Resampling, part 1

Jennifer Wilcock

STAT221 2020 S2

We are going to look at several methods known as resampling methods. These methods include the bootstrap, the jack-knife, cross-validation, and permutation tests.

All resampling methods have the same basic idea, of using observed sample(s) of data and repeatedly sampling from the original observed sample(s) to obtain a large collection of re-samples, and then use the distribution of the resamples to tell us about the original sample.

Applications to classical statistical problems

Resampling methods are inherently computational, but directly complement classical approaches to the same problems.

We can use resampling methods for tasks where there are already parametric approaches, including:

- estimating confidence intervals
- hypothesis tests.

Classical approaches are typically *parametric*, computational approaches are typically *non-parametric* (just as we have already considered the difference between methods in regression that are parametric/analytical/classical and which are non-parametric/computational/modern).

Classical/parametric approaches work well and are very quick to use when the assumptions they make are true (or close to being true). Often the assumptions are not true however, so we need methods we can use in these cases.

- For example, if we have one large sample of observations, all sampled from the same population, and the plot of these observed values looks roughly Normal, then we can assume a 95% confidence interval for the true population mean is the observed sample mean $\pm$ 1.96 standard errors.

Modern/non-parametric approaches work well when the assumptions in parametric approaches don't hold, or when the maths is too hard/the problem is too complex to deal with in any other way.

- For example, we have a small sample of observations, all sampled from the same population, and the plot of these observed values doesn't look roughly Normal, then we can use the bootstrap to find a 95% confidence interval for the population mean.

Although we can also use non-parametric methods even when there are parametric methods available, generally we wouldn't because non-parametric solutions take longer to run and are less powerful.

Applications to inherently computational problems

Here, applications include:

- finding optimal values, for example of smoothing parameters

which we might do using, for example, cross-validation or the jack-knife.

The remainder of the course

In the remaining weeks we will look at:

- a review of hypothesis testing, including p-values and confidence intervals
- permutation tests (also known as randomisation tests)
- power of a test, and effect size
- bootstrapping
- if time permits, the jack-knife and cross-validation and applying these methods to finding optimal bandwidths in smoothing.

New Section: Hypothesis Testing

The basic idea

A statistical hypothesis is a conjecture about the distribution of some random variable.

For example, I might conjecture (or speculate, or believe but with no specific evidence to support the belief) that the mean weight of students at UC is 80kg.

I must make my conjecture *before* I see the evidence I use to test if the conjecture is true.

The idea with hypothesis testing is to formally examine two opposing hypotheses:

- null hypothesis, H_0; and
- alternative hypothesis, H_1.

These two hypotheses are "usually" **mutually exclusive and exhaustive**.

Only one of them can be true and the other false.

Setting up a hypothesis test

Statistical hypothesis testing is like a proof by contradiction, using data-based evidence.

There is an analogy with the criminal justice system:

- the null hypothesis states that the accused is innocent
- the burden of proof lies with the prosecution to provide enough evidence ("beyond reasonable doubt") to convince the jury to reject the null hypothesis.

In a hypothesis test, evidence is accumulated by collecting and analyzing sample data for the purpose of determining which of the two hypotheses is true and which of the two hypotheses is false.

Null hypothesis, H_0:

- default assumption that **null hypothesis is true**
- typically contains value for population parameters to be tested
- must contain a "simple hypothesis", so a single population value
- see if there is **evidence to contradict the null hypothesis**

Alternative hypothesis, H_1 (sometimes labelled H_a):

- opposite of null hypothesis
- often the **hypothesis of interest** that would like to be prove (the 'research' hypothesis)
- typically a "composite hypothesis", containing multiple values.

We then ask: Is there enough evidence in the data to refute the null hypothesis, in favour of the alternative?

They can be two-side or one-sided

Hypothesis tests can be two-sided or one-sided (also called two-tailed or one-tailed).

Two tailed tests are typically of this form:

- $H_0 : \mu = 0$
- $H_1 : \mu \neq 0$

Here the null includes the "simple hypothesis" of equality $\mu = 0$,

and the alternative is a "composite hypothesis", which does not include a single specific value.

Two tailed tests are not *directional*, but one tailed tests are directional, e.g.:

- $H_0 : \mu \leq 0$
- $H_1 : \mu > 0$

or

- $H_0 : \mu \geq 0$
- $H_1 : \mu < 0$

Again, in both cases the null includes the "simple hypothesis" of equality $\mu = 0$.

The need for scientific transparency

It is possible to test hypotheses like

- $H_0 : \mu = 0$
- $H_1 : \mu > 0$

which are not **mutually exhaustive**.

But it would be good scientific practice in this example to use $H_0 : \mu \leq 0$ instead, to be transparent in your study results.

This is because if one rejects the null of $\mu = 0$ in favour of the alternative $\mu > 0$ then one would also reject the null for any case of $\mu < 0$, as these possibilities are even more different to the alternative.

In general, the null and alternative should be mutually exclusive and exhaustive, as this usually avoids the lack of transparency in interpreting study results.

The testing procedure

A typical testing process has four steps:

1. Assume null hypothesis H_0 is true.
2. Use statistical theory to derive a statistic (function of the data) used to measure difference from the null hypothesis. Calculate this **test statistic** from the sample data.
3. Find the probability that the test statistic would take a value "as extreme or more extreme" than that actually observed. Referred to as the p-value for the test.
4. If probability from step 3 is:
    - high then sample test statistic is likely if null hypothesis is true, so we have "**insufficient evidence to reject the null**"; or
    - low the sample test statistic is unlikely if null hypothesis is true, so we have "**sufficient evidence to reject the null**"

An example might be doing a t-test.

p-values and rejecting the null hypothesis

Test statistic essentially measures the (in)compatibility between the null hypothesis and the sample data.

After calculating the test statistic, a p-value is obtained by comparing the observed test statistics to the distribution of the test statistic under the null hypothesis.

The p-value is a measure of how unusual the test statistic is, under the null hypothesis.

The **level of significance**, α, is specified before the experiment, to define rejection region:

- p-value $\leq \alpha \implies$ Reject H_0 at significance level α
- p-value $> \alpha \implies$ Do not reject H_0 at significance level α

The **rejection region** is the set of all test statistic values for which H_0 will be rejected.

Interpreting p-values

Smaller p-values indicate that the data provides stronger evidence against H_0.

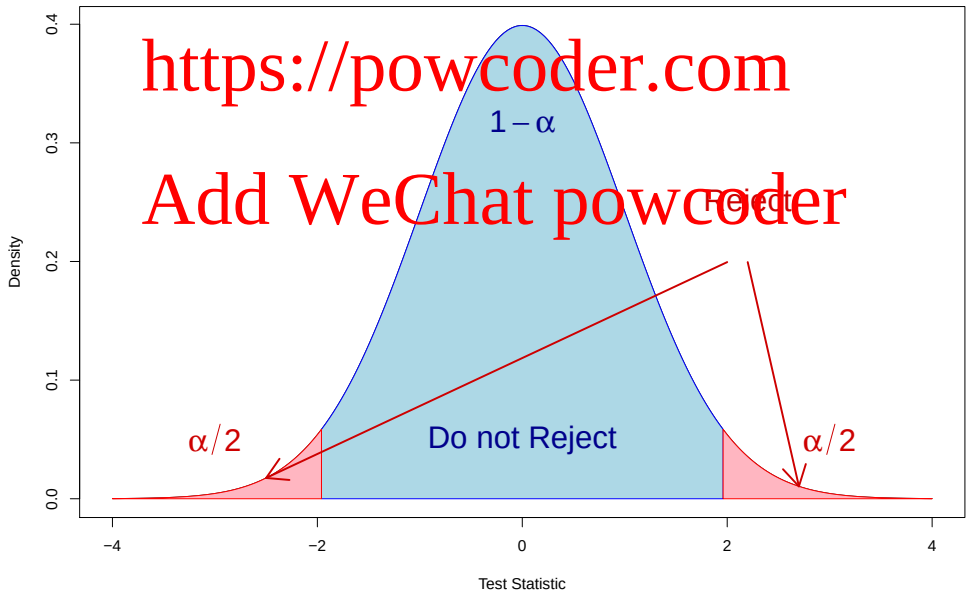
We commonly use a 'rule of thumb' for the interpretation of "small" p-values:

- p-value > 0.1 provides insufficient evidence against H_0;
- $0.05 <$ p-value ≤ 0.1 provides weak evidence against H_0;
- $0.01 <$ p-value ≤ 0.05 provides moderate evidence (or evidence) against H_0; and
- p-value ≤ 0.01 provides strong evidence against H_0.

