DATA2002

Permutation tests

Garth Tarr

THE UNIVERSITY OF SYDNEY

Lady tasting tea

Permutation tests for population means

Lady tasting tea

Lady tasting tea

Given a cup of tea with milk, a lady claims she can discriminate as to whether milk or tea was first added to the cup.

> How could we test this claim?
>
> What information would we need?

Lady tasting tea

Fisher proposed a preparing 8 cups of tea

- 4 cups where tea was added before milk
- 4 cups where milk was added before tea

The lady would then be randomly given the cups of tea and asked to identify the 4 where tea was added before milk.

We would then need to record:

- Which cups had tea or milk added first (**truth**).
- Which cups the lady claimed had tea or milk added first (**predicted**).

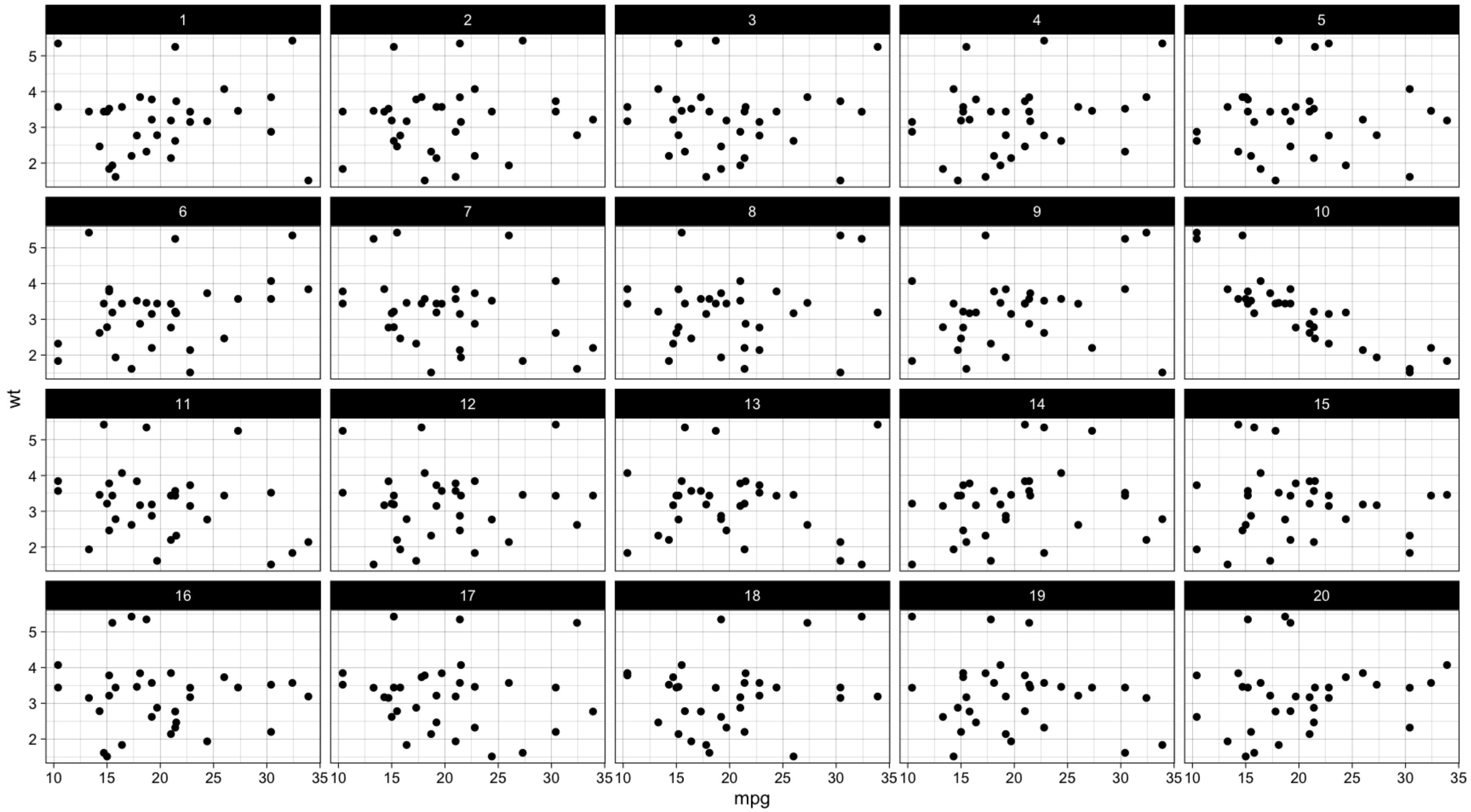
Ronald Fisher (1913)

