Basic Plotting

謝舒凱 Shu-Kai Hsieh

October 20, 2016

Contents

```
# clear R's memory
rm(list=ls())
# where R is currently looking
getwd()
# tell R where to look
setwd("~/Coding/R_lab/Linguistics Data Analysis with R/")
```

1 Introduction

This unit provides the necessary stimulus for understanding the gist that discrete and continuous data needs appropriate tools, and the validation may be seen through the distinct characteristics of such plots...this unit is also more closely related to Exploratory Analysis, and many visualization techniques here indeed are "exploratory" in nature. Thus, the current unit and the next complement mutually.

In R there are four main packages for producing graphics:

graphics : basic package already available in the R environment and is already loaded by default.

lattice : providing high-level functions for all the common plot types. Also available in R by default, but need to load it.

ggplot2 : the most modern system. 'gg' stands for 'grammar of graphics', which aims to break down graphs into component chunks.

grid : was developed to allow more flexible plotting, letting you draw things at a very low level, specifying where to draw each point, line, or rectangle.

This unit will mainly cover the details on effective data visualization:[1]

```
require(languageR)

## Loading required package: languageR

require(MASS)

## Loading required package: MASS
```

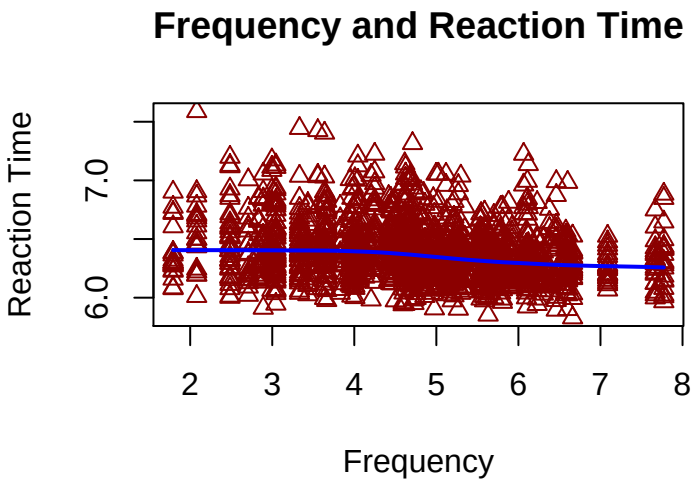
[1] Other related visualization packages such as ggvis will be introduced later.

```
require(lattice)

## Loading required package: lattice

require(ggplot2)

## Loading required package: ggplot2

set.seed(42)
small <- diamonds[sample(nrow(diamonds),1000),]
```

```
plot(lexdec$Frequency, lexdec$RT)
```

Within the `plot()` function, you can specify other optional arguments in order to modify the basic plot and change its look and/or add information to the figure.

```
with(lexdec,{
  plot(Frequency,RT,
  main = "Frequency and Reaction Time",
  xlab = "Frequency",
  ylab = "Reaction Time",
  type = "p",
  pch = 2,
  col = "dark red")

  lines(loess.smooth(Frequency, RT),
        col = "blue",
        lwd = 2)}
  )
```



main specifies the title of the plot.

xlab and ylab specify the x and y axes' labels.

pch gives a numeric argument defining the plotting symbol.

lwd gives the thickness of lines.

type gives the plotting type (dots, lines, mix, and so on).

col selects the color of the plot. Check `colors()`.

- `lines()` function to add a line representing the tendency of the data.
- `loess.smooth()` computes the point of a smooth curve, a curve describing the tendency of the data, and these points are then passed to the function lines that draws them in the plot.

Exercise: `plot()` is smart, try different variable combinations (e.g., `Word` and `RT`) and diffeerent parameters.

比較重要的是用變項的性質與要探究的關係來決定如何做圖觀察。

- Visualization of **categorical data** using a bar chart, dot chart, spine and mosaic plots, and the pie chart and its variants.

- Visualization of **continuous data** using a boxplot, histogram, scatter plot and its variants, and the Pareto chart.
- A very brief introduction to the rich school of `ggplot2`.

1.1 ggplot2

The syntax is a very different to other plotting code.

- Each plot is constructed with a call to the ggplot function, which takes a `data frame` as its first argument and an `aesthetic` as its second.

  An "aesthetic" is a dimension of a graph that we can perceive visually: the simplest example being the x and y axes. When we make a `scatterplot`, we choose one attribute to assign to the x axis, and one attribute to assign to the y axis. Other aesthetics we can use in a scatter plot are the color, size, and shape of the points in the graph: each of these aesthetics lets us communicate some dimension of the data, and understand complex relationships between them.

- ggplot2 recognizes the commands from `base` for changing the color and shape of the points, but also has its own set of more human-readable names. E.g, `shape` replaces `pch`.
- To split the plot into individual panels, we add a `facet`.

2 Visualization for Categorical Data

Reminder what is categorical data?

2.1 Bar charts

The bar charts may be obtained using two options. The function barplot, from the `graphics` library, is one way of obtaining the bar charts. The built-in examples for this plot function may be reviewed with `example(barplot)`

The second option is to load the package `lattice` and then use the `example(barchart)` function. The sixth plot, after you click for the sixth time on the prompt, is actually an example of the barchart function.

