Graphics in R

Ghana Data Science Workshop TADs

EPIDEMIOLOGY, ECONOMICS AND RISK ASSESSMENT (EERA)

Intro

Vizualising data is extremely important both in terms of data checking and understanding, but also as a very powerful way of describing your data/results to your audience (in a presentation, your thesis, in a paper etc).

One of the really exciting things about R is the amazing graphics that you can produce that are very quickly journal ready. They are infinitely superior to Excel, SAS, Stata, or Minitab. However there is a small hurdle to get over with R but this is made easier by learning the grammar of graphics and using the package ggplot2 in R.

NB: There are simpler ways to plot things with R, but are much uglier so this is worth the pain!

Load packages

So first of all you need to install the package and then run the library function to then run it in the current environment and start using its functions. **ggplot2** is actually part of **tidyverse** so another option is to just load tidyverse.

```
library(ggplot2)
#OR
library(tidyverse)

library(here) #for data import path
```

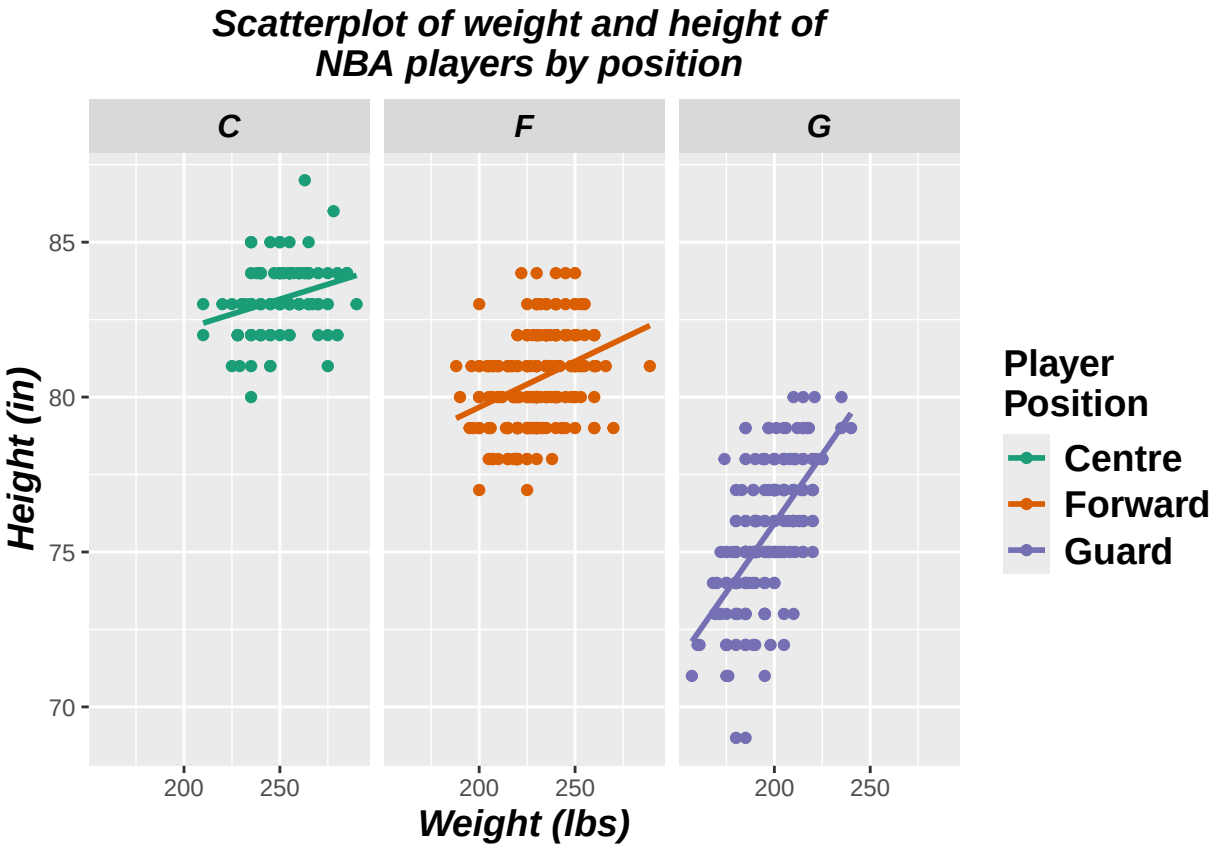
Import dataset

Lets import a dataset and see what we can do with it in ggplot. The "NBA.csv" dataset (http://www.stat.ufl.edu/~winner/datasets.html) contains the height, weight, age of NBA players and their positions. As we have done already have a look at the data frame and check how large it is and summarise it.

```
nba <- read_csv(here("data","NBA.csv"))
```

An example of a plot by ggplot

Now lets visualize the relationship between the Weight and Height of NBA players according to their position.