Dianne Cook

- World leader in data visualisation.
- Recent work in visual inference
- Myth busting and apophenia in data visualisation: is what you see really there?
- Academic supervisor of Hadley Wickham and Yihui Xie.
- @visnut
- Website: http://www.dicook.org/

Dianne Cook

```r
library(nullabor) # you probably need to run: install.packages("nullabor")
library(ggplot2)
lineup_df = lineup(null_permute('mpg'), mtcars, pos = 10)
qplot(mpg, wt, data = lineup_df, geom = 'point') + facet_wrap(~ .sample) + theme_linedraw()
```



Lady tasting tea - hypothesis

For Fisher's experiment we were left with two categorical variables.

$$\text{Truth} = \{\text{Milk, Tea, Tea, Milk, Tea, Tea, Milk, Milk}\}$$

$$\text{Prediction} = \{\text{Milk, Tea, Tea, Milk, Tea, Tea, Milk, Milk}\}$$

Asking the question: **Are the ladies predictions independent of the truth?** or

$$H_0\text{: Lady cannot taste the difference} \;\; \text{vs} \;\; H_1\text{: Lady can taste the difference}$$

Our **test statistic** is the number of predictions she gets correct,

$$T = \text{Number of correctly tea before milk cups correctly identified}$$

Calculating significance

The order of the cups was random, therefore there are

$$8! = 8 \times 7 \times 6 \times 5 \times 4 \times 3 \times 2 \times 1 = 40,320$$

ways to order 8 cups of tea.

There are $\binom{8}{4} = 70$ ways to select which 4 cups of tea had the tea added before milk.

Look at all 70 ways of prediction vs truth and calculate how often we see a test statistic of 0, 1, 2, 3 or 4.

Number correct, t_0	0	1	2	3	4	
Number of ways to select	$\binom{4}{0}\binom{4}{4} = 1$	$\binom{4}{1}\binom{4}{3} = 16$	$\binom{4}{2}\binom{4}{2} = 36$	$\binom{4}{3}\binom{4}{1} = 16$	$\binom{4}{4}\binom{4}{0} = 1$	70
Probability: $P(T = t_0)$	$\frac{1}{70}$	$\frac{16}{70}$	$\frac{36}{70}$	$\frac{16}{70}$	$\frac{1}{70}$	1

$P(T = 4) = \frac{1}{70} = 0.014$ (see Fisher's exact test in Lecture 9).

Permutations

We could also consider all 40,320 different orderings (permutations) of 8 cups of tea.

```r
# install.packages("arrangements")
library(arrangements)
permute_8 = permutations(8)
```

```r
head(permute_8, 10)
```

```
##       [,1] [,2] [,3] [,4] [,5] [,6] [,7] [,8]
##  [1,]    1    2    3    4    5    6    7    8
##  [2,]    1    2    3    4    5    6    8    7
##  [3,]    1    2    3    4    5    7    6    8
##  [4,]    1    2    3    4    5    7    8    6
##  [5,]    1    2    3    4    5    8    6    7
##  [6,]    1    2    3    4    5    8    7    6
##  [7,]    1    2    3    4    6    5    7    8
##  [8,]    1    2    3    4    6    5    8    7
##  [9,]    1    2    3    4    6    7    5    8
## [10,]    1    2    3    4    6    7    8    5
```

```r
tail(permute_8, 10)
```

```
##          [,1] [,2] [,3] [,4] [,5] [,6] [,7] [,8]
## [40311,]    8    7    6    5    3    2    1    4
## [40312,]    8    7    6    5    3    2    4    1
## [40313,]    8    7    6    5    3    4    1    2
## [40314,]    8    7    6    5    3    4    2    1
## [40315,]    8    7    6    5    4    1    2    3
## [40316,]    8    7    6    5    4    1    3    2
## [40317,]    8    7    6    5    4    2    1    3
## [40318,]    8    7    6    5    4    2    3    1
## [40319,]    8    7    6    5    4    3    1    2
## [40320,]    8    7    6    5    4    3    2    1
```

