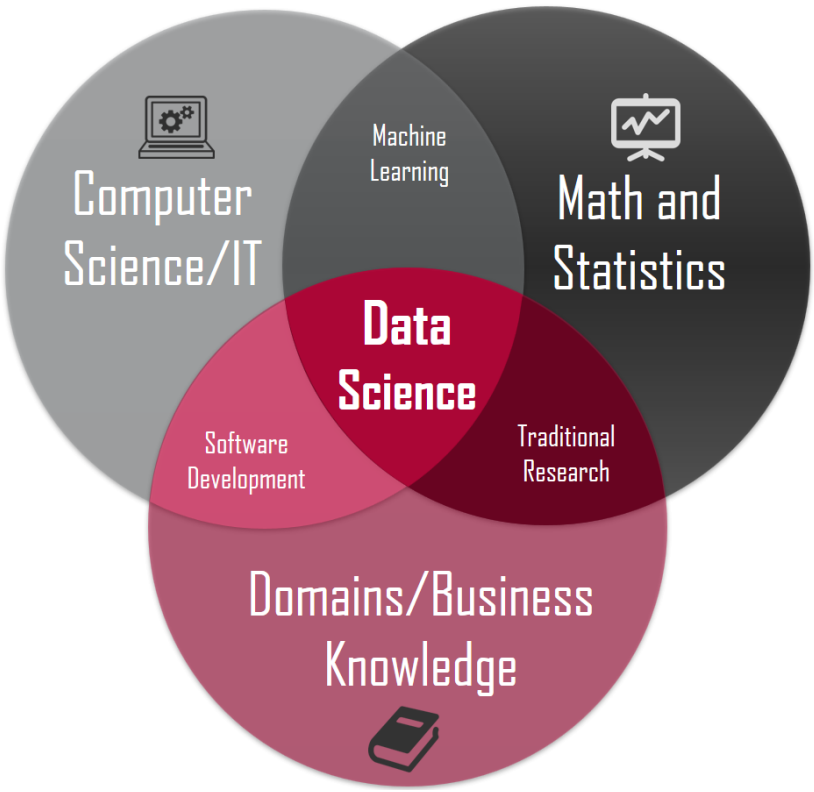
Michael Barber Following
Data scientist working in "data science for social good" industries
Jan 14 · 27 min read

Data science concepts you need to know! Part 1

This is the first post in a 5-part series that will outline some of the core concepts in data science:



Notably we will be covering the statistics bucket from the diagram above and trying to make it as intuitive and maths-lite as possible.

My background is in big pharma, academia (a PhD from Oxford's Physical and Theoretical Chemistry department) and "data science for social good" industries across southern and eastern Africa.

This series of posts aims to introduce and quickly develop some core concepts in data science and data analysis, with a specific focus on areas that I feel are overlooked or treated briefly in other materials. Therefore, these posts will suit data scientists looking to brush up on

statistical theory, data analysts looking for thorough training materials, and data managers who want to ask better questions from their data and their staff.

Some of the topics we will cover in this series include:

- *Non-parametric methods for power and sample size calculations*
- *The design and analysis of A/B tests (aka randomized trials)*
- *How to effectively explore data trends*
- *How to implement, assess and improve linear models*

These posts were adapted from a series of training materials I developed for the international NGO, One Acre Fund, which provides micro-loans to smallholder farmers in east Africa.

. . .

Aims of this post (#1):

- Understand what determines the shape of distributions.
- Understand how non-normality impacts hypothesis testing and power calculations
- Understand how to use Monte-Carlo methods for non-parametric sample size/power calculations

These posts are made with R-markdown, which allows the presentation of R code, text writing, and results, all in one place. R-markdown is especially useful for report writing as it makes analysis transparent and reproducible.

. . .

Motivation:

Statistics is essentially the study of distributions. A distribution is a way to tie together the frequency of a value, with the actual value that is observed. You can imagine for example, if we were measuring the height of our employees, then we are likely to find most people are 1.5 to 1.8m tall, however there will be a few people who are taller or

shorter than this. When we talk about data like this, we are really talking about distributions:

```
set.seed(111)
hist(rnorm(1000, mean=1.75, sd=0.5), xlab="Employee height
(m)", main="Employee height",
breaks=c(0,0.5,1,1.5,2,2.5,3,3.5,4),col="black")
```

The above histogram is our statement in graphical terms; we have the value of the observation along the X-axis (the values are grouped into "bins") and the count, or frequency (i.e. the number of times we observe that value) along the Y. Sometimes we might also plot the *probability of an observation* along the Y axis.

Reading the above histogram, we can say that the most common height is 1.5 to 2m, with decreasing frequency at more extreme values (for example, there are fewer people at 0.5–1m and 2.5–3m than at 1.5–2m). More specifically, by reading the X~Y axes, we can say there are ~400 people who are 1.5 to 2m tall, and ~200 who are 2 to 2.5m tall.

Every analysis must start with an understanding of the underlying data and its distribution. The *shape* of the distribution is going to determine:

- The analysis methods we use (should we use a T-test, or a Welch-test?)

- The sample size (how many individuals do we need to be able to detect an effect?)
- The size of our errors (how "spread-out" is the data?)
- Statistical significance (i.e. whether an intervention/product is successful or not, and how big its effect was)

> *Ultimately then, the distribution will determine whether we can assign significance and causality to our tested intervention.*

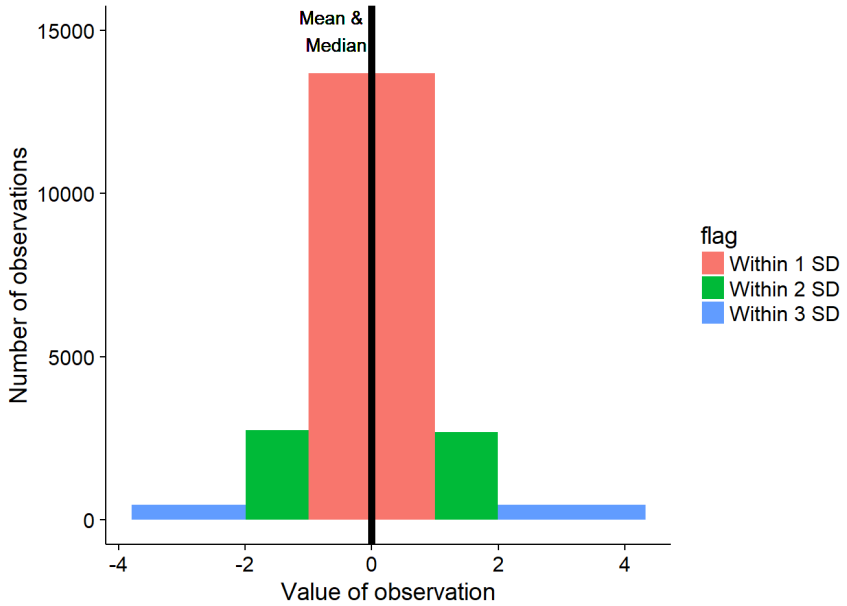
. . .

The normal distribution

The above histogram is from a normal distribution, one of the most common distributions you will see. Let's examine this distribution in a little more depth. In a normal (and only a normal distribution) we expect our mean **and** median to be at the center (tallest point) of the distribution, we likewise expect the distribution to be roughly symmetrical about the mean, as below:

```
#lets generate some example normal distributions
set.seed(111)# by setting a seed we can make sure that R
draws random numbers reproducibly
#simulate a random normal distribution
norm1 <- rnorm(20000, mean=0,sd=1)

normdf <- data.frame("val"=norm1, "flag"=0)

#which is a logical, it will evaulate a condition: here
"which val is greater than 1sd of norm OR less than -1sd of
norm"
# note that | means OR and & means and
sd2 <- which( normdf$val > 1* sd(norm1) | normdf$val <
-1*sd(norm1) )
sd3 <- which( normdf$val > 2*sd(norm1) | normdf$val <
-2*sd(norm1) )
sd1 <- which( normdf$val < 1*sd(norm1) & normdf$val >
-1*sd(norm1) )

#now lets use the index made above to reset the values in
flag to the following
normdf$flag[sd1] <- "Within 1 SD"
normdf$flag[sd2] <- "Within 2 SD"
normdf$flag[sd3] <- "Within 3 SD"
normdf$flag <- as.factor(normdf$flag)
```

```
#lets plot these...
#as a histogram
ggplot(normdf, aes(x=val, fill=flag)) +
geom_histogram(binwidth=.5, bins=100, alpha=1, breaks=c(-
Inf,min(subset(normdf, flag=="Within 3
SD")$val),min(subset(normdf, flag=="Within 2
SD")$val),min(subset(normdf, flag=="Within 1 SD")$val),
max(subset(normdf, flag=="Within 1 SD")$val),
max(subset(normdf, flag=="Within 2
SD")$val),max(subset(normdf, flag=="Within 3 SD")$val),
Inf)) + geom_vline(aes(xintercept = mean(val), colour="Mean
& Median"),color='black',size=2) +
geom_text(aes(x=mean(norm1)-0.6,y=15000,label="Mean & \n
Median")) + xlab("Value of observation") + ylab("Number of
observations")
```



A normal distribution is defined by 2 metrics, a mean and a standard deviation. For a normal distribution we expect 68% of values to lie within 1 standard deviation, 95% to lie within 2 standard deviations and 99.7% to lie within 3 standard deviations (as above).

