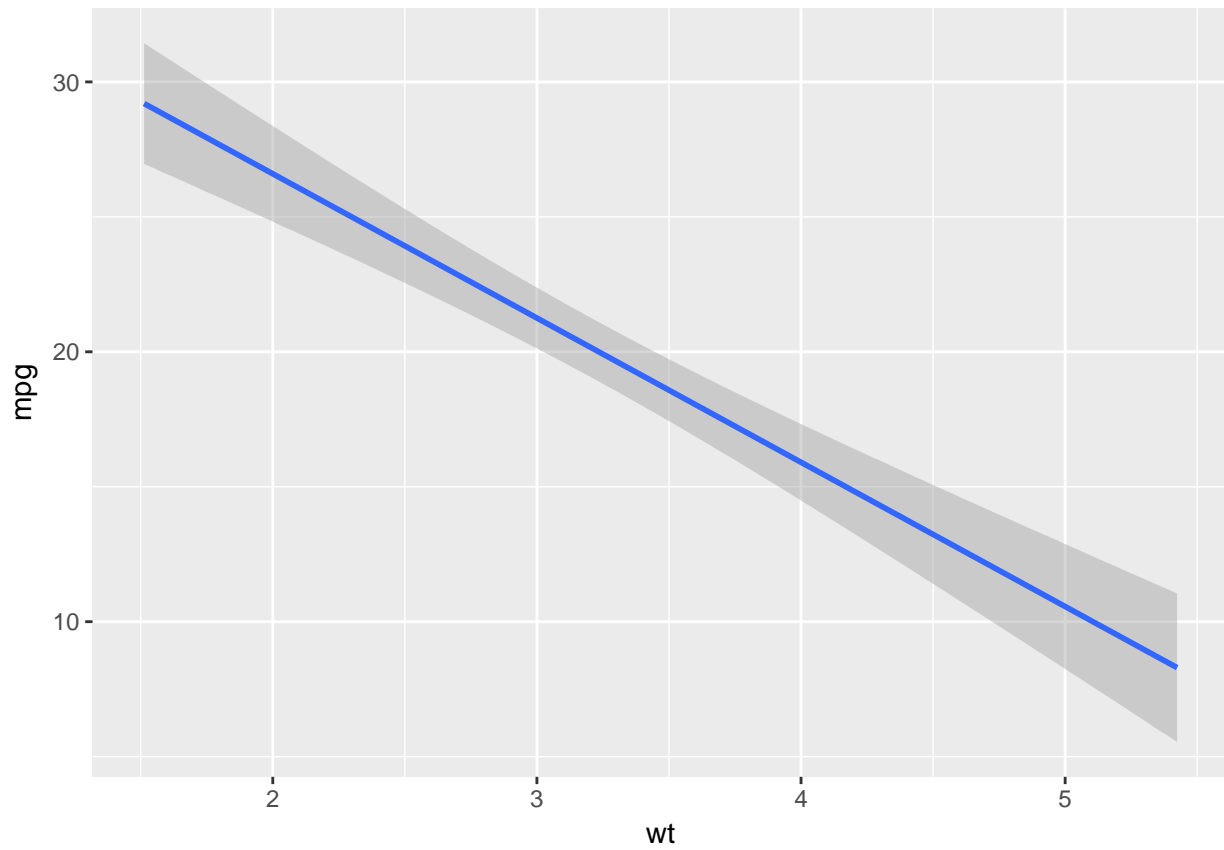




Relationship Across Variables

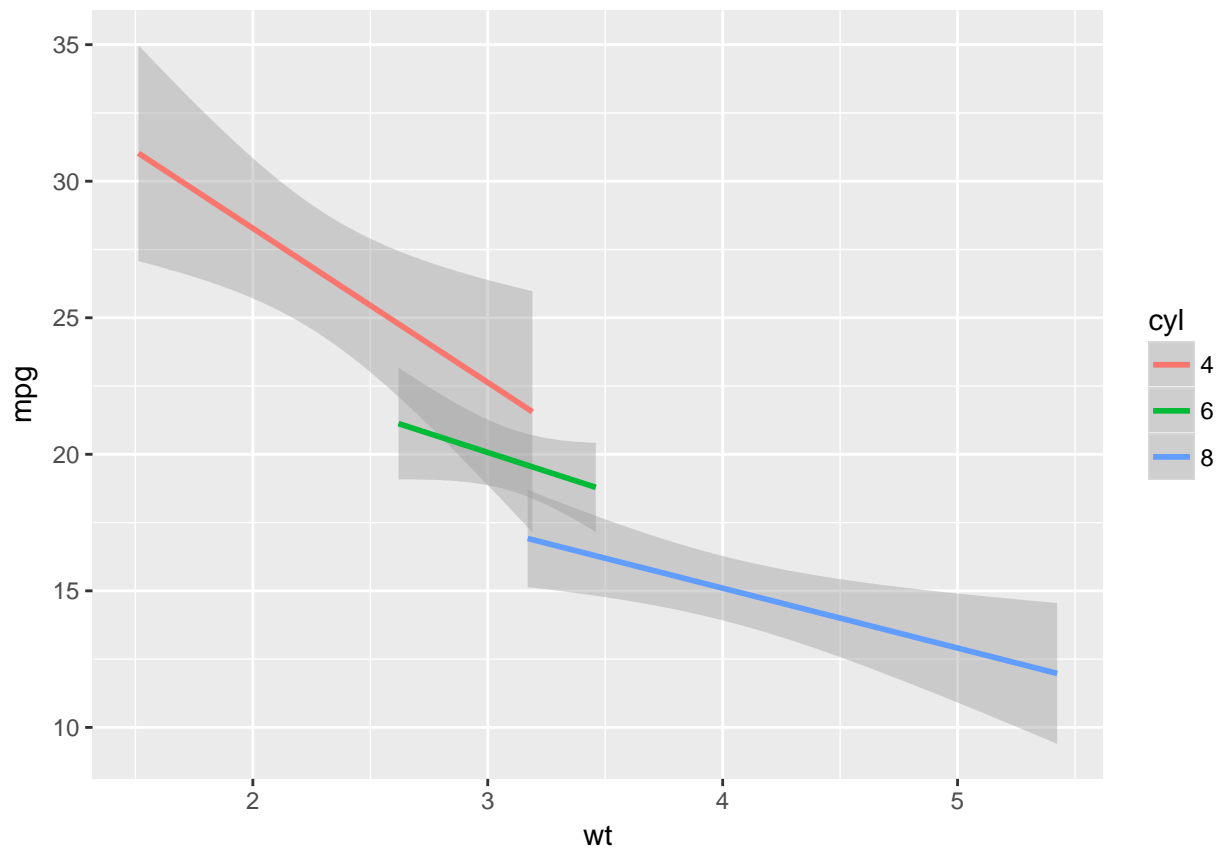
Using the regression, we can clearly see that the heavier a vehicle is the worst Miles per gallon it is going to have.

```
ggplot(mtcars, aes(wt, mpg)) + stat_smooth(method="lm")
```

