UNIVERSITY OF EXETER
Q-STEP CENTRE

Q-Step

A step-change in quantitative social science skills

Funded by the Nuffield Foundation, ESRC and HEFCE

An Introduction to R

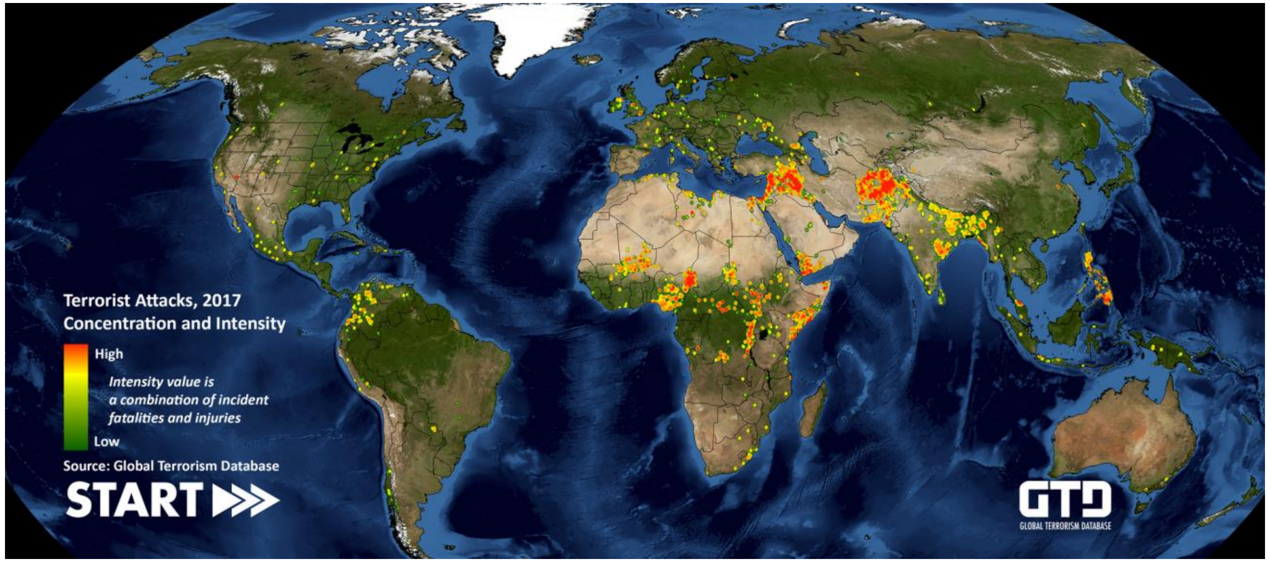
Q-Step Workshop – 09/01/2019

Lewys Brace
l.brace@Exeter.ac.uk

The Global Terrorism Database (GTD)

http://www.start.umd.edu/gtd/

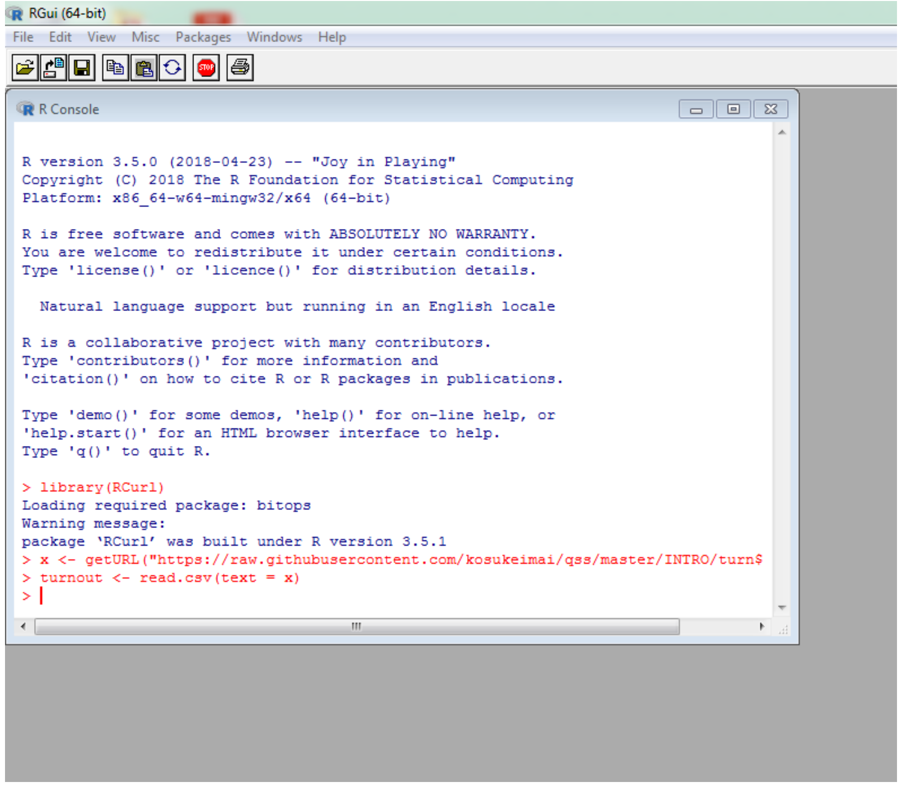
- Open-source database.
- Includes terrorist events between 1970-2017 (annual updates).
- Includes domestic and international incidents.

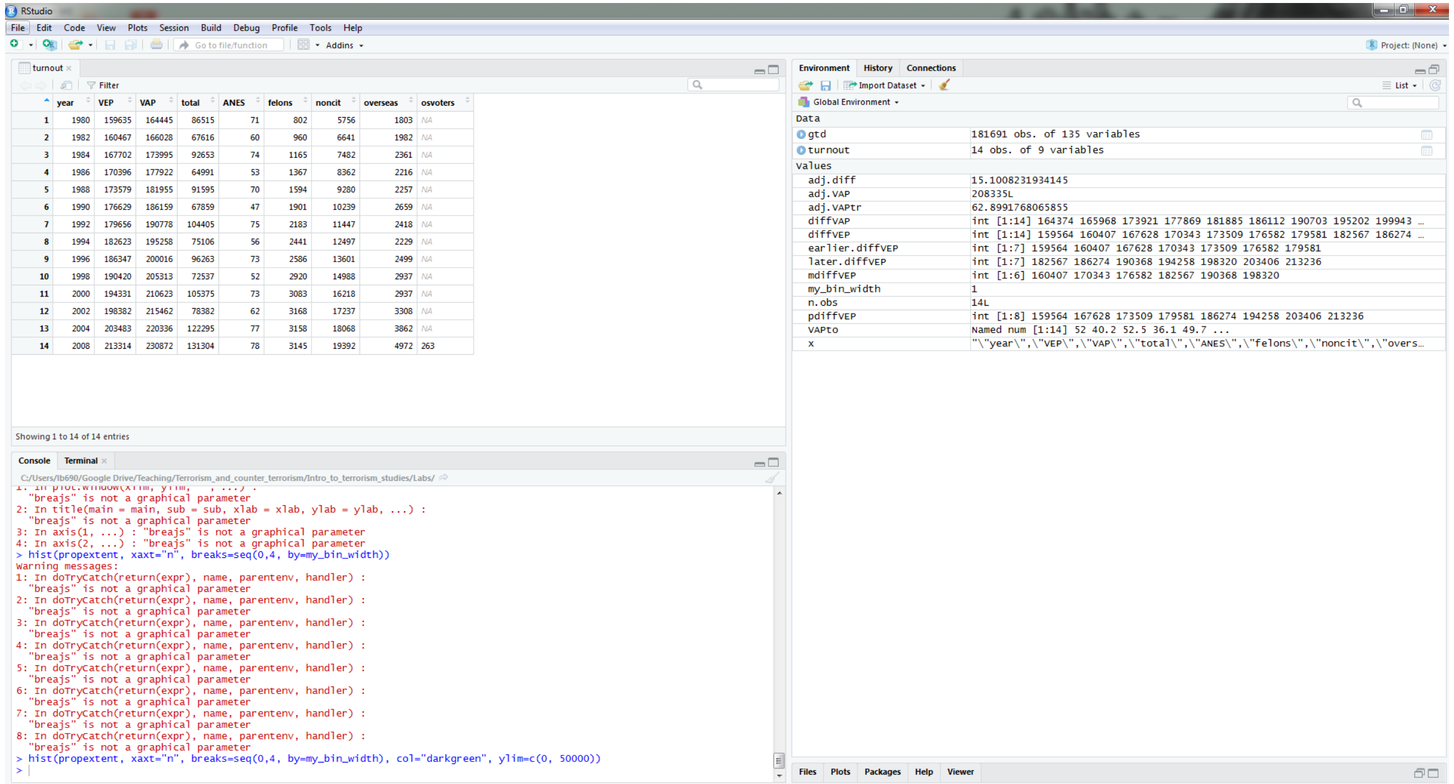


What is R?

- An open-source package for statistical analysis.
- Widely supported.
- The initial ‘learning curve’ for R is steep, it is worth learning in the long run for a host of reasons.
- You will get stuck at times, we all do.
- Thus, your best friend will be….

R and R Studio





R Can be downloaded from:
https://www.r-project.org/

R Studio can be downloaded from:
https://www.rstudio.com/products/rstudio/download/

Note: You will need to install base R before you can use R Studio.

Opening the dataset and loading it into R

- First, unzip the folder and save the entire GTD as a csv file.
- Then, we'll load it into R.
- This is most commonly done by:

Assignment operator

```
> data <- read.csv("C:/User/Documents/dataset.csv", header=T)
```

- You can also load in data files in the following manner:

```
> data <- read.csv(file.choose(), header=T)
```

- Load the complete GTD into R and assign it to the variable 'entire_gtd'.

The dataframe

- R has now loaded the csv data file and created a dataframe out of it.
- These are commonly abbreviated to df.
- It's the default way that data-focused software deals with data.
- For example:

Case ID	Variable one	Variable two	Variable 3
1	123	ABC	10
2	456	DEF	20
3	789	XYZ	30

Exploring a data set in R

- R offers a lot of functions with which to explore datasets:

  - summary(x) = Provide summary statistics for dataframe.

  - head(x) = Displays the top 6 records in the dataframe.

  - tail(X) = Displays the bottom 6 records of the dataframe.

  - nrow(X) = Displays the number of rows in the dataframe.

  - ncol(X) = Displays the number of columns (variables) in the dataframe.

  - dim(X) = Returns the dimension of the dataframe (the number of rows and number of columns).

Note: in each command, replace X with the name of your dataframe or the column od data you're interested in.

Exercise 2 - Exploring the dataset

- We now have a dataset loaded into the R environment as a dataframe.
- There are some natural questions to ask about these data:
    1. What is the *unit of analysis* (what does each row represent)?
    2. How many *cases* are in the dataset?
    3. How many *variables* are in the dataset?

1. What is the unit of analysis?

```
> head(entire_gtd)
       eventid iyear imonth iday approxdate extended resolution country          country_txt region
1 1.97000e+11  1970      7    2                    0                 58 Dominican Republic      2
2 1.97000e+11  1970      0    0                    0                130             Mexico      1
3 1.97001e+11  1970      1    0                    0                160        Philippines      5
4 1.97001e+11  1970      1    0                    0                 78             Greece      8
5 1.97001e+11  1970      1    0                    0                101              Japan      4
```

- So, the unit of analysis is terrorist events.
- In other words, each row represents a different terrorist incident.

2. How many cases are in the dataset?

```
> nrow(entire_gtd)
[1] 181691
```

3. How many variables are in the dataset?

```
> ncol(entire_gtd)
[1] 135
```

- You could have equally done:

```
> dim(entire_gtd)
[1] 181691    135
```

Exploring variables

- You can use the $ symbol to address a single column (variable) in a dataframe:

```
> length(entire_gtd$eventid)
[1] 181691
```

- Equally, you could use the attach() function, which then allows you to refer to a variable without having to reference the dataframe:

```
> attach(entire_gtd)
> min(eventid)
[1] 1.97e+11
```

- R has a number of different functions for exploring individual variables:
  - min(X) = Displays the minimum value.
  - max(X) = Displays the maximum value.
  - sum(X) = Sums the input data range.
  - mean(X) = Provides the arithmetic mean of the input range. Similar functions exist for the median, variance, std dev, etc.
  - unique(X) = Displays all of the different values for the input data range.
  - length(X) = Returns the number of non-missing records in a variable.
  - which(X) = Returns the indices of elements that are TRUE, given a criteria

- You can use logical operators in conjunction with some functions in order to explore data:

  - How many non-missing observations do we have in 'event id'?:

```
> length(eventid)
[1] 181691
```

  - Are any of the incident months greater than 12?:

```
> any(imonth > 12)
[1] FALSE
```

  - Which observations involve North America?:

```
> which(region_txt == "North America")
  [1]     2     6     8     9    10    11    12    14    15    18
 [24]    37    38    39    41    42    43    44    45    46    47
```

Logical operators:

==	Is equal to
!=	Is not equal to
>	Greater than
<	Less than
>=	Greater than or equal to
<=	Less than or equal to
is.na(X)	is a missing value
is.null(X)	Is a null value

Exercise 3 – Combining functions

1. Work out how many terrorist incidents took place in Egypt.
   - Hint: You may need to concatenate two functions.

- min(X) = Displays the minimum value.
- max(X) = Displays the maximum value.
- sum(X) = Sums the input data range.
- mean(X) = Provides the arithmetic mean of the input range. Similar functions exist for the median, variance, std dev, etc.

