MONASH University

MONASH BUSINESS SCHOOL

Introduction and Motivation

Data Visualisation and Analytics

Anastasios Panagiotelis
Lecture 1

Housekeeping

Welcome

- Lecturer: Anastasios Panagiotelis
- Email:
  - anastasios.panagiotelis@monash.edu
- Lecture slides and tutorial exercises.
- Data Visualisation: A practical introduction by Kieran Healey

My Slides

- My slides are designed to be interactive.
- They are best viewed in a web browser
  - Google Chrome and Firefox work best.
- Sometimes interactive plots won't look right or you will get stuck.
  - Just refresh (press F5).
- For a pdf version of the slides, from within Google Chrome simply print. There will be an option to print to pdf.

A History Lesson

Cholera

- In 1854 there was a breakout of the cholera disease in London killing 616 people.
- At the time it was speculated that the disease was carried in the air.
- A physician called John Snow was sceptical and began to collect data.

Snow's map

Snow's map

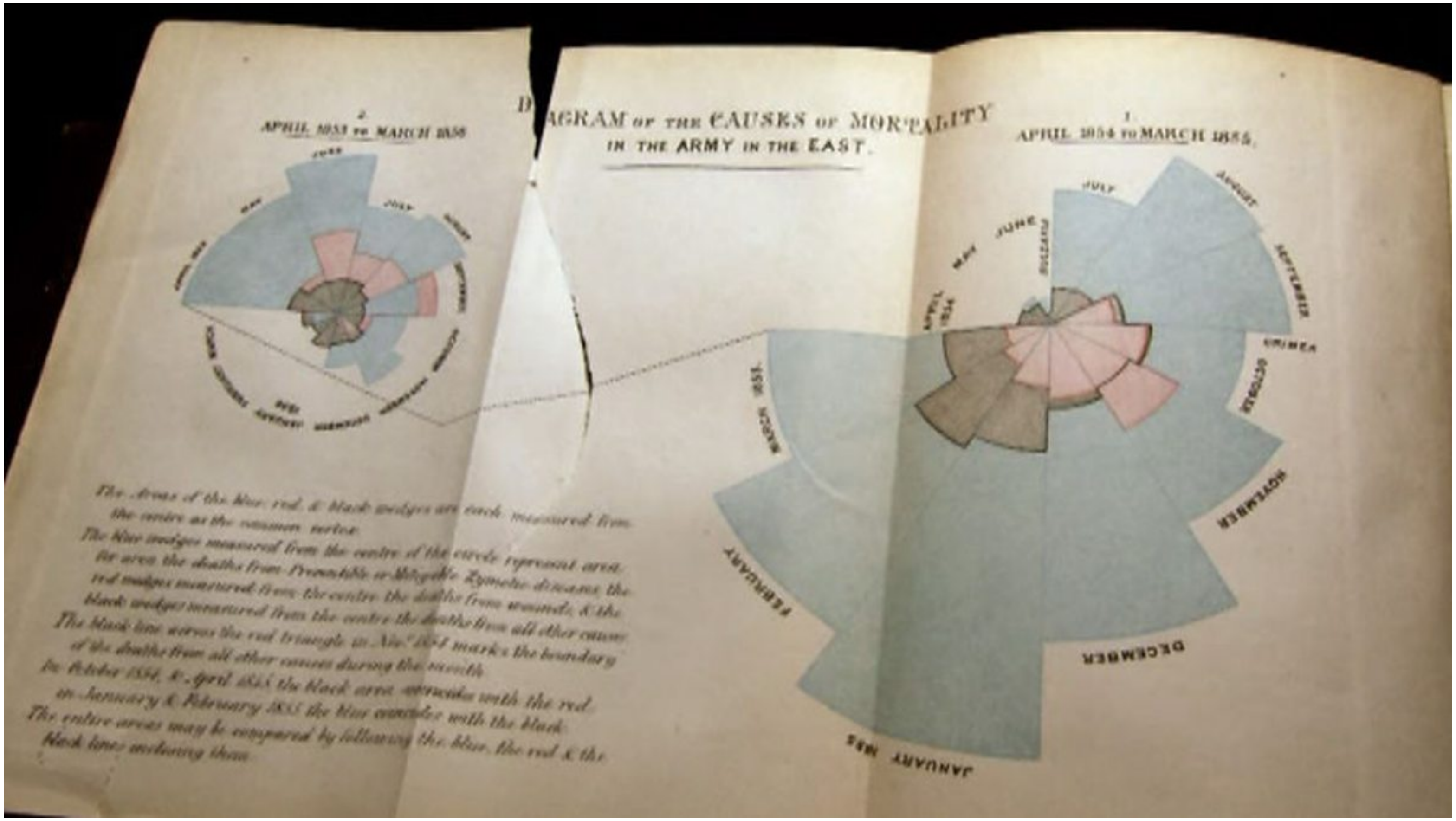
Consequences

- The map showed that cholera was more prevalent around a water pump on Broad Street.
- The pump was closed down.
- Eventually it was established that cholera is a water-borne disease.