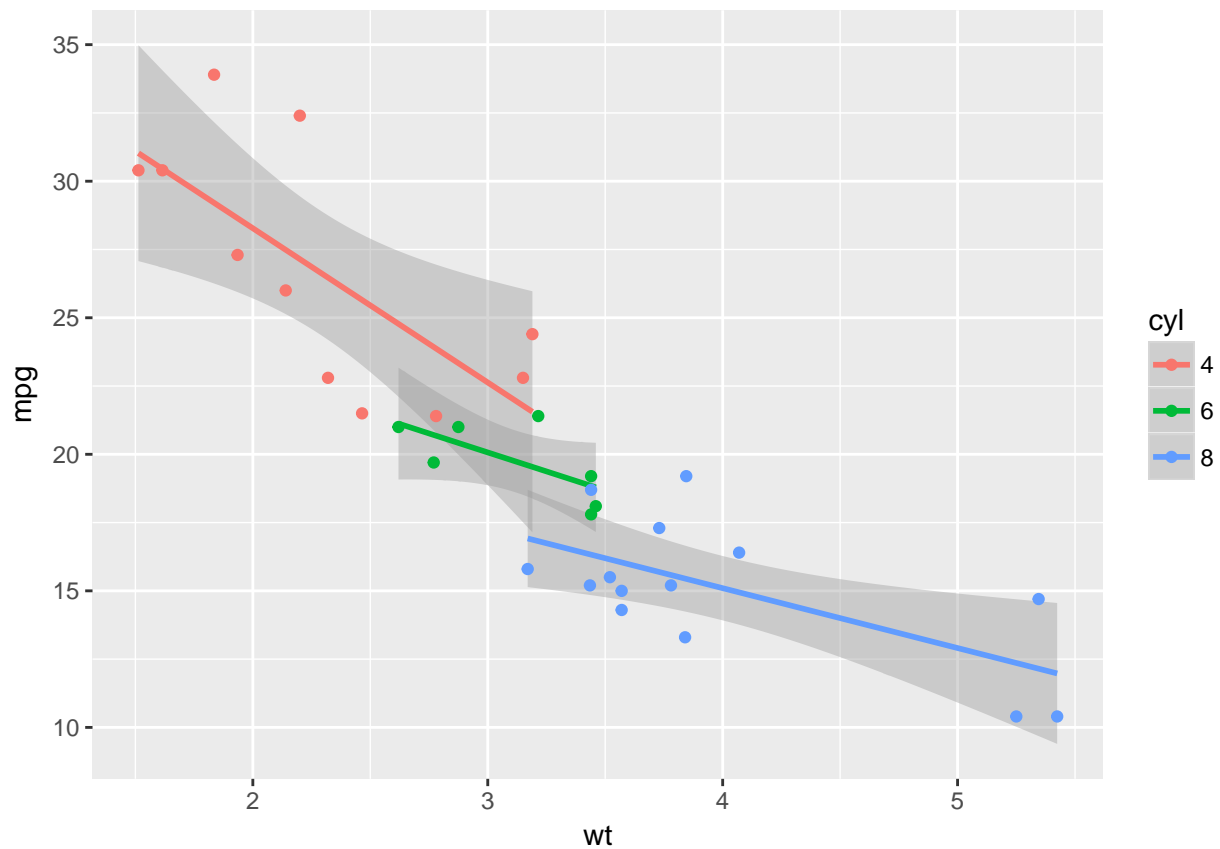


Furthermore, we can divide the regression by the number of cylinders. We can see that the negative correlation is still present, but is not the same for every category.

```
p <- ggplot(mtcars, aes(wt, mpg, color=cyl)) + stat_smooth(method="lm")
p
```