Note that the standard deviation of a distribution is proportional to the width of the distribution.

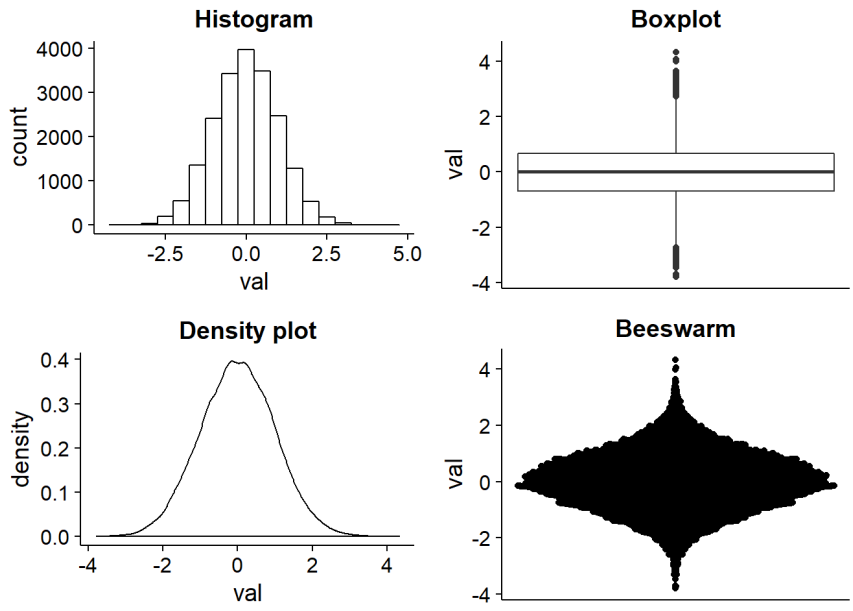
> *When we present distributions, our aim is to make the graph as simple as possible whilst still being true to the data.*

. . .

Presenting data

We can present distributions in other formats, not just histograms:

```
#
#histo
p1 <- ggplot(normdf, aes(x=val)) +
geom_histogram(binwidth=.5, colour="black", fill="white") +
ggtitle("Histogram")

#density
p2 <-ggplot(normdf, aes(x=val)) + geom_density() +
ggtitle("Density plot")

#boxplot
#the theme argument removes the x-axis as we have no
information to show here
p3 <- ggplot(normdf, aes(x=1, y=val)) + geom_boxplot() +
ggtitle("Boxplot") + xlab("")  +
theme(axis.title.x=element_blank(),
        axis.text.x=element_blank(),
        axis.ticks.x=element_blank())
#p3 <- ggdraw(add_sub(p3, "Boxplot"))

#finally a beeswarm plot
library(ggbeeswarm)

p4 <- ggplot(normdf) +  geom_beeswarm(aes(x=1,y=val)) +
ggtitle("Beeswarm") + theme(axis.title.x=element_blank(),
        axis.text.x=element_blank(),
        axis.ticks.x=element_blank())

#this is a funtion defined in the first block which allows
us to put more than one graph on the same page
multiplot(p1,p2,p3,p4, cols=2)
```



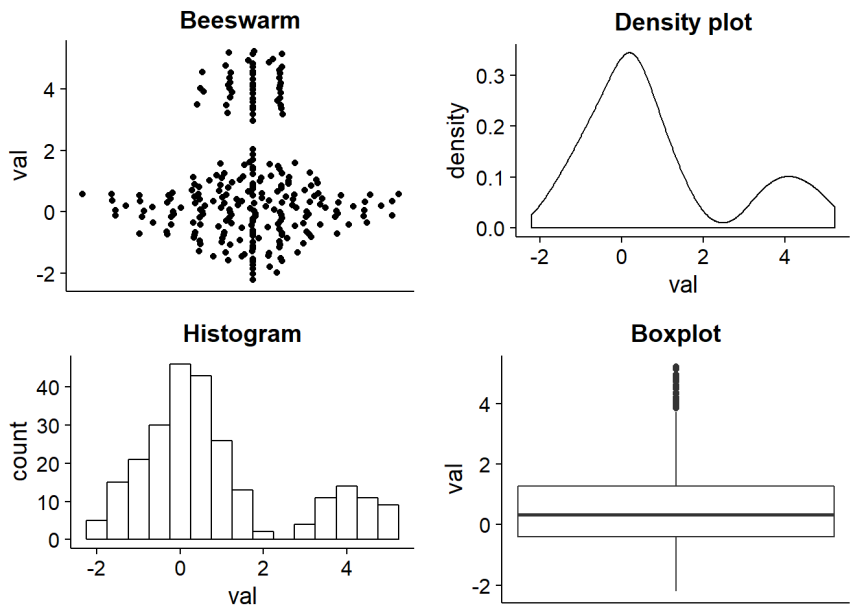
We are showing the exact same data above just in different ways:

- The **histogram** (top left) we are familiar with. Note that changing the number of bins of a histogram (i.e. how many bars do we want) can drastically change and alter our interpretation of the data.
- The **boxplot** (top right) has our values on the Y-axis, the thick center line represents the median, the upper and lower box boundaries represent the first and third quartile (respectively), the "whiskers" extending from the box represent the maximum and minimum values and finally the dots are (presumed) outliers. Box plots are informative but can hide clustered data.
- The **density plot** is similar to our histogram. However the density plot is smoothed and the y-axis now represents the fraction of total counts (from 0 to 1) rather than the actual counts. The sum of the area under the density plot is 1.
- The **beeswarm** plot shows every data point in the data frame. The width of the plot is proportional to the number of observations, we can tell the normality of the distribution as the plot is thicker around 0 (our mean) and then gets symmetrically smaller north and south of the mean. Whilst box plots can hide clustering, and density and histograms are dependent on a good bin width, a bee swarm can clearly show clustered data but is a little ugly to look at.

To illustrate this, let us show some clustered data (also known as a binomial distribution, note this is different to a normal distribution):

```
#
#lets make two seperate distributions and join them:
set.seed(122)
nor1 <- rnorm(200, mean=0,sd=0.8)
nor2 <- rnorm(50, mean=4,sd=0.7)

clusterdf <- data.frame("val"=c(nor1,nor2), "flag"=0)

# a beeswarm plot

p4 <- ggplot(clusterdf) +  geom_beeswarm(aes(x=1,y=val)) +
ggtitle("Beeswarm") + theme(axis.title.x=element_blank(),
        axis.text.x=element_blank(),
        axis.ticks.x=element_blank())


#histo
p1 <- ggplot(clusterdf, aes(x=val)) +
geom_histogram(binwidth=.5, colour="black", fill="white") +
ggtitle("Histogram")
```

```
#density
p2 <-ggplot(clusterdf, aes(x=val)) + geom_density() +
ggtitle("Density plot")

#boxplot
#the theme argument removes the x-axis as we have no
information to show here
p3 <- ggplot(clusterdf, aes(x=1, y=val)) + geom_boxplot() +
ggtitle("Boxplot") + xlab("")  +
theme(axis.title.x=element_blank(),
        axis.text.x=element_blank(),
        axis.ticks.x=element_blank())
#p3 <- ggdraw(add_sub(p3, "Boxplot"))



#this is a funtion defined in the first block which allows
us to put more than one graph on the same page
multiplot(p4,p1,p2,p3, cols=2)
```

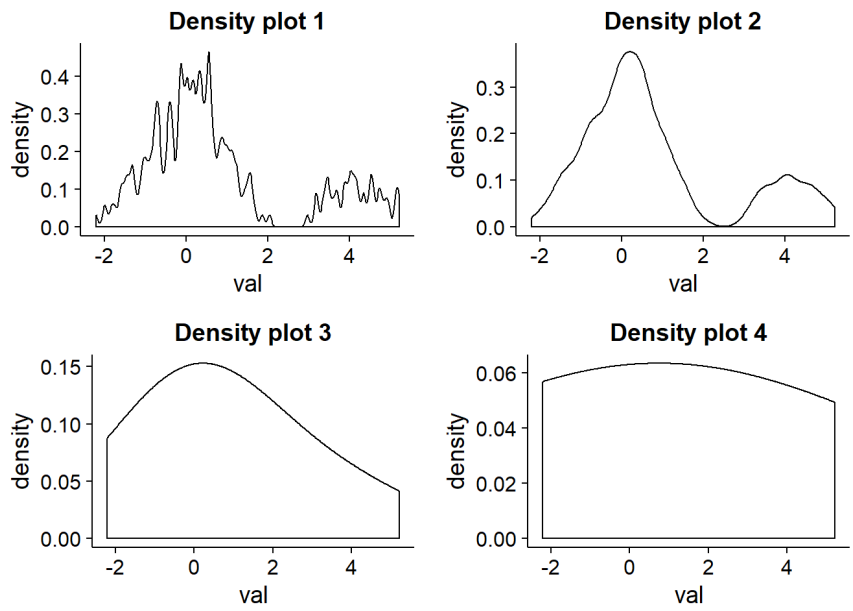


From the bee swarm we can see that there is a large cluster of data around ~0, but also a smaller cluster around ~4. The density plot and histogram pick this up but if we just examined the box plot we would not be able to see this clearly.