Load the dataset on severity counts for the JDT software from the RSADBE package with data(`Severity_Counts`). Also, check for this data.

3 Visualization techniques for continuous (dependent) variable data

3.1 Boxplot

The boxplot is based on five points: minimum, lower quartile, median, upper quartile, and maximum. The median forms the thick line near the middle of the box, and the lower and upper quartiles complete the box. The lower and upper quartiles along with the median, which is the second quartile, divide the data into four regions with each containing equal number of observations. The median is the middle-most value among the data sorted in the increasing (decreasing) order of magnitude. On similar lines, the lower quartile may be interpreted as the median of observations between the minimum and median data values.

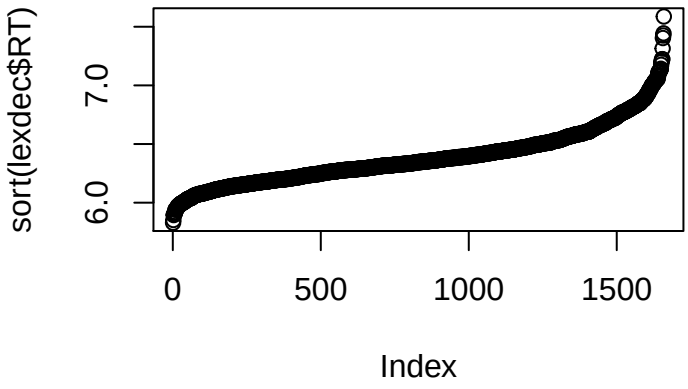
Try `example(boxplot)` and `example(bwplot)` from the graphics and lattice packages.

Example: One variable

```
#require(languageR)
colnames(lexdec)
```

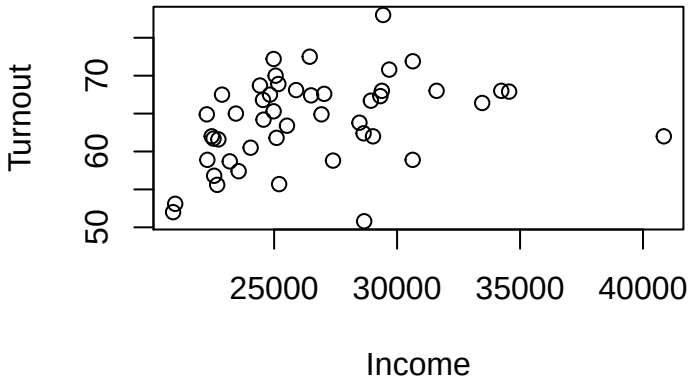
```
##  [1] "Subject"        "RT"             "Trial"          "Sex"
##  [5] "NativeLanguage" "Correct"        "PrevType"       "PrevCorrect"
##  [9] "Word"           "Frequency"      "FamilySize"     "SynsetCount"
## [13] "Length"         "Class"          "FreqSingular"   "FreqPlural"
## [17] "DerivEntropy"   "Complex"        "rInfl"          "meanRT"
## [21] "SubjFreq"       "meanSize"       "meanWeight"     "BNCw"
## [25] "BNCc"           "BNCd"           "BNCcRatio"      "BNCdRatio"
```

```
plot(sort(lexdec$RT)) # plot((lexdec$RT)) ?
```




```
# plot(quantile(lexdec$RT)) # quantile
```

```
boxplot(lexdec$RT)
```

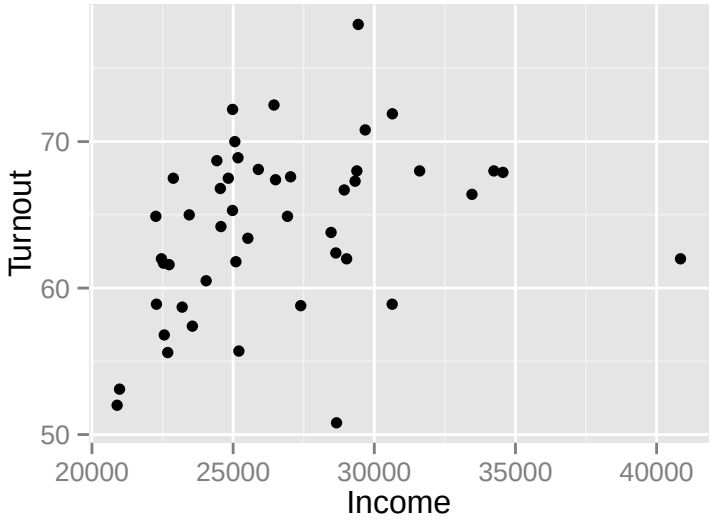


Example: Two variable

```
boxplot(meanSize, data=lexdec, notch = TRUE)
```

```
## Error in boxplot(meanSize, data = lexdec, notch = TRUE): 找不到物件'meanSize'
```

```
boxplot(lexdec$meanSize,lexdec$meanWeight, notch = TRUE, main = "Boxplot for the meanSize and meanWeig
```




```
#bwplot(lexdec$meanSize,lexdec$meanWeight, notch = TRUE)
```

Figure **??** shows a scatterplot of `Word` versus `Frequency` for the `ratings` data.