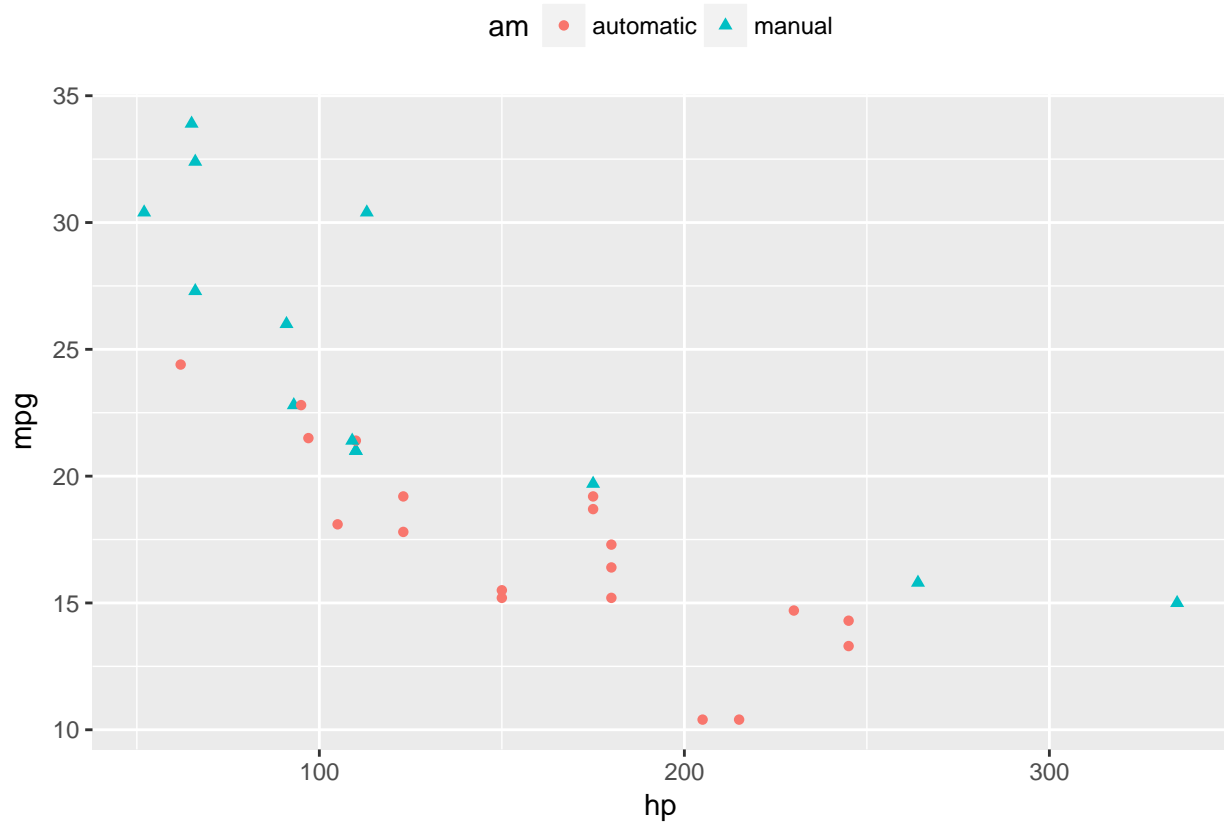
```
p + geom_point()
```



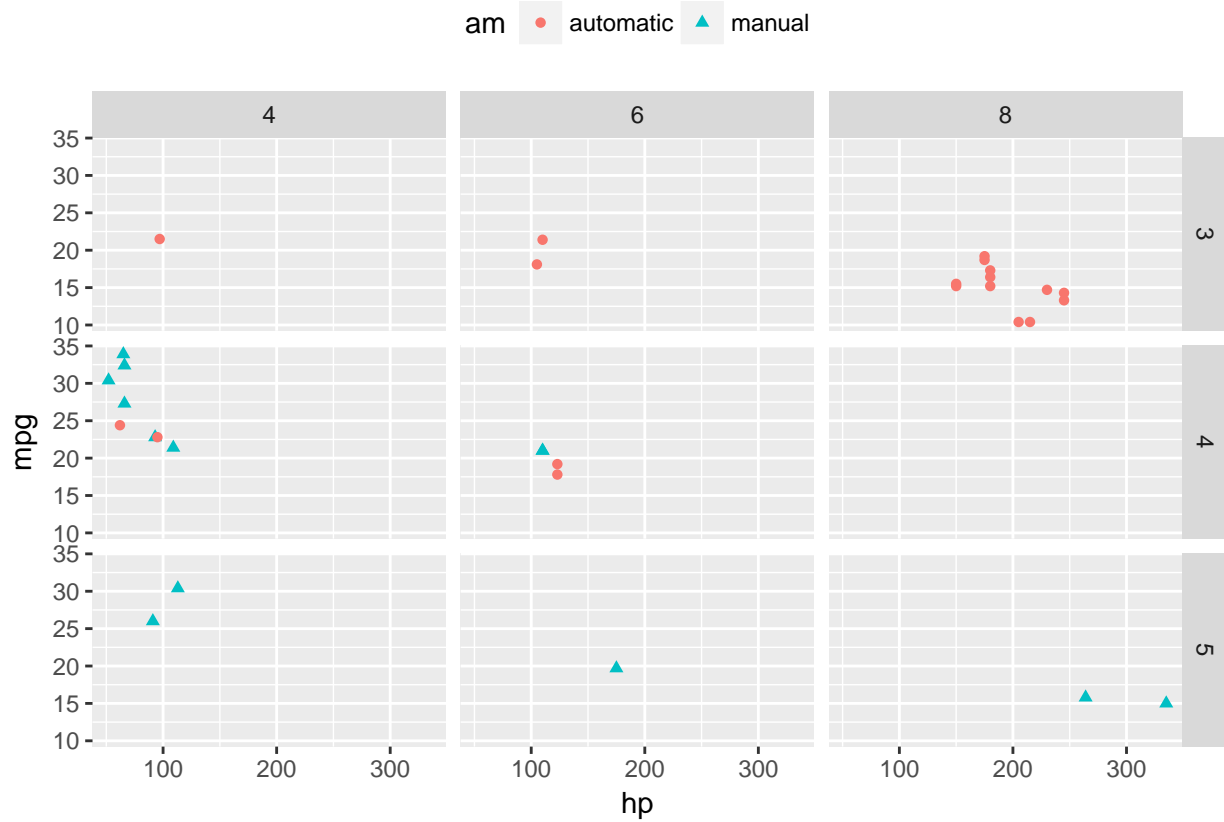
Complex Patterns

We can stratify the data to observe more complex patterns.

```
p <- ggplot(mtcars, aes(hp, mpg, color=am, shape=am)) + geom_point()
p <- p + theme(legend.position="top")
p
```



```
p + facet_grid(gear~cyl)
```



dplyr

The package dplyr is very fast for large datasets. It has main functions/verbs to work with data.table that we are going to explore here. The functions are filter, select, mutate, arrange, and summarise.

```
library(dplyr)
```

Group by and Summarise

```
summarise(mtcars, mean_mpg=mean(mpg), mean_wt=mean(wt))
```

```
##   mean_mpg mean_wt
## 1 20.09062 3.21725
```

```
temp <- group_by(mtcars, cyl)
temp1 <- summarise(temp, mean_mpg=mean(mpg), mean_wt=mean(wt))
temp1
```

```
## # A tibble: 3 × 3
##   cyl mean_mpg mean_wt
##   <fctr>   <dbl>   <dbl>
## 1     4 26.66364 2.285727
## 2     6 19.74286 3.117143
## 3     8 15.10000 3.999214
```

```
# one could alternatively chain the operation using the pipe operation %>%
# mtcars %>% group_by(cyl) %>% summarise(mean_mpg=mean(mpg), mean_wt=mean(wt))
```

Select

```
mtcars %>% select(mpg, cyl) %>% head()
```

```
##           mpg cyl
## Mazda RX4    21.0   6
## Mazda RX4 Wag 21.0   6
## Datsun 710    22.8   4
## Hornet 4 Drive 21.4   6
## Hornet Sportabout 18.7   8
## Valiant      18.1   6
```

Arrange

```
mtcars %>% select(cyl,am,wt) %>% arrange(cyl,am,wt) %>% head()
```

```
##   cyl   am   wt
## 1   4 automatic 2.465
## 2   4 automatic 3.150
## 3   4 automatic 3.190
## 4   4   manual 1.513
## 5   4   manual 1.615
## 6   4   manual 1.835
```

Mutate

Let's define heavy as more than 3000 lbs.

```
temp1 <- mtcars %>% mutate(heavy=factor(ifelse(wt < 3, "Light", "Heavy")))
str(temp1)
```

```
## 'data.frame': 32 obs. of 12 variables:
## $ mpg : num 21 21 22.8 21.4 18.7 18.1 14.3 24.4 22.8 19.2 ...
## $ cyl : Factor w/ 3 levels "4","6","8": 2 2 1 2 3 2 3 1 1 2 ...
## $ disp : num 160 160 108 258 360 ...
## $ hp : num 110 110 93 110 175 105 245 62 95 123 ...
## $ drat : num 3.9 3.9 3.85 3.08 3.15 2.76 3.21 3.69 3.92 3.92 ...
## $ wt : num 2.62 2.88 2.32 3.21 3.44 ...
## $ qsec : num 16.5 17 18.6 19.4 17 ...
## $ vs : Factor w/ 2 levels "0","1": 1 1 2 2 1 2 1 2 2 2 ...
## $ am : Factor w/ 2 levels "automatic","manual": 2 2 2 1 1 1 1 1 1 1 ...
## $ gear : Factor w/ 3 levels "3","4","5": 2 2 2 1 1 1 1 2 2 2 ...
## $ carb : Factor w/ 6 levels "1","2","3","4",...: 4 4 1 1 2 1 4 2 2 4 ...
## $ heavy: Factor w/ 2 levels "Heavy","Light": 2 2 2 1 1 1 1 1 1 1 ...
```

Filter

```
temp1 %>% filter(heavy=="Heavy") %>% head()
```

```
##   mpg cyl  disp  hp drat   wt  qsec vs      am gear carb heavy
## 1 21.4   6 258.0 110 3.08 3.215 19.44 1 automatic    3    1 Heavy
## 2 18.7   8 360.0 175 3.15 3.440 17.02 0 automatic    3    2 Heavy
## 3 18.1   6 225.0 105 2.76 3.460 20.22 1 automatic    3    1 Heavy
## 4 14.3   8 360.0 245 3.21 3.570 15.84 0 automatic    3    4 Heavy
## 5 24.4   4 146.7  62 3.69 3.190 20.00 1 automatic    4    2 Heavy
## 6 22.8   4 140.8  95 3.92 3.150 22.90 1 automatic    4    2 Heavy
```

Real World Problems

Parkinsons

We will try to predict the UPDRS score of the patient given their age, gender and different measures.