The code for this plot is the following!!

```
ggplot(nba, aes(x = Weight, y = Height, colour = Pos)) +
geom_point() +
stat_smooth(method = "lm", se = FALSE) +
scale_colour_brewer(palette="Dark2",
                    name  ="Player \nPosition",
                    breaks=c("C", "F", "G"),
                    labels=c("Centre", "Forward", "Guard")) +
facet_grid(. ~ Pos) +
labs(x = "Weight (lbs)",
     y = "Height (in)",
     title = "Scatterplot of weight and height of \n NBA players by position") +
theme(axis.title = element_text(colour = "black", size = 14, face = "bold.italic"),
strip.text = element_text(colour = "black", face = "bold.italic", size = 12),
plot.title = element_text(colour = "black", size = 14, face = "bold.italic", hjust = 0.5),
legend.title = element_text(colour="black", size=14, face="bold"),
legend.text = element_text(colour="black", size = 14, face = "bold") )
```

The ggplot grammar

It might look a bit scary, but we will now explain how to build a ggplot step by step by understanding the ggplot grammar. Basically, ggplots are composed of **building blocks** that are added to the plot one after the other using the + sign. The diagram below shows the most important building blocks. We start building a plot from the bottom!

1. DATA

Anything you try to plot with ggplot needs to belong to a dataframe. The variables we want to visualize belong to the *NBA* dataset.

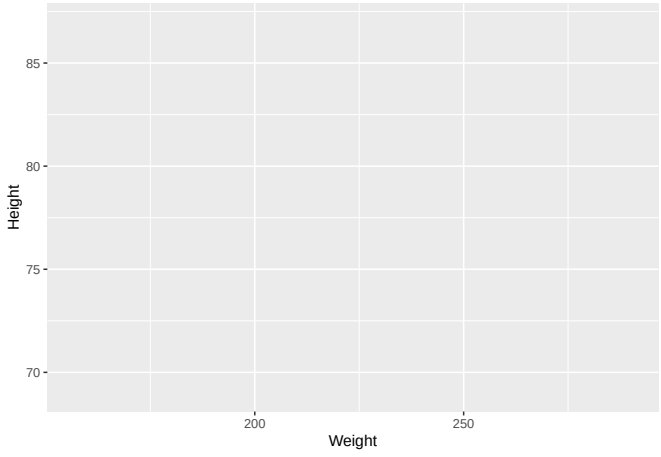
```
head(nba)
```

```
## # A tibble: 6 x 7
##       X Player              Pos   Height Weight   Age Age21
##   <dbl> <chr>               <chr>  <dbl>  <dbl> <dbl> <chr>
## 1     1 "Nate\xcaRobinson"  G         69    180    29 >21
## 2     2 "Isaiah\xcaThomas"  G         69    185    24 >21
## 3     3 "Phil\xcaPressey"   G         71    175    22 >21
## 4     4 "Shane\xcaLarkin"   G         71    176    20 <=21
## 5     5 "Ty\xcaLawson"      G         71    195    25 >21
## 6     6 "John\xcaLucas III" G         71    157    30 >21
```

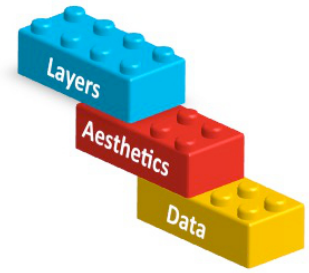
2. Aesthetics mapping

Aesthetics refer to the variables we want to see. In this case weight and height! So let's start building our plot using the *ggplot* function.

```
ggplot(data = nba, aes(x = Weight, y = Height))
```