The decision rule is made by comparing the p-value with the significance level α.

As with confidence intervals, the significance level is chosen by the application, typical values are $\alpha = 0.1$, 0.05 or 0.01.

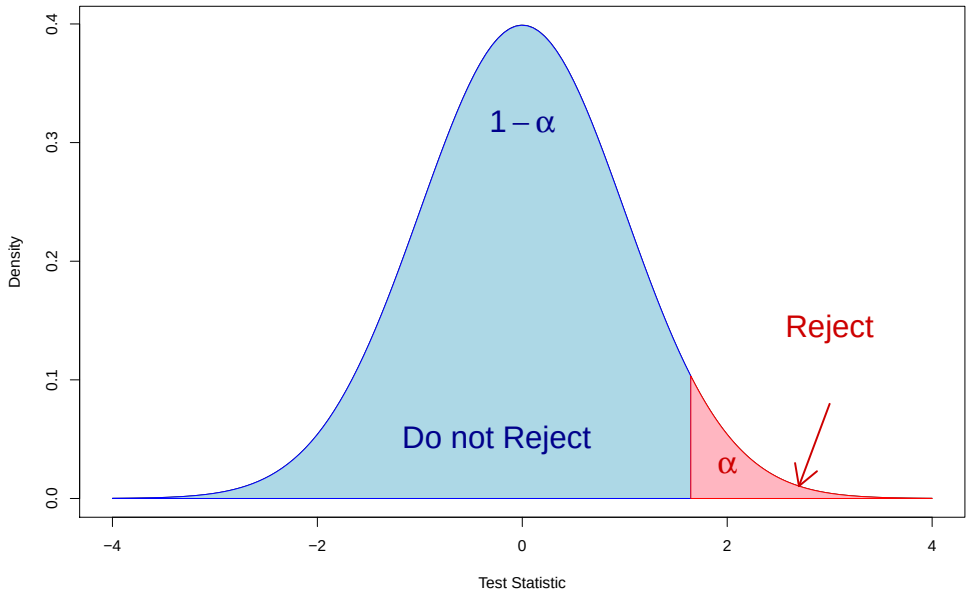
Rejection regions: Two sided test



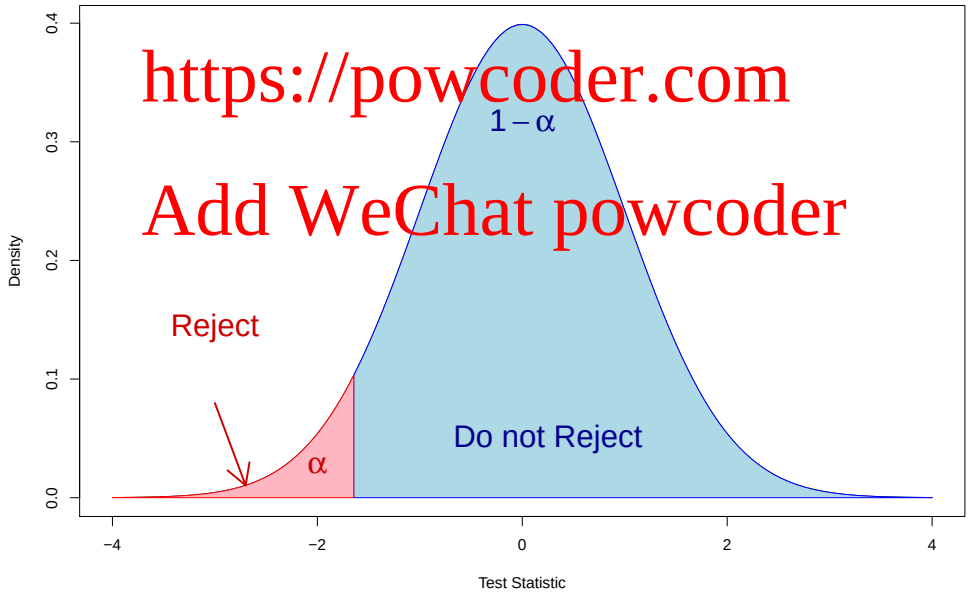
Red rejection region area is significance level α, i.e. probability of rejection if null is true.

Blue non-rejection region area is remainder $1 - \alpha$, i.e. probability of non-rejection if null is true.

Rejection regions: One-sided upper tail test



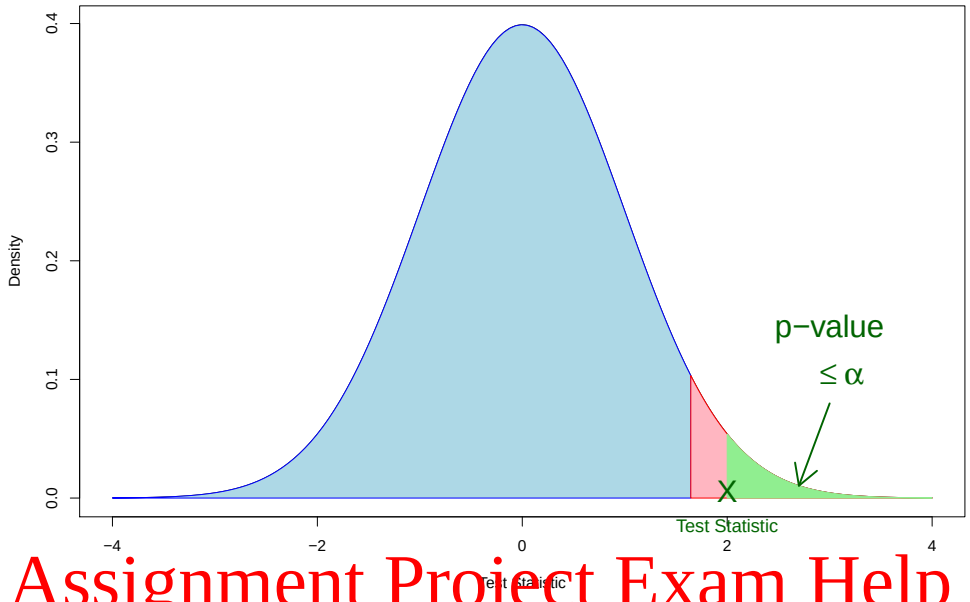
Rejection regions: One-sided lower tail test



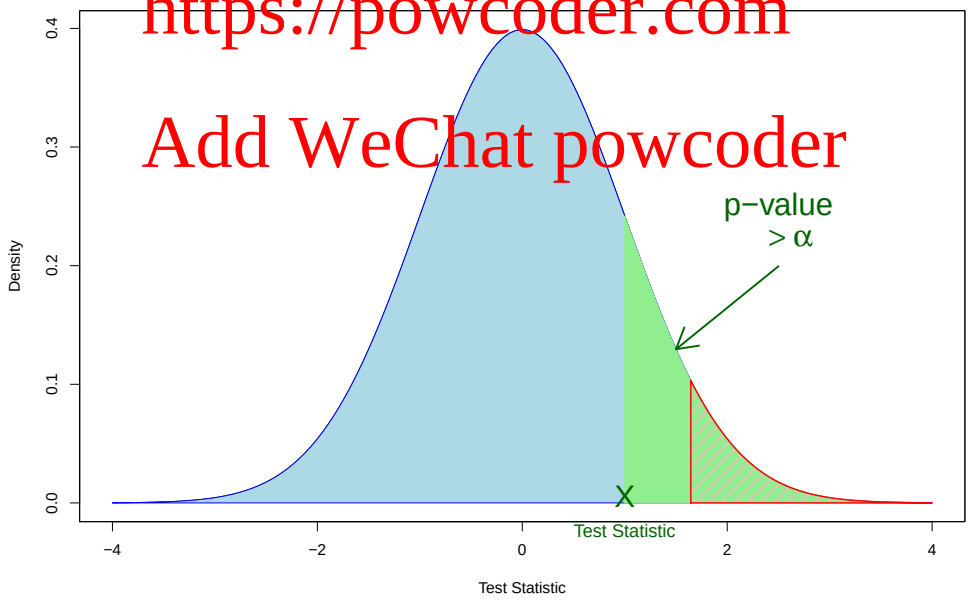
Potential results from one-sided tests

Let's focus on the p-value defined by the upper tail probability, i.e. the probability of getting that value of test statistic or something more extreme under the null hypothesis.