- Data visualisation saves lives!

Crimean War

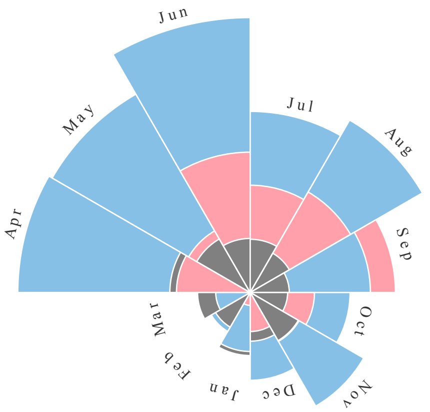
- At the same time Great Britain was at war against Russia in the Crimean peninsula.
- Florence Nightingale is famous as a nurse who treated the wounded soldiers.
- She also advocated British Parliament for more sanitary conditions in military hospitals.
- She knew the power of using data visualisation.

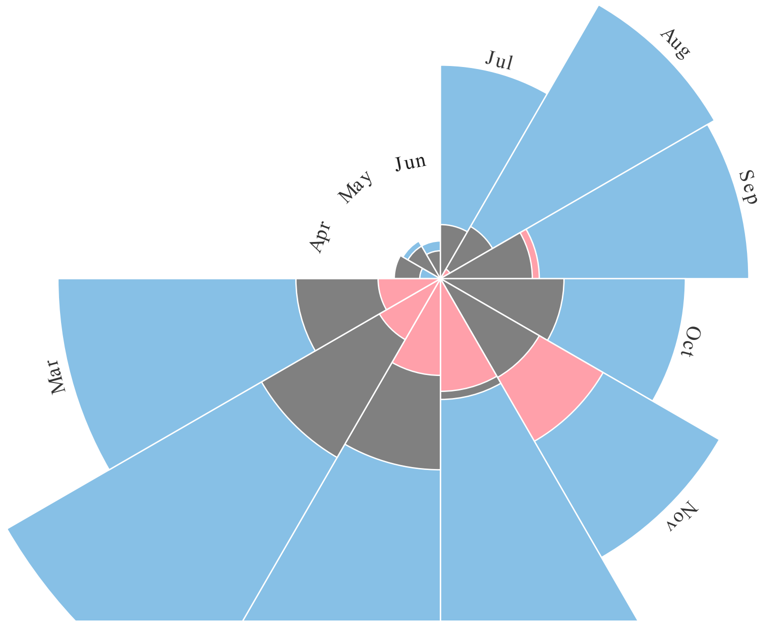
Nightingale's Rose chart



The Areas of the blue, red, & black wedges are each measured from
the centre as the common vertex.
The blue wedges measured from the centre of the circle represent area
for area the deaths from Preventible or Mitigable Zymotic diseases, the
red wedges measured from the centre the deaths from wounds, & the
black wedges measured from the centre the deaths from all other causes.
The black line across the red triangle in Nov^r 1854 marks the boundary
of the deaths from all other causes during the month.
In October 1854, & April 1855, the black area coincides with the red;
in January & February 1855, the blue coincides with the black.
The entire areas may be compared by following the blue, the red & the
black lines enclosing them.