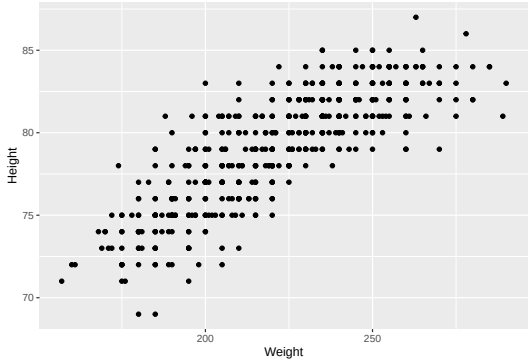


3. Layers



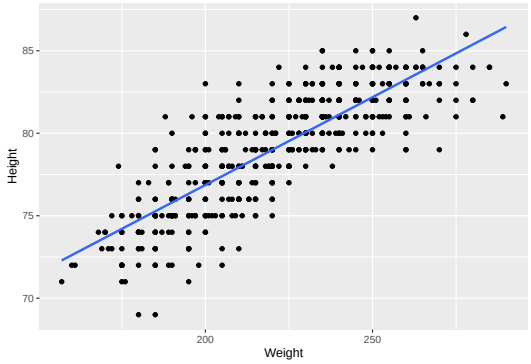
In order to see something on our plot we need to add layers. Layers include geometric elements (geoms) and statistical transformations (stats). Since we want to build a scatterplot our first layer will be a layer of points (geom_point):

```
ggplot(data = nba, aes(x = Weight, y = Height)) +
  geom_point() # Layer 1
```

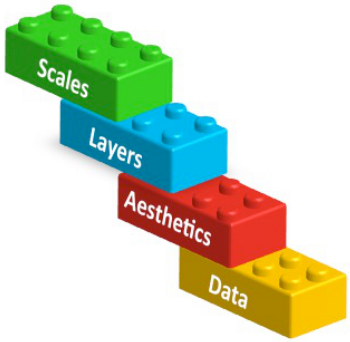


We also want to see the statistical relationship between weight and height so we will add a regression line as our second layer.

```
ggplot(data = nba, aes(x = Weight, y = Height)) +
  geom_point() + # Layer 1
stat_smooth(method = "lm", se = FALSE) # Layer 2
```

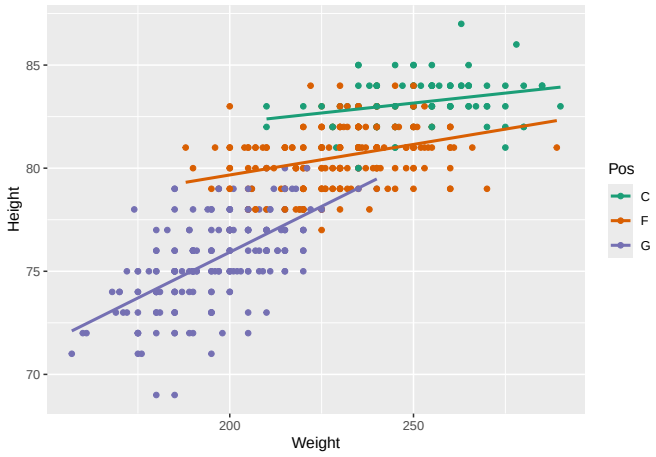


4. Scales



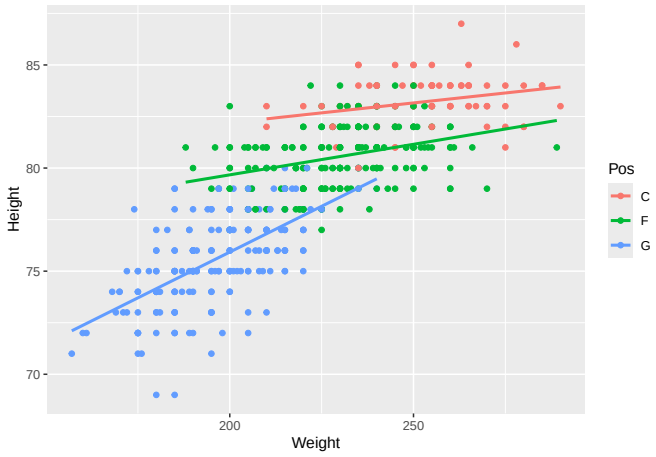
As we said at the beginning we are also interested in the position of each player. As a first step we want to colour each point by the player's position. Scales map values in the data space to values in an aesthetic space. This can be colour, size or shape. This will also automatically create a legend to explain the colours on the plot.

```
ggplot(data = nba, aes(x = Weight, y = Height, colour = Pos)) + # added colour = POS
  geom_point() +
stat_smooth(method = "lm", se = FALSE) +
scale_colour_brewer(palette="Dark2") # scale maps aes data values (x and y) to aes colour
```