With p-value$\leq \alpha$:



With p-value$> \alpha$:



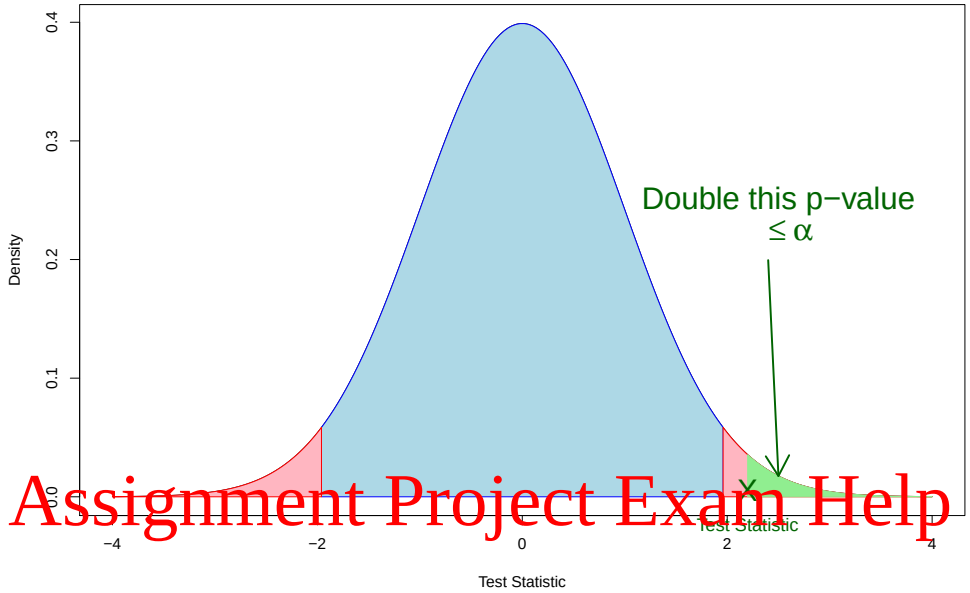
If the test statistic is in the red rejection region area, the p-value is less than significance level α.

If the test statistic is in the blue non-rejection region area, the p-value is greater than significance level α.

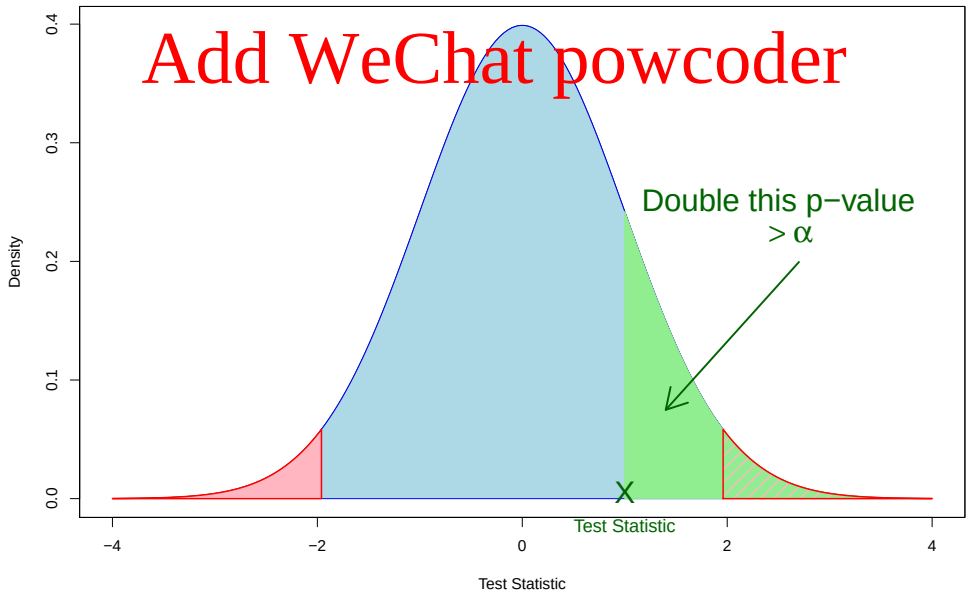
Potential Results From Two-sided Test Results

In general for a two-sided test, the significance level cut-off is split between the two tails ($\alpha/2$ per tail). Thus, the p-value is commonly defined by doubling the minimum of the upper and lower tail p-values.

With p-value$\leq \alpha$:



With p-value$> \alpha$:



If the test statistic is in one of the two $\alpha/2$ red rejection regions, the p-value is less than the significance level α.

If the test statistic is in the blue non-rejection region area, the p-value is greater than the significance level α.

Critical values & decision rule

The cut-off value(s) for the test statistic which define the rejection regions are defined as the *critical value(s)* of the test.

These are often denoted:

- X_α and $X_{1-\alpha}$ for a one sided test for the lower and upper tail respectively
- $X_{\alpha/2}$ and $X_{1-\alpha/2}$ for a two-sided test

where the subscript represents the relevant upper or lower tail probability.

Therefore, the decision rule can equivalently be based on the:

- p-value; or the
- critical value,

and both give the same conclusions.

Possible errors when testing

Consider the possible outcomes of hypothesis testing:

	H_0 **Not Rejected**	H_0 **Rejected**
H_0 **true**	Right Decision $(1-\alpha)$	Type I Error (α)
H_1 **true**	Type II Error (β)	Right Decision $(1-\beta)$

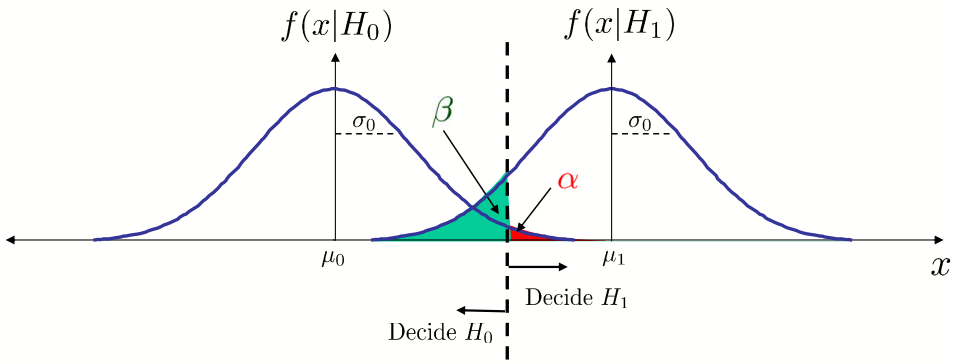
Type I error: To wrongly reject null when it is true, often called *false positive*.

Type II error: To wrongly not reject the null when it is false, *false negative*.

A way to remember these: the fable of the boy who called wolf: A shepherd boy looked after all the sheep of the village, watching them in the hills. For fun, he came down to the village many times and shouted 'wolf, wolf', and the villagers would go into the hills to protect the sheep, only to find the boy was lying (the villagers made a Type I error - the truth was that there was no wolf). One day a wolf really did arrive and so the boy ran down to the village and cried 'wolf, wolf', but the villagers ignored him and the wolf ate all their sheep (the villagers made a Type II error).