Attribute Information:

#	Attribute Description
1.	subject# - Integer that uniquely identifies each subject
2.	age - Subject age
3.	sex - Subject gender '0' - male, '1' - female
4.	test_time - Time since recruitment into the trial. The integer part is the number of days since recruitment.
5.	motor_UPDRS - Clinician's motor UPDRS score, linearly interpolated
6.	total_UPDRS - Clinician's total UPDRS score, linearly interpolated
7.	Jitter(%),Jitter(Abs),Jitter:RAP,Jitter:PPQ5,Jitter:DDP - Several measures of variation in fundamental frequency
8.	Shimmer,Shimmer(dB),Shimmer:APQ3,Shimmer:APQ5,Shimmer:APQ11,Shimmer:DDA - Several measures of variation in amplitude
9.	NHR,HNR - Two measures of ratio of noise to tonal components in the voice
10.	RPDE - A nonlinear dynamical complexity measure
11.	DFA - Signal fractal scaling exponent

#	Attribute Description
12.	PPE - A nonlinear measure of fundamental frequency variation

data.table

The data.table package provides a very efficient back end for dplyr. I recommend using it if you are working with large datasets. Particularly “fread” to read in datasets.

```
library(data.table)
```

```
# A Tsanas, MA Little, PE McSharry, LO Ramig (2009)
# 'Accurate telemonitoring of Parkinson.s disease progression by non-invasive speech tests',
# url <- "https://archive.ics.uci.edu/ml/machine-learning-databases/
# parkinsons/telemonitoring/parkinsons_updrs.data"
```

```
#parkinson_dat <- fread(url)
#names(parkinson_dat)[1] <- "subject"
```

```
#write.csv(parkinson_dat, file="parkinson_dat.csv")
```

```
parkinson_dat <- fread("parkinson_dat.csv")
```

```
str(parkinson_dat)
```

```
## Classes 'data.table' and 'data.frame': 5875 obs. of 23 variables:
```

```
## $ V1 : chr "1" "2" "3" "4" ...
```

```
## $ subject : int 1 1 1 1 1 1 1 1 1 ...
```

```
## $ age : int 72 72 72 72 72 72 72 72 72 ...
```

```
## $ sex : int 0 0 0 0 0 0 0 0 0 ...
```

```
## $ test_time : num 5.64 12.67 19.68 25.65 33.64 ...
```

```
## $ motor_UPDRS : num 28.2 28.4 28.7 28.9 29.2 ...
```

```
## $ total_UPDRS : num 34.4 34.9 35.4 35.8 36.4 ...
```

```
## $ Jitter(%) : num 0.00662 0.003 0.00481 0.00528 0.00335 0.00353 0.00422 0.00476 0.00432 0.00496
```

```
## $ Jitter(Abs) : num 3.38e-05 1.68e-05 2.46e-05 2.66e-05 2.01e-05 ...
```

```
## $ Jitter:RAP : num 0.00401 0.00132 0.00205 0.00191 0.00093 0.00119 0.00212 0.00226 0.00156 0.002
```

```
## $ Jitter:PPQ5 : num 0.00317 0.0015 0.00208 0.00264 0.0013 0.00159 0.00221 0.00259 0.00207 0.00253
```

```
## $ Jitter:DDP : num 0.01204 0.00395 0.00616 0.00573 0.00278 ...
```

```
## $ Shimmer : num 0.0256 0.0202 0.0168 0.0231 0.017 ...
```

```
## $ Shimmer(dB) : num 0.23 0.179 0.181 0.327 0.176 0.214 0.445 0.212 0.371 0.31 ...
```

```
## $ Shimmer:APQ3 : num 0.01438 0.00994 0.00734 0.01106 0.00679 ...
```

```
## $ Shimmer:APQ5 : num 0.01309 0.01072 0.00844 0.01265 0.00929 ...
```

```
## $ Shimmer:APQ11: num 0.0166 0.0169 0.0146 0.0196 0.0182 ...
```

```
## $ Shimmer:DDA : num 0.0431 0.0298 0.022 0.0332 0.0204 ...
```

```
## $ NHR : num 0.0143 0.0111 0.0202 0.0278 0.0116 ...
```

```
## $ HNR : num 21.6 27.2 23 24.4 26.1 ...
```

```
## $ RPDE : num 0.419 0.435 0.462 0.487 0.472 ...
```

```
## $ DFA : num 0.548 0.565 0.544 0.578 0.561 ...
```

```
## $ PPE : num 0.16 0.108 0.21 0.333 0.194 ...
```

```
## - attr(*, ".internal.selfref")=<externalptr>
```

```
parkinson_dat$sex <- factor(parkinson_dat$sex)
```

```
levels(parkinson_dat$sex) <- c("male", "female")
```

Inspecting the distribution of subjects based on gender.


```
parkinson_dat %>% distinct(subject, .keep_all = TRUE) %>% group_by(sex) %>%
  summarise(count=n(), mean_age=mean(age))
```

```
## # A tibble: 2 × 3
##   sex count mean_age
##   <fctr> <int>   <dbl>
## 1 male    28 64.82143
## 2 female  14 63.57143
```