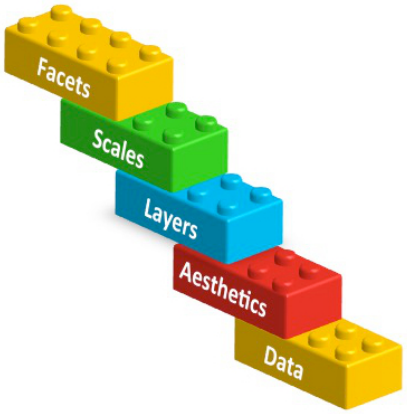


NB: You can obtain a similar plot just by adding a colour variable in the aesthetics. Scales give you the ability to have control over the colours chosen.

```
ggplot(data = nba, aes(x = Weight, y = Height, colour = Pos)) +
  geom_point() +
stat_smooth(method = "lm", se = FALSE)
```

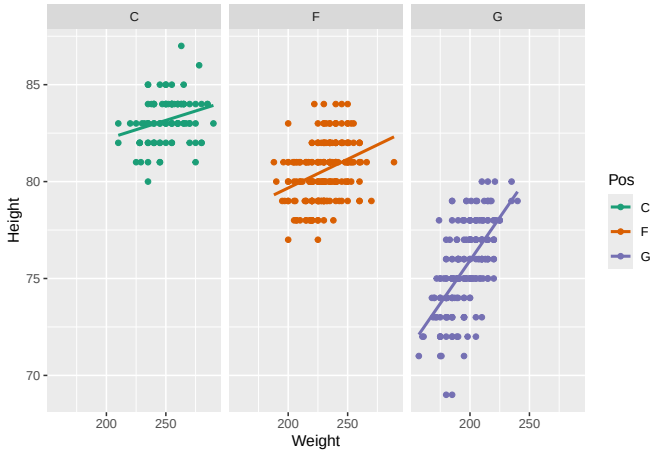


5. Facets



If you remember the original plot, we actually wanted to see a separate plot for each player position. Using facets we can display our data split by the chosen variable, in this case position.

```
ggplot(data = nba, aes(x = Weight, y = Height, colour = Pos)) +
  geom_point() +
stat_smooth(method = "lm", se = FALSE) +
scale_colour_brewer(palette="Dark2") +
facet_grid(. ~ Pos) # split grid by the variable Pos
```



6+7. Themes and other useful tricks!

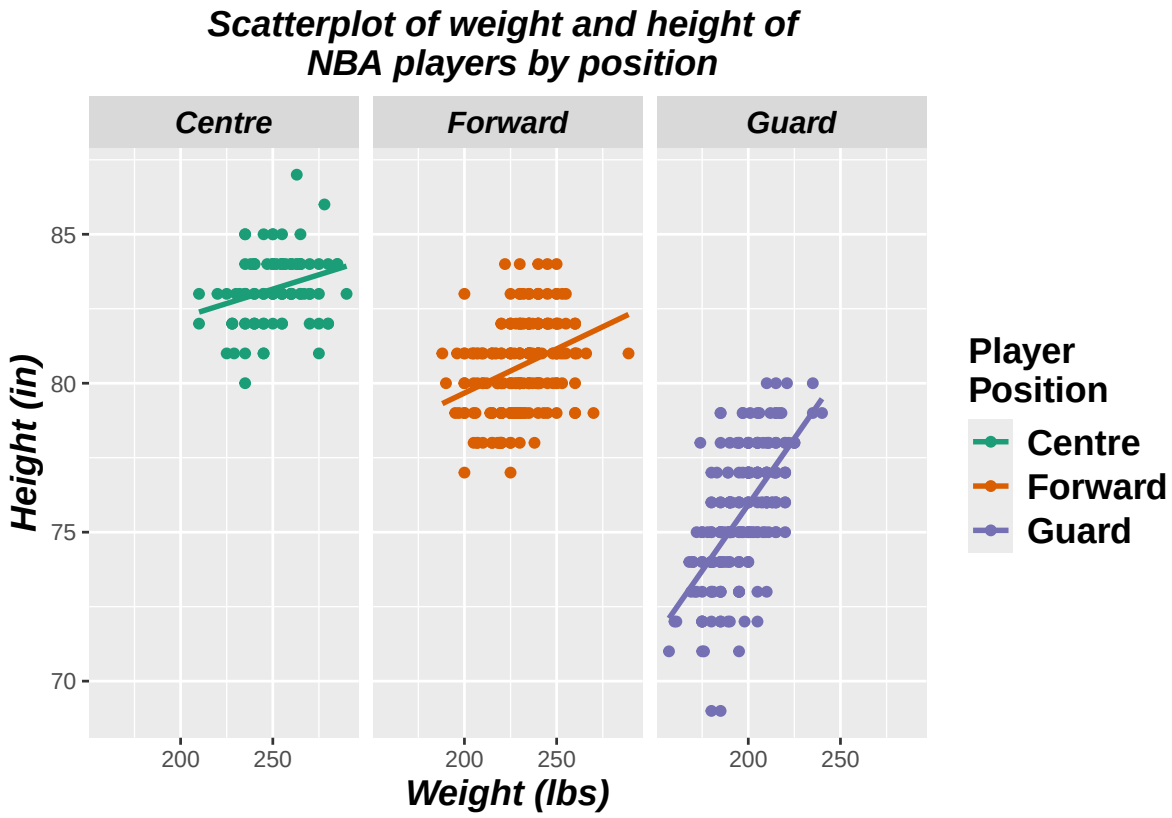
ggplot is very flexible and you can adjust pretty much every aspect of the plot to your preference. In our original plot we had added a plot title and changed the x and y axis labels using the **labs** function and the **title**, **x** and **y** arguments respectively as shown in the code below.