Start of lecture 9

Significance and power



The *power of the test* is the probability of correctly rejecting the null, $(1 - \beta)$.

The *significance level* is the probability of making a Type I error (α), and the *confidence level* is $(1 - \alpha)$.

It's important to realise that:

- rejecting H_0 does *not* mean that we have proved H_0 is false
- conversely, failing to reject H_0 does *not* mean that we have proved H_0 to be true.

As a result, failure to reject H_0 represents an inconclusive result.

Ideally, a statistical test should have small α and small β.

In practice, it is easy to control α but not β.

There is always a trade-off between the power of a test and the likelihood of false positives.

There is some statistical testing controversy ...

There is some controversy over hypothesis testing (see Wikipedia on "Statistical Hypothesis Testing"), especially over interpretation and language.

In the **Fisher** school you either 'Reject' or 'Do Not Reject' the null, you never accept anything!

Under the **Neyman-Pearson** school you accept H_1 in the rejection region and you accept H_0 in the do not reject region.

However, although you 'accept' one of the two hypotheses, this doesn't mean it is in fact true, only that you carry on under the assumption that it is - so it effectively becomes your working hypothesis of the truth.

Parametric and nonparametric tests

- **Parametric Tests**: rely on theoretical distribution of the test statistic under the null hypothesis, typically based on asymptotic theory (e.g. Central Limit Theorem) or assumptions about the distribution of the sample data (e.g., normal population)
- **Nonparametric Tests**: do not necessarily assume that the sample data are drawn from any particular distribution

New Section: Permutation tests

The most common application of these nonparametric tests is for comparing two samples.

Two-sample permutation test

Suppose we have samples collected from two different populations, with values denoted by

$$X = \{X_1, \ldots, X_n\}$$

and

$$Y = \{Y_1, \ldots, Y_m\}.$$

We assume:

- the two samples are independent of each other; and
- the data in each individual sample are iid (independent and identically distributed), meaning that each set of data points are randomly sampled from a single population.

Hypotheses are framed in terms of population values, so if X is a random sample from population (distribution) F_X, and Y from F_Y, then in general, we want to test the hypotheses:

$$H_0 : F_X = F_Y \qquad \text{against} \qquad H_1 : F_X \neq F_Y$$

When samples are collected, they are 'labelled' with the name of the population they are sampled from.

For example: Suppose you buy 4 regular and 3 expensive premium lightbulbs, and you want to test whether the premium lightbulbs have a longer lifetime than the regular ones.

The lifetime measured in 10,000 hours are:

- Regular bulbs: 90, 11, 94, 118
- Expensive bulbs: 197, 107, 752

Here the 'labels' are 'Regular' and 'Expensive', and we could instead have written the data as:

90	Regular
11	Regular
197	Expensive
94	Regular
107	Expensive
752	Expensive
118	Regular,

so that each observation is individually labelled.

Returning to the general case: under the null hypothesis, both samples X and Y come from the same population.

Therefore, under the null we can consider the combined sample (X, Y) as coming from a single population.

Thus, under the null hypothesis, any labels are entirely arbitrary.

Therefore, we can consider randomly rearranging (or 'permuting') the labels in order to evaluate the distribution of the test statistic.

So it is the *labels* that are permuted, and we don't touch the values of the observations. And after 'permuting' we have exactly the same number of observations as before, and the values are exactly the same, but the labels have been rearranged.

We can visualise this using the software InzightVIT (downloadable from https://inzight.nz) using the data in the file packaging.csv.

Start of lecture 10

The permutation test distribution

Under the null hypothesis, any statistical function of the original data (X, Y) will have same distribution as when the same function is applied to the permuted dataset (X^*, Y^*).

For example, under the null the distribution of $\bar{X} - \bar{Y}$ will be the same as the distribution of $\bar{X}^* - \bar{Y}^*$.

The '**sampling distribution**' of the test statistic under the null hypothesis is therefore found by calculating that statistic for a very large number of permuted datasets (X^*, Y^*), where the labels have been randomly allocated.

The actual test statistic obtained from the original data (X, Y) is then compared to this sampling distribution.

If the value of the test statistic from (X, Y) is 'unusual' in the permutation sampling distribution of (X^*, Y^*), then this provides evidence against the null hypothesis.

Why use permutation tests?

Permutation tests are often used as a substitute for standard parametric tests (like those studied in STAT101) when the usual assumptions might fail, or when the sample size is sufficiently small that asymptotic theory (e.g. Central Limit Theorem) may not hold.

For example, a permutation test may be used to replace a two sample t-test when the normal population assumption doesn't hold.

You can still use the t-test statistic to compare the two groups when the population is not normal, but the problem is that without the normal assumption the distribution of the t test statistic is unknown - it is not likely to follow a Student-t distribution as was used in STAT101.

Permuting the original values many times gives us a distribution of the t-test statistic under the null hypothesis, without assuming the population is normal.

First example: Doing a two-sample permutation test in R

Suppose you buy 4 regular and 3 expensive lightbulbs.

You want to test whether the expensive lightbulbs have a longer lifetime than the regular ones:

- this would be a **one sided (upper tail) test**.

We can't use a parametric test because:

- such lifetimes are not normally distributed and can be highly skewed (the exponential distribution is commonly used for lifetimes), and further
- the samples are sufficiently small that it is not appropriate to assume any asymptotic models (because, for example, the Central Limit Theorem may not hold).

Therefore we'll use a permutation test instead of a parametric t-test, but we still use the t-test statistic!!!

The observed lifetimes measured in units of 10,000 hours are:

```
x = c(90, 11, 94, 118)
y = c(197, 107, 752)
```

The steps in our permutation test are:

```
# Sample size
nx = length(x)
ny = length(y)

# Combined dataset and corresponding labels
xy = c(x, y)
xylabels = c(rep("x", nx), rep("y", ny))

# t-test statistic for original sample
teststat = (mean(y) - mean(x)) / sqrt(var(x)/nx + var(y)/ny)
teststat
```

```
## [1] 1.348412
```

```
# Number of permutation samples to take
N = 10000
set.seed(1)

# Create vector to store permutation t-statistics
tstat = numeric(N)

for (i in 1:N) {
  newlabels = sample(xylabels, replace = F) # sample labels without replacement

  # permute the data and obtain the t-test statistic
  newx = xy[newlabels == "x"]
  newy = xy[newlabels == "y"]
  tstat[i] = (mean(newy) - mean(newx))/sqrt(var(newx)/nx + var(newy)/ny)
}

# Calculate upper tail one-sided p-value
# (proportion of permuted test statistics above observed value)
pvalue = mean(tstat >= teststat)
pvalue
```

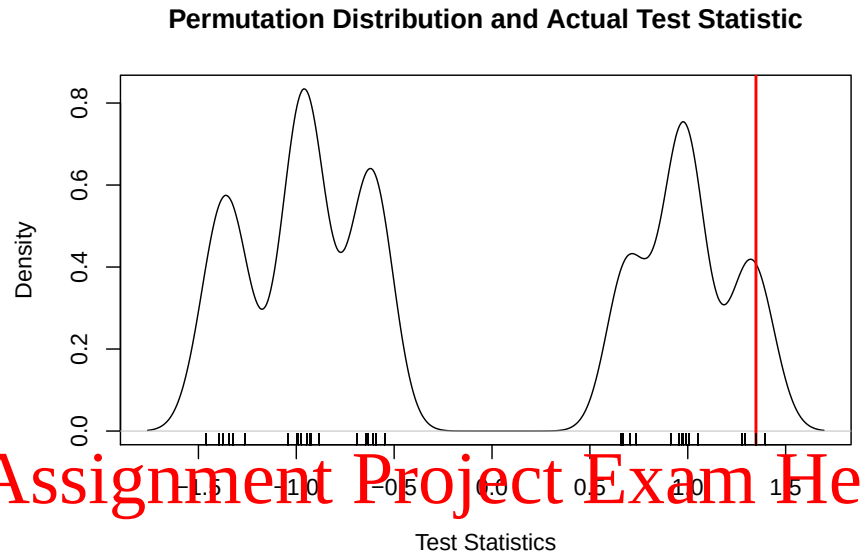
```
## [1] 0.0562
```

```
# Output mean difference, t-test statistic and p-value
print(c(mean(y) - mean(x), teststat, pvalue))
```

```
## [1] 273.750000   1.348412   0.056200
```

Now we can plot the 1000 test statistics (to get the t-distribution) and compare the original test statistic against this distribution:

```
# Plot permutation sampling distribution using kernel density estimate
plot(density(tstat, bw=0.1), xlab="Test Statistics",
     main="Permutation Distribution and Actual Test Statistic")
abline(v = teststat, lwd = 2, col = "red")
rug(tstat)
```



Notice that the sampling distribution of the test statistic is nothing like the usual t distribution.