Nightingale's Rose chart





- Blue areas: Preventible deaths.
- Red areas: Deaths from battle wounds.
- Black areas: Other causes.

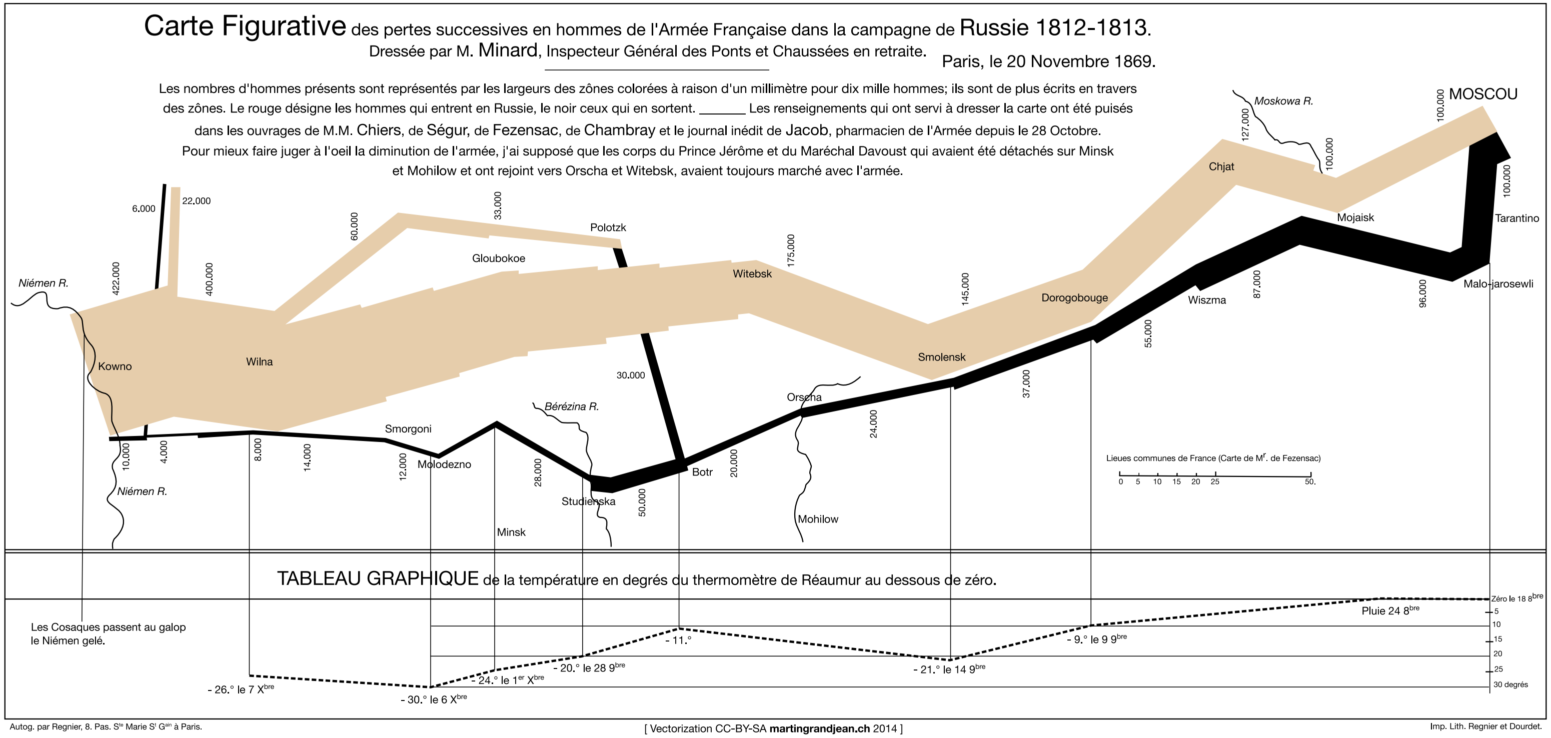
Aftermath

- The improved sanitation at military hospitals was eventually implemented in civilian hospitals.
- Data visualisation saves lives.
- Florence Nightingale became the first female member of the Royal Statistical Society.

Napoleon

- In 1812 Napoleon thought it was a good idea to invade Russia.
- This campaign was a disaster for the French.
- Engineer Charles Joseph Minard captured the extent of this catastrophe using visualisation.

Minard's plot



Minard's plot

- This visualisation provides information on 6 variables in one chart.
  - Number of troops.
  - Whether troops advance or retreat.
  - Temperature and time.
  - Longitude and latitude.
- Despite the clear message that invading Russia in winter is a bad idea, some people did not learn this lesson.

Why visualisation?

- Gain insights from data.
- Overview of large datasets.
- Search for:
  - Trends
  - Relationships
  - Irregularites
- In business data visualisation is a crucial tool to support decision making.

Modern Times

The R language

- Modern software tools make visualisation easily accessible.
- In this unit we will use the **R programming language** which can be downloaded here.
- Exact details of installing R will depend on whether you use Windows, Mac or Linux.
- A great tool for both new and experienced users of R is **RStudio** which can be downloaded here.

Using R

- To keep track of your workflow use a **script**:
  - You can open a new script by typing Ctrl+Shift+N
  - You can run a single line of code by pressing Ctrl+Enter
  - You can run a whole script by pressing Ctrl+Shift+S or Ctrl+Shift+Enter
- You can save scripts to run them anytime.
- Scripts allow you to keep analysis **replicable** which is important in research and business.

Variables in R

- In R everything is stored in a **variable**. Here the word variable has a slightly different meaning to the usual statistical meaning.