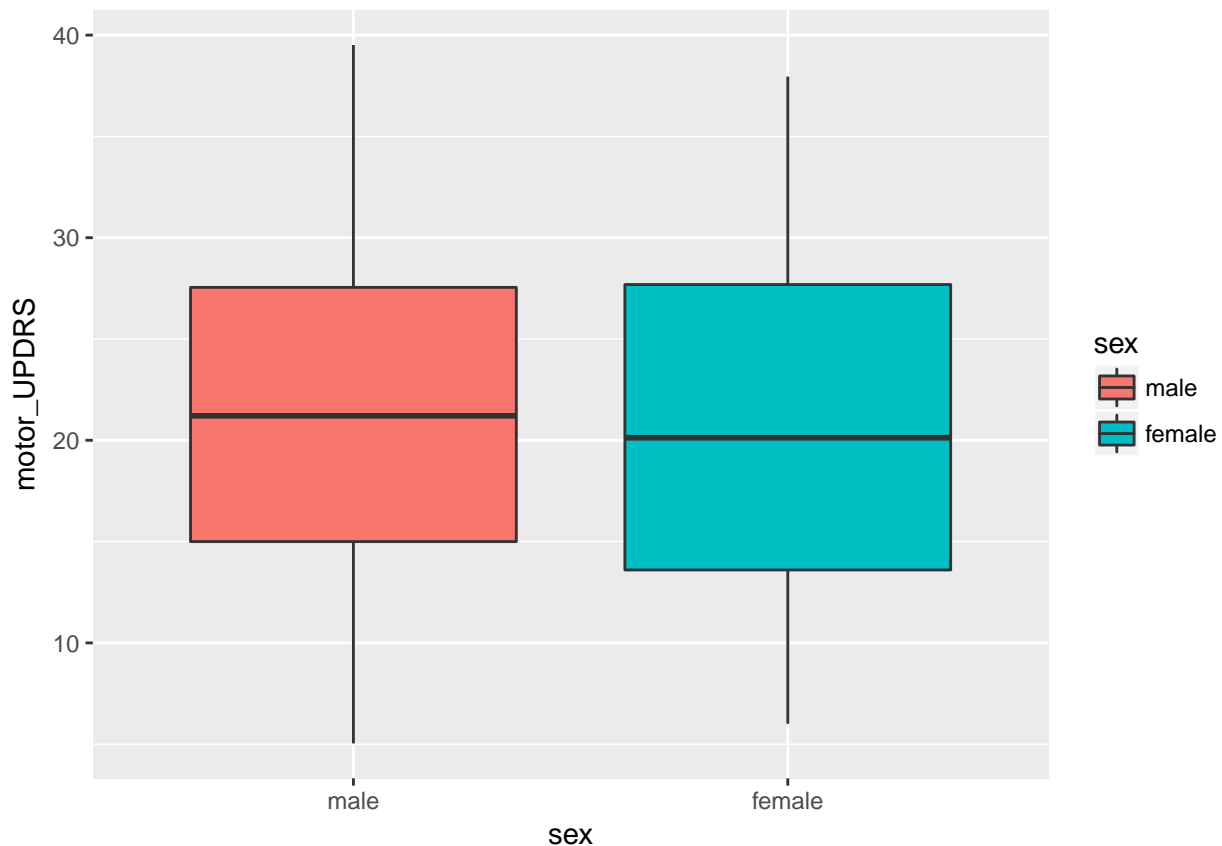
Inspecting the distribution of older subjects based on gender.

```
parkinson_dat_old <- parkinson_dat %>% filter(age >= 65)
parkinson_dat_old %>% distinct(subject, .keep_all = TRUE) %>% group_by(sex) %>%
  summarise(count=n(), mean_age=mean(age))
```

```
## # A tibble: 2 × 3
##   sex count mean_age
##   <fctr> <int>   <dbl>
## 1 male    16 70.56250
## 2 female   7 72.14286
```

Inspecting the score distribution based on gender.

```
library(ggplot2)
ggplot(parkinson_dat, aes(sex, motor_UPDRS, fill=sex)) + geom_boxplot()
```

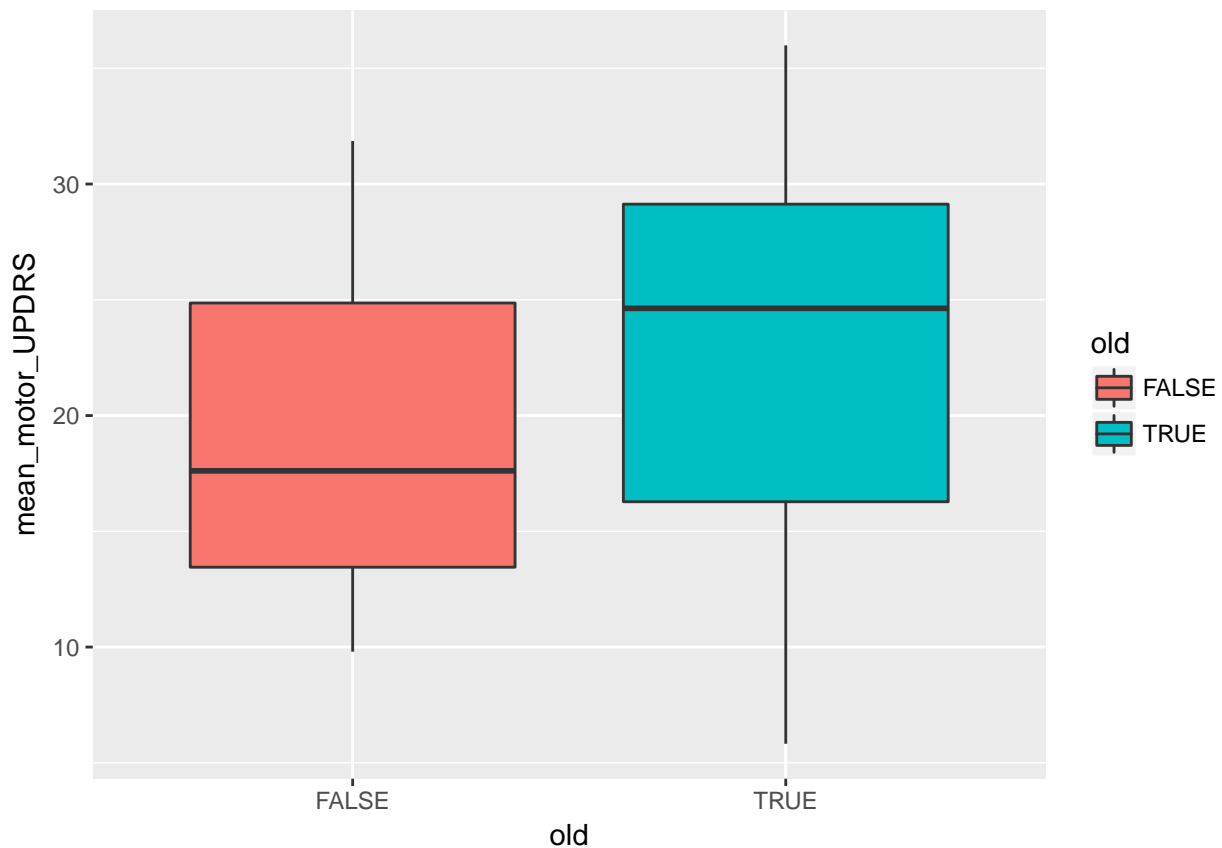


```
new_parkinson_dat <- parkinson_dat %>% group_by(subject) %>%
  summarise(mean_motor_UPDRS = mean(motor_UPDRS), age=mean(age))
new_parkinson_dat
```

```
## # A tibble: 42 × 3
##   subject mean_motor_UPDRS age
##   <int>      <dbl> <dbl>
## 1       1      31.89893   72
## 2       2      13.81254   58
## 3       3      27.12478   57
## 4       4      15.79082   74
## 5       5      31.63260   75
## 6       6      27.53169   63
## 7       7      16.04706   72
## 8       8      19.88702   73
## 9       9      18.31236   68
## 10      10      13.42442   58
## # ... with 32 more rows
```