- unique(X) = Displays all of the different values for the input data range.
- length(X) = Returns the number of non-missing records in a variable.
- which(X) = Returns the indices of elements that are TRUE, given a criteria

Answer:

```
> length(which(country_txt == "Egypt"))
[1] 2479
```

Exercise 4 – Bringing the basics together

1. Calculate the percentage of successful incidents and the percentage of unsuccessful incidents in the data set.

$$percentage = \frac{Total\ number\ of\ observations\ that\ fulfil\ criteria}{Total\ number\ of\ observations} * 100$$

```
> percent_successful <- (length(which(success == 1))/length(success)) *100
> percent_successful
[1] 88.95983
> percent_unsuccessful <- 100 - percent_successful
> percent_unsuccessful
[1] 11.04017
```

Sub-setting data

- Sometimes, you'll want to look at a subset of the entire dataframe.
- R's subset function creates a new dataframe, which is a subset of the entire dataframe, in accordance with the conditions you specify.
- For example, if we wanted to look at the GTD data just for Russia:

```
> russia <- subset(gtd, subset=gtd$country_txt=="Russia")
```

country	var2	var3	var4
Romania	123	312	798
Romania	536	123	753
Russia	545	654	951
Russia	666	456	645
Russia	896	897	289
Rwanda	235	448	143
Rwanda	232	556	561

Country	Var2	Var3	Var4
Russia	454	654	951
Russia	666	456	645
Russia	896	897	289

Graphical data exploration

- Graphing data is one of the most important parts of data analysis, and is the first thing you should do after cleaning your data.
- Graphing data offers a number of functions:
  1. It allows your to explore your data, by allowing you to easily identify unusual values and often aiding in the decision of what kind of analysis to subject the data to.
  2. Enables you to ensure that your chosen model is a realistic fit to the data.
  3. Allows you to communicate the nature of your data by summarising numerical information.

Graphs for EDA

- EDA should always be carried out prior to any formal statistical analysis.
- EDA allows you to check your data for:
  1. To ensure you have meaningful data.
  2. Detect errors that were made during data entry.
  3. To detect patterns in the data that may go undetected by the statistical analysis that you use.
  4. To ensure that the assumptions of your data are met.
  5. To interpret the departures from the assumptions.
  6. To detect unusual values, which are known as ‘outliers’.
- Graphing is the most important part of this exploration.

How to present data

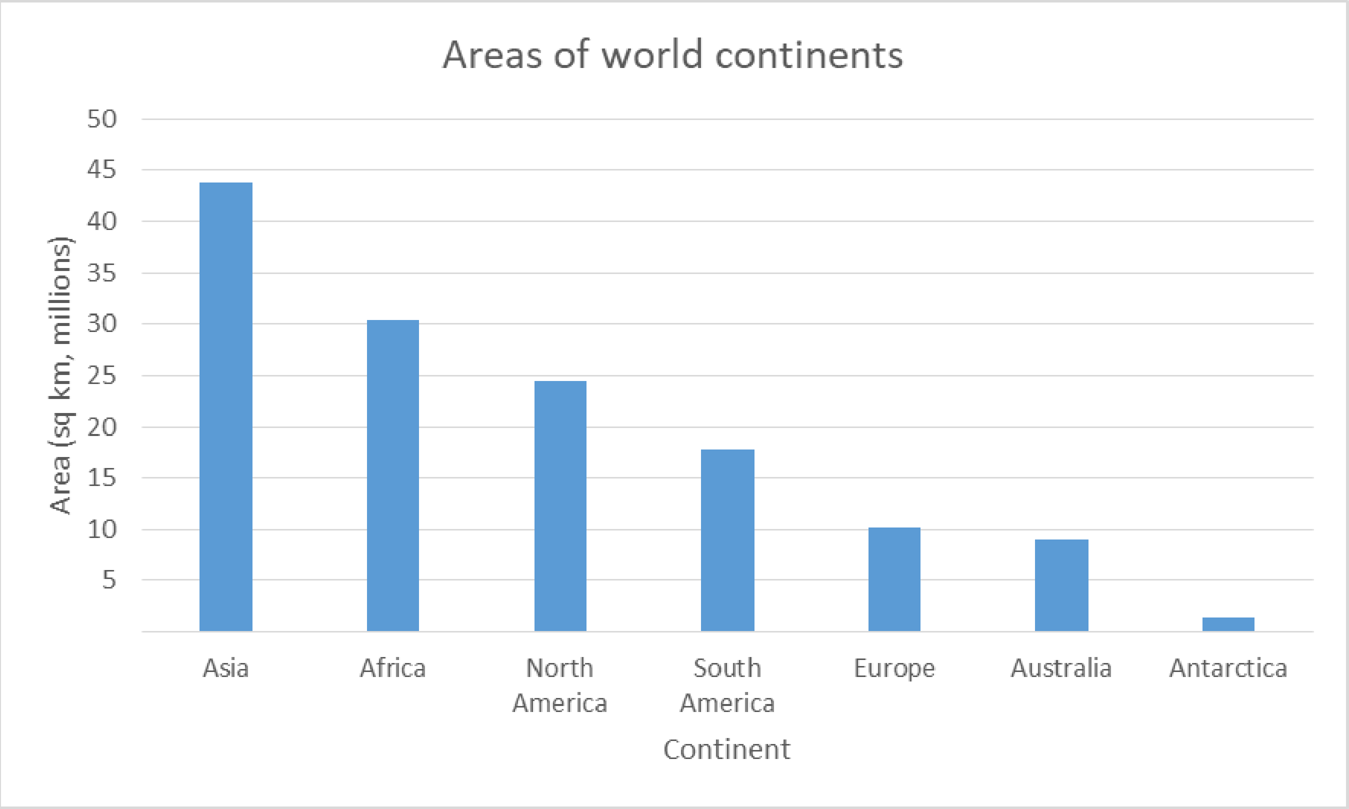
Graphs should, among other things:

- Show the data.
- Induce the reader to think about the data being presented, rather than focusing on the graph itself; i.e. what colour the bars are, etc.
- Avoid distorting the data.
- Present many numbers with minimal ink.
- Make large data sets, if that’s what you’re working with, coherent.
- Encourage the reader to compare different pieces of data.
- Reveal data.

Bar charts

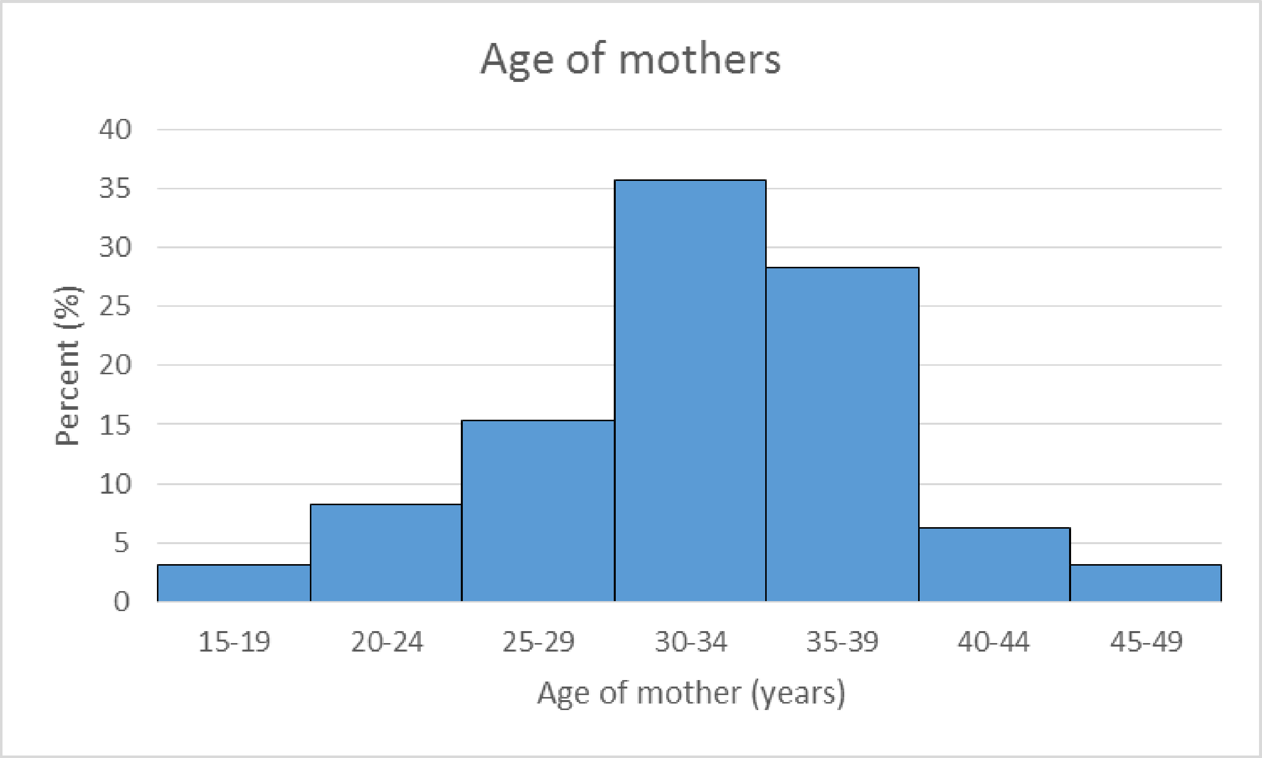
- It provides an instant picture of the relative sizes of different categories.

- It is considered good practice to list the categories in size order, as they are here.
- It is often worth simplifying the numbers on the axis; i.e. '30,000000km^2' becomes '30 millionkm^2'. This is true for most graph type.



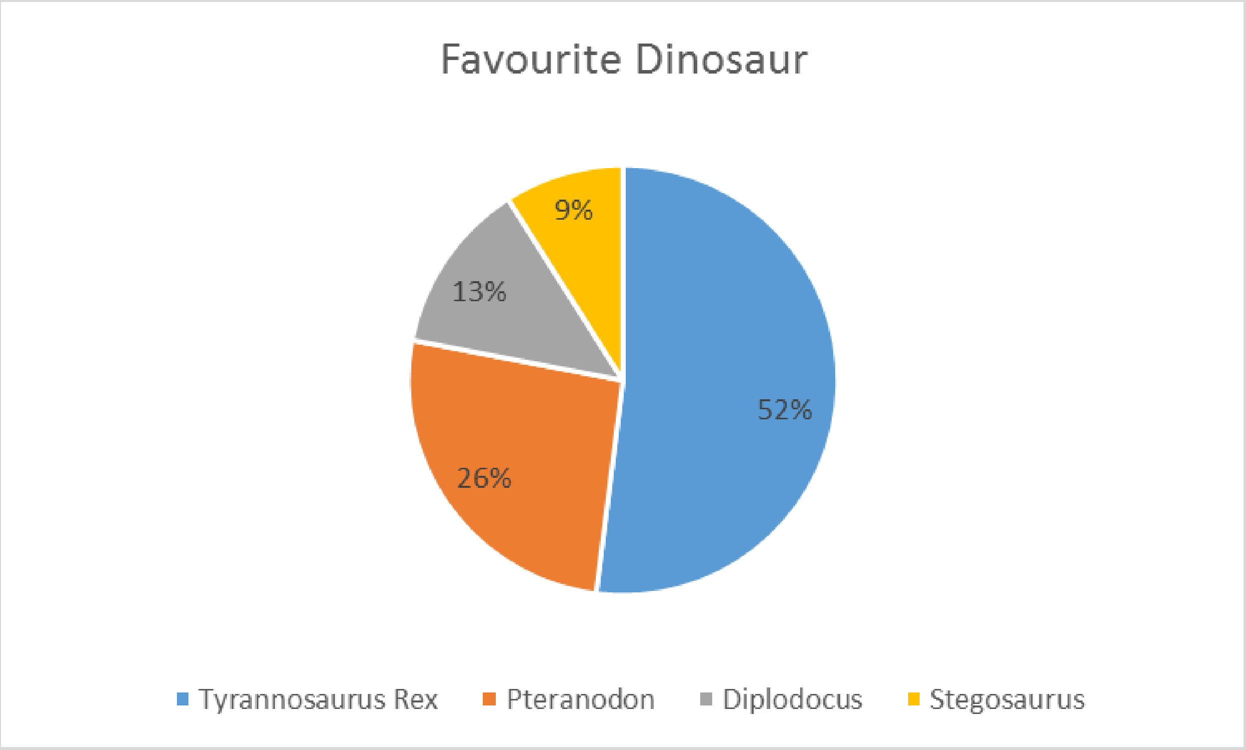
Histograms

- Here, all of the bars are touching, indicating that horizontal axis represents a continuous number scale.
- Often is to summarise a distribution.



Pie charts

- Frowned upon in some circles, but total valid if used correctly.
- They are valid if you have a complete set, the component parts of which add up to 100% of the 'pie'



- To calculate the amount of the pie that a Pteranodon, which constitutes 26%:

$$360 * \frac{26}{100} = 93.6$$

Pteranodon pie section = 93.6°

Choosing the correct graph

- Three things to consider when deciding which graph to use:
  1. The type of data being represented.
  2. The type of statistical judgement which you hope the graph will help you to make.
  3. And…

Discrete and continuous data

Discrete:

Investigation	Typical items of data
Types of wildflower	Bluebell, marigold, meadow-sweet
Household size	1 person, 2 people, 3 people, 4 people
Environmental problems	Oil spills, acid rain, dog fouling
Vehicle colour	Red, green, blue

Continuous:

Investigation	Typical items of data
Babies' birth weight	3120g, 3760g, 2700g
Temperature survey	18.6◦c, 21.4◦c, 19.0◦c
Survey of commuting times	23 mins, 11 mins, 70 mins

Types of data

- Categorical (entities are divided into distinct categories):
  - Binary variable (There are only two categories (i.e. dead or alive).
  - Nominal variable: There are more than two categories (i.e. whether someone is an omnivore, vegetarian, vegan, or fruitarian).

- Ordinal variable:
  - -The same as a nominal variable, but the categories have a logical order (i.e. whether a student got a fail, pass, merit, or distinction in an exam).