Finally, altering the bins of the density plot or histogram can also change our interpretation of the data:

```
#density
p1 <-ggplot(clusterdf, aes(x=val)) + geom_density(bw=0.05) +
ggtitle("Density plot 1")
p2 <-ggplot(clusterdf, aes(x=val)) + geom_density(bw=0.2) +
```

```
ggtitle("Density plot 2")
p3 <-ggplot(clusterdf, aes(x=val)) + geom_density(bw=2) +
ggtitle("Density plot 3")
p4 <-ggplot(clusterdf, aes(x=val)) + geom_density(bw=6) +
ggtitle("Density plot 4")


multiplot(p1,p3,p2, p4, cols=2)
```



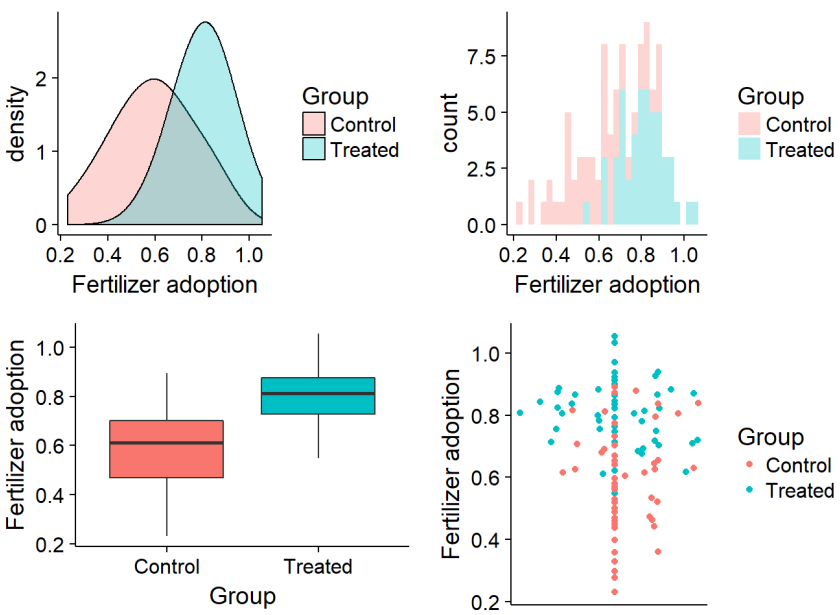
Our interpretation of the data is changed dramatically depending on what plot we examine, even though the data is the same! How could we learn anything about our data from plot 4? Plot 1 is likely too much detail, whilst plot 3 and 4 are too little detail (notice that plot 4 totally hides the second cluster of data).

Plot 2 **is both simple and true to the data**. It is therefore the best plot here. Note that the only way to figure out the best plot is to examine the underlying data. I would therefore recommend using either a bee swarm or a density plot with a small bin width (or high number of bins) to first examine the data in as much detail as possible, before producing a simplified version of the data.

Efficiently comparing distributions

Usually we will want to compare distributions to see if there is an obvious difference, for example when comparing treatment and control groups. My preferred way to do this is using either density plots or box plots. Let's look at some examples below where we are measuring the effect of a treatment on fertilizer adoption amongst farmers in Rwanda:

```
#lets make some data
set.seed(122)
nor1 <- data.frame("val"=rnorm(50, mean=0.6,sd=0.15),
"Group"="Control")
nor2 <- data.frame("val"=rnorm(50, mean=0.8,sd=0.1),
"Group"="Treated")

#bind the data by rows (e.g. stack one ontop of the other)
dat <- rbind(nor1,nor2)


#Density plots with semi-transparent fill
g1 <- ggplot(dat, aes(x=val, fill=Group)) +
geom_density(alpha=.3, bw=0.1) + xlab("Fertilizer adoption")

# boxplot and remove redundant legend.
g2 <- ggplot(dat, aes(x=Group, y=val, fill=Group)) +
geom_boxplot() +
    guides(fill=FALSE) + ylab("Fertilizer adoption")

g3 <- ggplot(dat, aes(x=val, fill=Group)) +
geom_histogram(alpha=.3, bw=0.1) + xlab("Fertilizer
adoption")

g4 <- ggplot(dat, aes(x=val, fill=Group)) +
geom_beeswarm(aes(x=0, y=val, colour=Group)) +
ylab("Fertilizer adoption") +
theme(axis.title.x=element_blank(),
        axis.text.x=element_blank(),
        axis.ticks.x=element_blank())


multiplot(g1,g2,g3,g4, cols=2)
```

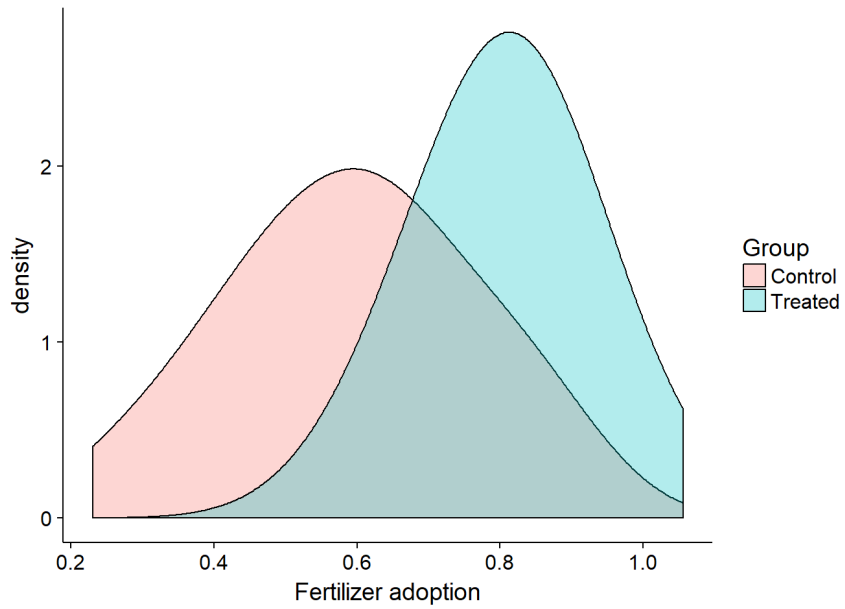


We can see that the density plot and box plot are quite good for viewing differences between groups. However the bee swarm plot and histogram are too busy to be able to quickly draw conclusions (though they are useful in other scenarios, such as seeing outliers). Therefore this data should be presented as a density or box plot to fulfill our criteria of being **simple to read but true to the data** .

Significant and non significant differences

So we now have a descriptive graph of our treatment and control groups.

```
ggplot(dat, aes(x=val, fill=Group)) + geom_density(alpha=.3,
bw=0.1) + xlab("Fertilizer adoption")
```



The two groups look different, but how can we be more sure, and how can we *quantify* our uncertainty?

Hypothesis testing

Statistical testing exists because we only ever have a **sample** of a **population** (see glossary). For example, we might measure the adoption of fertilizer in a sample of east African farmers during a randomized trial. But we are trying to infer something about **ALL** east African farmers (i.e. generalizing from our sample to the whole population). We want to be able to understand effects on the actual

populations so we can make decisions about a program intervention; *therefore we want to compare populations, but we only have samples*. Hypothesis testing allows us to test the differences in the underlying populations given a few key assumptions.

Hypothesis tests have certain assumptions underlying them. Some example assumptions for commonly used tests are:

- **T-test**: Assumes a normal-like distribution. This test was made in the 1950s for small sample sizes, more robust tests are now available which should be used
- **Paired T-test**: A T-test (as above), but used for measuring the **same subject** before and after treatment (e.g. measuring farmer planting compliance before and after a leafleting campaign)
- **Two-sample T-test:** A T-test used for **two different groups** e.g. comparing maize yields of treated farmers and control farmers.
- **Welch test:** An improvement on the T-test, which is more robust to unequal variances (e.g. more robust to having samples where the width of the density plot is different). We can call the Welch test with the argument "var.equal=FALSE" in the t.test() R function. Note that a Welch test is only valid for normally distributed data (like the T-test).