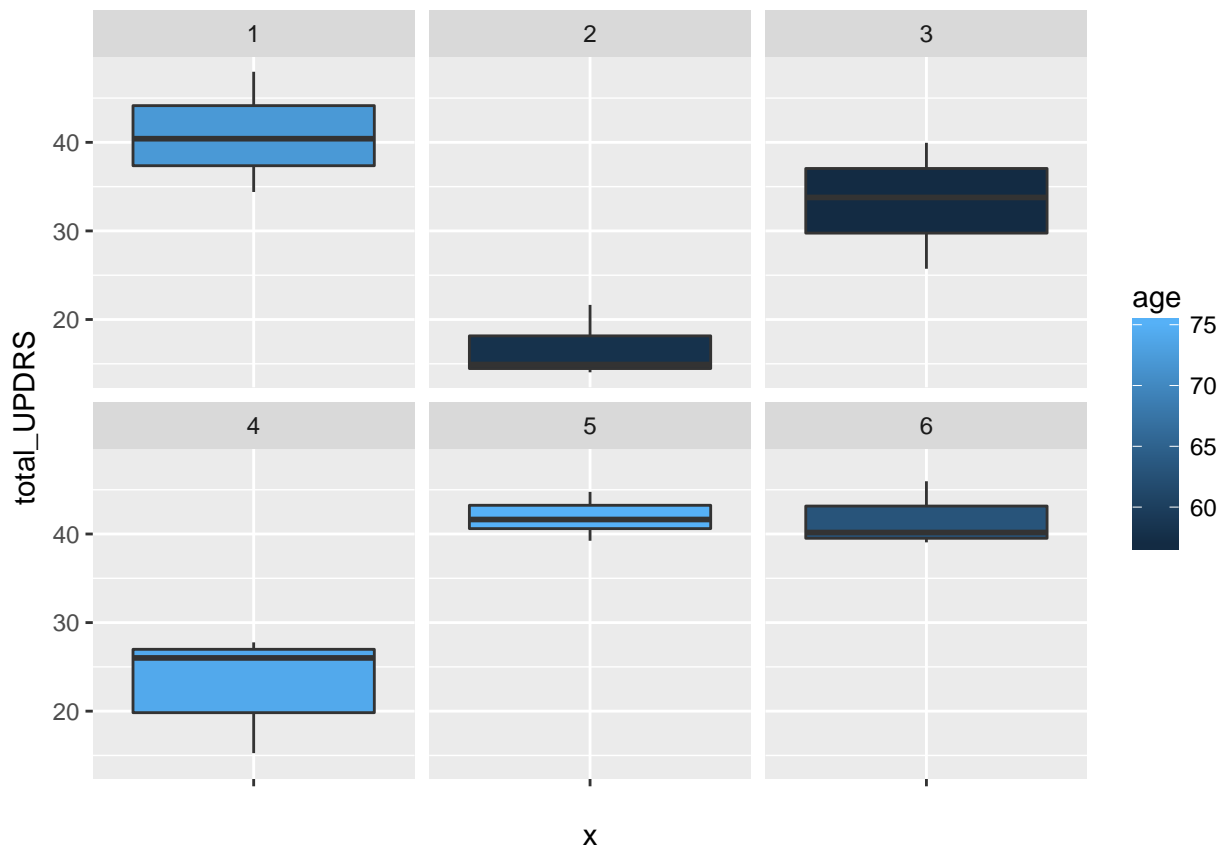
Looking at the score distribution for people over/under 65 years old.

```
new_parkinson_dat$old <- new_parkinson_dat$age >= 65
ggplot(new_parkinson_dat, aes(old, mean_motor_UPDRS, fill=old)) +
  geom_boxplot() #+ ggtitle("my title") + xlab("x lab") + ylab("y lab")
```



Looking at the score distribution of the first 6 subjects and filling the boxplot based on their age.

```
parkinson_dat_sub <- subset(parkinson_dat, subject<=6)
ggplot(parkinson_dat_sub, aes("", total_UPDRS, fill=age)) + geom_boxplot() +
  facet_wrap(~subject, ncol = 3)
```



Advanced ggplot

```
parkinson_dat_select <- parkinson_dat %>% select(total_UPDRS,PPE,DFA)
head(parkinson_dat_select)
```

```
##      total_UPDRS      PPE      DFA
## 1:      34.398 0.16006 0.54842
## 2:      34.894 0.10810 0.56477
## 3:      35.389 0.21014 0.54405
## 4:      35.810 0.33277 0.57794
## 5:      36.375 0.19361 0.56122
## 6:      36.870 0.19500 0.57243
```

```
library(reshape2)
```