3.2 Histograms

3.3 Line charts

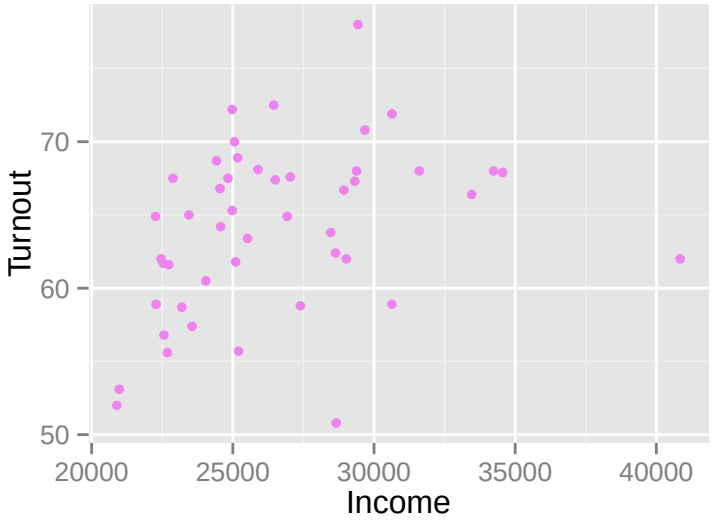
- For exploring how a continuous variable changes over time, a line plot often provides more insight than a scatterplot, since it displays the connections between sequential values.
- In base, line plots are created in the same way as scatterplots, except that they take the argument type = "l". In ggplot2, swapping a scatterplot for a line plot is as simple as swapping `geom_plot` for `geom_line`.
- In this case of drawing two lines, `geom_ribbon` can plot two lines and the contents in between.

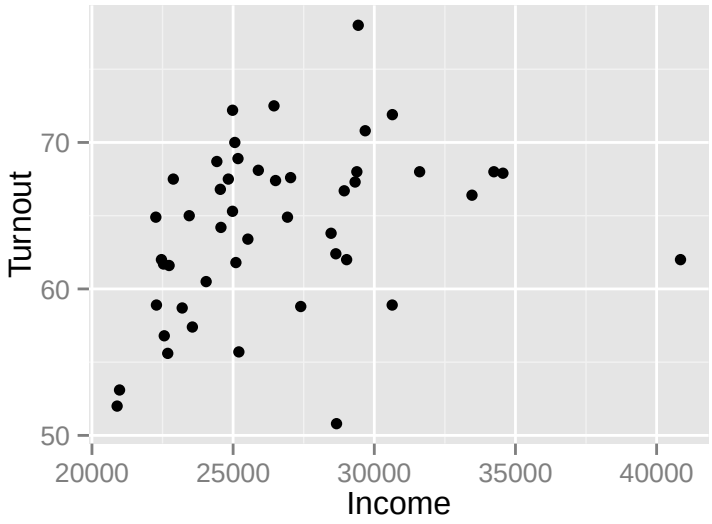
```
with(
  economics,
  plot(date, psavert, type = "l", ylim = c(0, max(psavert)))
)
```

爲什麼不要 **pie:**

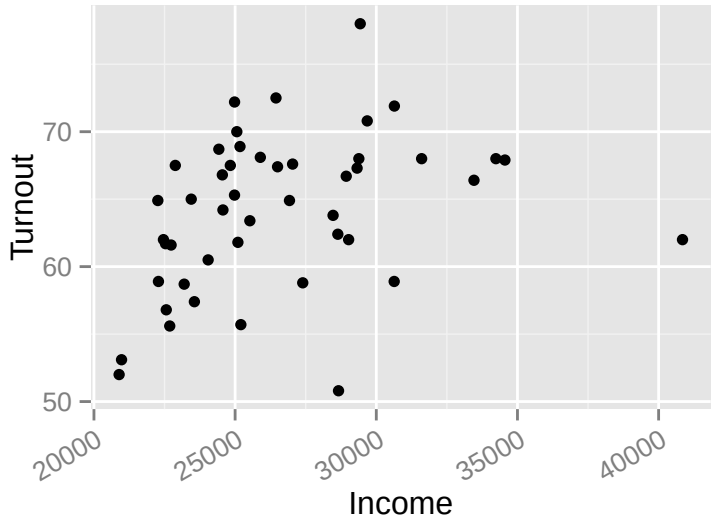
why pie chart = lie chart

```
f = function(i){
  pie(i);
  barplot(i)
}

par(mfcol = 2:3)
f(16:20)
f(c(20,20,19,21,20))
f(20:16)
```