Permutations

Use the `permuations()` function on the `truth` vector:

```
truth = c("milk","tea","tea","milk","tea","tea","milk","milk")
permute_guess = permutations(truth)
permute_guess[92,]
```

```
## [1] "milk" "tea"  "tea"  "milk" "milk" "milk" "tea"  "tea"
```

We can check if a particular sequence of tea cups is **identical** to the true sequence:

```
identical(truth, truth)
```

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

```
identical(permute_guess[92,], truth)
```

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

Exact p-value

We can calculate the exact p-value by looking across all permutations:

```
B = nrow(permute_guess)
check_correct = vector("numeric", length = B)
for(i in 1:B) {
  check_correct[i] = identical(permute_guess[i,], truth)
}
c(sum(check_correct), mean(check_correct))
```

```
## [1] 576.00000000   0.01428571
```

The p-value is the same as we get using Fisher's exact test!

Approximate p-value

Often it's not feasible to consider all $n!$ permutations, so we can `sample()` a selection of them.

```
set.seed(123)
truth = c("milk","tea","tea","milk","tea","tea","milk","milk")
B = 10000
result = vector(length = B) # initialise outside the loop
for(i in 1:B){
  guess = sample(truth, size = 8, replace = FALSE) # does the permutation
  result[i] = identical(guess, truth)
}
mean(result)
```

```
## [1] 0.0146
```

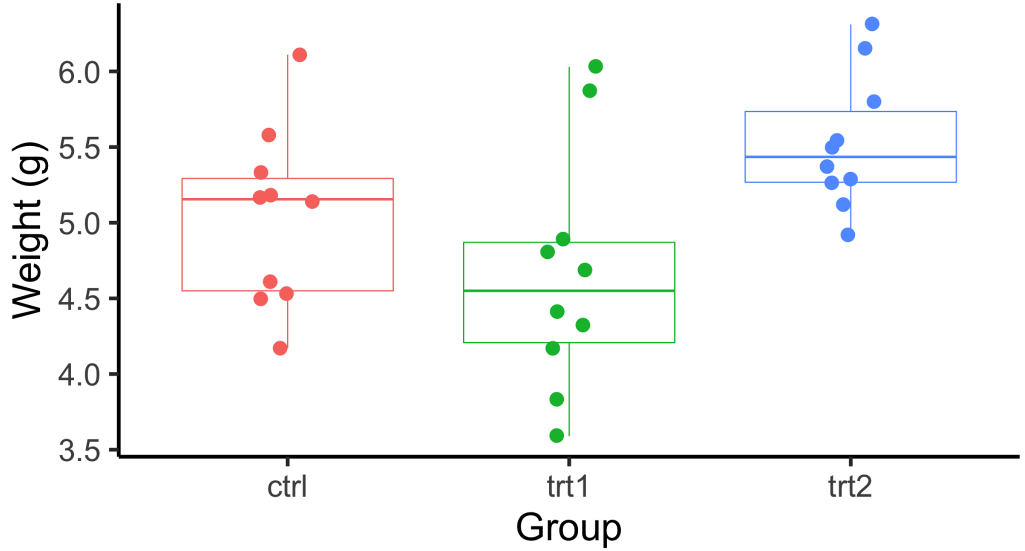
Pretty close to the exact p-value.

Permutation tests for population means

Plant growth

The `PlantGrowth` data has results from an experiment to compare yields (as measured by dried weight of plants) obtained under a control and two different treatment conditions (Dobson, 1983; Table 7.1).