- Continuous (entities get a distinct score):
  - Interval variable: Equal intervals on the variable represent equal differences in the property being measured (i.e. the difference between 6 and 8 is equivalent to the difference between 13 and 15).
  - Ratio variable: The same as an interval variable, but the ratios of scores on the scale must also make sense (i.e. a score of 16 on an anxiety scale means that the person is, in reality, twice as anxious as someone scoring 8).

Class test

Data item	Type of data: D or C?
M27, M5, M1	Discrete
1929, 1930, 1931	Continuous
1.2m, 1.3m, 1.4m	Continuous
4.5, 4.6, 4.7	Continuous
Ant, Butterfly, Bee	Discrete

Graphs in R

- R has a base graphics package.
- But ggplot2 is the library that most people use when graphing in R.
- This involves becoming familiar with R packages.

R packages

- Check which libraries are already installed:

In:
```
> library()
```

Out:
```
Packages in library '\\isad.isadroot.ex.ac.uk/UOE/User/R/win-library/3.5':

assertthat              Easy Pre and Post Assertions
backports               Reimplementations of Functions Introduced Since R-3.0.0
base64enc               Tools for base64 encoding
BH                      Boost C++ Header Files
```

- ggplot2 is installed automatically as part of the default R installation.
- Not all packages are loaded during R start up; ggplot2 is one of these.
- We load a package into R by:

```
> library(ggplot2)
```

Frequency table

- The propextent variable measures the amount of property damage that resulted from a terrorist incident.
- For data like these, sometimes the easiest way to get an understanding of what is being shown is through a frequency table:

Extent of Property Damage
(propextent; propextent_txt)
Categorical Variable

If "Property Damage?" is "Yes," then one of the following four categories describes the extent of the property damage:

1 = Catastrophic (likely ≥ $1 billion)
2 = Major (likely ≥ $1 million but < $1 billion)
3 = Minor (likely < $1 million)
4 = Unknown

```
> prop_freq_table <- transform(table(propextent))
> prop_freq_table
  propextent  Freq
1          1     6
2          2   909
3          3 43304
4          4 19846
```

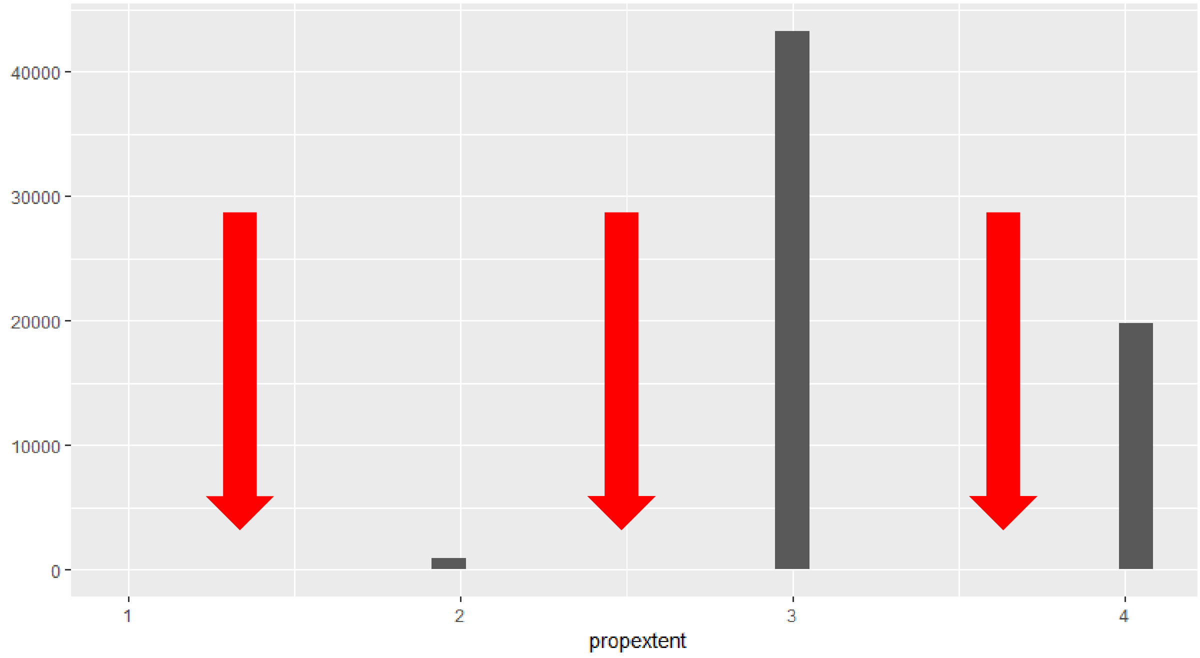
- This is useful, but sometimes we require a graphical representation of this information.

Histograms

- Histograms are a great way to visualise the frequency distribution of a variable.
- Creating histograms in R is easy and done with the hist() function.

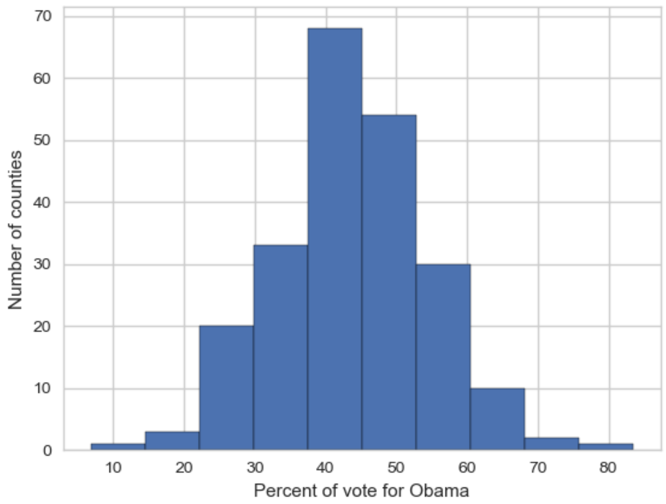
```
> ggplot(entire_gtd, aes(x=propextent))+geom_histogram()
```

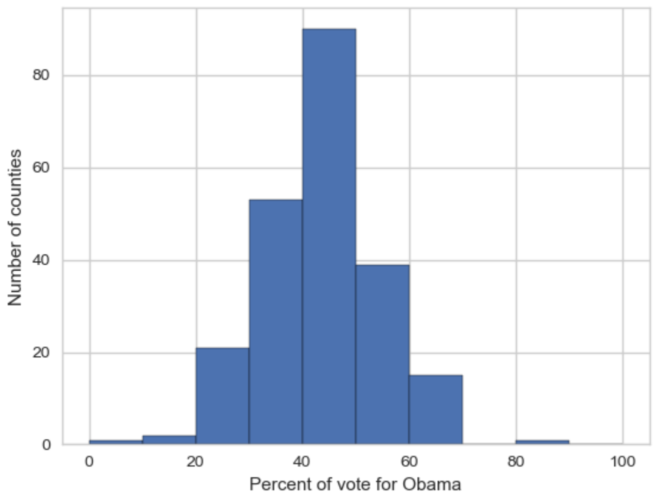
- What is the problem with this graph?



Bins

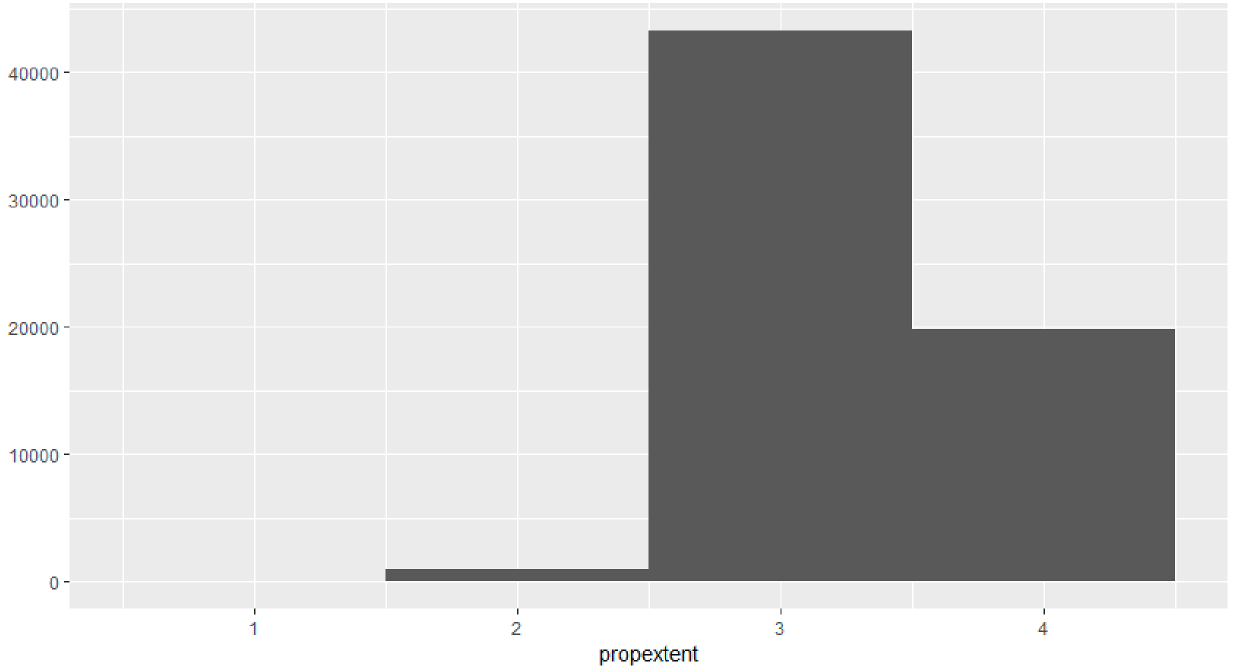
- This is because R is using its default *bins*, which aren't quite right for the data we've used as input.
- Binning is important to think about when producing histograms in any language.
- As an example, the two histograms below show the **exact** same data, but the graphs have different bin values.





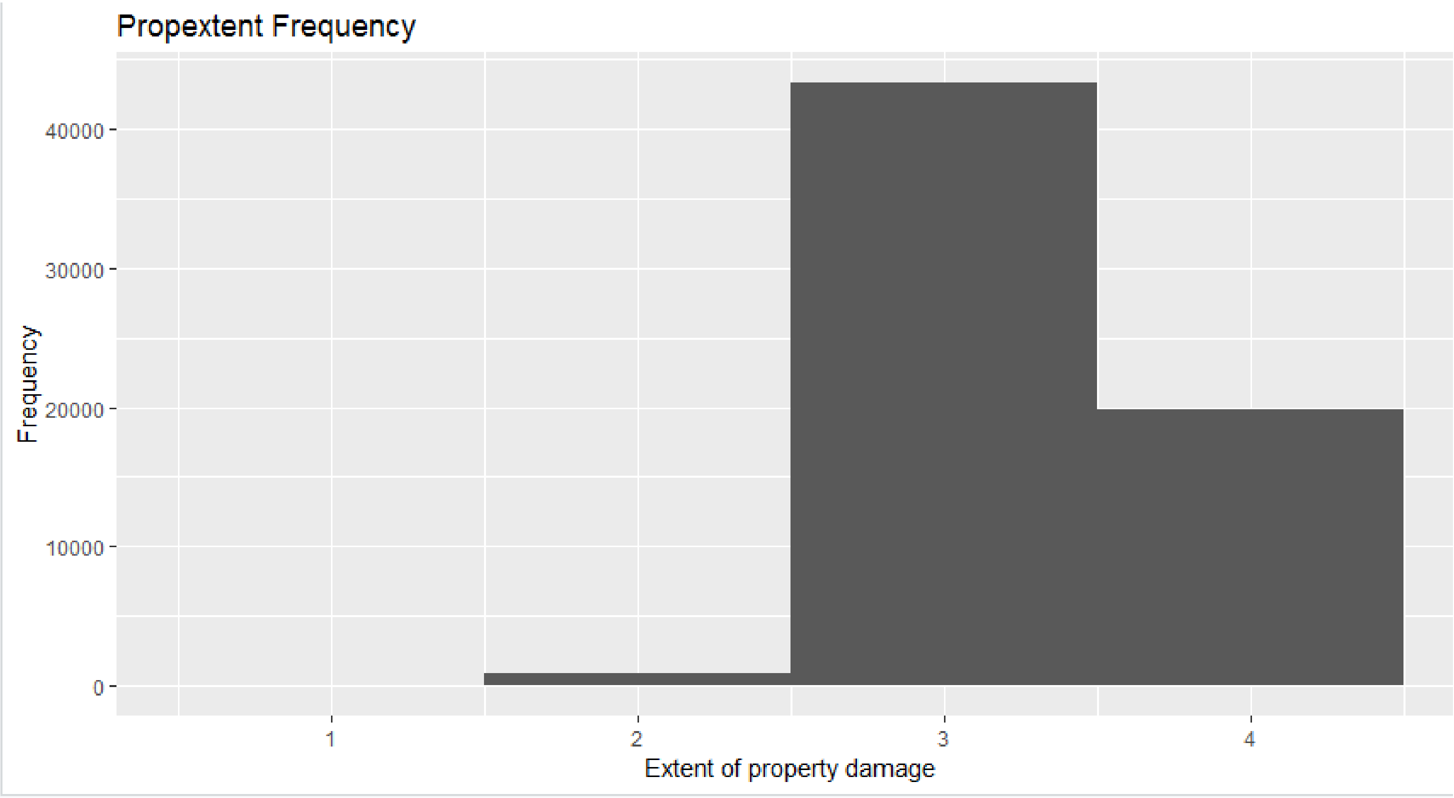
- So, we re-run the piece of code we just created, but this time, we specify the bin width:

```
> ggplot(entire_gtd, aes(x=propextent))+geom_histogram(binwidth=1)
```