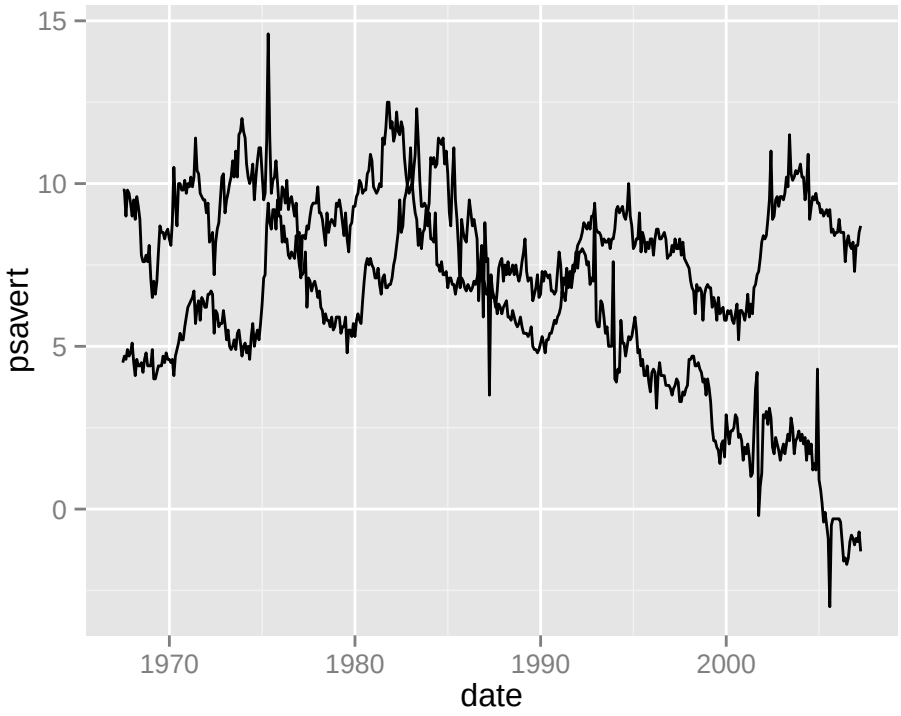




```
# ggplot2
ggplot(economics, aes(date, psavert)) +
geom_line()
```




```
# drawing multiple lines
ggplot(economics, aes(date)) +
  geom_line(aes(y = psavert)) +
  geom_line(aes(y = uempmed))
```




```
ggplot(economics, aes(date, ymin = psavert, ymax = uempmed)) +
  geom_ribbon(color = "black", fill = "white")
```

4 Visualizing two or more variables

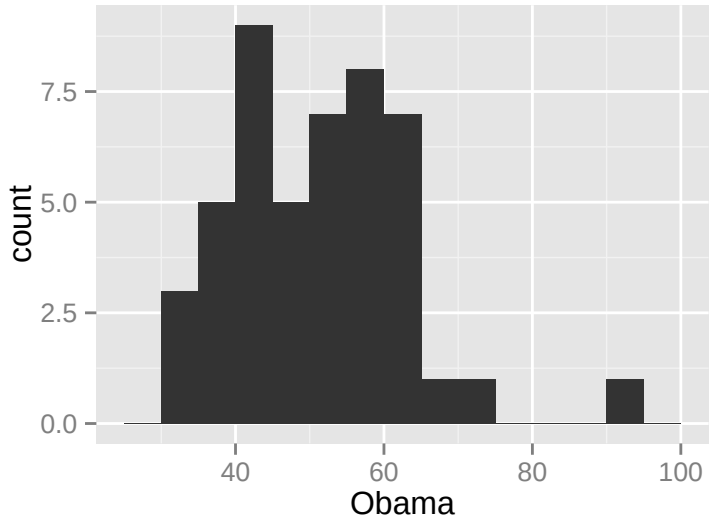
4.1 Correlation

Let's return to the data `verbs` dataset in `LanguageR`. Let's display in a table and plot the animacy patterns by type of Recipient (NP or PP):

```
verbs.xtabs <-xtabs(~AnimacyOfRec + RealizationOfRec,
                    data = verbs,
                    subset = AnimacyOfTheme!= "animate"
                    #data = verbs[verbs$AnimacyOfTheme != "animate"]
                  )
```

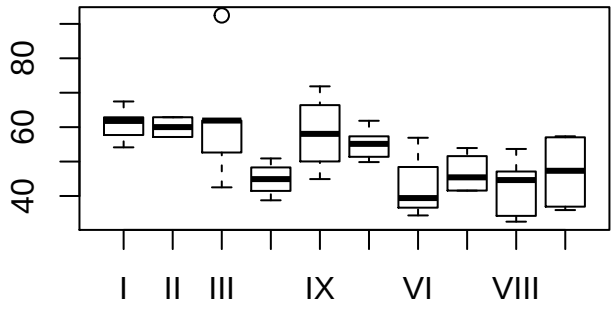
We could plot the same data as a barplot:

```
par(mfrow = c(2,2))
barplot(verbs.xtabs, legend = rownames(verbs.xtabs), beside = T)
barplot(verbs.xtabs, legend = c("anim", "inanim"))
par(mfrow = c(1,1))
```



4.2 Mosaic plots

```
verbs.xtabs <- xtabs(~AnimacyOfRec+AccessOfRec+RealizationOfRecipient,
                     data = dative)

mosaicplot(verbs.xtabs,main="dative")
```