How do we know when to use each test? And how do we test the assumptions of these tests?

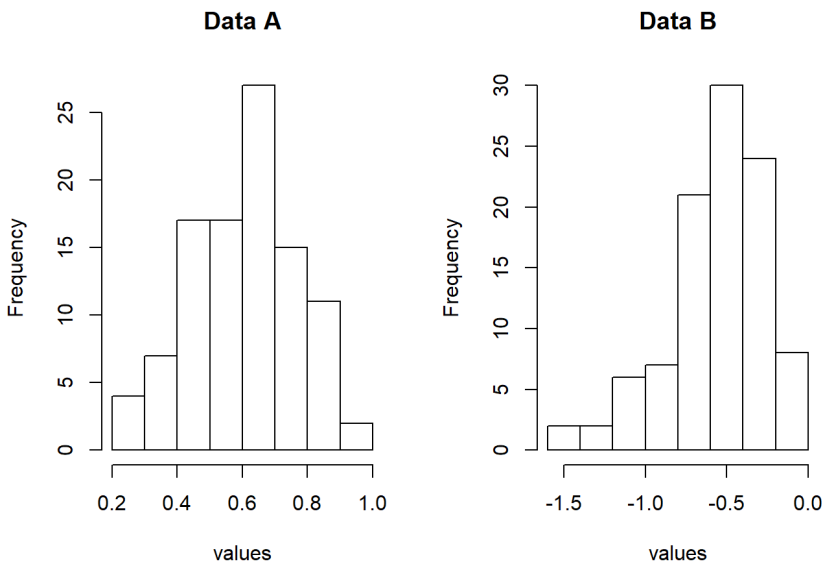
Normality assumptions

Many hypothesis tests and sample size calculations assume normally distributed data (remember, this means the data is symmetrical about the mean). However, very often we would like to conduct hypothesis tests on non-normal data. We will examine hypothesis tests for non-normal data shortly, but first, how can we know if our data is normally distributed? There are a number of approaches here, I will present two of the more common methods.

Let's take the following data sets:

```
#
#lets make some data
set.seed(122)
dat <- data.frame("val"=rnorm(100, mean=0.6,sd=0.15))
```

```
#this lets us plot graphs next to each other
par(mfrow = c(1, 2))


#this is a simple histogram
# i call the column named "val" from dataframe named "dat"
with the $val addition
hist(dat$val, main="Data A", xlab="values")


hist(log(dat$val), main="Data B",xlab="values")
```

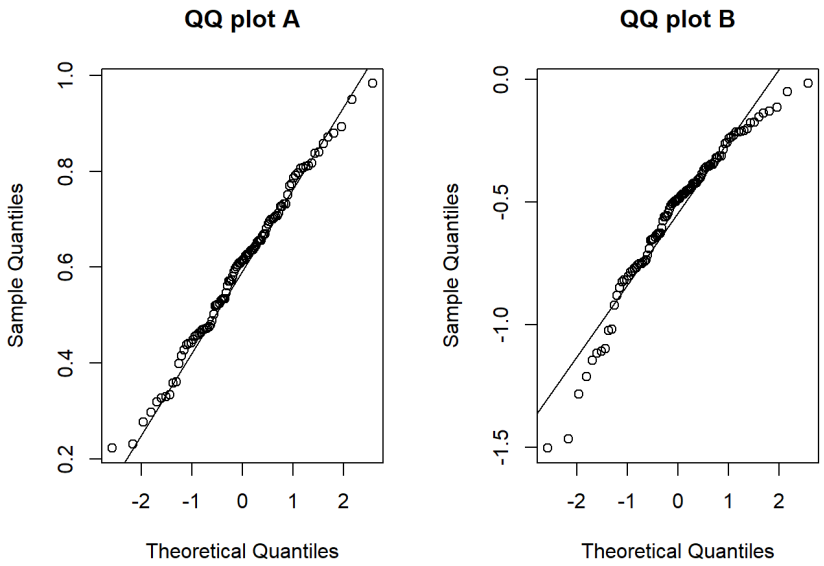


Hopefully it is clear that data A might be normal, but data B certainly looks non-normal (note the clear asymmetry)!

To test the normality more thoroughly,we can plot an ideal normal distribution and then our data on top (this is known as a QQ-plot), and then calculate the R-squared between the relationships:

```
#
#plot data against ideal normal
#note that the subset command means we will only take the
val column of data with Group=="Control"
# subset(dat, Group=="Treated") would return the whole
dataframe where the column "Group" is equal to "Treated",
adding the $val allows us to access the val column directly
par(mfrow = c(1, 2))

qqnorm(dat$val, main="QQ plot A")
qqline(dat$val)
```

```
qqnorm(log(dat$val), main="QQ plot B")
qqline(log(dat$val))
```




```
#get Rsquared
qn=qqnorm(log(dat$val), plot.it=FALSE)
rsq_B <- cor(qn$x,qn$y)

#get Rsquared
qn=qqnorm(dat$val, plot.it=FALSE)
rsq_A <- cor(qn$x,qn$y)
```

The QQplot plots a perfect normal distribution as a solid diagonal line and then our actual data as markers (circles) on top of that.

We can see from the plots that our data A follows a normal (the solid diagonal line) distribution well. However examining the data in plot B we can see substantial deviations from the normal line, these deviations are clearly not random but follow a trend (random deviations would occur equally as many times on either side of the line).

To quantify this we can extract the R-squareds (we'll learn more on R-squared in later lessons, but it essentially measures how close our points are to that diagonal line on a scale from 0 to 1). The R-squared values are 0.997 and 0.974 for A and B respectively. It is enough to look at the data to realize plot B is non-normal, but we can also pick a cutoff,

such as an R-squared of 0.975 and call anything below this non-normal and anything above this normal.

Another approach is to use the Shapiro test, which will conduct a hypothesis test on whether our data is normal or not:

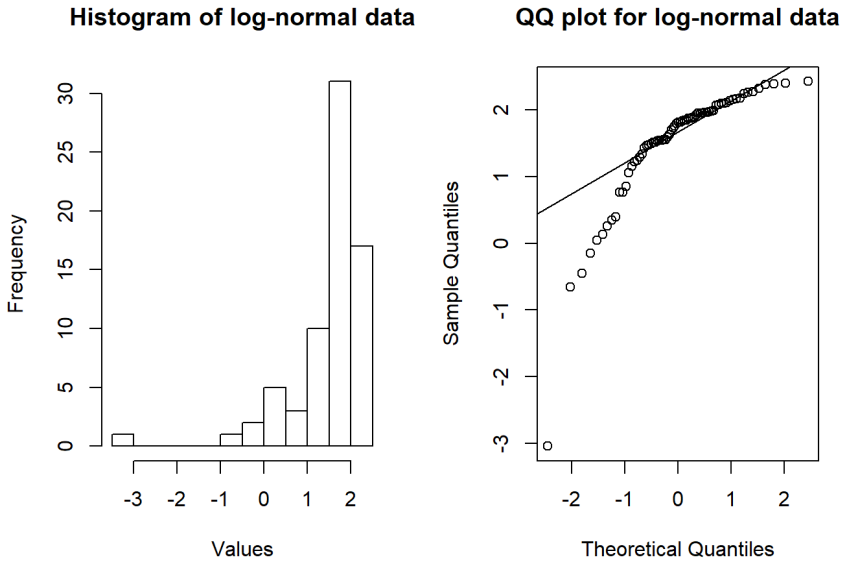
```
shapiro.test(dat$val)
```

```
        Shapiro-Wilk normality test

data:  dat$val
W = 0.99227, p-value = 0.8402
```

The key part here is the P-value, which is 0.84 for data A. The higher the p-value the more normal the distribution. Often we will use a P-value cutoff of 0.1 to ensure normality. So anything with a p-value of <0.1 we would class as non-normal. This approach is complementary to the above, and I would recommend both plotting a QQ-plot (as above) **and** running a Shapiro test (and reporting both in any report).

Let's look at the QQ-plot and Shapiro test for some non-normal (specifically log-normal) data:

```
non_norm <- log(rnorm(75, mean=5,sd=3))


par(mfrow = c(1, 2))
hist(non_norm, main="Histogram of log-normal data",
xlab="Values")
qqnorm(non_norm, main="QQ plot for log-normal data")
qqline(non_norm)
```




```
        Shapiro-Wilk normality test

data:  non_norm
W = 0.76001, p-value = 2.375e-09
```

The above distribution has a Shapiro test P-value of 2.4e-09 and the QQ-plot shows *systematic deviations* from the normal line. This strongly suggests our data B is non- normal.