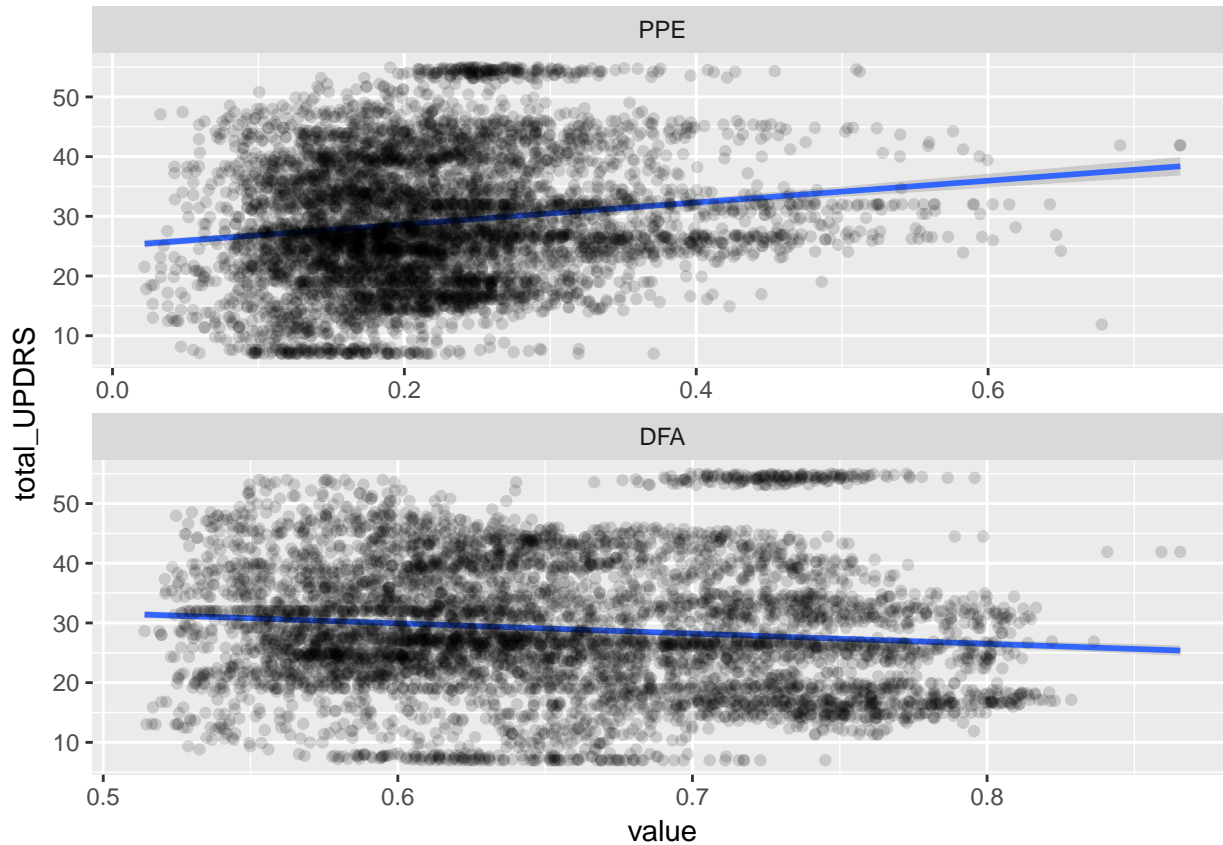
```
##
## Attaching package: 'reshape2'
## The following objects are masked from 'package:data.table':
##
##      dcast, melt
```

```
parkinson_dat_melt <- melt(parkinson_dat_select, id="total_UPDRS")
head(parkinson_dat_melt)
```

```
##      total_UPDRS variable  value
## 1:      34.398      PPE 0.16006
## 2:      34.894      PPE 0.10810
## 3:      35.389      PPE 0.21014
```

```
## 4:      35.810      PPE 0.33277
## 5:      36.375      PPE 0.19361
## 6:      36.870      PPE 0.19500
```

```
ggplot(parkinson_dat_melt, aes(value, total_UPDRS)) + stat_smooth(method = "lm") +
  geom_point(alpha=0.15) + facet_wrap(~variable, scales="free", ncol=1)
```



Regression

We can do a linear regression to predict the UPDR score from the age, sex, PPE, and DFA variables.

```
mymodel <- lm(total_UPDRS~age+sex+PPE+DFA+0, parkinson_dat) # + 0 is for no intercept, coherent with the
summary(mymodel)
```

```
##
## Call:
## lm(formula = total_UPDRS ~ age + sex + PPE + DFA + 0, data = parkinson_dat)
##
## Residuals:
##      Min       1Q   Median       3Q      Max
## -26.193  -7.441  -1.549   7.402  25.376
##
## Coefficients:
##      Estimate Std. Error t value Pr(>|t|)
## age      0.32456    0.01498   21.67  <2e-16 ***
## sexmale  21.42927    1.68034   12.75  <2e-16 ***
## sexfemale 19.21671    1.64638   11.67  <2e-16 ***
## PPE     21.54042    1.56344   13.78  <2e-16 ***
```

Breast Cancer

Attribute Information:

#	Attribute Domain
1.	Sample code number id number
2.	Clump Thickness 1 - 10
3.	Uniformity of Cell Size 1 - 10
4.	Uniformity of Cell Shape 1 - 10
5.	Marginal Adhesion 1 - 10
6.	Single Epithelial Cell Size 1 - 10
7.	Bare Nuclei 1 - 10
8.	Bland Chromatin 1 - 10
9.	Normal Nucleoli 1 - 10
10.	Mitoses 1 - 10
11.	Class: (2 for benign, 4 for malignant)

```
breast_cancer <- fread("breast_cancer.csv")
str(breast_cancer)
```

```
breast_cancer$class <- factor(breast_cancer$class)
levels(breast_cancer$class) <- c("benign", "malignant")
str(breast_cancer)
```

```
## Classes 'data.table' and 'data.frame': 699 obs. of 12 variables:
## $ V1 : chr "1" "2" "3" "4" ...
## $ id_number : int 1000025 1002945 1015425 1016277 1017023 1017122 1018099 1018561 1033078 1
## $ clump_thickness : int 5 5 3 6 4 8 1 2 2 4 ...
## $ cell_size : int 1 4 1 8 1 10 1 1 1 2 ...
## $ cell_shape : int 1 4 1 8 1 10 1 2 1 1 ...
## $ marginal_adhesion: int 1 5 1 1 3 8 1 1 1 1 ...
## $ single_epithelial: int 2 7 2 3 2 7 2 2 2 2 ...
## $ bare_nuclei : chr "1" "10" "2" "4" ...
## $ bland_chromatin : int 3 3 3 3 3 9 3 3 1 2 ...
## $ normal_nucleoli : int 1 2 1 7 1 7 1 1 1 1 ...
## $ mitoses : int 1 1 1 1 1 1 1 1 5 1 ...
## $ class : Factor w/ 2 levels "benign","malignant": 1 1 1 1 1 2 1 1 1 1 ...
## - attr(*, ".internal.selfref")=<externalptr>
```