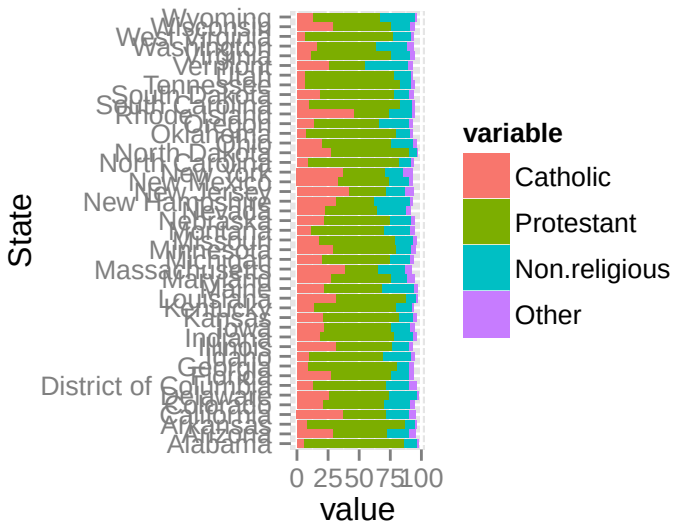


4.3 Scatterplot

散佈圖

For the relation between two numerical variables with many different values.

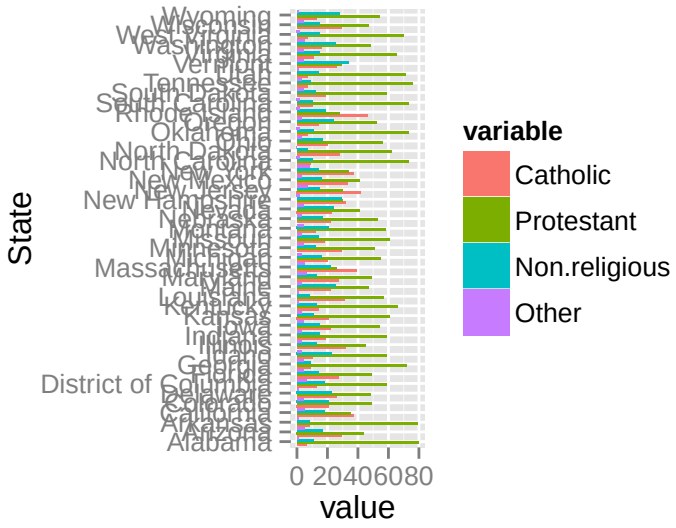
```
attach(ratings)
plot(Frequency,FamilySize)
lines(lowess(Frequency,FamilySize),col=gray(.1))
```




```
#lines(lowess(FamilySize~Frequency),col=gray(.1))
```

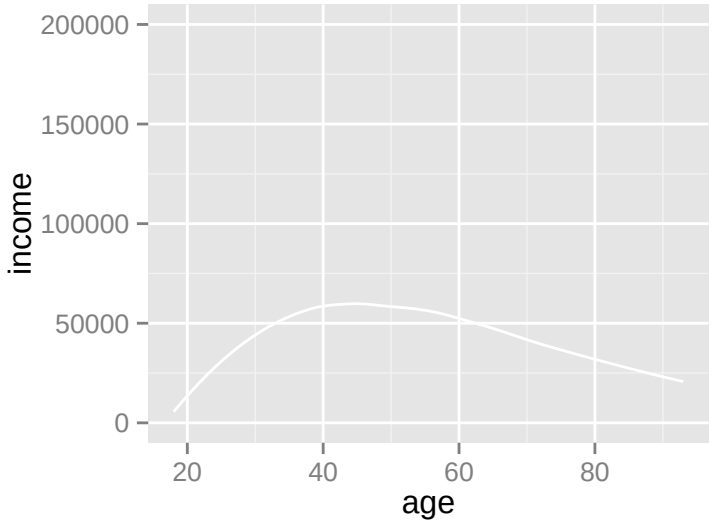
We can also plot the actual words instead of points:

```
plot(FamilySize~Frequency,type="n")
text(Frequency,FamilySize,as.character(Word),cex=0.7)
```



Note: you can jitter the words (so that they are not overlapping so much) using `jitter()`.

```
plot(FamilySize~Frequency,type="n")
text(Frequency,jitter(FamilySize,6),as.character(Word),cex=0.7)
```