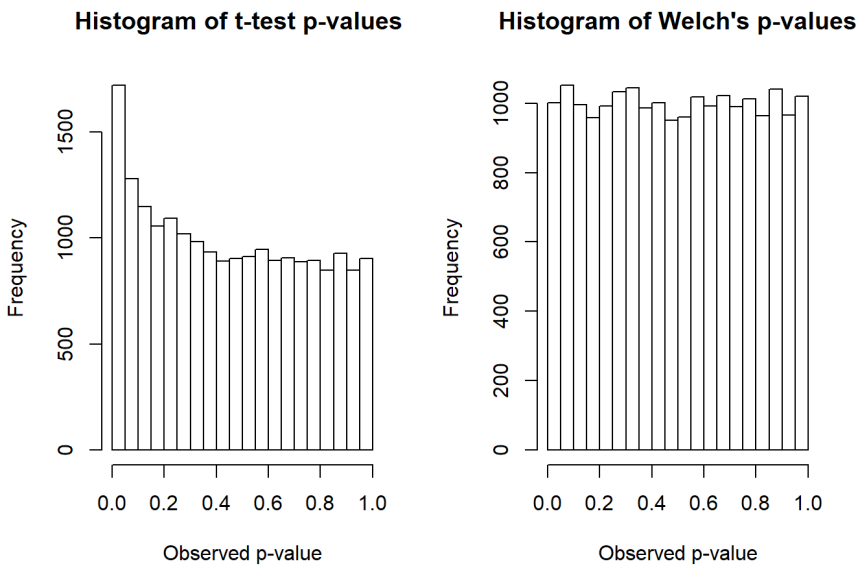
Hypothesis testing normal distributions

Once we are convinced that our data is normally distributed (we'll tackle non-normal data later) we can run some hypothesis tests. The T-test is commonly used as a "go-to" hypothesis test but here I want to convince you to use the Welch test instead (remember, to access the Welch test in R we call the t.test() function but add the argument **var.equal=FALSE**). The Welch test is better suited to unequal sample sizes and unequal variances, which we will often come across in the real world.

Below, I have run a simulation to demonstrate the danger of using T-tests over Welch tests with unequal variance data:

```
#set up vars
nSims <- 20000 #number of simulations
```

```
p1 <-c()
p2 <- c()

#create variables for dataframe
catx<-rep("x",38)
caty<-rep("y",22)
condition<- c(catx,caty)

#run simulations
for(i in 1:nSims){ #for each simulated experiment
  sim_a<-rnorm(n = 38, mean = 0, sd = 1.11) #simulate
participants condition a
  sim_b<-rnorm(n = 22, mean = 0, sd = 1.84) #simulate
participants condition b
  p1[i]<-t.test(sim_a,sim_b, alternative = "two.sided",
var.equal = TRUE)$p.value #perform the t-test and store p-
value
  p2[i]<-t.test(sim_a,sim_b, alternative = "two.sided",
var.equal = FALSE)$p.value #perform the Welch test and store
p-value
}


par(mfrow = c(1, 2))
hist(p1, main="Histogram of t-test p-values ", xlab=
("Observed p-value"))
hist(p2, main="Histogram of Welch's p-values", xlab=
("Observed p-value"))
```



We are here plotting a histogram of P-values derived from the two tests after running a simulation many many times. We will remember that the p-value is *the probability that the difference between groups is observed purely by chance (i.e. there is no real difference between groups)* . The two populations here have the same mean, this means that we

should accept the null hypothesis and reject the alternative hypothesis (see glossary).

As we know there is no real difference between the populations, we should accept the null hypothesis and see a flat distribution of p-values (we should see a p-value of <0.05 5% of the time by definition, as a p-value of 0.05 implies a 5% chance of a false result in frequentist statistics). Note that in a real situation we wouldn't already know if there was or was not a difference between populations!

We see that the right hand histogram of the Welch test results is flat (this is good, it matches what we know about the data). However, the T-test (left hand histogram) performs very poorly here and reports more smaller p-values. This means the chances of us wrongly concluding that *there **is** a significant difference between the groups* is much higher. To be precise, we are 72 % more likely to make an incorrect conclusion if we use the T-test rather than the Welch test here!!! This could lead to accepting program or product innovations which actually have no effect, wasting time and money.

The moral of the story is to **use the right test at the right time**. If the data is normal, then we should always use the Welch test (and ignore the T-test) as there are fewer assumptions for the Welch test vs. the T-test. If the data is non-normal then we need a very different test.

Hypothesis testing non-normal distributions

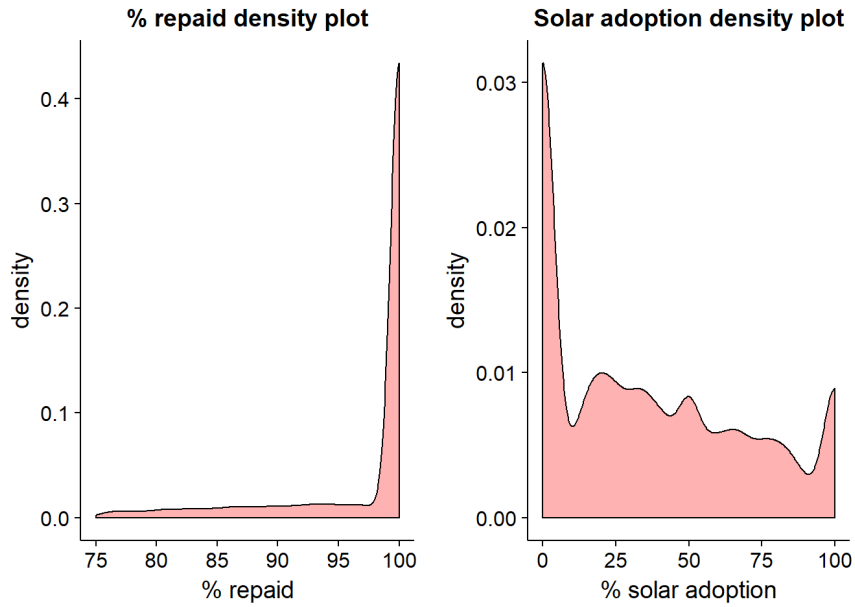
The above holds for normally distributed data. But what about non-normal data?

We are often confronted with non-normal data in real life. Any scenario where our values will approach 0 or 100% are likely non-normal. For example, if we look at loan repayment at an established bank, or if we look at the adoption of a very popular (or unpopular) product.

Some examples of non-normal distributions are shown for farmers in Rwanda below (loan repayment percentage and solar light adoption rates):

```
df <- read.csv("./Sample_data.csv")
df$X..Repaid <- df$TotalRepaid/df$TotalCredit * 100
df$solar_A <- df$solar_A *100
d1 <- ggplot(df, aes(x=X..Repaid)) + geom_density(alpha=0.3,
```

```
fill="red") + ggtitle("% repaid density plot")  + xlab("%
repaid") + xlim(75,100)
d2<- ggplot(df, aes(x=solar_A)) + geom_density(alpha=0.3,
fill="red") + ggtitle("Solar adoption density plot")  +
xlab("% solar adoption")

multiplot(d1,d2,cols=2)
```



We can clearly see from the density plot that our data is non-normal in both cases. But let us use the diagnostics outlined above to make sure:

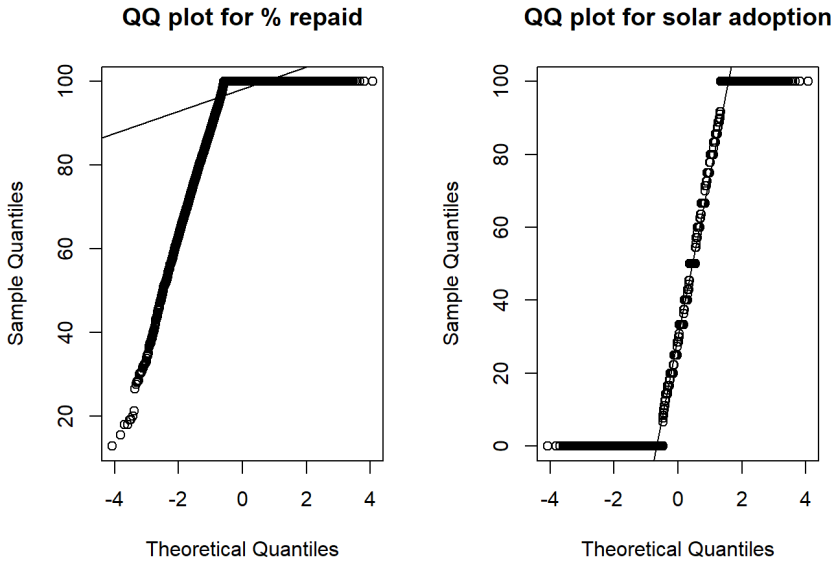
```
par(mfrow = c(1, 2)) qqnorm(df$X..Repaid, main="QQ plot for
% repaid") qqline(df$X..Repaid) qqnorm(df$solar_A, main="QQ
plot for solar adoption") qqline(df$solar_A)
```

```
#Shapiro test has an upper limit of 5000, so we will
randomly take a subset of data to test instead
shapiro.test(sample(df$X..Repaid, size=1000))
```

```
Shapiro-Wilk normality test data: sample(df$X..Repaid, size
= 1000) W = 0.54519, p-value < 2.2e-16
```

```
shapiro.test(sample(df$solar_A, size=1000))
```

```
Shapiro-Wilk normality test data: sample(df$solar_A, size =
1000) W = 0.87404, p-value < 2.2e-16
```




```
    Shapiro-Wilk normality test

data:  sample(df$X..Repaid, size = 1000)
W = 0.54519, p-value < 2.2e-16
```

```
    Shapiro-Wilk normality test

data:  sample(df$solar_A, size = 1000)
W = 0.87404, p-value < 2.2e-16
```

WOW! Those are some terrible looking QQplots, and some really low P-values! The data is far from normal.