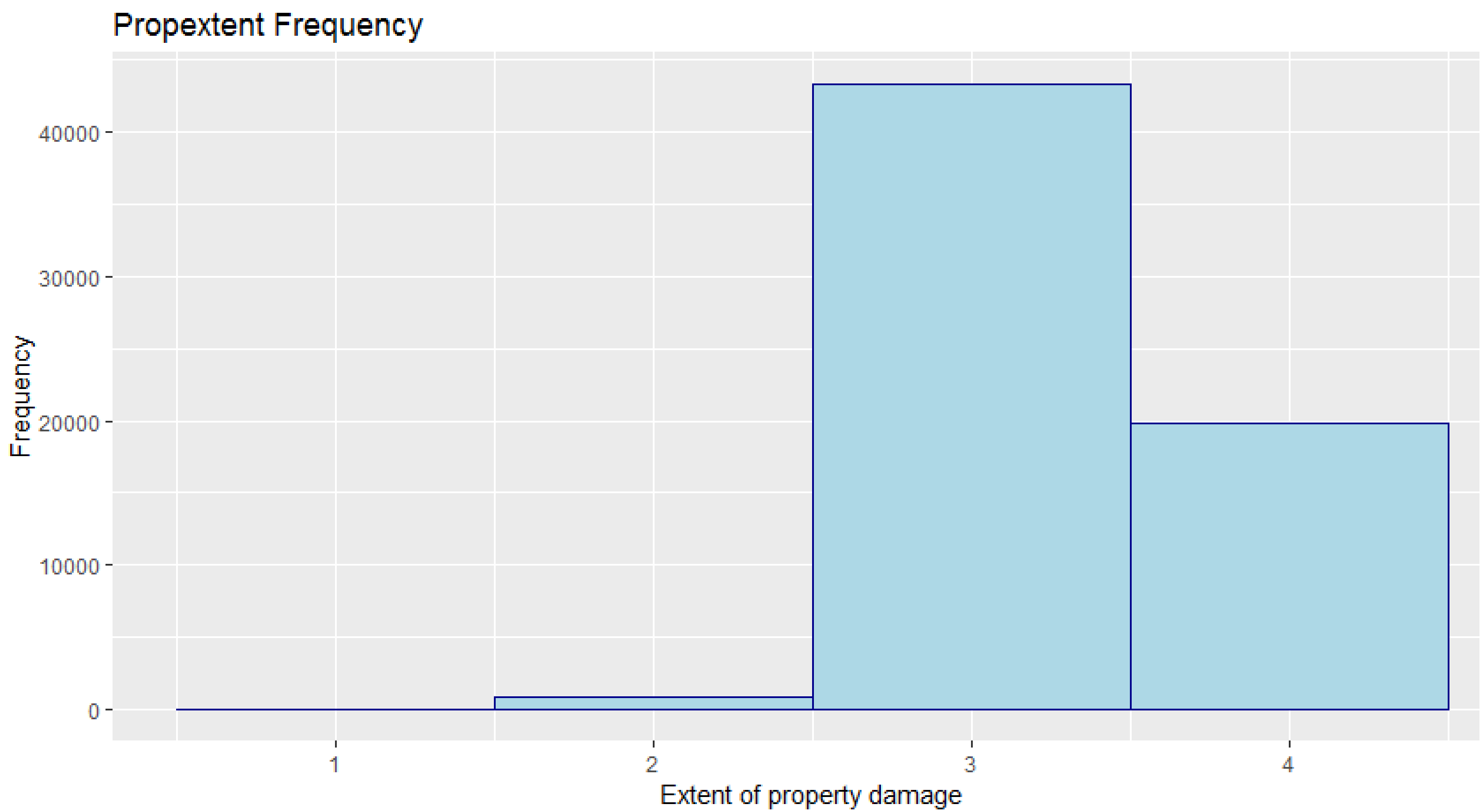
- Let's add a title and customise the x and y axis labels:

```
> ggplot(entire_gtd, aes(x=propextent))+geom_histogram(binwidth=1)+
geom_histogram(binwidth=1)+ggtitle("Propextent Frequency")+labs(y="Frequency", x="Ext
ent of property damage")
```



Making it pretty

```
> ggplot(entire_gtd, aes(x=propextent))+geom_histogram(binwidth=1, color="darkblue", fill="lightblue")
  +ggtitle("Propextent Frequency")+labs(y="Frequency", x="Extent of property damage")
```

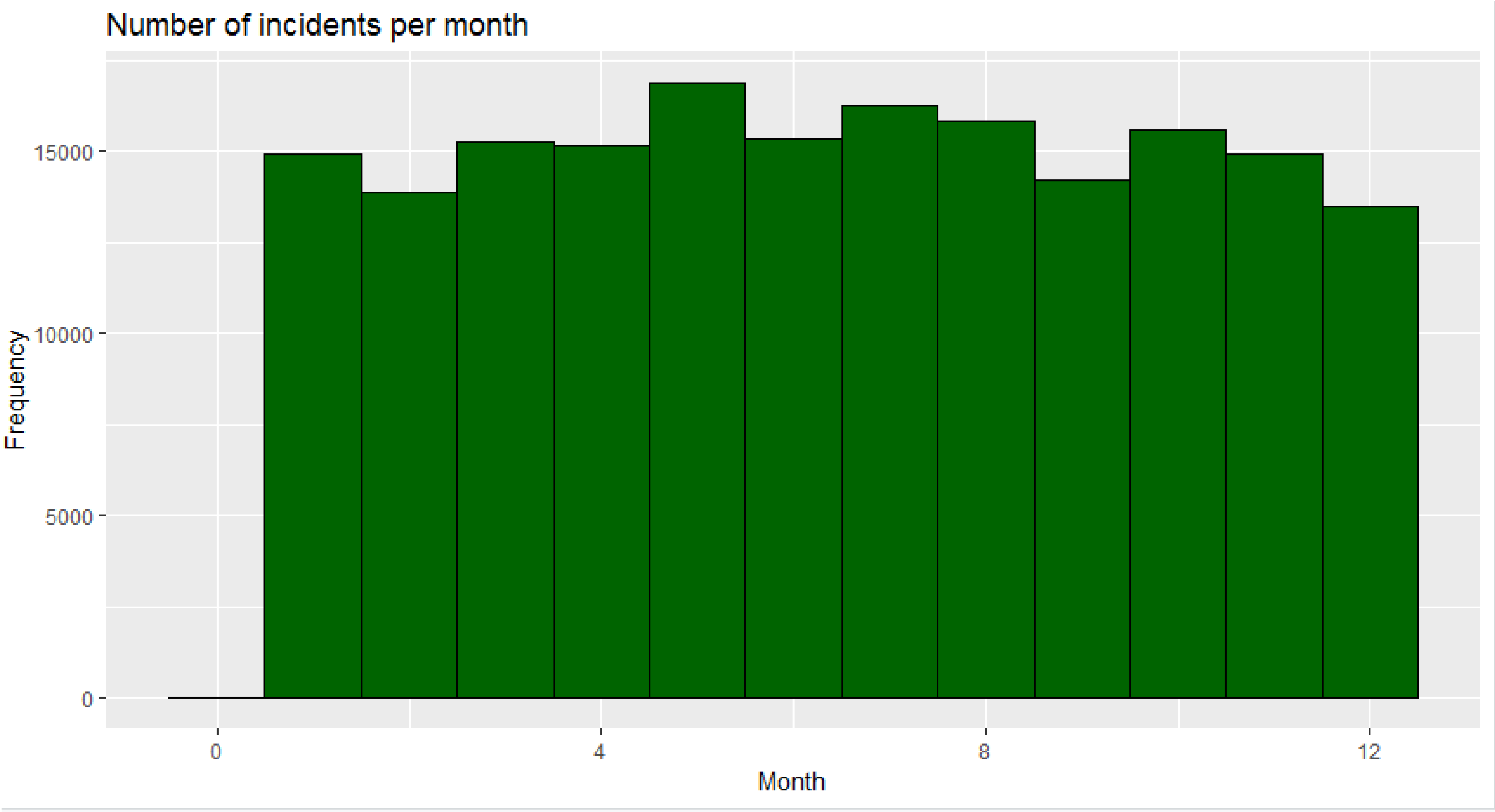


Summarising

- A crucial human skill is to be selective about the data that we choose to analyse and, where possible, to summarise this information as concisely as possible.
- There are two useful questions to think about when summarising data:

    1. What is a typical, or average, value of the data?
    2. How widely spread out are the data values?

A distribution of incidents over months



What is a distribution?

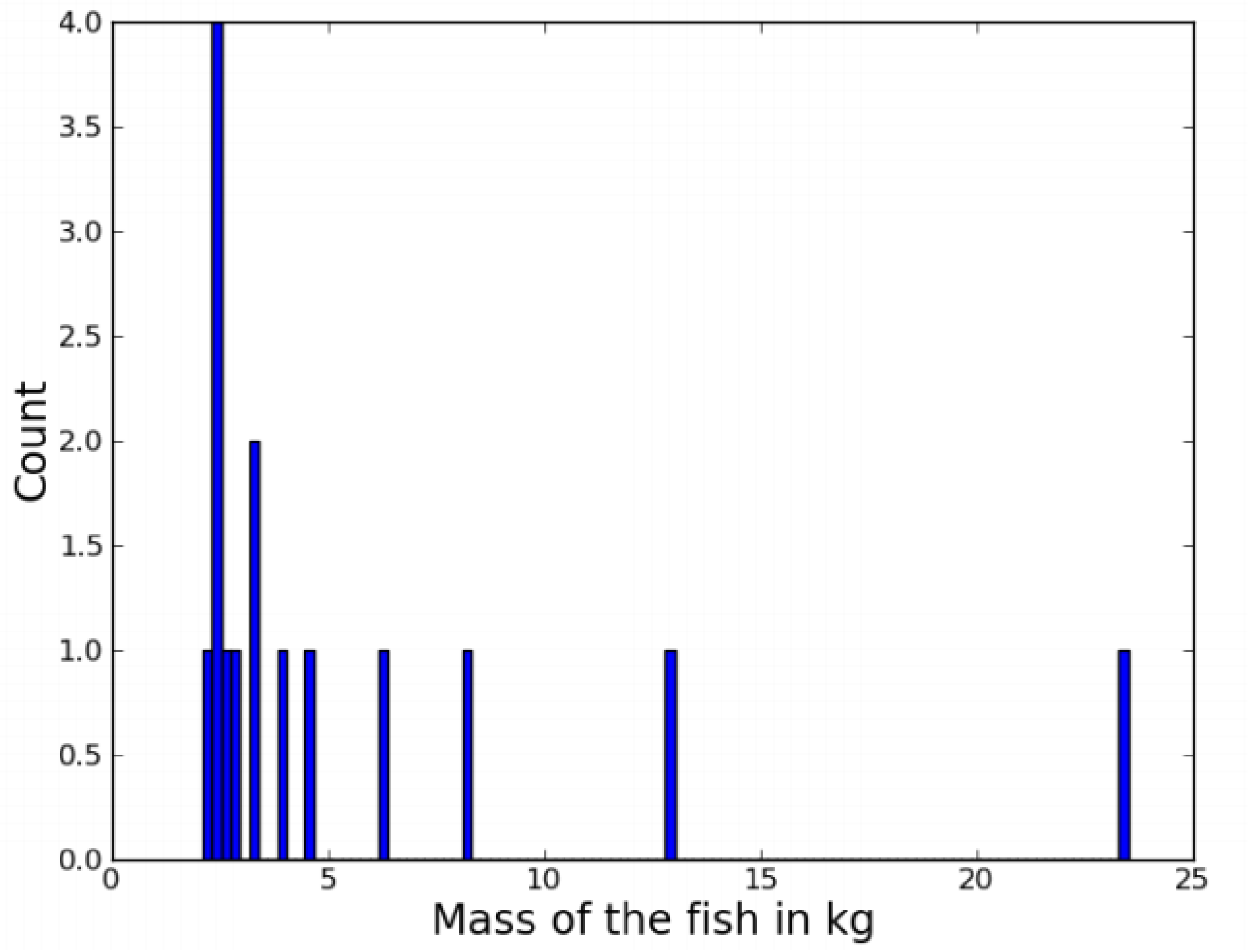
- For now, think of a distribution as just a set of measurements, numbers, or data points.
- We could talk about a distribution as being the result of a game of heads and tails being played multiple times; i.e. the results obtained when tossing a coin many times.
- Another distribution might be the final exam scores of every student in a particular school system.
- A third would be the predictions of global population in 2100 from 50 runs of a demographic simulation, each with a different random seed value.

An example

- Let's grab a sample distribution to work with:

  [2.1, 2.4, 2.4, 2.4, 2.4, 2.6, 2.9, 3.2,
  3.2, 3.9, 4.5, 6.3, 8.2, 12.8, 23.5]

- These 15 numbers could be anything.
- Let's say that they represent the mass in kilograms of some fish we've caught from a specific location.
- Note that our scales are accurate to the nearest 100g.



Questions to ask about the distribution

Some natural questions to ask about any distribution are:

- What's a typical score?
- What's the range of the scores?
- How spread-out are the scores?
- Are the scores distributed evenly/symmetrically?

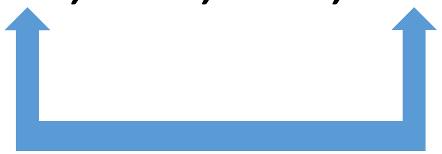
What is a typical score?

- How heavy is a typical fish? In other words, where do our fish mass scores tend to cluster on the number line?
- We need a number to represent the ‘middle’ score, or a *measure of central tendency*.
- Options include:
  - The *mean,* also known as the *average*, or the *arithmetic mean*.
  - The *median*, the middle score in the range of data.
  - The *mode*, the most frequently occurring score.