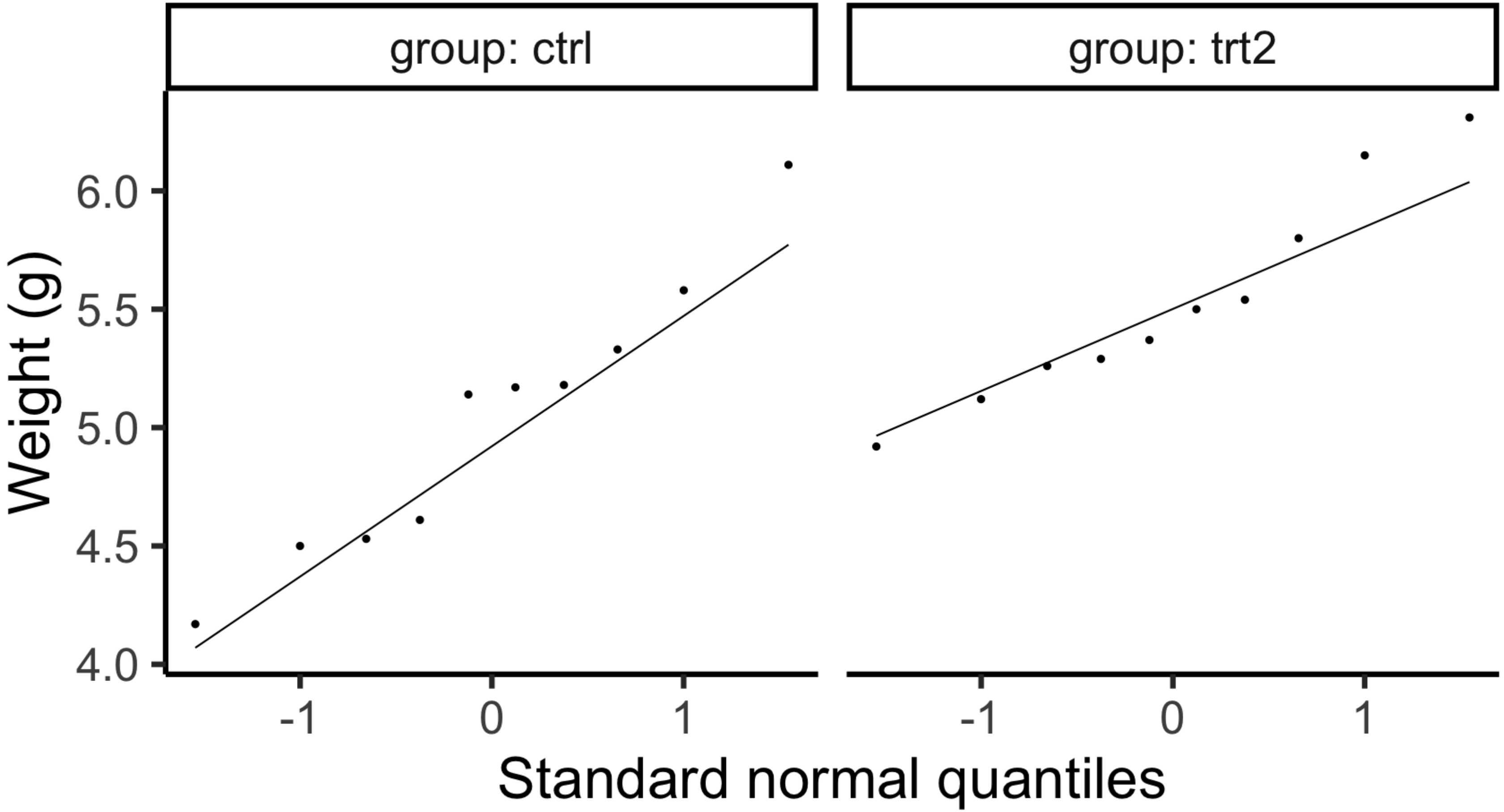
```
# built into R, load it into the environment
data("PlantGrowth")
library(tidyverse)
PlantGrowth %>% ggplot() +
  aes(y = weight, x = group, colour = group) +
  geom_boxplot(coef = 10) +
  geom_jitter(width = 0.1, size = 5) +
  theme(legend.position = "none") +
  labs(y = "Weight (g)", x = "Group")
```



We want to compare the **control** group to the **treatment 2** group.

```
dat = PlantGrowth %>% filter(group %in% c("ctrl", "trt2"))
```

```r
dat %>%
  ggplot() + aes(sample = weight) +
  geom_qq() + geom_qq_line() + facet_grid(cols = vars(group), labeller = "label_both") +
  labs(y = "Weight (g)", x = "Standard normal quantiles")
```



Two-sample t-test

```r
t.test(weight ~ group,
       data = dat,
       var.equal = TRUE)
```

```
##
##      Two Sample t-test
##
## data:  weight by group
## t = -2.134, df = 18, p-value = 0.04685
## alternative hypothesis: true difference in means between group ctrl and group trt2 is not equal to 0
## 95 percent confidence interval:
##  -0.980338117 -0.007661883
## sample estimates:
## mean in group ctrl mean in group trt2
##              5.032              5.526
```

Wilcoxon rank-sum test

```r
wilcox.test(weight ~ group, data = dat)
```

```
##
##      Wilcoxon rank sum exact test
##
## data:  weight by group
## W = 25, p-value = 0.06301
## alternative hypothesis: true location shift is no
```