- In R, think of variables as little boxes or envelopes with names on them.
- We can put a number into these boxes, or words or matrices or entire blocks of data or even other boxes.

Assigning Variables

- How to store the number 1 in a variable `a` and the number 2 in a variable `b`?

```
a<-1
b=2
```

- Note that you can use either `<-` or `=` to assign variables.

Seeing results

There are a few options for looking at what is stored in a variable

```
print(b)
```

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

```
b
```

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

```
str(b)
```

```
##  num 2
```

Character variables

- We can store more than just numbers in a variable.
- Try to store your own name in a variable called `name`.

```r
name<-'Anastasios'
str(name)
```

```
##  chr "Anastasios"
```

- You must use apostrophes otherwise R will look for a variable called `Anastasios`.

Workspace

- All variables are kept in the **workspace**. You can see what is in your workspace by using the command

```
ls()
```

```
## [1] "a"    "b"    "name"
```

Clear Workspace

- You can clear the workspace using

```
rm(list=ls())
```

- If you try ls() again the workspace will be empty.
- In RStudio you can also see all the variables in the *Environment* tab.
- It is worth clearing the workspace at the beginning of every script.

Working directory

- If you try to read data from your hard drive, or save plots or data then the concept of a **working directory** is important. To check your working directory type

```
getwd()
```

```
## [1] "/home/anastasios/Documents/Teaching/Data Visualisation and Analytics 2
```

- To change the working directory use `setwd`

```
setwd("/home/anastasios/Documents")
```

Basic arithmetic in R

- Basic arithmetic is fairly simple. Try `a+b`. Also we will put this in a new variable called `z`.

```
z<-a+b
str(z)
```

```
##  num 3
```

- To subtract use `-`, to multiply use `*`, to divide `/` and to take powers use `^`.