The mode

- The mode is merely the data value that is most common, or most frequent, in the data set.
- In our fish example:

2.1, 2.4, 2.4, 2.4, 2.4, 2.6, 2.9, 3.1, 3.2, 3.9, 4.5, 6.3, 8.2, 12.8, 23.5



mode = 2.4

Calculating mode in R

- R doesn’t have a built in function for calculating the mode.
- We therefore create a function to do it for us.

```
> Mode <- function(x) {
+     ux <- unique(x)
+     ux[which.max(tabulate(match(x, ux)))]
+ }
```

- If there is a calculation that you know you’re going to have to do regularly, creating a function to do it for you is a great time saver.
- You can call a function that you created the same way in which you call the built-in R functions:

```
> Mode(weaptype1)
[1] 6
```

The mean ($\bar{x}$)

- Calculating the mean = $\frac{sum\ of\ all\ data\ values}{number\ of\ data\ values}$

- In our example, we have the weight of 15 fish:
  2.1, 2.4, 2.4, 2.4, 2.4, 2.6, 2.9, 3.2, 3.2, 3.9, 4.5, 6.3, 8.2, 12.8, 23.5
- The sum of these fish weights is 82.8.
- There are 15 values, so:

$$\bar{x} = \frac{82.8}{15}$$

$$\bar{x} = 5.52$$

Calculating the mean in R

- The mean can be calculated easily in R with the `mean()` function.
- Remember, we first want to make sure that we don't have any missing values in the data we are exploring:

```
total_ransom_amt <- ransomamt[!is.na(ransomamt)]
```

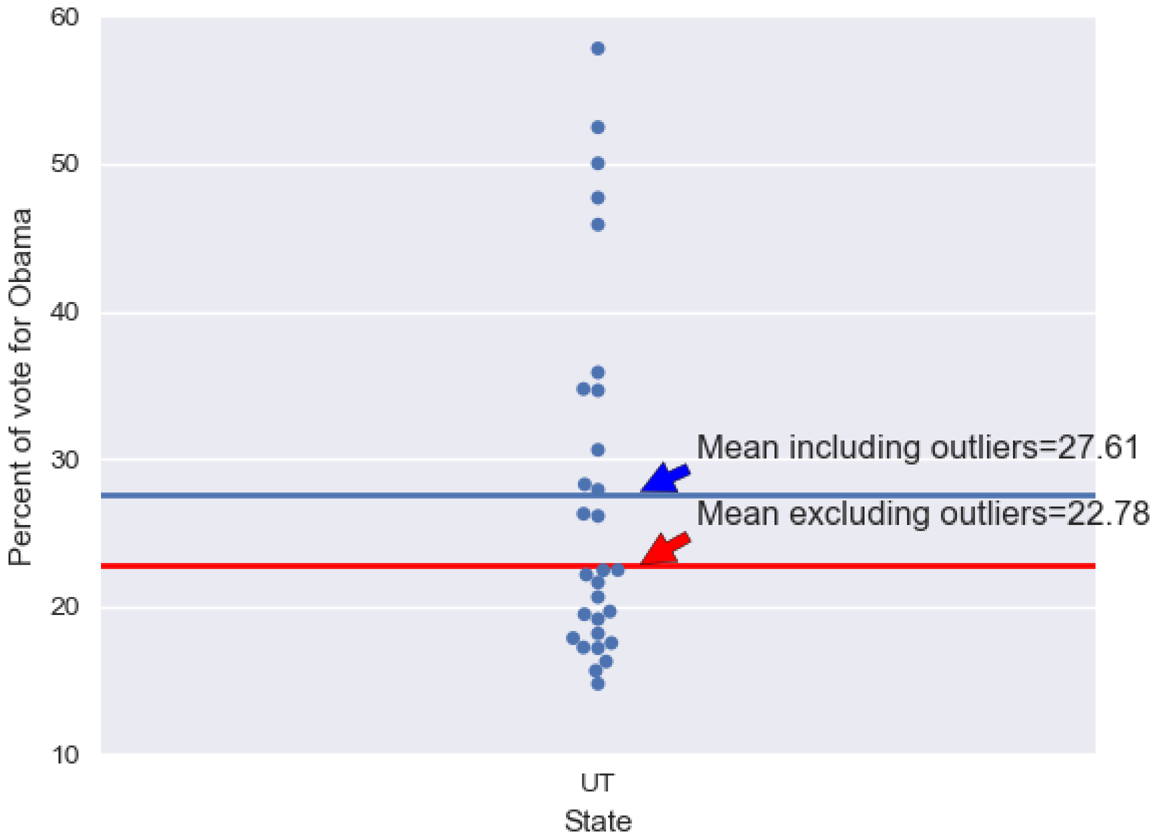
```
> mean(total_ransom_amt)
[1] 3172530
```

- When printing the value to the screen, you can use the `prettyNum()` function to make the result more readable:

```
> prettyNum(mean(total_ransom_amt), big.mark=",")
[1] "3,172,530"
```

Outliers

- Outliers are data points whose value is far greater or far less than the rest of the data points.
- If we take a look at the data set on the right, we see that there are 5 outliers.
- Outliers, or extreme values, heavily influence the value of the mean.



The median

- To calculate the median, order all data values in regards to their values from smallest to largest. The median value is then the middle value in the range of scores.
- In our fish example:

2.1, 2.4, 2.4, 2.4, 2.4, 2.6, 2.9, 3.1, 3.2, 3.9, 4.5, 6.3, 8.2, 12.8, 23.5

median = 3.1

- If the data set has an even number of data values, you take the middle two numbers and then take the mean of these two numbers.
- If we add another data point into our fish example:

2.1, 2.4, 2.4, 2.4, 2.4, 2.6, 2.9, 3.1, 3.2, 3.9, 4.5, 6.3, 8.2, 12.8, 23.5, 23.6

$$\text{median} = \frac{3.1+3.2}{2}$$

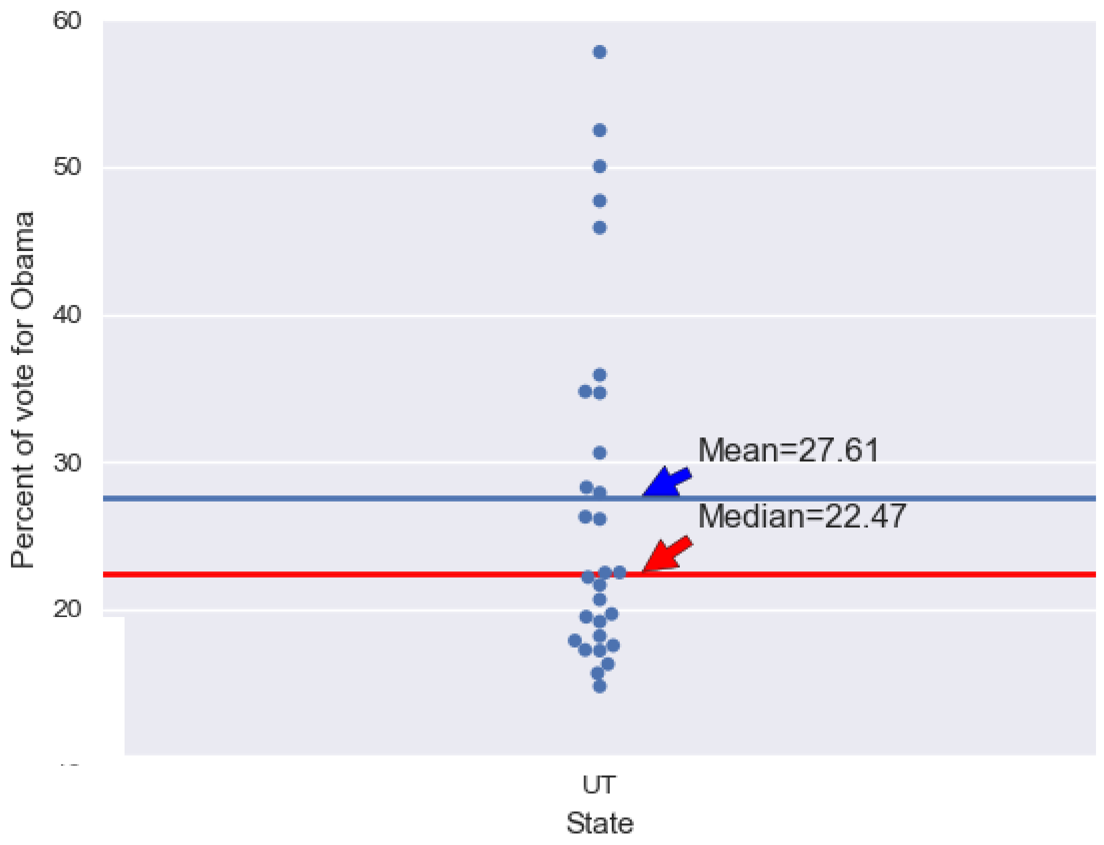
$$\text{median} = \frac{6.3}{2}$$

$$\text{median} = 3.15$$

Calculating the median in R

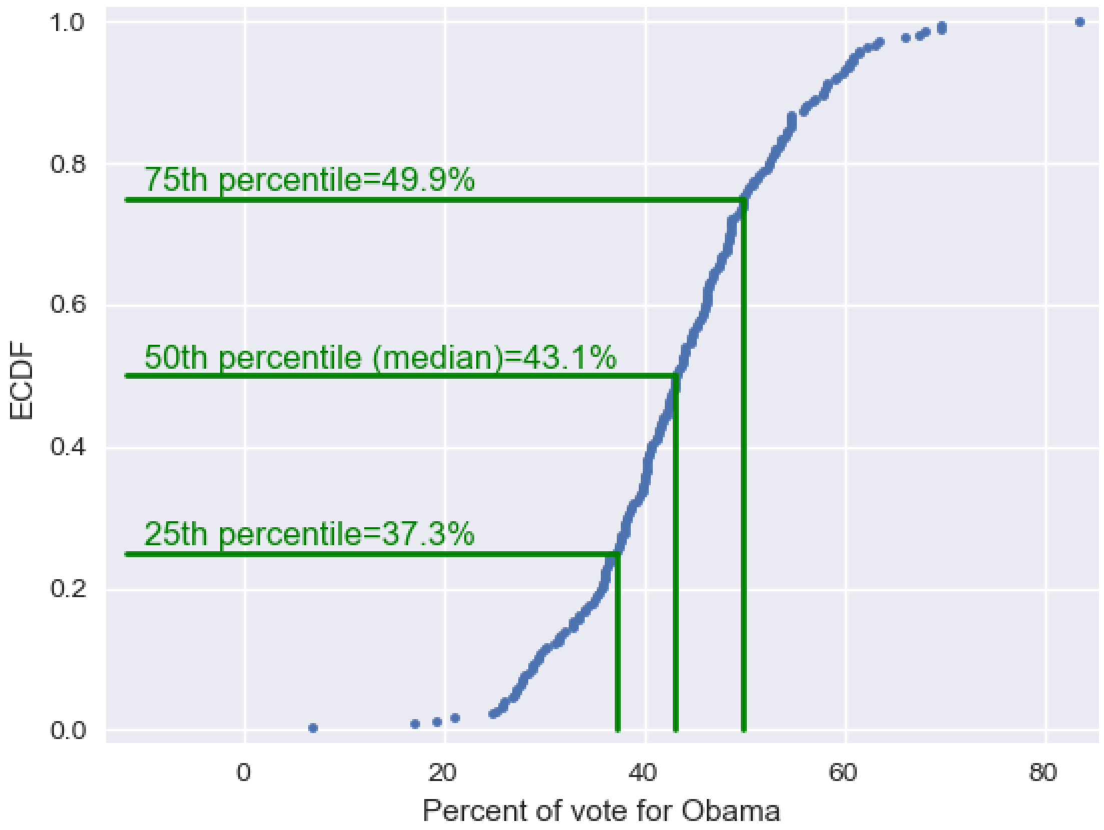
- We can see here that the median is not ‘tugged up’ by the 5 outliers because it is immune to extreme outliers.
- This is because the median is calculated in regards to the relative positon of the data points and not the data values themselves.
- Calculating the median in R is done in the same way as the mean:

```
> median(total_ransom_amt)
[1] 15000
> prettyNum(median(total_ransom_amt), big.mark=",")
[1] "15,000"
```



Percentiles

- The median is a special name for the 50th percentile.
- This means that 50% of the data are less than the median.
- Similarly, the 25th percentile is the data point that is equal to 25% of the data.
- And so on for 75th percentile, etc.