Extracting information from `t.test` objects

```
tt = t.test(weight ~ group, data = dat, var.equal = TRUE)
tt
```

```
##
##      Two Sample t-test
##
## data:  weight by group
## t = -2.134, df = 18, p-value = 0.04685
## alternative hypothesis: true difference in means between group ctrl and group trt2 is not equal to 0
## 95 percent confidence interval:
##  -0.980338117 -0.007661883
## sample estimates:
## mean in group ctrl mean in group trt2
##              5.032              5.526
```

```
names(tt)
```

```
##  [1] "statistic"   "parameter"   "p.value"     "conf.int"
##  [5] "estimate"    "null.value"  "stderr"      "alternative"
##  [9] "method"      "data.name"
```

```
tt$statistic
```

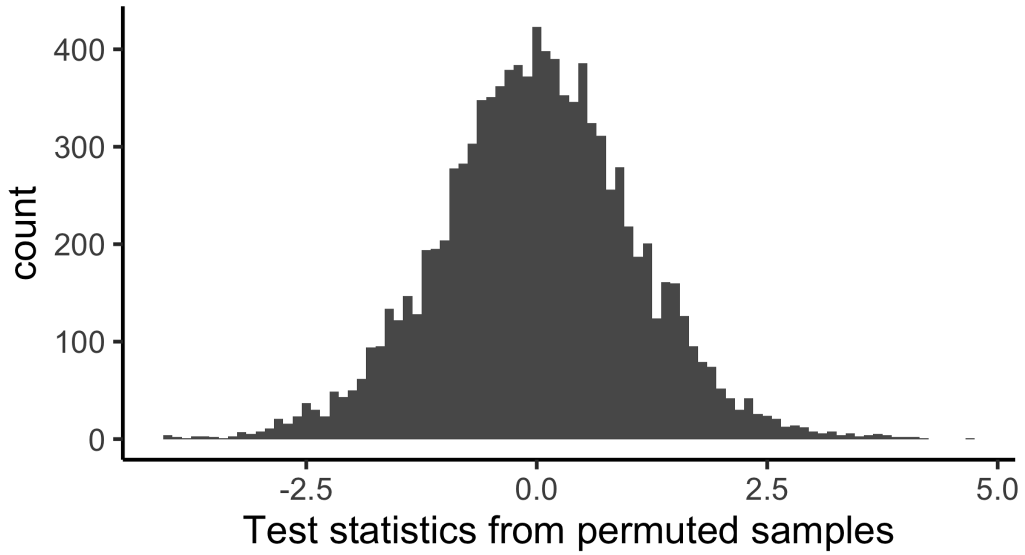
```
##        t
## -2.13402
```

Permutation test

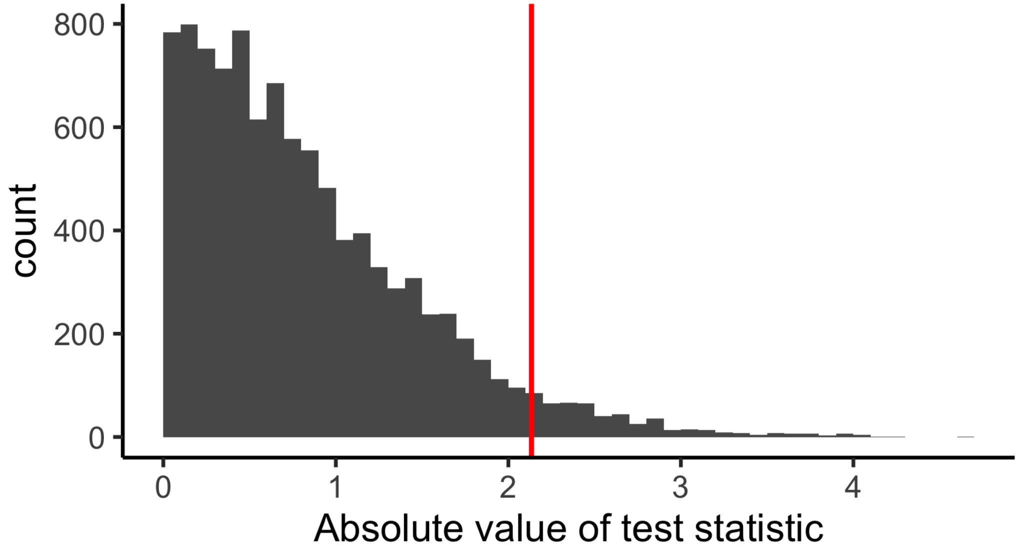
Permute the **class labels** (many times) and see what values we get for the t-test statistic.

```
B = 10000 # number of permuted samples we will consider
permuted_dat = dat # make a copy of the data
t_null = vector("numeric", B) # initialise outside loop
for(i in 1:B) {
  permuted_dat$group = sample(dat$group) # this does the permutation
  t_null[i] = t.test(weight ~ group, data = permuted_dat)$statistic
}
```

```
t_null %>% data.frame() %>%
  ggplot() +
  aes(x = t_null) +
  geom_histogram(binwidth = 0.1) +
  labs(
    x = "Test statistics from permuted samples"
  )
```



```r
data.frame(abs_t_null = abs(t_null)) %>%
  ggplot() +
  aes(x = abs_t_null) +
  geom_histogram(binwidth = 0.1,
                 boundary = 0) +
  geom_vline(xintercept = abs(tt$statistic),
             col = "red", lwd = 2) +
  labs(
    x = "Absolute value of test statistic"
  )
```