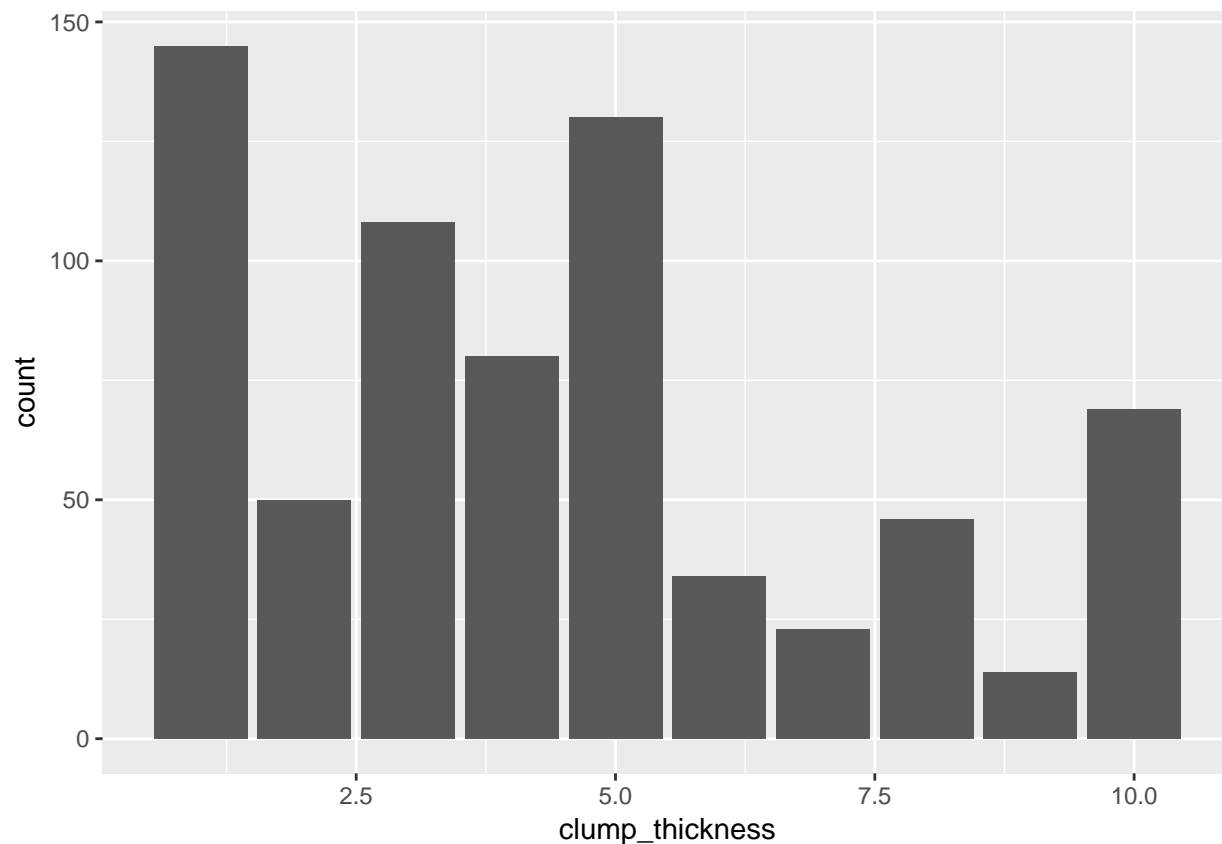
Let us see how many tumor of each class is present in our dataset

```
breast_cancer %>% group_by(class) %>% summarise(count=n())
```

```
## # A tibble: 2 × 2
##   class count
##   <fctr> <int>
## 1 benign 458
## 2 malignant 241
```

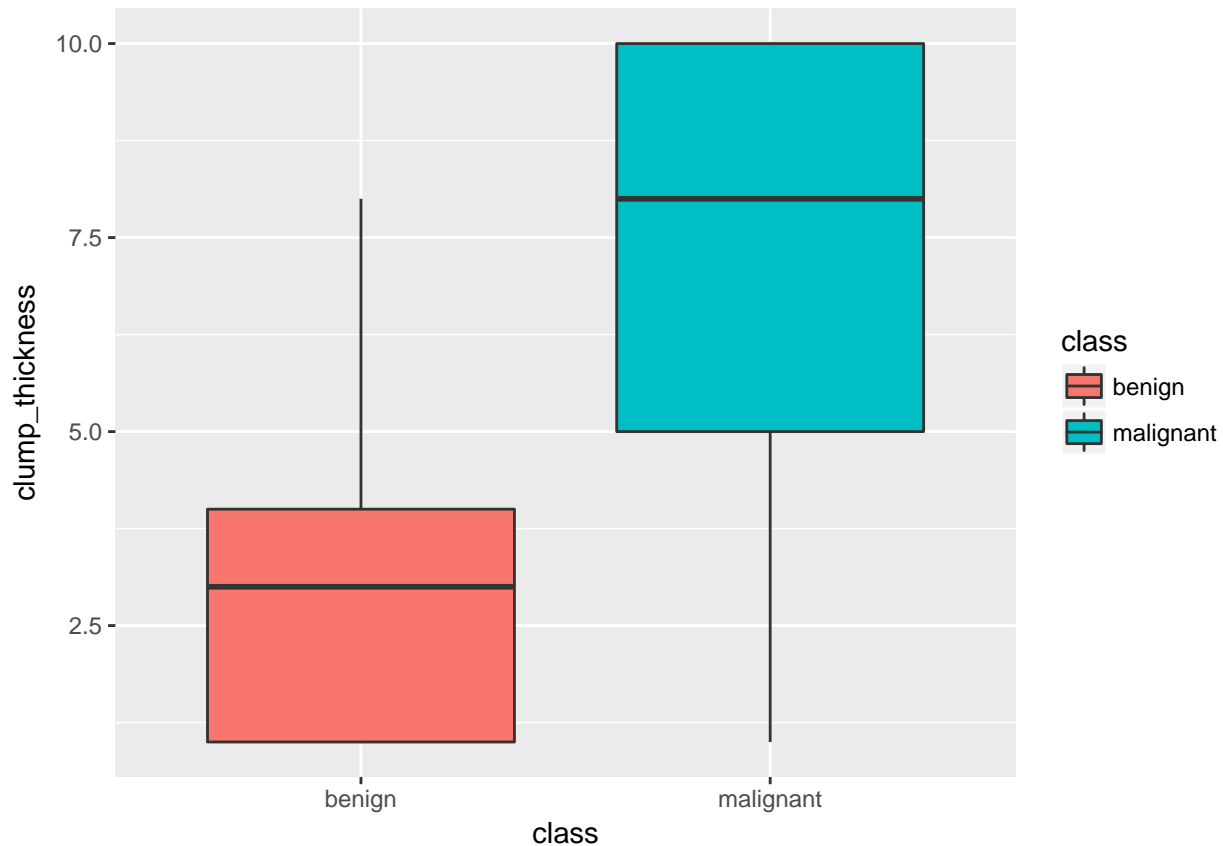
We can look at the distribution of the clumb thickness using a barplot.

```
ggplot(breast_cancer, aes(clump_thickness)) + geom_bar()
```



Using ggplot to inspect the distribution of the clump's thickness given the tumor's class.

```
ggplot(breast_cancer, aes(class, clump_thickness, fill=class)) + geom_boxplot()
```



Classification

Let us use a generalized linear model to classify the tumor into malignant and benign.

```
model <- glm(class~clump_thickness, family = "binomial", data=breast_cancer)
summary(model)
```

```
##
## Call:
## glm(formula = class ~ clump_thickness, family = "binomial", data = breast_cancer)
##
## Deviance Residuals:
##      Min       1Q   Median       3Q      Max
## -2.1986  -0.4261  -0.1704   0.1730   2.9118
##
## Coefficients:
##              Estimate Std. Error z value Pr(>|z|)
## (Intercept)   -5.16017    0.37795  -13.65  <2e-16 ***
## clump_thickness  0.93546    0.07377   12.68  <2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## (Dispersion parameter for binomial family taken to be 1)
##
```

```
##      Null deviance: 900.53  on 698  degrees of freedom
## Residual deviance: 464.05  on 697  degrees of freedom
## AIC: 468.05
##
## Number of Fisher Scoring iterations: 6
```

Ressources

Datacamp

<https://www.datacamp.com/>

R for Data Science

<http://r4ds.had.co.nz/>

Advanced R

<http://adv-r.had.co.nz/>