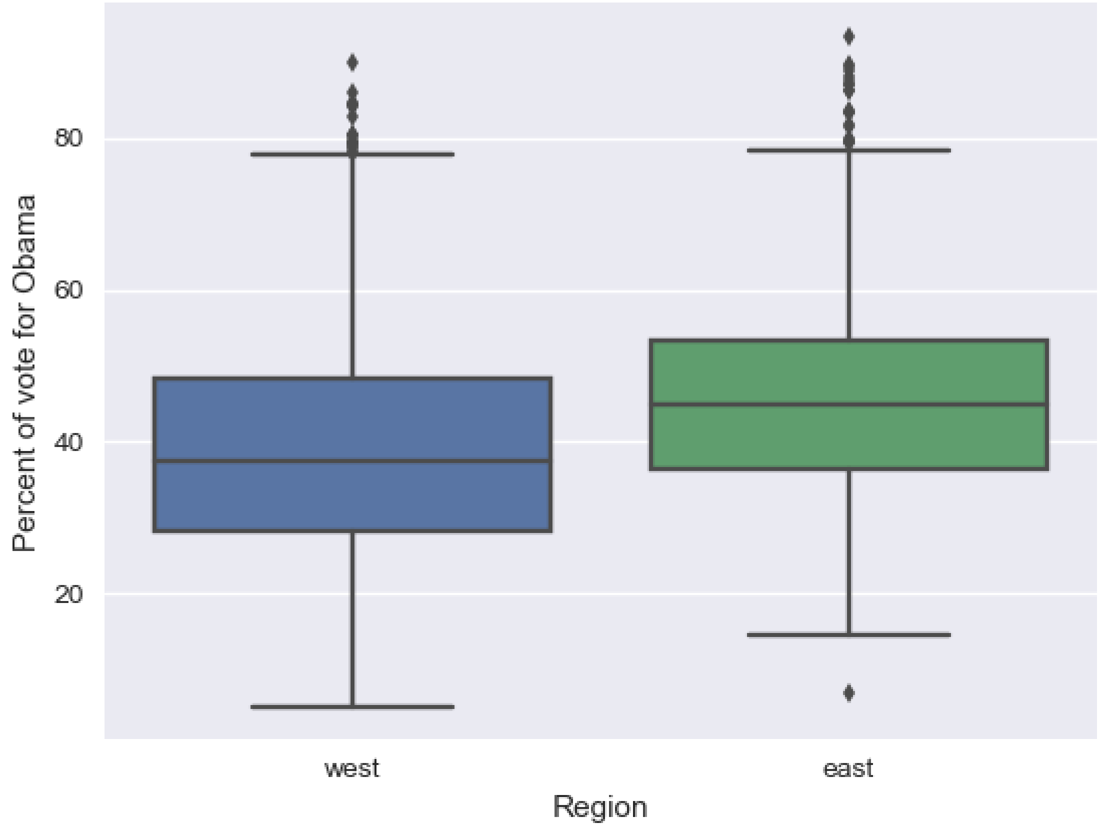
- Percentiles are useful summary statistics.
- They can easily be computed in R with the `summary()` function:

```
> summary(total_ransom_amt)
      Min.    1st Qu.     Median       Mean    3rd Qu.       Max.
       -99          0      15000    3172530     400000 1000000000
```

- We now have three summary statistics.
- However, the point of summary statistics is to keep things concise, but we are starting to get a lot of different numbers here.
- This is where quantitative EDA meets graphical EDA.

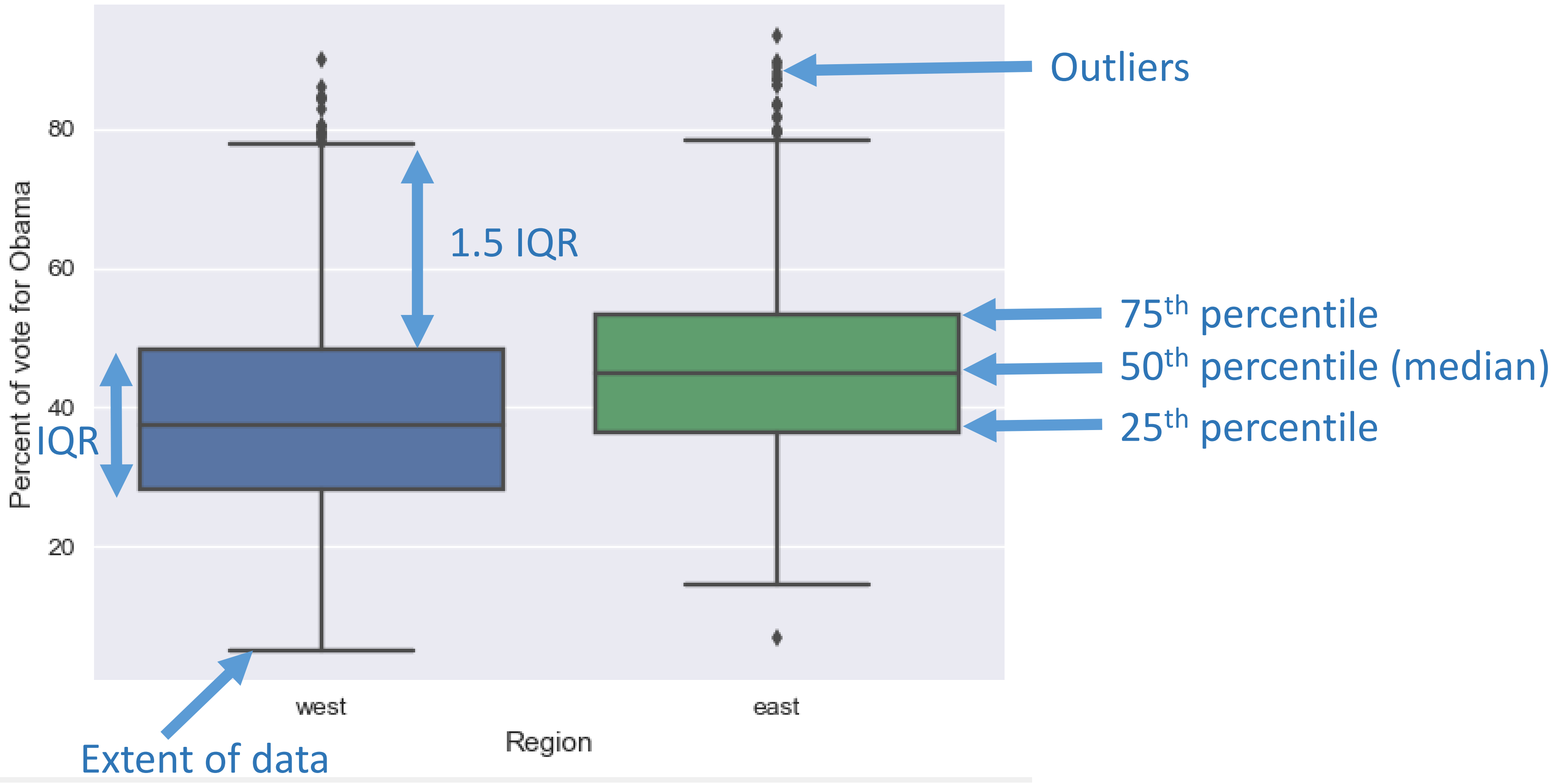
Box plots

- These were invented by John Tukey in order to display salient features of a dataset based on percentiles.



What does a box plot show?

- Interquartile range (IQR) = middle 50% of the data.
- The whiskers extend a distance of 1.5 times the IQR, or the extent of the data; whichever is less extreme.
- All points outside of the whiskers are plotted as individual points, which is the common criterion for defining an outlier.



Plotting a box plot in R

- Ensure that ggplot2 is in your working space:

```
> library(ggplot2)
```

- We need some data to plot. Here we'll is the 'imonth' variable again.
- It's easier to plot a dataframe than a vector, so we convert our variable of interest into a df:

```
> df <- data.frame(imonth)
```

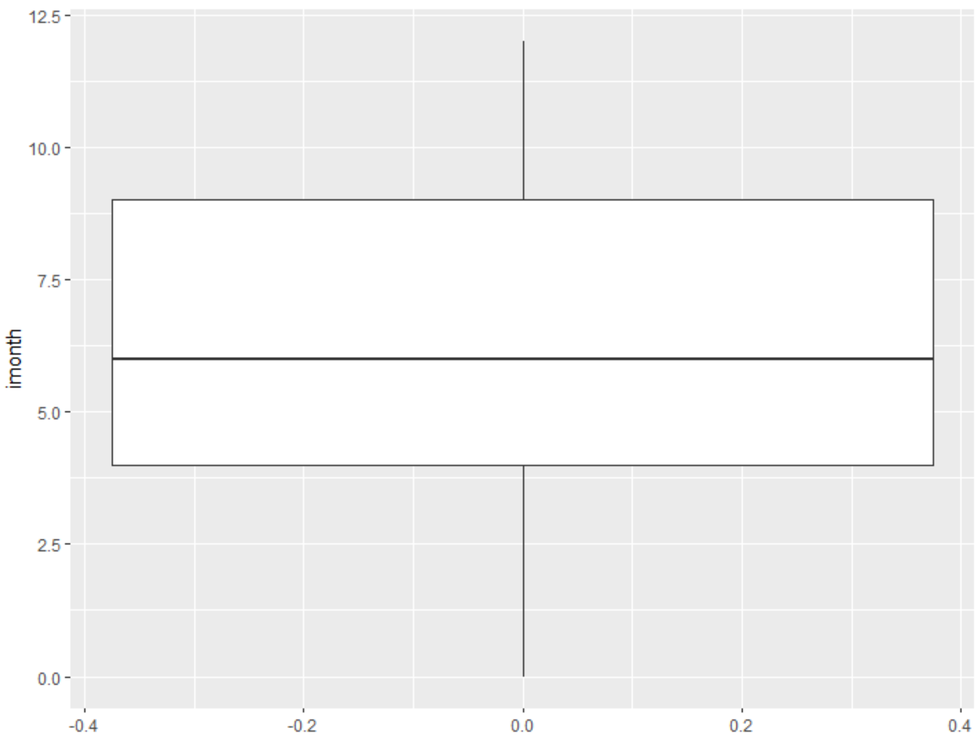
- We then feed in the df to the following command:

Where to get the data from

Which column of the df do you want too plot?

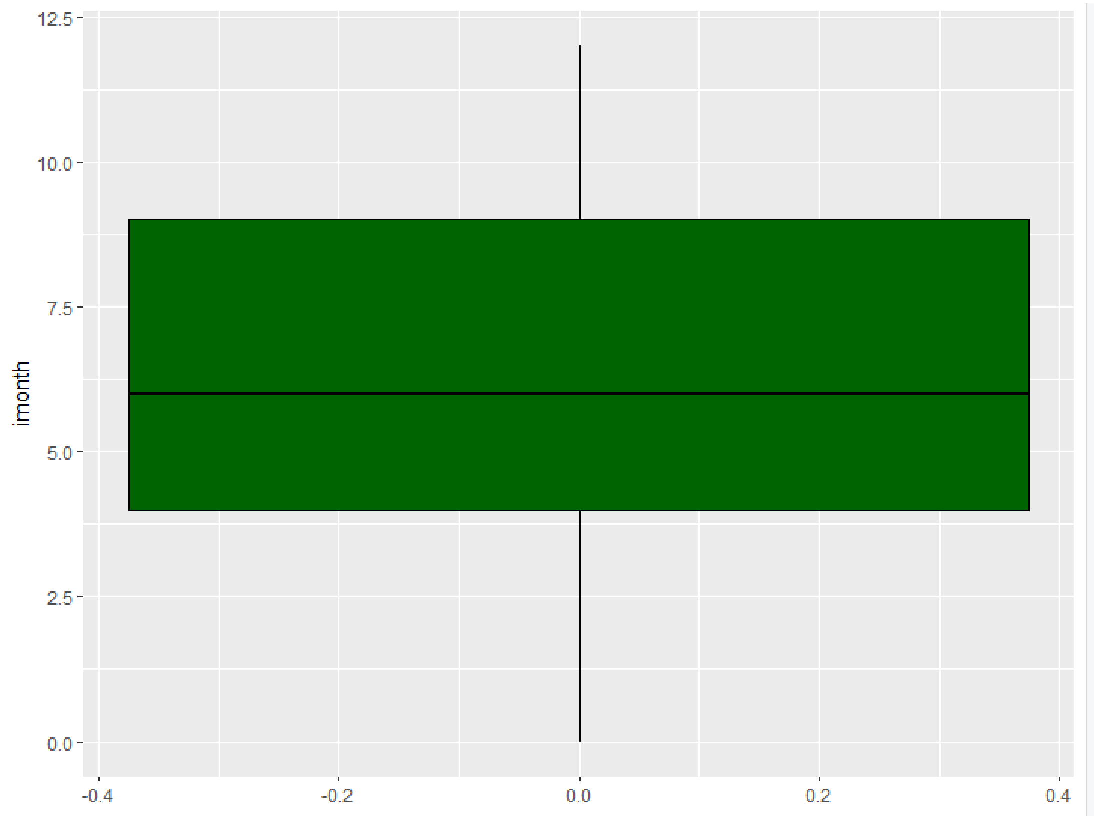
We want a box plot.

```
> ggplot(df, aes(y=imonth)) + geom_boxplot()
```