The observed test statistic (vertical line) is out in the upper tail, so is unusual under the null hypothesis.

The conclusions of this test are:

The above is a one-sided upper tail test with hypotheses:

- $H_0 : \mu_Y \leq \mu_X$ or equivalently $\mu_Y - \mu_X \leq 0$
- $H_1 : \mu_Y - \mu_X > 0$

The observed difference in means is $\mu_Y - \mu_X = 273.5$.

The test statistic (`teststat`) is 1.348412.

The p-value is 0.056200.

There is therefore insufficient evidence to reject the null hypothesis at the $\alpha = 5\%$ level.

The critical value for a permutation test can also be obtained using the following code:

```
alpha = 0.05

# Sort permutation t-statistics and extract upper (1-alpha)% quantile
tstat = sort(tstat)
tcrit = tstat[ceiling((1 - alpha) * N)]
tcrit
```

```
## [1] 1.348412
```

Second example: Two-sample, two-sided permutation test

Consider making the above example a two-sided test with hypotheses:

- $H_0 : \mu_Y = \mu_X$ or equivalently $\mu_Y - \mu_X = 0$
- $H_1 : \mu_Y - \mu_X \neq 0$

The observed difference in means and the test statistic are the same.

But the p-value will be doubled to give 0.1126, so there is insufficient evidence to reject the null hypothesis at $\alpha = 5\%$ (or even at the $\alpha = 10\%$ level).

Permutation test p-values revisited

The above formula for calculating the p-value is not quite correct, since when carrying out the hypothesis test, you are also assuming the **observed test statistic is actually from the null distribution**.

Thus this datum should be included in calculating the p-value, and we need to add one to the numerator and denominator:

```
# Calculate correct upper tail one-sided p-value
pvalue = (sum(tstat >= teststat) + 1) / (N + 1)
pvalue
```

```
## [1] 0.05629437
```


However, for most practical purposes this usually makes no difference to the conclusion from the test, except in 'marginal cases' or where the sample size is small.

'*Marginal significance*' is terminology used when the p-value is only a little smaller than the significance level.

Third example: Two-sample, one-sided, permutation test with more data

The first example used a very small dataset, with 7 observations in total, to do a one-sided test. Here we will repeat this but using a larger dataset (now with a total of 30 observations, but unequal sample sizes).

The earlier examples used real data for lifetimes where lifetimes are known to follow an exponential distribution. This time we will simulate some data from two different exponential distributions.

```
# Simulate larger samples from Exponential populations with lambda=1 and lamba=0.5
nx = 10
ny = 20
set.seed(1)
x = rexp(nx, rate = 1)
y = rexp(ny, rate = 0.5)

# Combined dataset and corresponding labels
xy = c(x,y)
xylabels = c(rep("x", nx), rep("y", ny))

# t-test statistic for original sample
teststat = (mean(y) - mean(x)) / sqrt(var(x)/nx + var(y)/ny)

# Number of permutation samples to take
N = 1000

# Create vector to store permutation t-statistics
tstat = numeric(N)
for (i in 1:N) {
  newlabels = sample(xylabels, replace = F) # sample labels without replacement
```

```
  # permute data and obtain t-test statistic
  newx = xy[newlabels == "x"]
  newy = xy[newlabels == "y"]
  tstat[i] = (mean(newy) - mean(newx)) / sqrt(var(newx)/nx + var(newy)/ny)
}

# Calculate upper tail one-sided p-value
pvalue = (sum(tstat >= teststat) + 1) / (N + 1)

# Sort permutation t-statistics and extract upper (1-alpha)% quantile
alpha = 0.05
tstat = sort(tstat)
tcrit = tstat[ceiling((1 - alpha) * N)]

# Output mean different, test statistic, critical value and p-value
print(c(mean(y) - mean(x), teststat, tcrit, pvalue))
```

```
## [1] 1.56737386 2.67828573 2.04469990 0.00999001
```

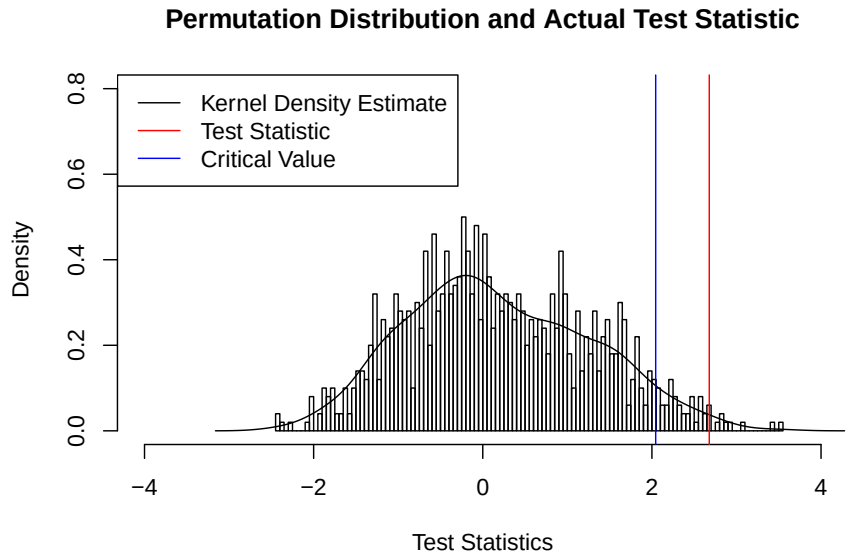
Conclusions:

The p-value of 0.01 is less than the significance level (0.05), so there is evidence that the mean of the $Y \sim Exponential(\lambda = 0.5)$ population is greater than for $X \sim Exponential(\lambda = 1)$.

The sample difference in means is 1.57.

The test statistic is 2.67 which is well above the critical value of 2.04.


```
# Plot permutation sampling distribution using kernel density estimate
hist(tstat, breaks = 100, freq = FALSE, xlab = "Test Statistics",
     ylim = c(0, 0.8), xlim = c(-4,4),
     main = "Permutation Distribution and Actual Test Statistic")
lines(density(tstat, bw = 0.25))
abline(v = teststat, col = "red")
abline(v = tcrit, col = "blue")
legend("topleft",
       legend = c("Kernel Density Estimate", "Test Statistic", "Critical Value"),
       lty = 1, col = c("black", "red", "blue"))
```



Notice that with larger samples from each population that the sampling distribution for the test statistic is now much smoother and close to symmetric.

In fact, it is now close to the Student-t distribution, which is assumed in the classical parametric testing approach demonstrated in STA1101.

Classical t-test in R, using the same data as the third example

It is easy to carry out a classical t-test in R:

```
# This is a one sided t-test with mean(y) > mean(x) as alternative hypothesis
# (hence "greater" option)
t.test(y, x, alternative = "greater")
```



```
##
##  Welch Two Sample t-test
##
## data:  y and x
## t = 2.6783, df = 26.274, p-value = 0.006297
## alternative hypothesis: true difference in means is greater than 0
## 95 percent confidence interval:
##  0.5696045       Inf
## sample estimates:
## mean of x mean of y
## 2.4099961 0.8426222
```

In this example, we have **unequal sample sizes** and **unequal variances**.

Under these criteria it is called 'Welch's two sample t-test'.