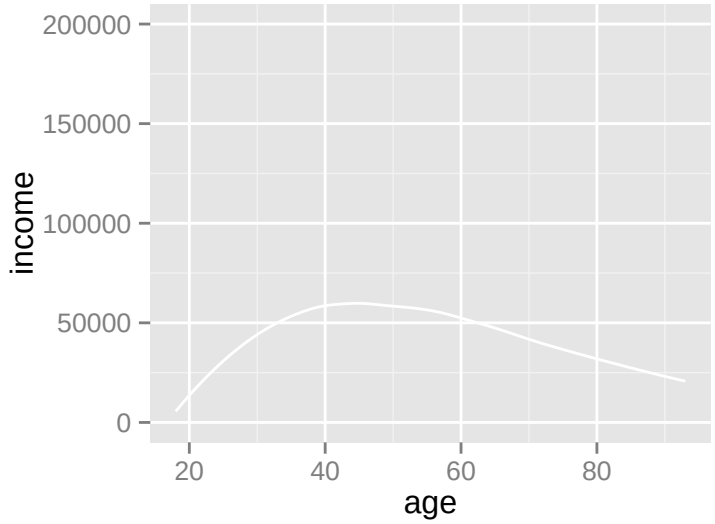

```
detach(ratings)
```

Using ggplot2

`qplot()` 比 `ggplot()` 使用上還要簡潔。

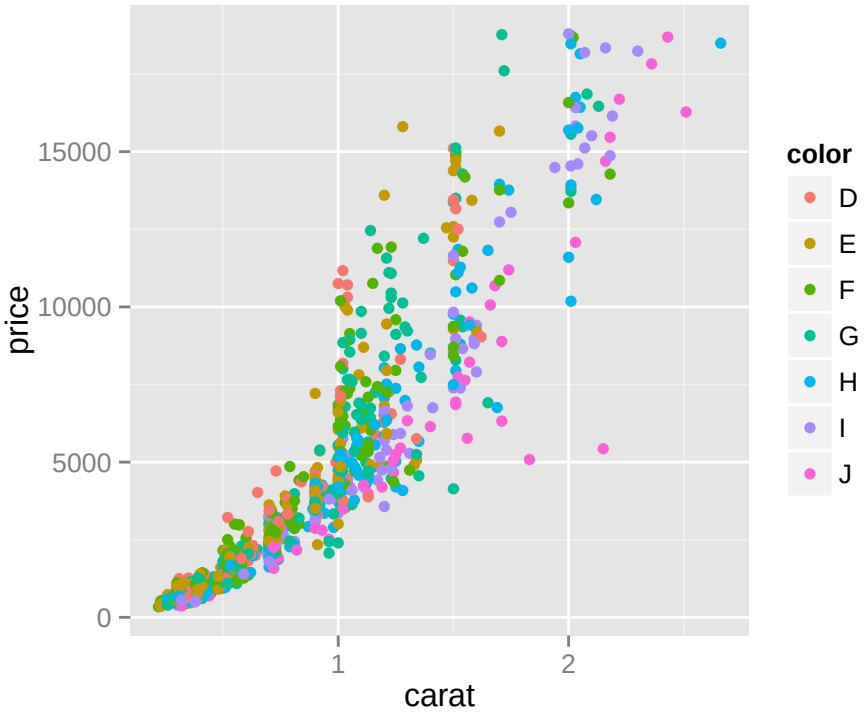
Using `diamonds`, create a scatterplot where we put `carat`, or `weight`, on the x axis and `price`, in dollars, on the y axis.

```
# ggplot(diamonds, aes(x=carat, y=price)) +
#   geom_point()
# qplot(carat, price, data = diamonds)
qplot(log(carat), log(price), data = diamonds)
```

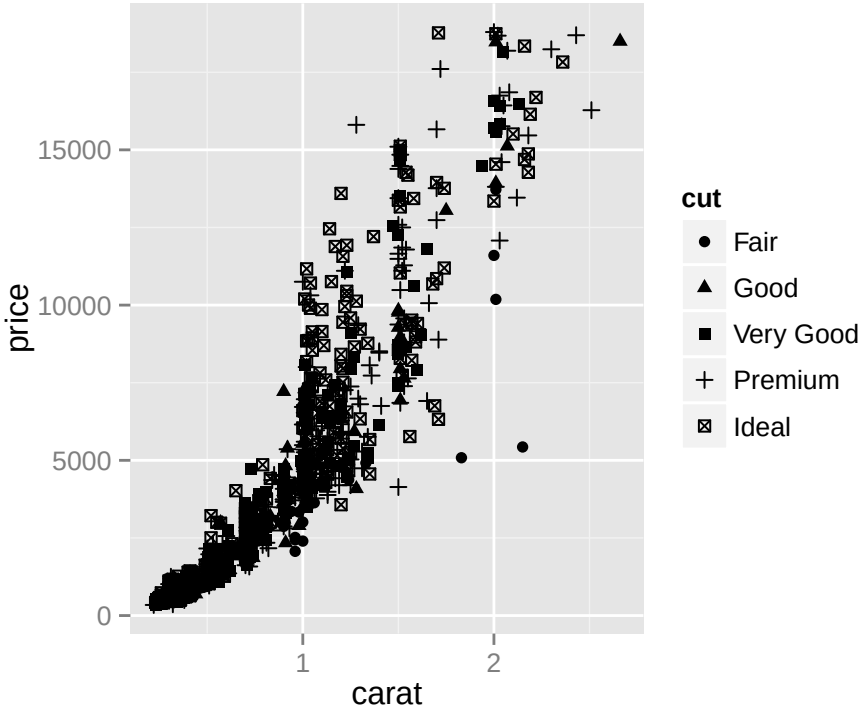


The first big difference when using qplot instead of ggplot comes when you want to assign **colours, sizes** or **shapes** to the points on your plot.

```
par(mfrow=c(2,2))
qplot(carat, price, data = small, colour = color)
```