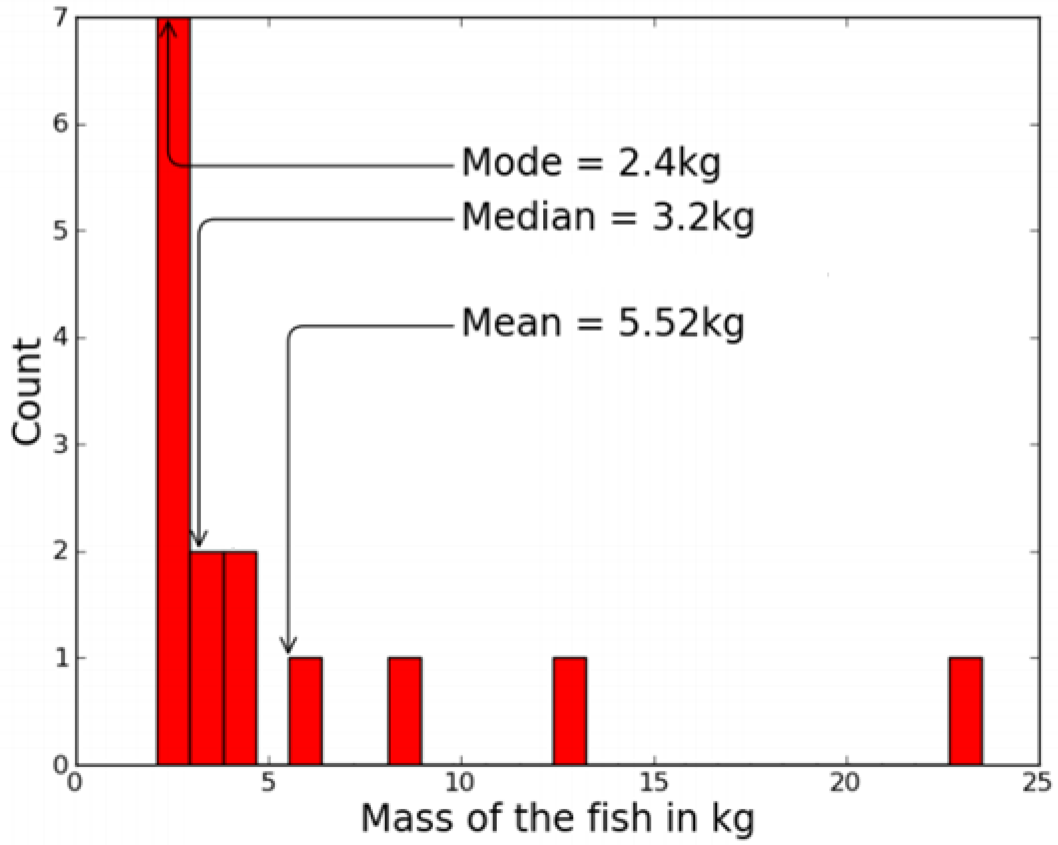
Making it pretty

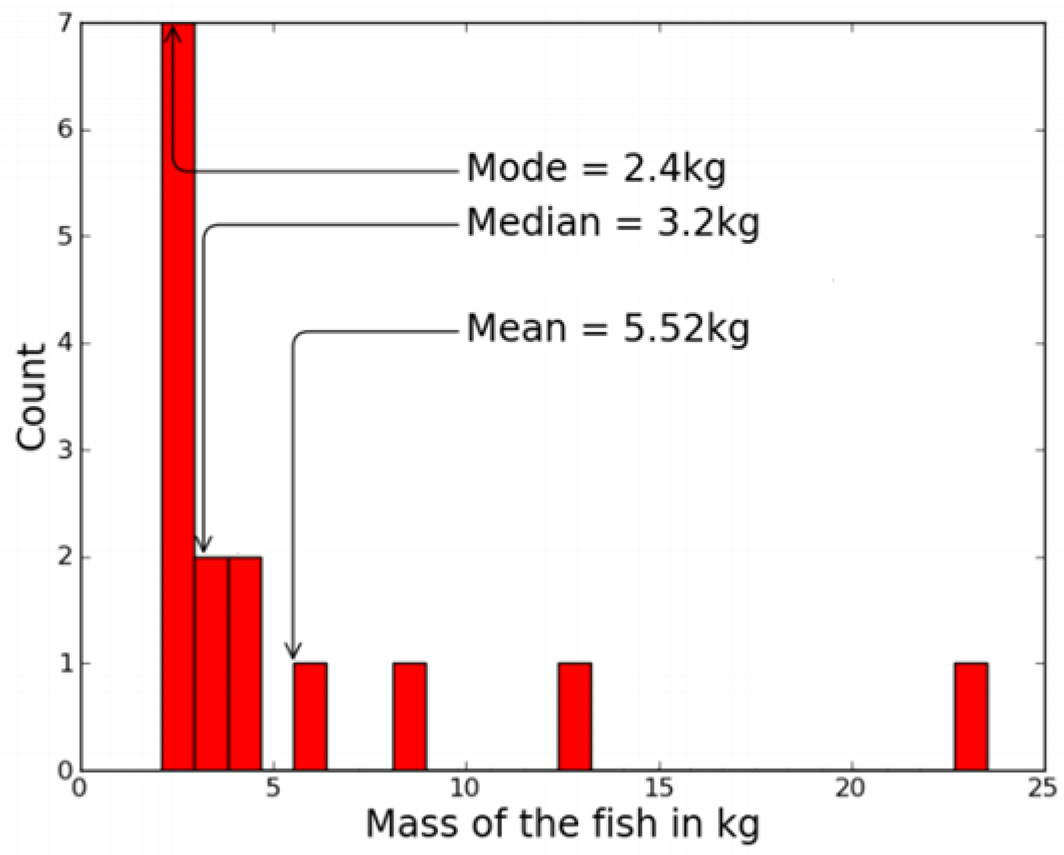
```
> ggplot(df, aes(y=imonth)) + geom_boxplot(fill="darkgreen", colour="black")
```

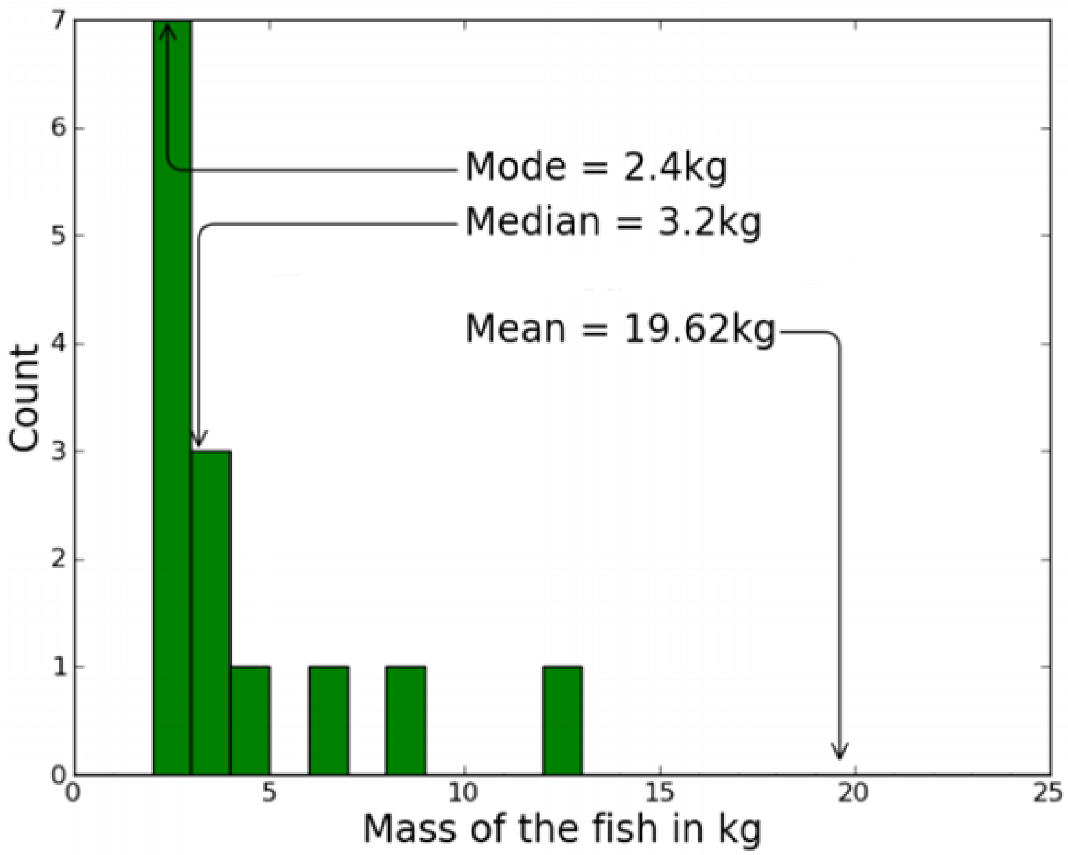


So… Which measure is best?

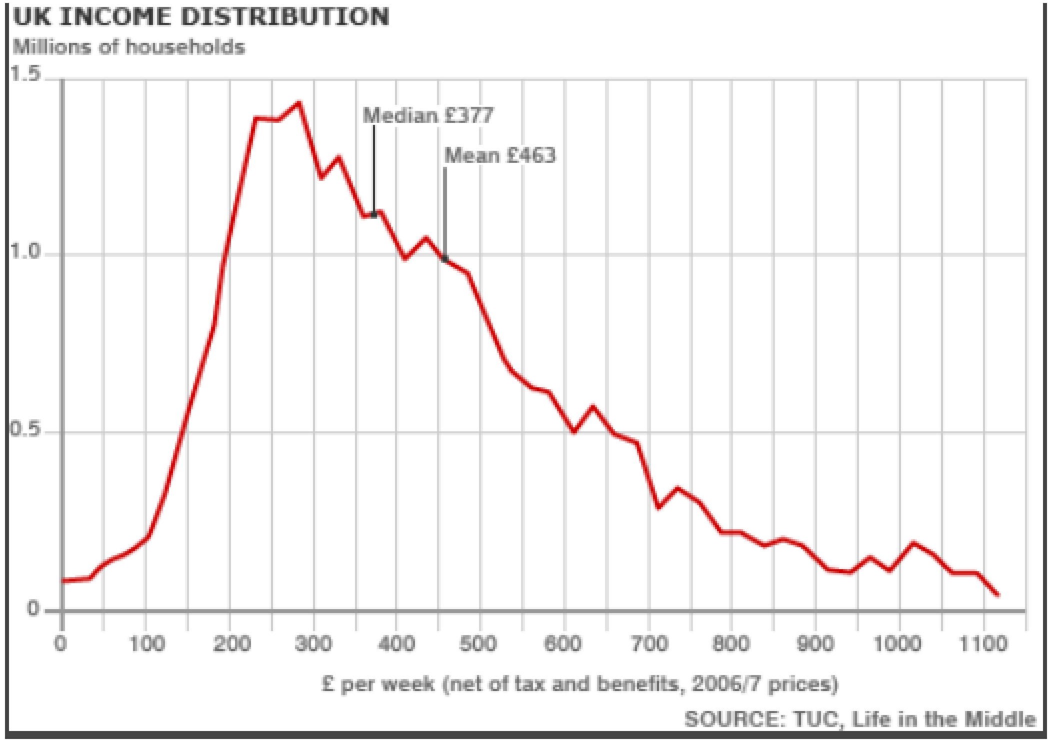
- There's no perfect way to report the central tendency of a distribution. Each of the measures tell us something.
- The median (central score) is a good all-purpose measure, and is not overly influenced by values at the extremes.
- The mean is very popular because of some of its mathematical properties, as we will see later on. But it's very sensitive to outliers (extreme values).
- As another example, look what happens if the last fish in our fish distribution is a 235kg monster.




Mode = 2.4kg
Median = 3.2kg
Mean = 5.52kg
Count
Mass of the fish in kg


Mode = 2.4kg
Median = 3.2kg
Mean = 19.62kg
Count
Mass of the fish in kg

- If the distribution is lop-sided, as for income (below) and our fish weights example, the median "feels better" as a measure of typicality.
- But the mean is the expected value of the distribution: useful if we want to calculate the expected tonnage of 1000 fish.

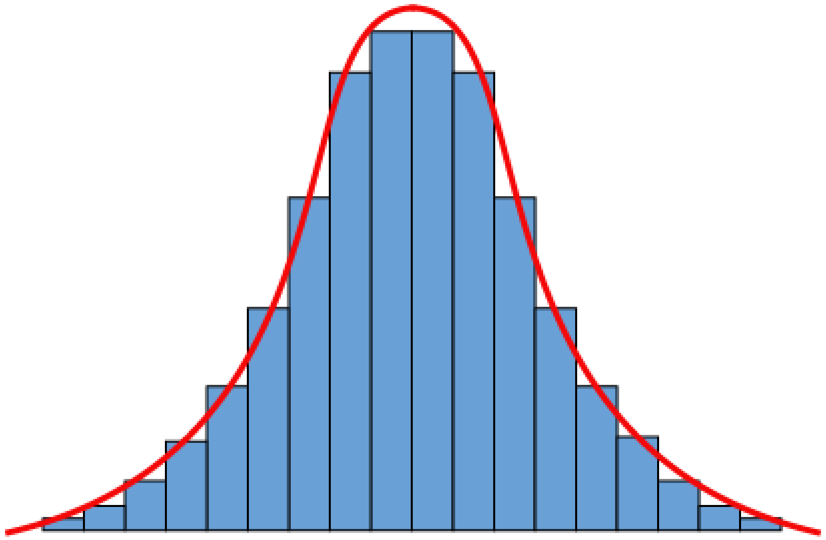


The mean is not "in the middle"

- There is a common misconception that confuses the mean with the median in everyday language:

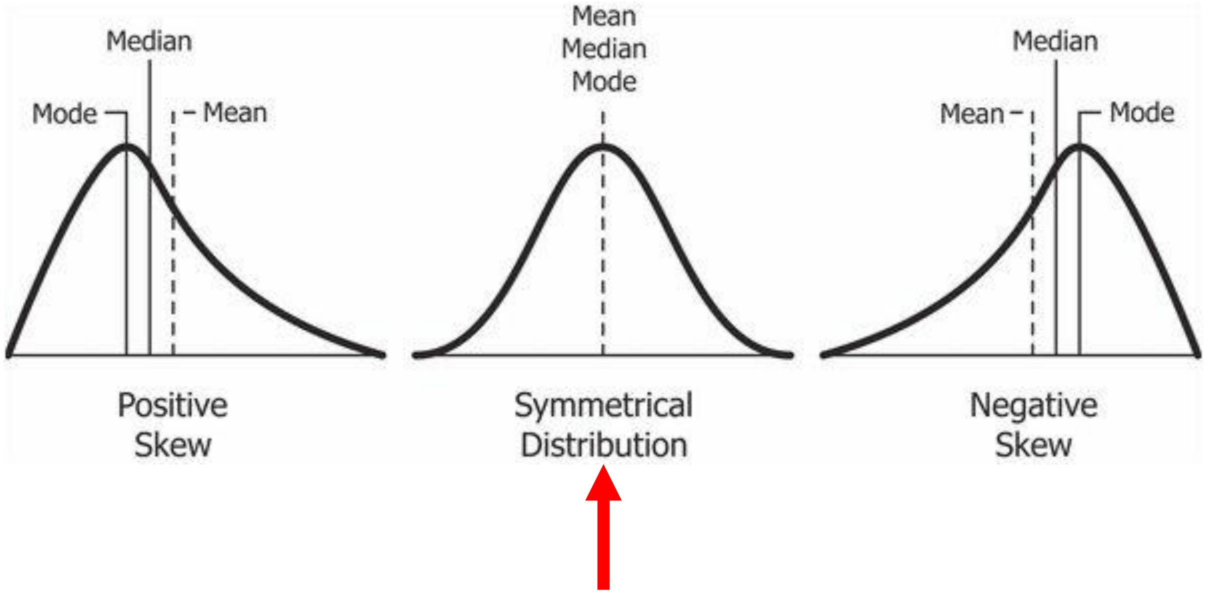
"You know how stupid the average person is? Well, 50% of people are dumber than that!"