The t-test statistic for evaluating whether the population means are different is:

$$t = \frac{\overline{Y} - \overline{X}}{s_{\overline{Y}-\overline{X}}} \tag{1}$$

where the standard deviation of the difference in the means:

$$s_{\overline{Y}-\overline{X}} = \sqrt{\frac{s_X^2}{n_X} + \frac{s_Y^2}{n_Y}}. \tag{2}$$

Note that s_*^2 are the unbiased estimators of the variance of the two samples and n_* are the sample sizes.

Under the null hypothesis, the distribution of the test statistic is approximately a Student-t distribution with the degrees of freedom calculated using the 'Welch-Satterthwaite equation':

$$\mathrm{df} = \frac{(s_X^2/n_X + s_Y^2/n_Y)^2}{(s_X^2/n_X)^2/(n_X - 1) + (s_Y^2/n_Y)^2/(n_Y - 1)}. \tag{3}$$

Let's check out the classical t-test in R to make sure we understand how all the output is calculated:

```
# One sided t-test with mean(y) > mean(x) as alternative hypothesis
t.test(y, x, alternative = "greater")
```

```
##
##  Welch Two Sample t-test
##
## data:  y and x
## t = 2.6783, df = 26.274, p-value = 0.006297
## alternative hypothesis: true difference in means is greater than 0
## 95 percent confidence interval:
##  0.5696045       Inf
## sample estimates:
## mean of x mean of y
## 2.4099961 0.8426222
```


The test statistic $t = 2.6783$ is the same as that calculated above using equation (1).

The degrees of freedom in output is calculated using equation (3):


```
# degrees of freedom for Welch's test
ttestdf = (var(x)/nx+var(y)/ny)^2 / ((var(x)/nx)^2/(nx-1)+(var(y)/ny)^2/(ny-1))
ttestdf
```

```
## [1] 26.2741
```


The standard deviation is calculated using equation (2):

```
# standard deviation for Welch's test
ttestsd = sqrt((var(x)/nx + var(y)/ny))
ttestsd
```

```
## [1] 0.5852153
```

The t-test statistic is calculated using equation (1):

```
# $t$-test statistic for Welch's test
tteststat = (mean(y)-mean(x)) / ttestsd
tteststat
```

```
## [1] 2.678286
```

The p-value for one-sided upper tail test is given by upper tail probability from Student-t distribution:

```
# p-value for Welch's test
pt(tteststat, df = ttestdf, lower.tail = FALSE)
```

```
## [1] 0.00629742
```

It is also worth noting that the permutation test p-value of 0.00999001 and that from Welch's test 0.00629742 are similar, demonstrating that the Student-t approximation is fairly good even for such small samples from non-normal populations (ie the t-test is robust against non-normality).

Start of lecture 11

Permutation test exact p-values

The sampling distribution from the original small samples on page 14 is very different from that for the larger samples on page 17.

In fact, for the small samples it is not even close to being symmetric.

Why is so oddly behaved? The key reason is that:

- it is a **discrete distribution**, so that
- only certain permutations of the sample are possible, so that
- there are limited values that the test statistic and p-value can take.

This motivates the idea of an '**exact test**' where only the limited number of permutations of the actual sample data (from each group) are considered.

We could do this instead of doing many randomly chosen permutations for which many will simply be repeated (as happened in the first example).

The hypothesis test could use combinations only

The test statistic in the first example will be the same for many permutations, e.g.

- $X = \{90, 11, 94, 197\}$ and $Y = \{118, 107, 752\}$
- $X = \{90, 11, 197, 94\}$ and $Y = \{118, 107, 752\}$
- $X = \{90, 197, 11, 94\}$ and $Y = \{118, 107, 752\}$
- etc.

will all give same test statistic, since we are comparing $\bar{X}$ to $\bar{Y}$.

Therefore, only the unique permutations (in terms of which observations are allocated to each group) are relevant.

These unique permutation are called the **combinations**.

For two groups the number of combinations is given by:

$$\left(\begin{array}{c} n_x + n_y \\ n_x \end{array} \right) = \frac{(n_x + n_y)!}{n_x!\, n_y!}.$$

'$n_x + n_y$ choose n_x'.

However, these can still be referred to as 'permutation tests'.

Note that if the test statistic does depend on the orderings, then this simplification will not hold. So the simplification doesn't hold in general.

In our small example there are $\left(\begin{array}{c} 7 \\ 4 \end{array} \right) = 35$ combinations.

Fourth example: An 'Exact' two-sample test

The `choose` function gives the total number of combinations:

```
x = c(90, 11, 94, 118)
y = c(197, 107, 752)

# Sample size
nx = length(x)
ny = length(y)

# Combined dataset and corresponding labels
xy = c(x, y)
xylabels = c(rep("x", nx), rep("y", ny))

# t-test statistic for original sample
teststat = (mean(y) - mean(x)) / sqrt(var(x)/nx + var(y)/ny)

# Number of unique permutation samples to take
N = choose(nx + ny, nx)
N
```

```
## [1] 35
```

So there are 35 combinations for this exact test. R has a function `combn` which will produce a list of these unique combinations (selecting nx from $nx + ny$ possibilities): check the help file using `?combn`.

An aside on how this function works:


```
letters[1:4]
```

```
## [1] "a" "b" "c" "d"
```



```
combn(letters[1:4], 2) # all combinations of a,b,c,d taken 2 at a time
```

```
##      [,1] [,2] [,3] [,4] [,5] [,6]
## [1,] "a"  "a"  "a"  "b"  "b"  "c"
## [2,] "b"  "c"  "d"  "c"  "d"  "d"
```

which gives 6 unique combinations.

```
1:4
```

```
## [1] 1 2 3 4
```

```
combn(1:4, 2) # all combinations of 1,2,3,4 taken 2 at a time
```

```
##      [,1] [,2] [,3] [,4] [,5] [,6]
## [1,]    1    1    1    2    2    3
## [2,]    2    3    4    3    4    4
```

```
combn(4, 2) # all combinations of 1,2,3,4 taken 2 at a time (the same as before)
```

```
##      [,1] [,2] [,3] [,4] [,5] [,6]
## [1,]    1    1    1    2    2    3
## [2,]    2    3    4    3    4    4
```

Returning to our test, we can use `combn(7, 4)` to tell us the observations to label as x, with the remainder labelled as y:

```
# Obtain all unique permutations
nx + ny
```

```
## [1] 7
```

```
nx
```

```
## [1] 4
```

```
allpermutations = combn(nx + ny, nx) # all combinations of 1:7 taken 4 at a time
allpermutations
```

```
##      [,1] [,2] [,3] [,4] [,5] [,6] [,7] [,8] [,9] [,10] [,11] [,12] [,13]
## [1,]    1    1    1    1    1    1    1    1    1     1     1     1     1
## [2,]    2    2    2    2    2    2    2    2    2     2     3     3     3
## [3,]    3    3    3    3    4    4    4    5    5     6     4     4     4
## [4,]    4    5    6    7    5    6    7    6    7     7     5     6     7
##      [,14] [,15] [,16] [,17] [,18] [,19] [,20] [,21] [,22] [,23] [,24]
## [1,]     1     1     1     1     1     1     1     2     2     2     2
## [2,]     3     3     3     4     4     4     5     3     3     3     3
## [3,]     5     5     6     5     5     6     6     4     4     4     5
## [4,]     6     7     7     6     7     7     7     5     6     7     6
##      [,25] [,26] [,27] [,28] [,29] [,30] [,31] [,32] [,33] [,34] [,35]
## [1,]     2     2     2     2     2     2     3     3     3     3     4
## [2,]     3     3     4     4     4     5     4     4     4     5     5
## [3,]     5     6     5     5     6     6     5     5     6     6     6
## [4,]     7     7     6     7     7     7     6     7     7     7     7
```

End of the aside

Recall that the observations are:

```
xy
```

```
## [1]  90  11  94 118 197 107 752
```

We can now do the 'exact' test by calculating the test statistics for these 35 unique permutations only:

```
# Number of unique permutation samples to take
N = choose(nx + ny, nx)
N
```

```
## [1] 35
```

```
# Create vector to store permutation t-statistics
tstat = numeric(N)

for (i in 1:N) {
  # permute data and obtain t-test statistic
  newx = xy[allpermutations[, i]]
  newy = xy[-allpermutations[, i]]
  tstat[i] = (mean(newy) - mean(newx)) / sqrt(var(newx)/nx + var(newy)/ny)
}

# Calculate upper tail one-sided p-value
# (proportion of permuted test statistics above observed value)
pvalue = mean(tstat >= teststat)

# Sort permutation t-statistics and extract upper (1-alpha)% quantile
alpha = 0.05
```

```
tstat = sort(tstat)
tcrit = tstat[ceiling((1 - alpha) * N)]

# Output mean different, test statistic, critical value and p-value
print(c(mean(y) - mean(x), teststat, tcrit, pvalue))
```

```
## [1] 273.75000000   1.34841208   1.34841208   0.05714286
```

Notice that the p-value 0.05714286 is close to the 0.056200 obtained by the repeated permutations approach, on page 14.