However, we would still like to do hypothesis testing on such data.

How to test non-normal data

Now let's compare methods for testing an intervention which increases solar adoption by 1.5%. We will use the T-test (incorrect for non-normal data), the Welch test (also incorrect for non-normal data) and the **Wilcoxon test** (an appropriate test for non-normal data).

```
#load data
norm1 <- df$solar_A
```

```
norm2 <- norm1 *1.015
```

The T-test:

```
#t test
t.test(norm1,norm2, alternative = "two.sided", var.equal =
TRUE)
```

P-value = 0.1048

Welch test:

```
#welch test
t.test(norm1,norm2, alternative = "two.sided", var.equal =
FALSE)
```

P-value = 0.1048

Wilcoxon test:

```
wilcox.test(norm1 , norm2 ,paired=FALSE, correct=TRUE)
```

P-value = 1.55e-06

We can see that we get very different results! The T-test and Welch test report p-values of 0.105 and 0.105, which would indicate non-significance under a typical p-value threshold of 0.05. Whereas the Wilcoxon test indicates a p-value of 1.55e-06.

You can see that depending on what test we use, we come out with completely different results. The T-test and Welch Test would lead us to believe that the intervention was not useful, whilst the Wilcoxon test makes clear that the intervention does seem to have an effect. It is therefore important to **use the right test for the right data.** Imagine if we had been testing an intervention that was harmful to farmers but used the wrong test and so concluded the intervention was safe!

We can now add the Wilcoxon test to our library:

- **Wilcoxon-test**: A good test for non normal data with very few assumptions. We are testing the hypothesis that two datasets have different distributions. The main assumptions here are: 1) samples are randomly representative of population, 2) samples are independent of each other , 3) values have an order (e.g. 3 is more than 1, but we can't say that true is more than false).

Reporting data

We have now seen how to calculate robust P-values for data. However the way we report data is also dependent on the shape of the distribution. For normal data we can simply report the mean and the confidence intervals, for example *mean of 4.5 (CI: 1.5–7.5) Kgs per acre* (you could also report the mean with the standard error, as long as it is clear what you are reporting). Non-normal data is a little more tricky, and we should generally report the median and 1st and 3rd quartiles, e.g. *median of 4.5 (Q1 2.5 & Q3 6)*.

We will remember that the normal distribution is symmetrical, it is the **asymmetry** of non-normal distributions which prevents us reporting CIs, SEMs, or StDevs (as these are all based on symmetry).

A concept related to the hypothesis testing idea above is **statistical power**. Power is used in calculating sample sizes and also in interpreting the results of an RCT after a trial. A low power will mean our results are less reliable and we may be making false conclusions about interventions.

It is important that we use the correct power calculation in much the same way it is important we use the right hypothesis test. Power calculations are dependent on the data distribution and the most common power calculator used in R is fine for normally distributed data. This calculator requires an "effect size" which is a simple way to measure the effect of a treatment. The effect size is basically the difference in means between samples divided by the standard deviation of the samples. This means the units are in standard deviations. So an effect size of 1 means the data has shifted over an amount equal to 1 standard deviation. I have included a function below to calculate effect sizes using the Cohen's D equation:

```
cohen_d <- function(d1,d2) {

  m1 <- mean(d1, na.rm=TRUE)
  m2 <- mean(d2, na.rm=TRUE)
  s1 <- sd(d1, na.rm=TRUE)
  s2 <- sd(d2, na.rm=TRUE)
  spo <- sqrt((s1**2 + s2**2)/2)
  d <- (m1 - m2)/spo
  rpb <- d / sqrt((d**2)+4)
  ret <- list("rpb" = rpb, "effectsi" = d)
  return(ret)  }
```

Now we have a way to calculate effect size, let's use it to calculate the power for a study with normal data and 100 participants. We will use the function " pwr.t.test " from the pwr library. This function requires the following arguments:

- **h**—this is the effect size calculated by the Cohen's D equation above. We call it with the "$effectsi" after the object (see code for details).
- **n**—the sample size of each sample
- **sig.level**—this is the p-value cutoff we want to use in our analysis. This will usually be 0.05 or 0.1.
- **power**—we want to set this to NULL to get the function to return power to us.

All together we will call this: `pwr.t.test(n= 200, d= inp$effectsi, power=NULL, alternative="two.sided")` from the library "pwr".

Where the "length()" argument returns the number of elements in the list named "norm".

```
norm <- rnorm(200,40,10)
norm2 <- norm*1.1
inp <- cohen_d(norm, norm2)
```

We see here that we have a power of 0.97. Ideally we want power to be above 0.75–0.8 for a good probability of detecting an effect accurately.

Note that we can calculate the sample size needed for a power of 0.8 by filling in the power argument and setting **n** to NULL as below (it is a

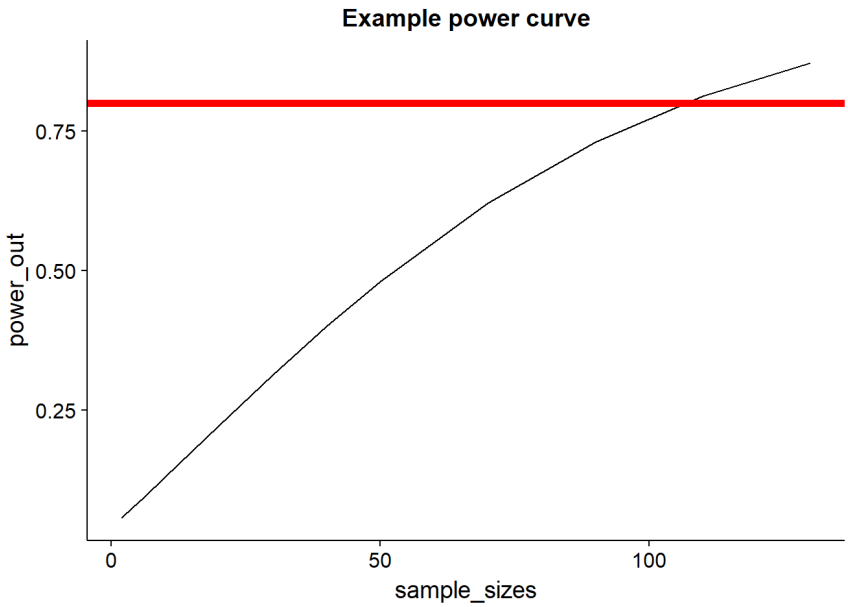
slightly different equation as we are assuming equally sized control and treatment groups):

```
pwr.t.test(n= NULL, d= inp$effectsi, power=0.8,
alternative="two.sided")
```

```
     Two-sample t test power calculation

              n = 106.3332
              d = 0.3859861
      sig.level = 0.05
          power = 0.8
    alternative = two.sided

NOTE: n is number in *each* group
```

We can make a loop to look at the power with different sample sizes, and then plot this with a line indicating where a power of 0.8 is.

```
power_out <- c() # initialise this empty so we can append to
it
sample_sizes <- c(2,4,6,10,20,30,40,50,70,90,110,130)
for(i in sample_sizes){
  x <- pwr.t.test(n= i, d= inp$effectsi, power=NULL,
alternative="two.sided")$power
  power_out <- c(power_out, x)

}

p <- ggplot() + geom_line(aes(x=sample_sizes, y= power_out))
+ geom_hline(yintercept = 0.8, color="red", size=2) +
ggtitle("Example power curve")
p
```