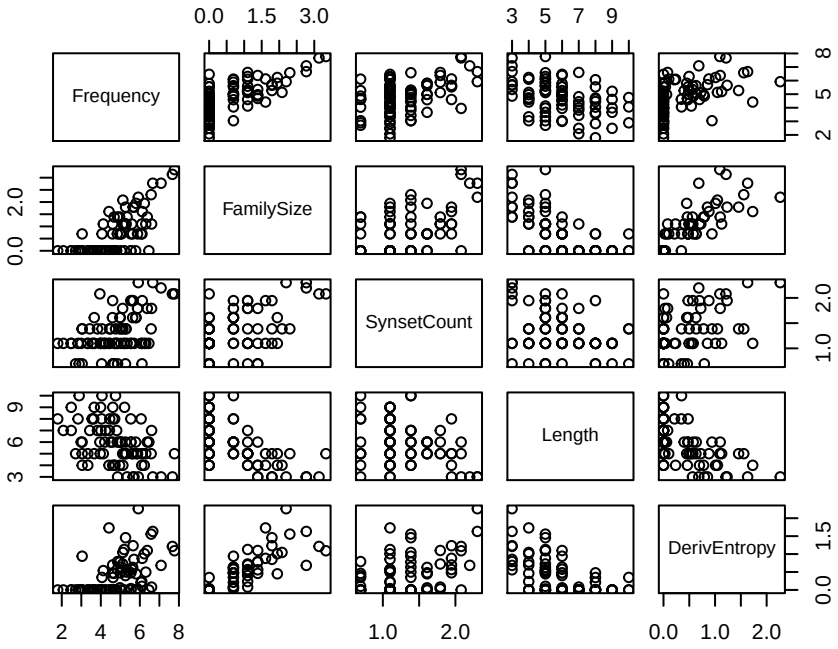



```
qplot(carat, price, data = small, shape = cut)
```



4.4 Scatterplot matrices: sploms

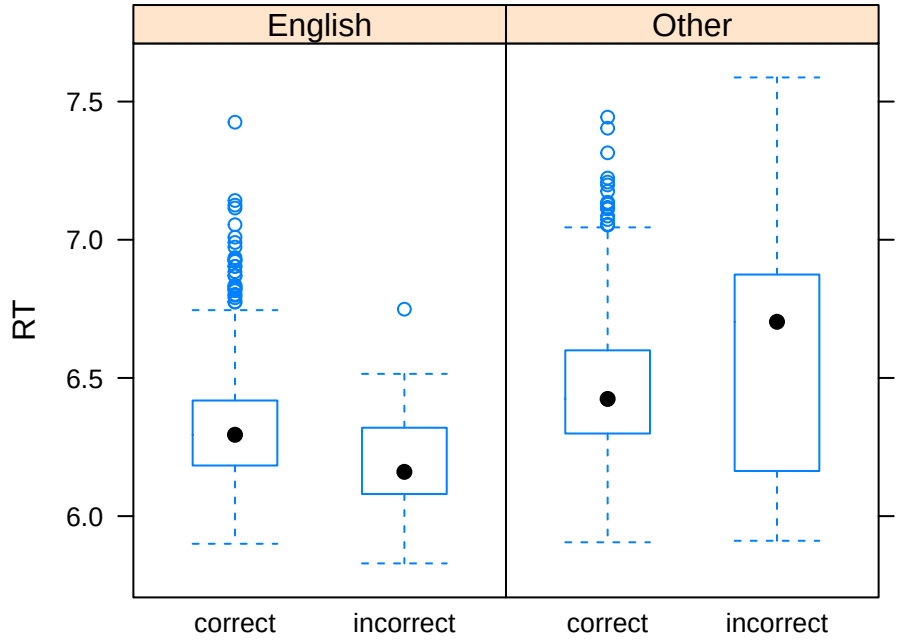
```
pairs(ratings[,-c(1,6:8,10:14)])
```



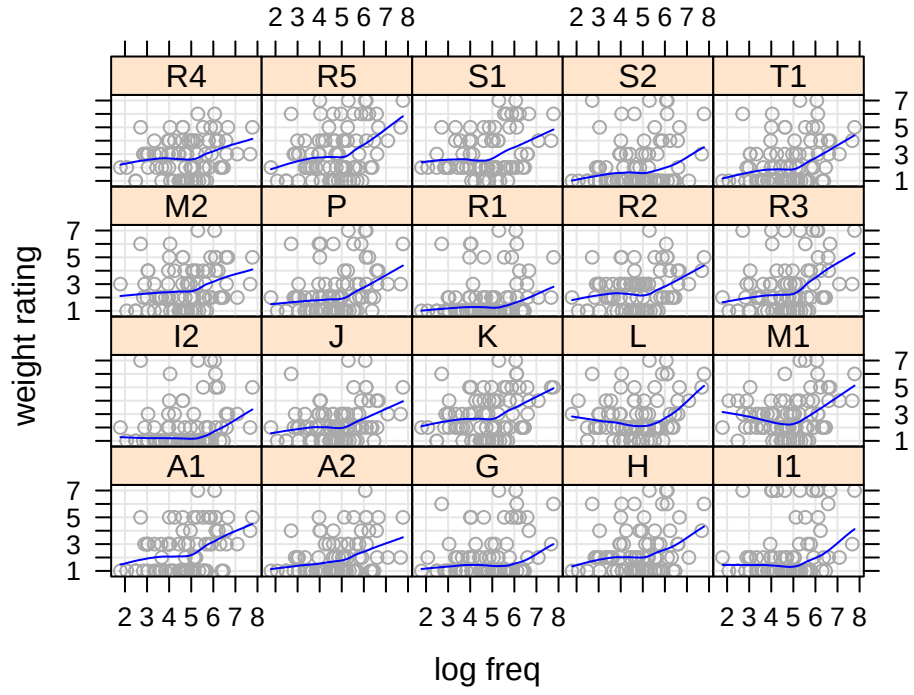
5 Trellis graphics

Trellis graphics give you very powerful visualization tools for studying grouped data. First load the library (install first if necessary).

```
# Now we look at xyplot that uses a lowess smoother; Baayen has written a function called xylowess.fnc:
library(lattice)
bwplot(RT~Correct|NativeLanguage,data=lexdec)
```




```
xylowess.fnc(Rating~Frequency|Subject,data=weightRatings,xlab="log freq",ylab="weight rating")
```