Monte Carlo and exact sampling

The two sampling approaches for getting the test statistics under the permutation scheme are referred to as:

- Monte Carlo sampling - random samples from possible permutations
- Exact test - only unique permutations used.

The former is an approximation to the latter, but is more practical for large samples which lead to too many permutations.

Exact p-values frequently come up in other areas of statistics that have discrete data, such as binomial distributions (testing for a proportion) and contingency tables.

Back to the example: An exact two-sample test

The unique values of the test statistic are given in the following table:


```
table(tstat)
```

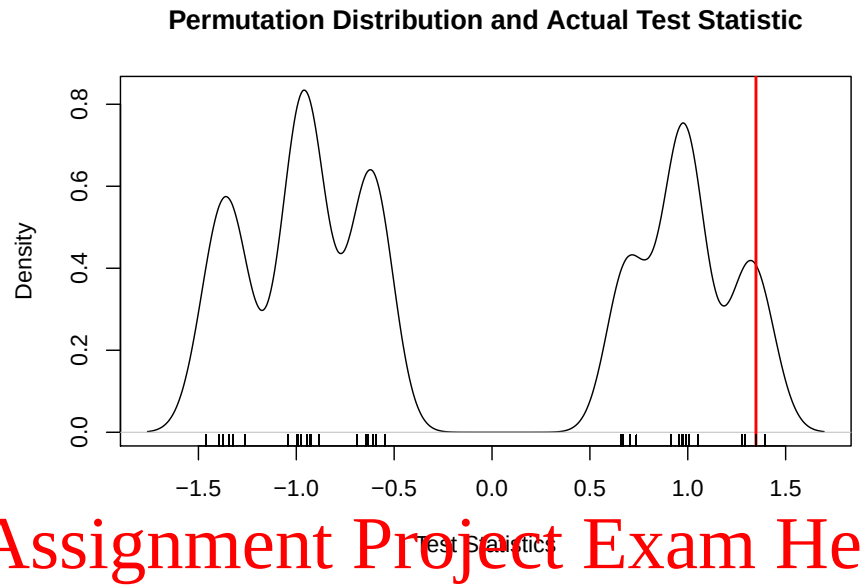


```
## tstat
##  -1.46012848312594  -1.39582759622129  -1.37666316011324
##                  1                  1                  1
##  -1.34261724030786  -1.32401252310949  -1.26435428150407
##                  1                  1                  1
##  -1.04313743925751 -0.998231311458993 -0.989246691069474
##                  1                  1                  1
## -0.973759375119742 -0.946744011035543 -0.931216941089718
##                  1                  1                  1
## -0.923810953579016  -0.88200187821866 -0.691211289507945
##                  1                  1                  1
## -0.645502648506479 -0.631610122091557 -0.607035509594958
##                  1                  1                  1
## -0.593276520708821 -0.548674435835161  0.657956193884955
##                  1                  1                  1
##  0.669239975188824  0.706253147387712  0.737987095868991
##                  1                  1                  1
##  0.912531923383827  0.955279265069289  0.968717372641221
##                  1                  1                  1
##  0.977926215782566  0.992123649633063   1.00587512834482
##                  1                  1                  1
##   1.05121904971437   1.27696549603357   1.29352200921875
##                  1                  1                  1
##    1.3484120784389   1.39620685221911
##                  1                  1
```

and these are the values of the 35 tick-marks in the rug on the plot on page 14.

Here's the plot from the full permutation test again (from page 14):

```
# Plot permutation sampling distribution using kernel density estimate
plot(density(tstat, bw=0.1), xlab="Test Statistics",
     main="Permutation Distribution and Actual Test Statistic")
abline(v = teststat, lwd = 2, col = "red")
rug(tstat)
```



Note that the critical value of the test statistic is one of these unique values.

What p-values are possible for an exact test?

Only a limited set of p-values are possible due to the discrete nature of the exact test. There are 35 combinations:


```
length(table(tstat))
```

```
## [1] 35
```

so a maximum of 35 p-values are possible.

Under the null hypothesis, p-values are uniformly distributed[1]. Therefore we know that the 35 p-values in this exact test take the values $\{i/35 : i = 1 \ldots, 35\}$ at each of the test statistic values:

```
(1:35)/35
```

```
##  [1] 0.02857143 0.05714286 0.08571429 0.11428571 0.14285714 0.17142857
##  [7] 0.20000000 0.22857143 0.25714286 0.28571429 0.31428571 0.34285714
## [13] 0.37142857 0.40000000 0.42857143 0.45714286 0.48571429 0.51428571
## [19] 0.54285714 0.57142857 0.60000000 0.62857143 0.65714286 0.68571429
## [25] 0.71428571 0.74285714 0.77142857 0.80000000 0.82857143 0.85714286
## [31] 0.88571429 0.91428571 0.94285714 0.97142857 1.00000000
```

Notice that the p-value of 0.05714286 for this example is second in the list.

However, you should be aware this is the maximum possible number of p-values. If the test statistic is not unique for each combination then a reduced set of p-values is possible.

[1] The reason for this is really the definition of alpha as the probability of a type I error. We want the probability of rejecting a true null hypothesis to be alpha, we reject when the observed p-value $< \alpha$, the only way this happens for any value of alpha is when the p-value comes from a uniform distribution. The whole point of using the correct distribution (normal, t, f, chisq, etc.) is to transform from the test statistic to a uniform p-value. If the null hypothesis is false then the distribution of the p-value will (hopefully) be more weighted towards 0.

The `Pvalue.norm.sim` and `Pvalue.binom.sim` functions in the `TeachingDemos` package for R will simulate several data sets, compute the p-values and plot them to demonstrate this idea, if you would like to investigate this further.

For example, if the dataset is $X = \{90, 90, 94, 118\}$ and $Y = \{197, 107, 752\}$ then the duplicated data values will lead to non-unique values for the test statistics.

p-values of 1! How?

What does a p-value of 1.0 mean?

When doing exact testing the p-values can be 1, meaning that your data is as consistent as possible with the null hypothesis.

This could happen, for example, if the null hypothesis is that the mean of X is at least the mean of Y (as it was in our small sample, one-sided test), and you observe $x_i > y_j$ for every combination of i and j.

Fifth example: p-value of 1

For example, if the dataset were $X = \{197, 107, 752, 118\}$ and $Y = \{90, 11, 94\}$.