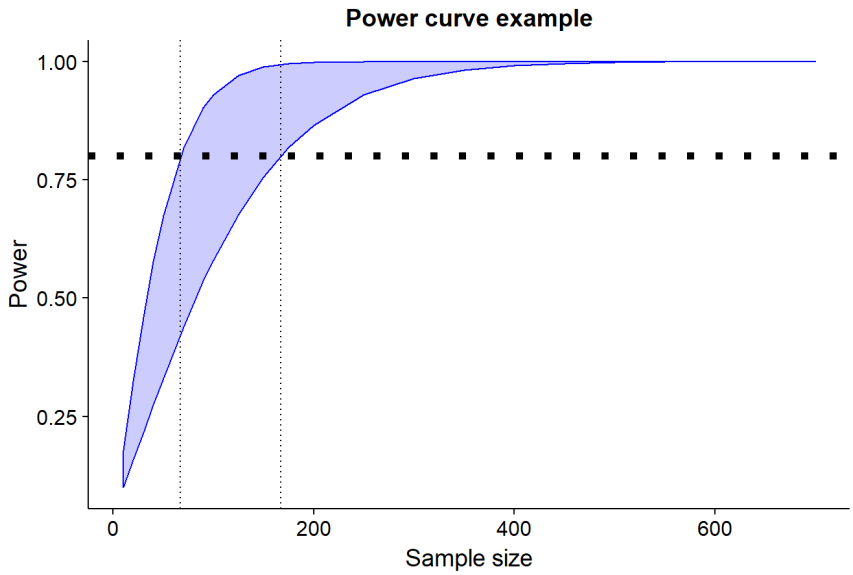


Plots like the one above help you to see what sample sizes and powers are reasonable. I would strongly recommend plotting power curves for a lower estimate of effect size (i.e. the smallest size effect you might see) and and upper estimate of effect size (i.e. the largest effect you might see).

To run through a complete example: let us imagine we want to run an RCT (or A/B test) that examines the effect of a new interest rate on fertilizer adoption. We have two groups, a control group, which has the same interest rate as before. And a treatment group which has an interest rate reduced by 3 points (e.g. 15% interest rather than 18%). Our understanding of the customer means we think we will see a 15 to 25% increase in fertilizer adoption with the treatment. We plot a power curve like follows *(please examine the code and make sure you understand it)*:

```
control <- rnorm(100, 0.5,0.2)
treat_lower_estimate <- control *1.15
treat_upper_estimate <- control *1.25

power_lw <- c() # initialise this empty so we can append to
it
power_hi <- c() # initialise this empty so we can append to
it
sample_sizes <-
c(10,20,30,40,50,70,90,100,125,150,175,200,250,300,350,400,4
50,500,550,600,700)
for(i in sample_sizes){

  lower_cohen <- cohen_d(control, treat_lower_estimate)
  a <- pwr.t.test(d = lower_cohen$effectsi , n=i,  sig.level
= 0.05, power = NULL)$power
```

```
  power_lw <- c(power_lw, a)

  upper_cohen <- cohen_d(control, treat_upper_estimate)
  b <- pwr.t.test(d = upper_cohen$effectsi , n=i,  sig.level
= 0.05, power = NULL)$power
  power_hi <- c(power_hi, b)

}


marker <- pwr.t.test(d = lower_cohen$effectsi , n=NULL,
sig.level = 0.05, power = 0.8)$n
marker2 <- pwr.t.test(d = upper_cohen$effectsi , n=NULL,
sig.level = 0.05, power = 0.8)$n
ggplot() + geom_ribbon(aes(x=sample_sizes, ymin= power_lw,
ymax=power_hi), alpha=0.2, colour="blue", fill="blue")  +
xlab("Sample size") + ylab("Power") + geom_vline(xintercept
= marker, linetype="dotted" ) + geom_hline(yintercept=0.8,
linetype="dotted", size=2) + geom_vline(xintercept = marker2
, linetype="dotted") + labs(title="Power curve example",
caption="Power curve indicating sample sizes needed for a
0.8 power (dotted lines)") + theme(plot.caption =
element_text(hjust = 0.5))
```



Power curve indicating sample sizes needed for a 0.8 power (dotted lines)

The above plot allows us to see that it would require a large sample size (n is approx. 200, so we would need 400 participants in this trial—200 control & 200 treatment) to detect a 15% change in adoption with a power of 0.8. Plotting the data like this allows more flexible decision making and is highly recommended.

Non-Normally distributed data and sample size/power

The above is all fine for normal power calculations. But what about non-normal powers?

We have a solution for this too!

I've made a function below ("MCpower") which will calculate the power for a non-normal distribution. It requires data (or expected data) for samples 1 and 2 (treatment and control) and a sample size (the "size" argument).

Please make sure you understand this code and are comfortable using it

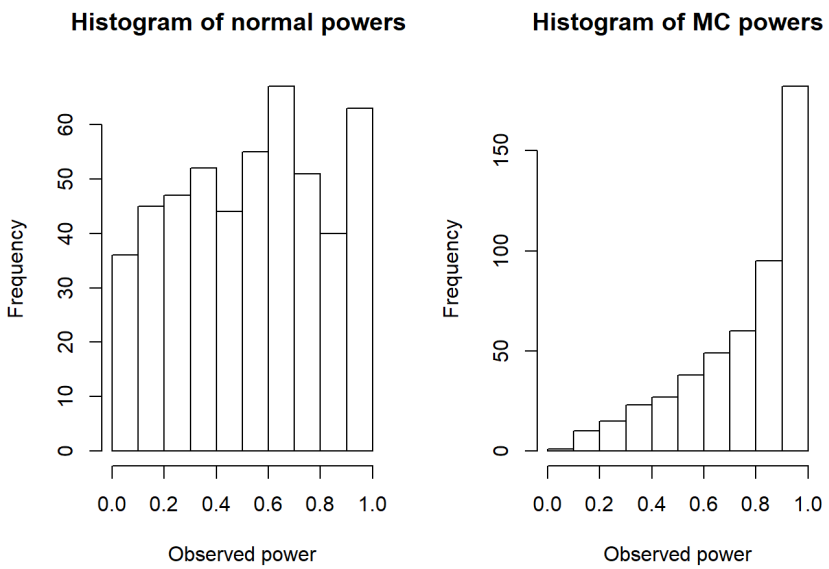
```
MCpower = function(sample1, sample2, size) {

  reps = 1000
    results  <- sapply(1:reps, function(r) {
        resample1 <- sample(sample1, size=size,
replace=TRUE)
        resample2 <- sample(sample2, size=size,
replace=TRUE)
        test <- wilcox.test(resample1, resample2,
alternative="two.sided",paired=FALSE, correct=TRUE)
        test$p.value
    })
    sum(results<0.05)/reps
}


#we use it like this
Non_norm_power <- MCpower(control, treat_upper_estimate,
100)
```

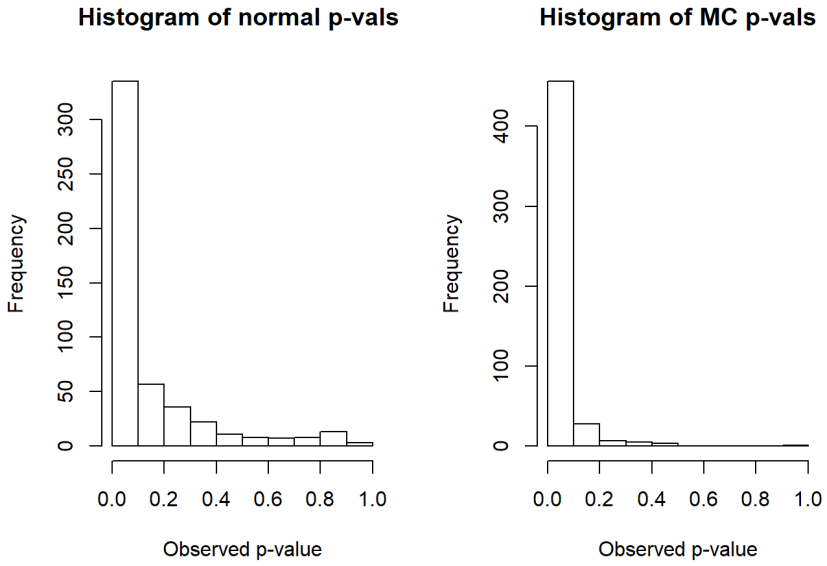
We can compare these approaches to understand why we need to use the correct test on the correct data. Let's run some simulations of experiments. The data below is non-normal, and there is a difference between the samples. Therefore we would expect to see a *statistically significant difference* . Note that whilst a normal power test or hypothesis test will test for means, the power calculations here will test for the overall similarity of distributions.