We also changed the title and labels of our legend by adding information to **scale** and the facet labels by adding information to **facet**.

Lastly, we used **theme** to change font size, colour and style of many elements of the plot. You can use theme to change pretty much everything you like on your plot.

Here is the whole code again, with some useful websites to understand each building block better!

```
ggplot(nba, aes(x = Weight, y = Height, colour = Pos)) +
# Visit http://docs.ggplot2.org/current/ for lists of geoms, stats etc
geom_point() +
stat_smooth(method = "lm", se = FALSE) +
# Visit http://docs.ggplot2.org/current/scale_brewer.html for more colour options
scale_colour_brewer(palette="Dark2",
                    name  ="Player \nPosition",
                    breaks=c("C", "F", "G"),
                    labels=c("Centre", "Forward", "Guard")) +
facet_grid(. ~ Pos, labeller=labeller(Pos = c("C"="Centre", "F"="Forward", "G"="Guard"))) +
labs(x = "Weight (lbs)",
     y = "Height (in)",
     title = "Scatterplot of weight and height of \n NBA players by position") +
# Visit http://docs.ggplot2.org/current/theme.html and
# http://www.sthda.com/english/wiki/ggplot2-themes-and-background-colors-the-3-elements
# for info about themes
theme(axis.title = element_text(colour = "black", size = 14, face = "bold.italic"),
strip.text = element_text(colour = "black", face = "bold.italic", size = 12),
plot.title = element_text(colour = "black", size = 14, face = "bold.italic", hjust = 0.5),
#legend.position = "none", # can add this line to remove legend from plot
legend.title = element_text(colour="black", size=14, face="bold"),
legend.text = element_text(colour="black", size = 14, face = "bold") )
```


Scatterplot of weight and height of
NBA players by position
Centre
Forward
Guard
Height (in)
85
80
75
70
200
250
Weight (lbs)
Player
Position
Centre
Forward
Guard

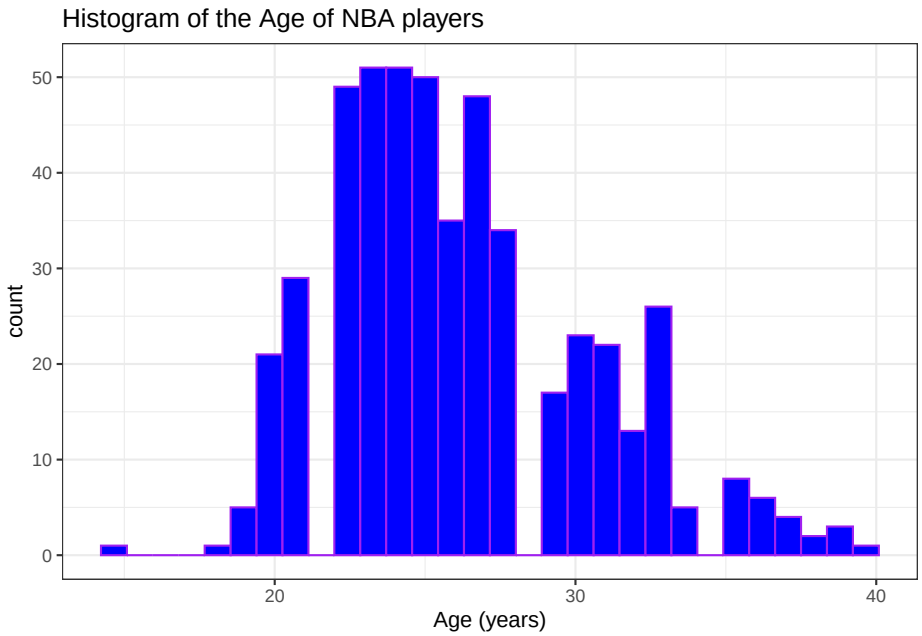
Other plot types

So far we have seen how to draw a scatter plot with ggplot. You can plot pretty much any type of plot you like and you can change that by changing the *geom* type used. There are a number of geoms and you can see these when you use Rstudio as you type. But some of the common ones are below and also in the cheat sheet provided.