- Not necessarily so!
- Mean and median are roughly equal in symmetrical distributions (right), but if intelligence is distributed in a lop-sided way (previous slide), it will be either more or less than 50% of people who are "dumber than average".



Skewness

- Skewness refers to the direction and degree of lop-sidedness in a distribution. Think of a left- or negative-skewed distribution as "right-walled", and vice versa.
- A common reason for a right-skewed distribution is when the values are bounded by zero at one end, e.g., for our fish weights, or for the duration of some event.



Normal distribution. Also known as the bell curve.

Measuring variation

- It's also natural to ask how wide a distribution is. In other words, how spread-out are the scores? How much variation is there?

The range

- We could note the minimum and maximum values: between 2.1kg and 23.5kg in the fish case.
- Using these values, we see that there is 21.4kg difference between the two values.
- This is called the *range*.
- It’s not a terrible summary of the spread in the fish data, but like the mean, the range is obviously very sensitive to extreme values.

Calculating the range in R

- Done in the same manner as calculating the mean, median, etc.

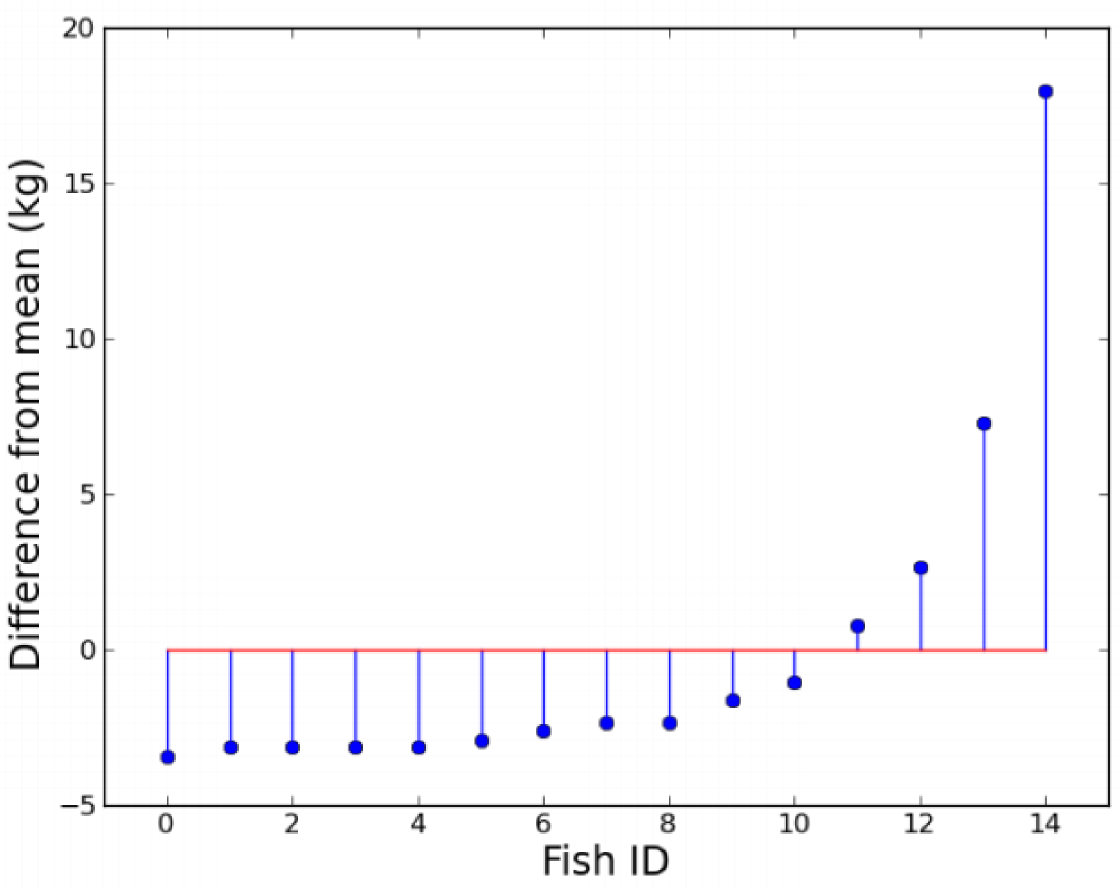
```
> data <- c(1, 1, 5, 6, 7, 3, 9, 8, 10, 4, 15, 3, 20)

> range(data)
[1]  1 20
```

- We could be a bit more sophisticated, and start looking at how far our scores differ from the mean.
- If we take each fish weight from our example, and subtract the mean of 5.52 from them, we get this new distribution of differences from the mean:

[-3.42, -3.12, -3.12, -3.12, -3.12, -2.92, -2.62, -2.32, -2.32, -1.62, -1.02, 0.78, 2.68, 7.28, 17.98]

- Could we use the average of these numbers to describe how much the distribution varies?



Average distance from the mean?

- The trouble is that the average distance from the mean is always zero, because the sum of the differences from the mean is zero. (Implicit in the definition of the mean.)
- OK, so ignore the signs and take the absolute value of the differences. After all, distance is not usually treated as a signed quantity...

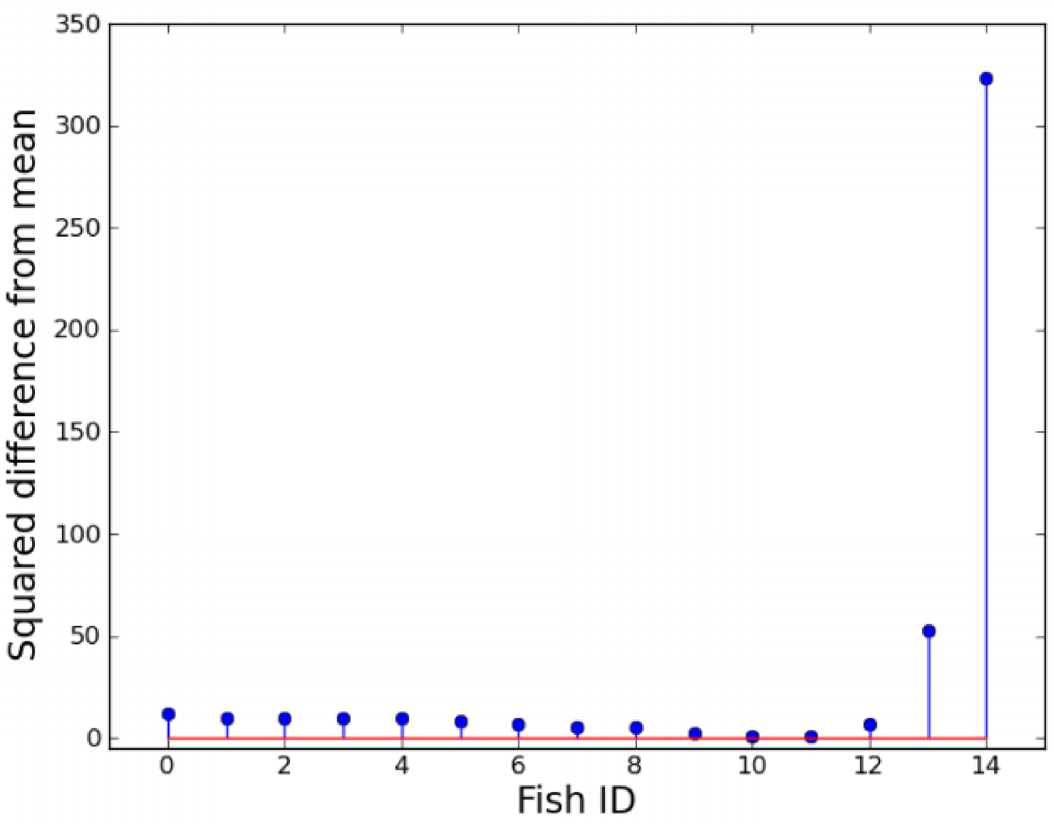
[−3.42, −3.12, −3.12, −3.12, −3.12, −2.92, −2.62, −2.32, −2.32, −1.62, −1.02, 0.78, 2.68, 7.28, 17.98]

- This can work. In the fish case, we get a value of 3.83.

- So, on average, a particular fish we catch is 3.83kg away from the overall average of the distribution.
- This is known as the *average absolute deviation.*
- It is a sensible-enough measure of variation, but it's not one that is popularly used.

The variance

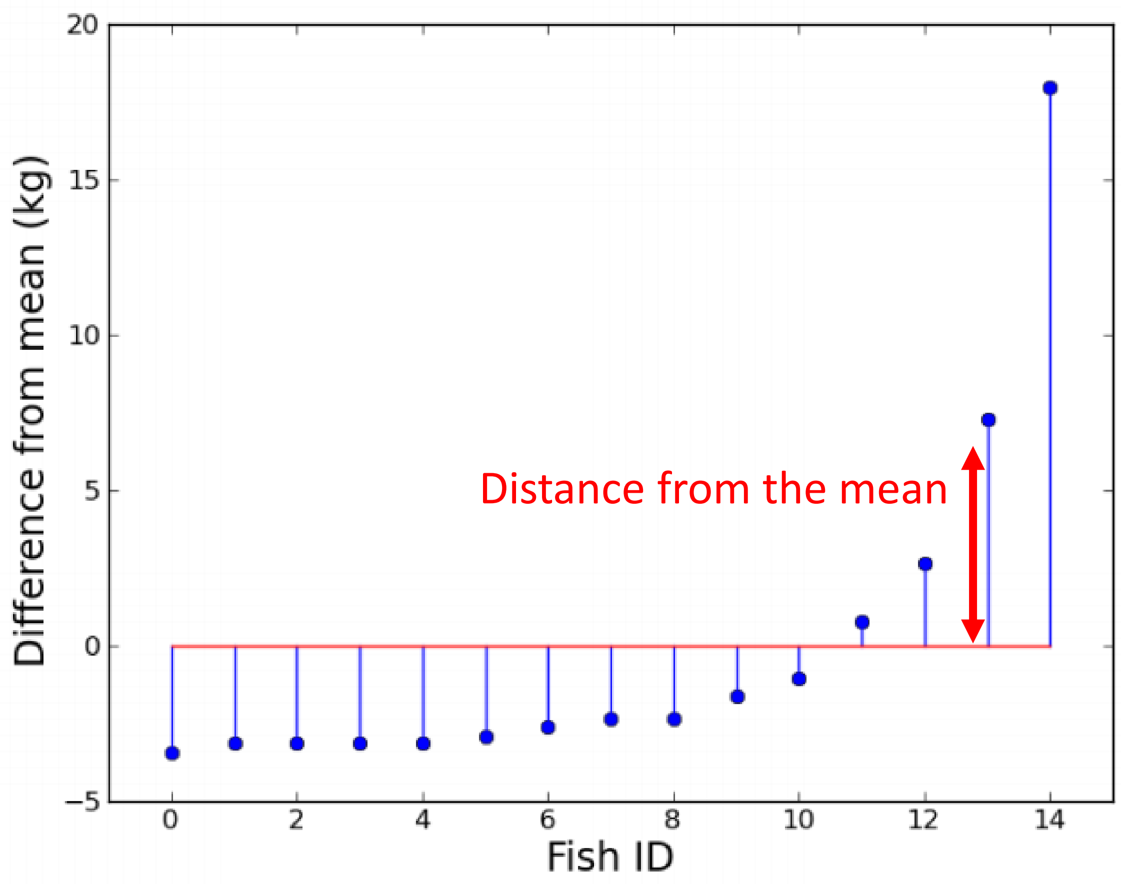
- Another way of getting around the average-deviation-adds-to-zero problem is to take the squares of the deviations.
- This move turns out to have a lot of useful properties.
- If we take the average of these squared differences from the mean, this is called the *variance*.
- In the fish case, the variance is 30.97.



Calculating the variance

- In other words, for each data point, we take the distance from the mean and square it.
- And then take the average of these squared values

$$variance = \frac{1}{n}\sum_{i=1}^{n}(x_i - \bar{x})^2$$



Calculating the variance in R

- Done the same was as the mean and median:

```
> var(total_ransom_amt)
[1] 9.12739e+14
```

- This is a good measure of variation: if someone else's fish catch had a variance of 40, we would know that their fish weights were more spread-out than ours.
- However, the variance is in the odd unit of kilograms squared...

The standard deviation

- We solve this problem by taking the square root of the variance to get a new measure called the standard deviation.
- We are now back in the familiar units of kilograms.
- The standard deviation in the fish case is 5.56kg, a bit higher than the mean absolute deviation of 3.83kg.
- The standard deviation is by far the most popular measure of variation in a distribution.

$$\sigma = \sqrt{\frac{1}{n}\sum_{i=1}^{n}(x_i - \bar{x})^2}$$

Calculating std dev in R.

- Done the same was as the mean, median, and variance:

```
> sd(total_ransom_amt)
[1] 30211571
> prettyNum(sd(total_ransom_amt), big.mark=",")
[1] "30,211,571"
```

Any questions?