Functions in R

- Apart from very simple arithmetic, variables in R are manipulated using a **function**.
- The input (also called argument) goes in parentheses, while the output can be assigned to a new variable.
- Some functions take more than one input. In this case separate by commas.

Example

- The function `sqrt` takes the square root.

```
rootb<-sqrt(b)
str(rootb)
```

```
##  num 1.41
```

What happens when you take a square root of something that is not a number?

```
rootname<-sqrt(name)
```

```
## Error in sqrt(name): non-numeric argument to mathematical function
```

Getting Help

- If you aren't sure what a function does, use R help. The easiest way is to simply use the `?`

```
?sqrt
```

- If you want to do something and do not know the name of the relevant function you can search using `??`. Try to find a function to do logarithms using

```
??logarithms
```

Comments

Anything after a `#` will not be executed by R.

```r
a<-1 # Set the variable a to 1
#x<-4 This line is not executed
str(a)
```

```
##  num 1
```

```r
str(x)
```

```
## Error in str(x): object 'x' not found
```

Comment multiple lines using Ctrl+Shift+C

Vectors

We can create a variable with multiple numbers or strings using the `c` function.

```
Consumption<-c(50,40,25,0)
str(Consumption)
```

```
##  num [1:4] 50 40 25 0
```

```
Drink<-c('Coke','Pepsi','Coke','Homebrand')
str(Drink)
```

```
##  chr [1:4] "Coke" "Pepsi" "Coke" "Homebrand"
```

Vector

These variables are example of a **vector**. Sometimes when we apply a function to a vector, we apply the function to each element.

```
logcons<-log(Consumption)
str(logcons)
```

```
##  num [1:4] 3.91 3.69 3.22 -Inf
```

Vectors

Other functions take a vector as an input and return a single number as the output

```
meancons<-mean(Consumption)
str(meancons)
```

```
##  num 28.8
```

Inf and NaN

There are *special* values that numeric variables can take. These are `Inf` and `-Inf` for positive and negative infinity and `NaN` for not a number. The presence of `NaN` indicates an error.

```r
log(-1)
```

```
## Warning in log(-1): NaNs produced
```

```
## [1] NaN
```

It is important to distinguish `NaN` from `NA`. The latter is used for missing data.

Lists

Another object common in R is known as a **list**. A list can contain completely different types of variables.

```r
alist<-list(w=name, x=Drink, y=Consumption)
```

elements of lists are accessed using `[[]]` or `$`

```r
alist[[1]]
```

```
## [1] "Anastasios"
```

Packages

R Packages

- A big advantage of R is the use of add-on packages, easily downloaded from an online repository called **CRAN**.
- Using a package involves two steps:
  - Download and install the package using the function `install.package` (do once).
  - Load the package using `library` function (include at beginning of script).
- Both these steps can also be done in RStudio through the *Packages* tab.

Options in installing packages

- If you have not already done so, download, install and load the R package `ggplot2`

```r
install.packages('ggplot2')
```

To load the package

```r
library(ggplot2)
```

- By downloading the package you also download all of the help documentation.

The tidyverse

- When you have time, download the `tidyverse` package
- The `tidyverse` is a collection of packages.
  - `readr` is used for reading in data.
  - `dplyr` and `tidyr` is used for manipulating data into an easy to use format.
  - `ggplot2` is used for visualisation.

Anscombe's quartet

Plotting data

- Anscombe's quartet is a synthetic dataset used to demonstrate the importance of data visualisation.
- We will also use it to learn some basic R.
- The data comes built into R.
- There are 4 pairs of x and y variables.