geom	description
geom_point	Points, eg. a scatterplot
geom_line	lines
geom_ribbon	Ribbons, y range with continuous x
geom_polygon	Polygon, a filled path
geom_pointrange	vertical line with point in the middle
geom_path	connect observations in original order
geom_histogram	Histograms
geom_text	Textural annotations
geom_violin	Violin plots
geom_map	Polygons from map

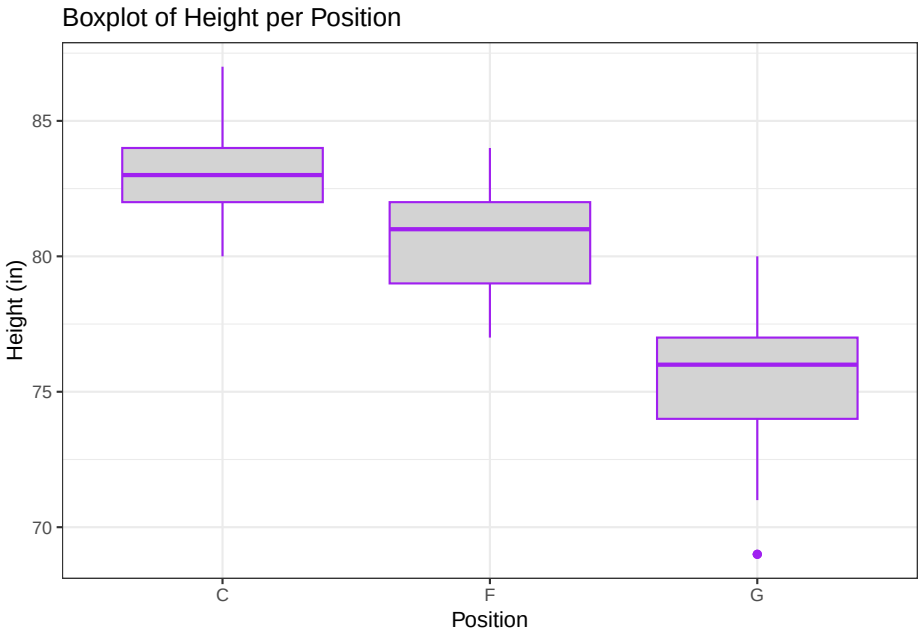
Histograms

```
p <- ggplot(data=nba, aes(x=Age)) +
      geom_histogram(fill="blue", colour="purple") +
      labs(title="Histogram of the Age of NBA players", x="Age (years)") +
      theme_bw()
p
```



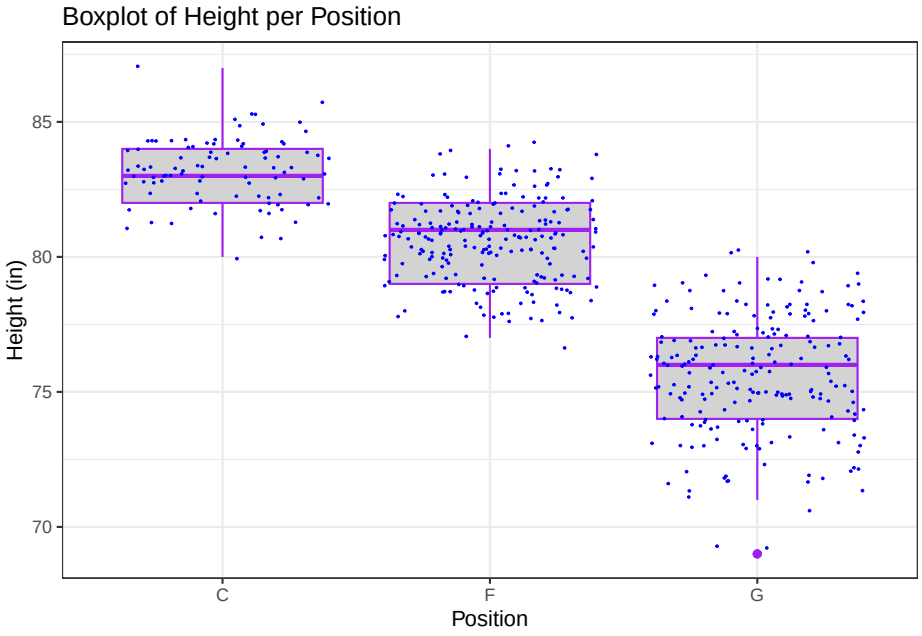
Boxplots

```
ggplot(data=nba) +
      geom_boxplot(aes(x=Pos, y=Height), fill="lightgrey", colour="purple") +
      labs(x="Position",
           y="Height (in)",
           title="Boxplot of Height per Position") +
# Set title and axis labels
      theme_bw()
```