So, as stated in the first example on page 14, which was a one-sided upper tail test with hypotheses:

- $H_0 : \mu_Y \leq \mu_X$ or equivalently $\mu_Y - \mu_X \leq 0$
- $H_1 : \mu_Y - \mu_X > 0$.

```
x = c(197, 107, 752, 118)
y = c(90, 11, 94)

# Sample size
nx = length(x)
ny = length(y)

# Combined dataset and corresponding labels
xy = c(x, y)
xylabels = c(rep("x", nx), rep("y", ny))

# t-test statistic for original sample
teststat = (mean(y) - mean(x)) / sqrt(var(x)/nx + var(y)/ny)

# Number of unique permutation samples to take
N = choose(nx + ny, nx)

# Obtain all unique permutations
allpermutations = combn(nx + ny, nx)

# Create vector to store permutation t-statistics
tstat = numeric(N)

for (i in 1:N) {
  # permute data and obtain t-test statistic
  newx = xy[allpermutations[, i]]
  newy = xy[-allpermutations[, i]]
  tstat[i] = (mean(newy) - mean(newx)) / sqrt(var(newx)/nx + var(newy)/ny)
}

# Calculate upper tail one-sided p-value
# (proportion of permuted test statistics above observed value)
pvalue = mean(tstat >= teststat)
pvalue
```

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

There is no evidence against the null.

Permutation test for correlation

In addition to two independent samples being tested with permutation tests, we might wish to test paired data $(X_1, Y_1), (X_2, Y_2), \ldots, (X_n, Y_n)$.

If the data can be assumed to come from a bivariate normal distribution (i.e., X and Y random variables are both normal, but might be correlated), then the classical parametric test is another t-test where

- $H_0 : \rho = 0$
- Test statistic $t = r\sqrt{\frac{(n-2)}{(1-r^2)}}$ which follows a Student-t distribution on $n-2$ degrees of freedom.

This test is implemented in R with the `test.cor` function.

If the Student-t distribution is inappropriate (i.e., the populations are not normal), can we do a permutation test?

The null hypothesis is that there is no correlation, which typically means no relationship between X and Y.

As usaual, the sampling distribution of the test statistics under the null is found by calculating that statistic for a large number of permuted datasets (X_*, Y_*) where the labels have been randomly allocated.

We could permute the indices on both the X and Y values, but it is sufficient to permute just one of the datasets, since this permutation will destroy any relationship/correlation between X and Y.

This can be done using the basic steps we have already followed in other examples:

1. use the `sample` function to permute the data;
2. recompute the correlation from the permuted data; and
3. see what proportion of the correlations are at least as large as the actual value in the original dataset.

Finally: Matched pairs t-test (or paired t-test) for the mean

The same permutation approach can be used for paired data, and is useful if you don't wish to assume normal populations.

In a matched pairs test, the difference $D_i = X_i - Y_i$, $i = 1, \ldots, n$, between the two paired observations is calculated. The null hypothesis is that $\overline{D} = 0$, and the alternative is that the mean of these differences $\overline{D}$ is significantly different to 0.

- $H_0 : \overline{D} = 0$
- $H_1 : \overline{D} \neq 0$

In correlation testing we could permute individual datapoints to destroy any relationship between X and Y. With paired data, however, we must retain the relationship between X and Y, so **we have to permute each of the n pairs together** to obtain the sampling distribution of the test statistic.

Sixth example: Matched pairs permutation test

```
# Simulate correlated samples from standard normals
n = 20
set.seed(2)
x = rnorm(n)
y = rnorm(n, 0.5*x + 0.5)
```

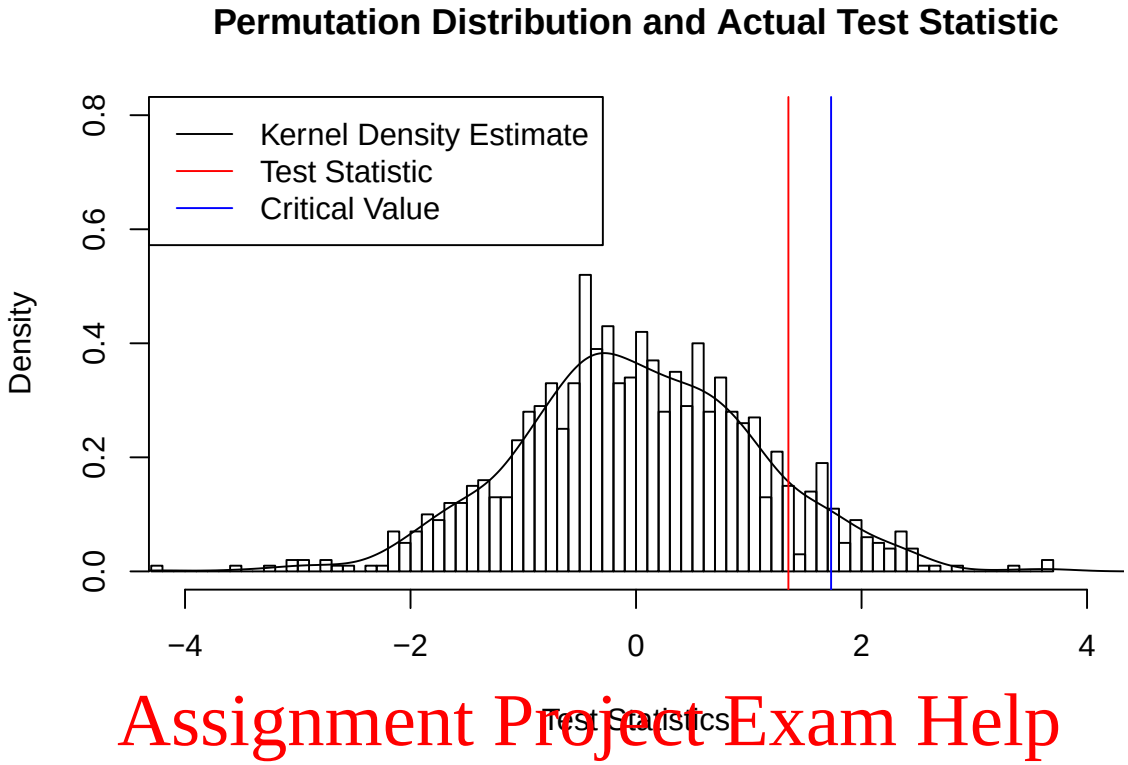
```r
# t-test statistic for original sample extracted from t.test() function
teststat = t.test(y, x, alternative = "greater", paired = TRUE)$statistic

# Number of permutation samples to take
N = 1000

# Create vector to store permutation t-statistics
tstat = numeric(N)
for (i in 1:N) {
  # decide which ones to interchange
  newlabels = sample(c(TRUE, FALSE), size = n, replace = TRUE)

  # permute the pairs where newlabels = TRUE, and obtain t-test statistic
  newx = x
  newy = y
  newx[newlabels] = y[newlabels]
  newy[newlabels] = x[newlabels]
  tstat[i] = t.test(newy, newx, alternative = "greater", paired = TRUE)$statistic
}

# Calculate upper tail one-sided p-value and critical value
pvalue = (sum(tstat >= teststat) + 1) / (N + 1)
alpha = 0.05
tstat = sort(tstat)
tcrit = tstat[ceiling((1 - alpha) * N)]

print(c(mean(y) - mean(x), teststat, tcrit, pvalue))
```



```
##                 t
## 0.4019877 1.3513186 1.7299853 0.0989011
```



```r
# Plot the permutation sampling distribution using kernel density estimate
hist(tstat, breaks = 100, freq = FALSE, xlab = "Test Statistics",
     ylim = c(0, 0.8), xlim = c(-4, 4),
     main = "Permutation Distribution and Actual Test Statistic")
lines(density(tstat, bw = 0.25))
abline(v = teststat, col = "red")
abline(v = tcrit, col = "blue")
legend("topleft",
       legend = c("Kernel Density Estimate", "Test Statistic", "Critical Value"),
       lty = 1, col = c("black", "red", "blue"))
```



Notice in the R code that the t-test statistic is extracted from the `t.test` function, rather than manually calculated.

The permutation test p-value, 0.0989011, is similar to that of the parametric test, 0.09623:

```
t.test(y, x, alternative = "greater", paired = TRUE)
```



```
##
##  Paired t-test
##
## data:  y and x
## t = 1.3513, df = 19, p-value = 0.09623
## alternative hypothesis: true difference in means is greater than 0
## 95 percent confidence interval:
##  -0.1123914        Inf
## sample estimates:
## mean of the differences
##               0.4019877
```

End of notes for Permutation tests. Next is 'Power'.