Anscombe's quartet

```
str(anscombe)
```

```
## 'data.frame':	11 obs. of  8 variables:
##  $ x1: num  10 8 13 9 11 14 6 4 12 7 ...
##  $ x2: num  10 8 13 9 11 14 6 4 12 7 ...
##  $ x3: num  10 8 13 9 11 14 6 4 12 7 ...
##  $ x4: num  8 8 8 8 8 8 8 19 8 8 ...
##  $ y1: num  8.04 6.95 7.58 8.81 8.33 ...
##  $ y2: num  9.14 8.14 8.74 8.77 9.26 8.1 6.13 3.1 9.13 7.26 ...
##  $ y3: num  7.46 6.77 12.74 7.11 7.81 ...
##  $ y4: num  6.58 5.76 7.71 8.84 8.47 7.04 5.25 12.5 5.56 7.91 ...
```

Summary stats

- We can find the mean of the final pair using the `mean` function.

```
xbar<-mean(anscombe$x4)
ybar<-mean(anscombe$y4)
str(xbar)
```

```
##  num 9
```

```
str(ybar)
```

```
##  num 7.5
```

Summary stats

- We can find the variance of the final pair using the `var` function.

```
vx<-var(anscombe$x4)
vy<-var(anscombe$y4)
str(vx)
```

```
##  num 11
```

```
str(vy)
```

```
##  num 4.12
```

Summary stats

- We can find the correlations between x and y using the `cor` function.

```
rxy<-cor(anscombe$x4,anscombe$y4)
str(rxy)
```

```
##  num 0.817
```

- There are two inputs or *arguments* to the function. Separate these using a `,`

Your turn

- If your birthday is from January to April:
  - Find the mean and variance of x1 and y1 and their correlation
- If your birthday is from May to August:
  - Find the mean and variance of x2 and y2 and their correlation
- If your birthday is from September to December:
  - Find the mean and variance of x3 and y3 and their correlation

Conclusions

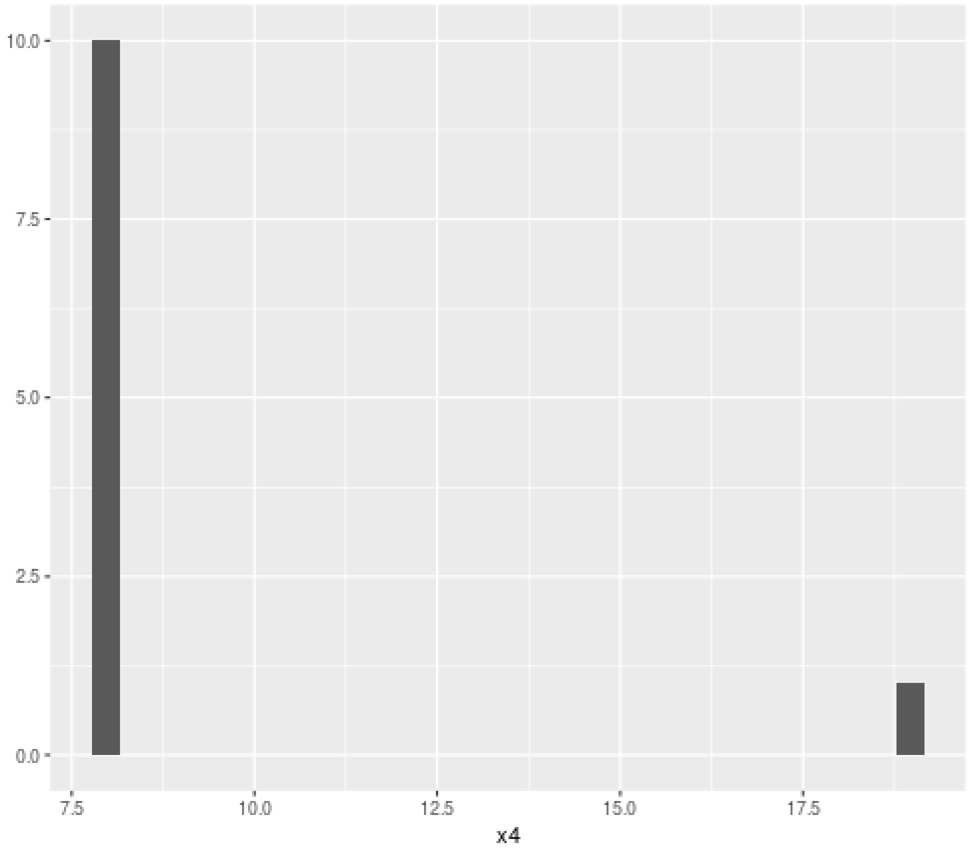
- Results
  - The means of all x variables are 9
  - The means of all y variables are 7.5
  - The variances of all x variables are 11
  - The variances of all y variables are 4.12
  - The correlation between x and y is 0.82
- Does this mean all datasets are equal?