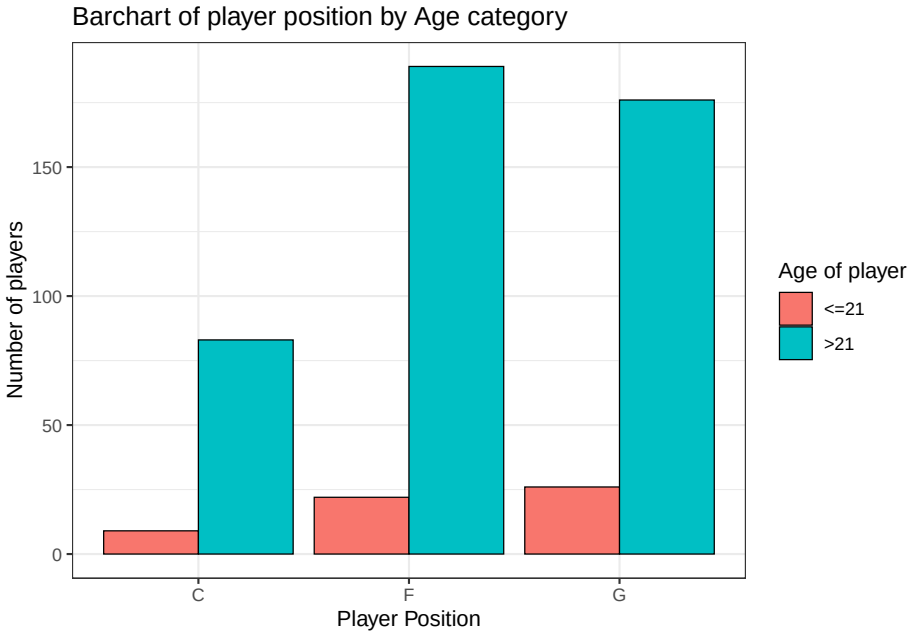
One of the things about box plots though is that you lose a lot of information so you can get fancy and overlay the raw dots like this...

```
ggplot(data=nba) +
geom_boxplot(aes(y=Height, x=Pos), fill="lightgrey", colour="purple") +
geom_jitter(aes(y=Height, x=Pos), colour="blue", size=0.2) + # geom_jitter spreads points
#out so that they don't overlap
labs(x="Position",
     y="Height (in)",
     title="Boxplot of Height per Position") +
theme_bw()
```



Barcharts

```
ggplot(data=nba, aes(x=Pos, fill=Age21)) +
    geom_bar(colour="black", stat="count",
             position=position_dodge(), # split bars by Age21 variable.
                                        #try commenting this line out
             size=.3) +                      # Thinner lines
    scale_fill_discrete(name="Age of player") +      # Set legend title
    labs(x = "Player Position", y = "Number of players",
         title = "Barchart of player position by Age category") + #Set labels text
    theme_bw() # Set theme
```