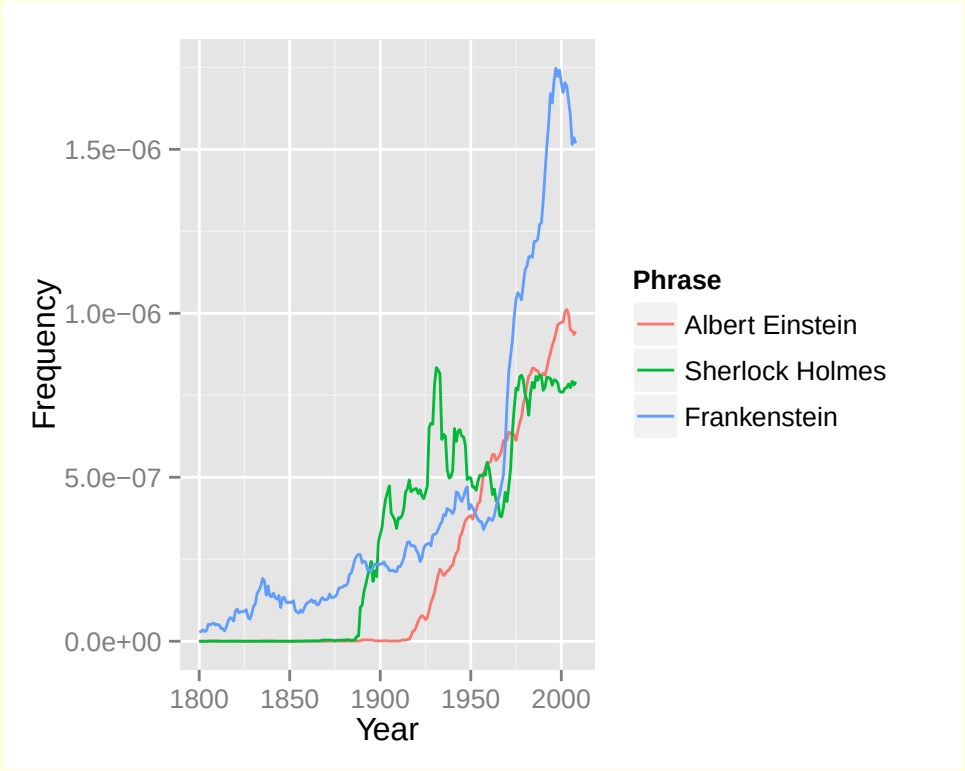


References

R Harald Baayen. Analyzing linguistic data. *A practical introduction to statistics using R*, 2008.

Exercise.1: Play with google book ngram corpus:

```
library(ngramr)
library(ggplot2)

ng <- ngram(c("Albert Einstein", "Sherlock Holmes", "Frankenstein"),year_start = 1800)


ggplot(ng,
       aes(x = Year,
           y = Frequency,
           colour = Phrase)) +
  geom_line()
```



Exercise.2 from (Baayen, 2008): The data set `english` in `LanguageR` lists lexical decision and word naming latencies for two age groups. Inspect the distribution of the naming latencies (`RTnaming`). First plot a histogram for the naming latencies with `truehist()`. Then plot the density. The voicekey registering the naming responses is sensitive to the different acoustic properties of a word' s initial phoneme. The column Voice specifies whether a word's initial phoneme was voiced or voiceless. Use `bwplot()` to make a trellis boxplot for the distribution of the naming latencies across voiced and voiceless phonemes with the age group of the subjects (`AgeSubject`) as grouping factor.

Exercise.3 from (Baayen, 2008):
The data set `moby` is a character vector with the text of Melville's *Moby Dick*. In this exercise, we consider whether **Zipf's law** holds for Moby Dick. According to Zipf's law (Zipf, 1949), the frequency of a word is inversely proportional to its rank in a numerically sorted list. The word with the highest frequency has rank 1, the word with the next highest frequency has rank 2, etc. If Zipf's law holds, a plot of log frequency against log rank should reveal a straight line. We make a table of word frequencies with `table()` - we cannot use `xtabs()`, because words is a vector and `xtabs()` expects a data frame—and sort the frequencies in reverse numerical order:

```
> moby.table = table(moby)
> moby.table = sort(moby.table, decreasing = TRUE)
> moby.table[1:5]
  moby
  the    of   and     a    to
  13655  6488  5985  4534  4495
```

We now have the word frequencies. We use the colon operator and `length()`, which returns the length of a vector, to construct the corresponding ranks:

```
> ranks = 1 : length(moby.table)
> ranks[1:5]
[1] 1 2 3 4 5
```

Make a scatterplot of log frequency against log rank.