Let's visualise

- Later on we will use the `ggplot` function to create figures.
- For now we can use a simple function within the `ggplot2` package called `qplot`.
- Simply tell `qplot` the variable(s) that you want to plot and the dataset.
- The `qplot` function tries to guess what type of plot you want.

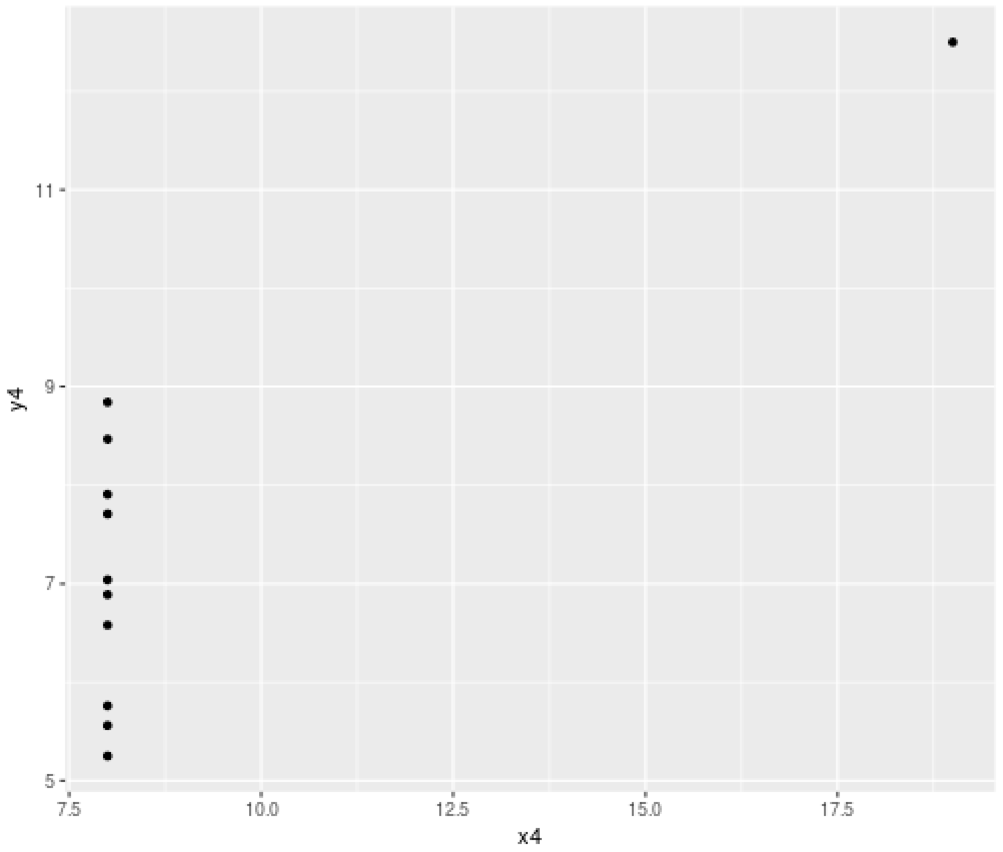
Histogram

```
qplot(x4,data = anscombe)
```