More info

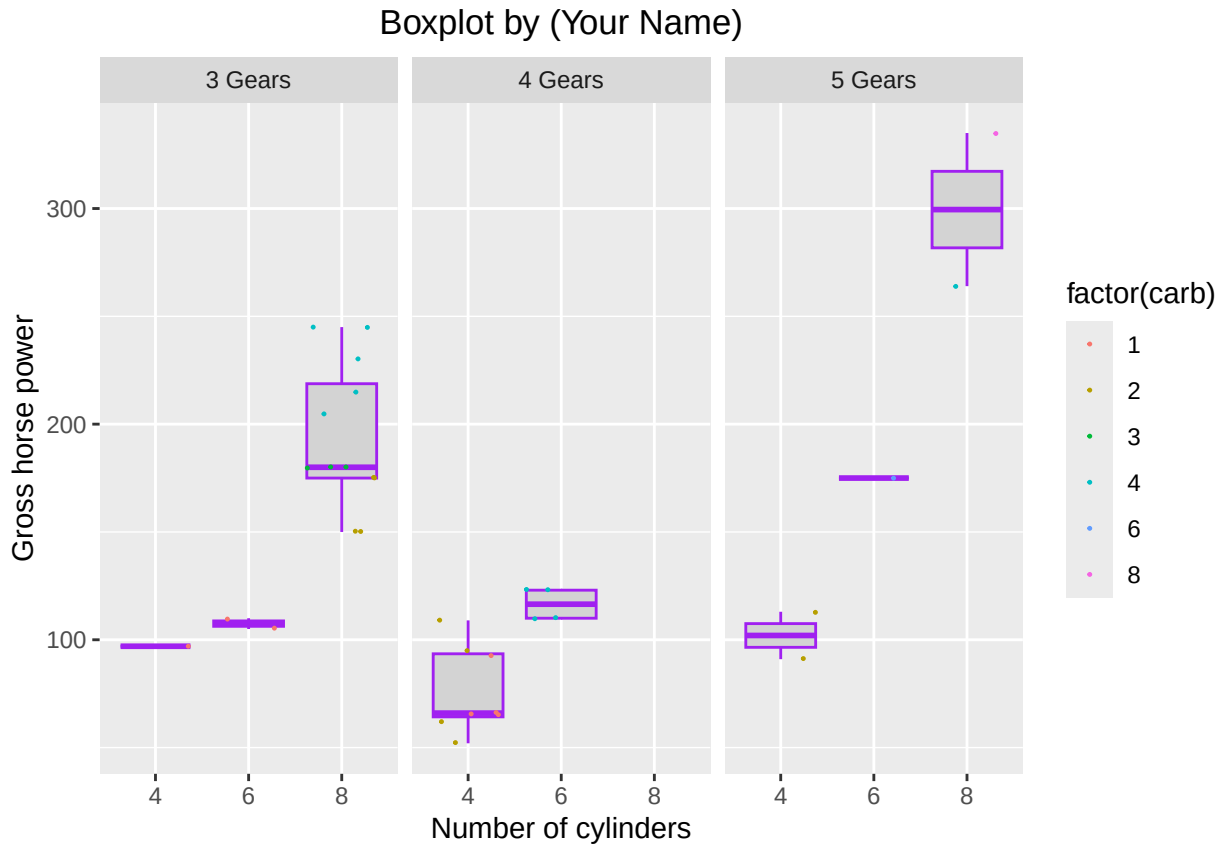
- R Cookbook Graphs
  http://www.cookbook-r.com/Graphs/
- Line plots tutorial
  http://t-redactyl.io/blog/2015/12/creating-plots-in-r-using-ggplot2-part-1-line-plots.html
- Bar plots tutorials
  http://t-redactyl.io/blog/2016/01/creating-plots-in-r-using-ggplot2-part-3-bar-plots.html
  http://t-redactyl.io/blog/2016/01/creating-plots-in-r-using-ggplot2-part-4-stacked-bar-plots.html
- Scatter plots tutorials
  http://t-redactyl.io/blog/2016/02/creating-plots-in-r-using-ggplot2-part-5-scatterplots.html
  http://t-redactyl.io/blog/2016/02/creating-plots-in-r-using-ggplot2-part-6-weighted-scatterplots.html
- Histograms tutorial
  http://t-redactyl.io/blog/2016/02/creating-plots-in-r-using-ggplot2-part-7-histograms.html
- Boxplots tutorial
  http://t-redactyl.io/blog/2016/04/creating-plots-in-r-using-ggplot2-part-10-boxplots.html

Exercises

Ex 1.

Using dataset *mtcars* plot a box-plot of Gross horse power (*hp*) against number of cylinders (*cyl*). Give the plot the title *"Boxplot by (Your Name)"*. Add the real horse power values using dots coloured by Number of carburetors (*carb*) faceted by Number of forward gears (*gear*). Change the labels of gear to "3 Gears", "4 Gears", "5 Gears".

Hint: To find more info about an R dataset try *?mtcars*.
Hint 2: Your plot should look like this!



Ex 2.

Using the same dataset plot a stacked bar chart of number of cylinders (*cyl*) by Transmission (*am*). Change colours corresponding to *am* manually to blue for 0 and red for 1. Change the legend labels to Autumatic and Manual and the legend title to Transmission.

Hint: Check the bar plots tutorial websites above for help
Hint 2: Your plot should look like this!