Let's run a simulation where we calculate the power needed to test the difference between two populations with different means:

```
#set up vars
nSims <- 500 #number of simulations

#initialise empty vectors for appending to
x1 <- c()
```

```
x2 <- c()
p1 <- c()
p2 <- c()

#run simulations
for(i in 1:nSims){ #for each simulated experiment
  set.seed(i)
  sim_a <- c(rnorm(150, mean=130,sd=20) )**2
  sim_b <- c(rnorm(150, mean=120,sd=35) )**2
  sim_b[which(is.nan(sim_b)==TRUE)] <- 1
  sim_a[which(is.nan(sim_a)==TRUE)] <- 1
  inp <- cohen_d(sim_a, sim_b) # effect size
  x1[i] <- pwr.t.test(d= inp$effectsi , n =length(sim_a),
sig.level = 0.05, power = NULL)$power
  x2[i] <- MCpower(sim_a, sim_b,  size=150)
  p1[i] <- t.test(sim_a,sim_b)$p.value
  p2[i] <- wilcox.test(sim_a,sim_b, paired=FALSE,
correct=TRUE)$p.value
}

#hist(sim_b, bins=20)
#shapiro.test(sim_b)
par(mfrow = c(1, 2))
hist(x1, main="Histogram of normal powers ", xlab=("Observed
power"), xlim =c(0,1))
hist(x2, main="Histogram of MC powers", xlab=("Observed
power"), xlim =c(0,1))
```



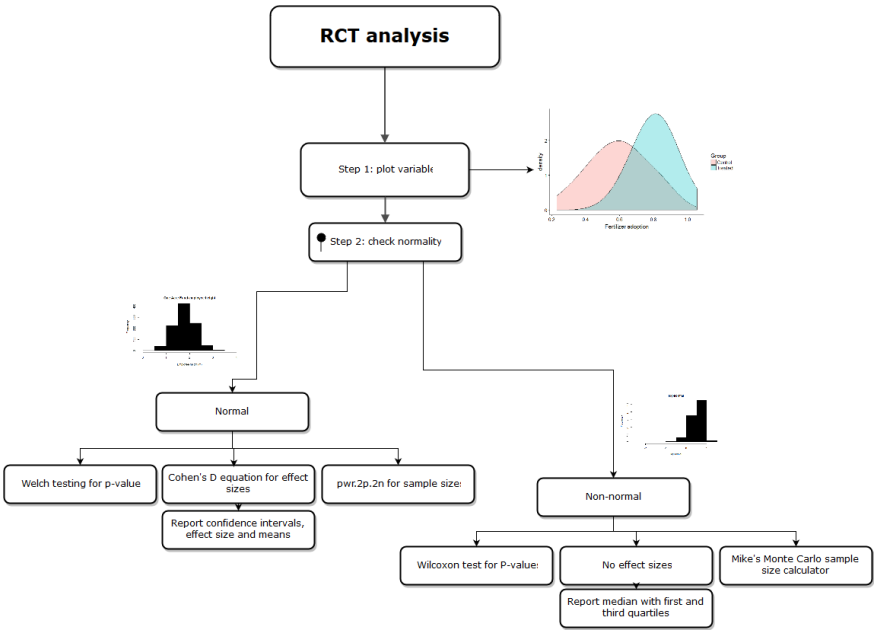


The mean p-value from several hundred simulations of t.testing is 0.133, and from MC testing is 0.0305 (see the histograms above for the distributions). We know now that the T-test assumes a normal-like distribution, which is incorrect here, whereas the MC method requires very few assumptions to be met. We can see that the MC method generally has higher power and often correctly identifies that there **is** a significant difference between these samples, unlike the T-test which performs poorly.

Note that for normal data, the results of normal power calculations and my MC method are quite similar. Comparing two distributions with a an effect size of 1 yields P-values of 0.012 and 0.014 and powers of 0.86 and 0.85 for the standard method and MC method respectively.

Summary

We have now seen how to plot and present data so that it is simple and true to the underlying data, we have also seen how to calculate p-values and powers for normal and non-normal data in an RCT, and what happens when we get these assumptions wrong. I have included a brief flow-chart to illustrate the decisions that need to be made:



The best practices when analysing an RCT:

- Even though you calculate the sample size before an RCT, it is important to calculate the power (which is related to sample size) after the RCT to make sure our study has sufficient power. Remember, low power means we are unlikely to detect an effect from an intervention, and generally makes our results less reliable. I would expect every RCT analysis to begin with power calculations for key variables

- We must also plot the distributions of key variables. In an RCT we will often have 2 or more groups (e.g. Control and treatment) to observe. I would therefore suggest plotting density plots or boxplots for each key variable at the start of the analysis. R-Markdown (which is what this lesson is written in) is very helpful here and will be a key focus of the AMP.

- Test our assumptions. We should run a Shapiro test on our data to make sure it is normal *before* deciding what method to use (e.g. a T-test or a Wilcoxon test).

- Always report the p-value with means and confidence intervals for normal distributed data OR with medians and first and third quartiles for non-normal data.

- Make sure graphs are **simple, but true to the data**.

Recap of statistical tests

- **T-test**: Assumes a normal-like distribution. This test was made in the 1950s for small sample sizes, more robust tests are now available which should be used
- **Paired T-test**: A T-test (as above), but used for measuring the **same subject** before and after treatment (e.g. measuring farmer planting compliance before and after a leafleting campaign)
- **Two-sample T-test:** A T-test used for **two different groups** e.g. comparing maize yields of treated farmers and control farmers.
- **Welch test:** An improvement on the T-test, which is more robust to unequal variances (e.g. more robust to having samples where the width of the density plot is different). We can call the Welch test with the argument "var.equal=FALSE" in the t.test() R function. Note that a Welch test is only valid for normally distributed data (like the T-test).
- **Wilcoxon-test**: A good test for non normal data with very few assumptions. We are testing the hypothesis that two datasets have different distributions. The main assumptions here are: 1) samples are randomly representative of population, 2) samples are independent of each other , 3) values have an order (e.g. 3 is more than 1, but we can't say that true is more than false).

. . .

Resources

- Instructions to setup/install R
- Interactive R basics tutorial—highly recommended
- Base R cheatsheet—highly recommended, print it and keep it on your desk!
- Non-interactive tutorial (if that's your thing)
- Youtube tutorials
- Cheatsheets—recommended for more advanced R users only
- StackOverFlow—for specific technical R questions

. . .

Glossary

- **Observation**: In statistics, this is the value of a variable at a particular time. We can also think of observations as rows of data in a dataframe
- **Variable**: A characteristic that can be observed. This might also be called a *feature* and can be thought of as a column in a dataframe (examples would be: gender, price of sugar, or harvest yields).
- **Normality**: A normal distribution is approximately symmetrical, and the mean must equal the median. The further a data point is from the mean the less likely it is to occur. Normal data is also known as "bell shaped data".
- **Variance and Standard deviation**: Variance is the width of a normal distribution. Standard deviation is the square root of the variance. We can think of this as how spread-out the data is, a high variance means lots of variability, a low variance means most observations will have very similar values.
- **Quartiles**: A quartile is another data metric. If I were to line up (in order) all the values for a variable then the first quartile would be the value at one quarter of the total length. Likewise the third quartile would be the value at three quarters length. For example, the data *1,3,6,8,10,14,15,18,20,22,25,26* has a first quartile of *6*, and a third quartile of *20*.
- **Population**: A large pool of potential data from which a sample is taken.
- **Sample**: A subset of a population for which we have data. For example, we might be studying a *sample* of 1000 Kenyan farmers but want to generalize to the whole Kenyan farmer *population*.
- **P-value**: The probability that the observed result could be generated purely by chance (i.e. the probability there is no real difference between *populations*)
- **Power**: The probability that we are able to detect a difference between two populations, given a certain sample size, distribution and error rate. Note that power also has an effect on how robust our results are.
- **Effect Size**: The standardized difference between 2 groups. It has the units *stdev*. So an effect size of 1 is equal to a difference of 1 stdev. between groups.

- **Hypothesis test**: a statistical test. Each hypothesis test has two parts, a *null hypothesis* which says that the two populations are the same (i.e. there is 0 difference) and an *alternative hypothesis* which says that there is a difference between populations Often the null hypothesis is called H0 and the alternative hypothesis is called H1.
- **Standard form / scientific notation**: we can write numbers with lots of zeros in standard form. e.g. 0.001 become 1e-3 and 1000 becomes 1e+3 (meaning 1 *10 to the power 3).
- **Histogram** : a graph of rectangles (also known as *bins*), the height of a bin is proportional to the number of observations and the width is proportional to the range of values encompassed.

. . .

Next post:

- Minimum detectable effects
- Planning an A/B test
- Intra cluster effects
- And more!

. . .

Originally published at michael-bar.github.io.

If you see any errors in these posts, please do leave a comment!
Likewise if anything requires further explanation, then comment!