Scatterplot

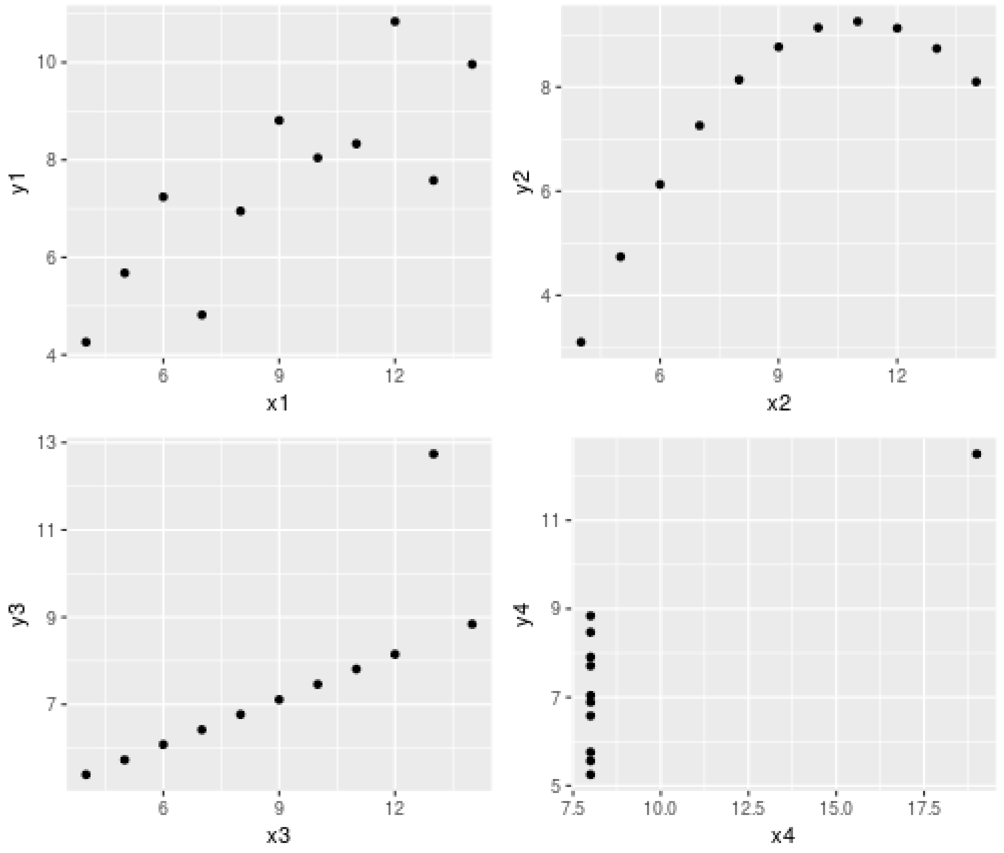
```
qplot(x4,y4,data = anscombe)
```



Your turn

- If your birthday is from January to April:
  - Plot histograms of x1 and y1 and a scatterplot.
- If your birthday is from May to August:
  - Plot histograms of x2 and y2 and a scatterplot.
- If your birthday is from September to December:
  - Plot histograms of x3 and y3 and a scatterplot.

All Results



y1
10
8
6
4
6
9
12
x1
y2
8
6
4
6
9
12
x2
y3
13
11
9
7
6
9
12
x3
y4
11
9
7
5
7.5
10.0
12.5
15.0
17.5
x4

Visualisation

- Although all four datsets have the same summary stats they are vastly different.
- These differences can easily be seen using visualisation.
- Always look at your data as part of an analysis.

Where now

- Clearly `qplot` is quite limited in what it is able to do.
- Over the next period we will consider:
  - More ways to plot a single variable.
  - More ways to plot relationships between two or more variables.
  - Visualising variables that are categorical.
- Before getting into those details we cover some general principles of good plotting.