What proportion of test statistics from randomly permuted data are more extreme than the test statistic we observed?

```r
mean(abs(t_null) >= abs(tt$statistic))
```

```
## [1] 0.0501
```

This is our permutation test p-value.

Permutation tests

- The two-sample t-test and the permutation test gave similar p-values, but this won't always be the case.
- The two-sample t-test is a **parametric** test where the test statistic is assumed to follow some distribution (a $t_{n_x+n_y-2}$ distribution).
- The permutation test considers the $(n_1+n_2)!$ permutations of the labels (or a random subset to save computation time) from a single instance of the data (the n_1+n_2 observations).
- The permutation test only assumes that the observations $X_1, X_2, \ldots, X_{n_x}, Y_1, Y_2, \ldots, Y_{n_y}$ are *exchangeable*, that is, swapping labels on observations keeps the data just as likely as the original[1].
- The permutation test may use the t-test **test statistic** but it does not use the t **distribution**.

Exchangeability means that the joint distribution is invariant to permutations of the values of each variable in the joint distribution, i.e, $f_{XYZ}(x=1, y=3, z=2) = f_{XYZ}(x=3, y=2, z=1)$. If this is not the case then counting permutations is not a valid way of testing the null hypothesis, as each permutation will have a different weight (probability/density). Permutation tests depend on each assignment of a given set of numerical values to your variables having the same density/probability. Read more here.

Latent heat of fusion

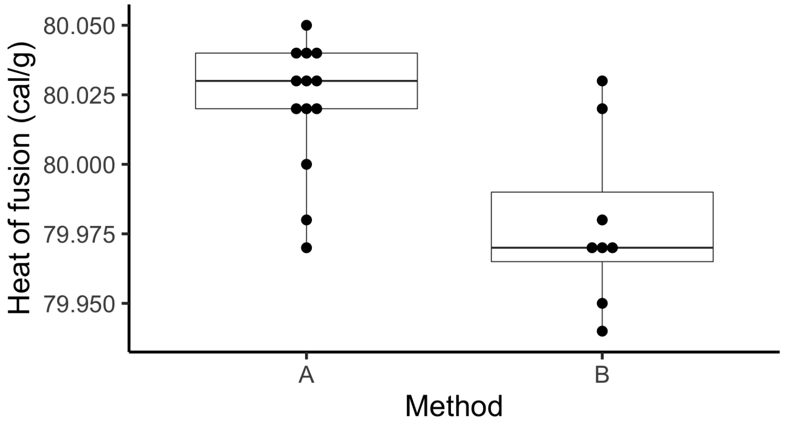
Natrella (1963; page 3-23) presents data from two methods that were used in a study of the latent heat latent heat of fusion of ice. Both method A (digital method) and Method B (method of mixtures) were conducted with the specimens cooled to -0.72°C. The data represent the change in total heat from -0.72°C to water at 0°C, in calories per gram of mass.

> Does the data support the hypothesis that the electrical method (method A) gives larger results?

```r
A = c(79.98, 80.04, 80.02, 80.04, 80.03, 80.03, 80.04,
      79.97, 80.05, 80.03, 80.02, 80.00, 80.02)
B = c(80.02, 79.94, 79.98, 79.97, 79.97, 80.03, 79.95,
      79.97)
heat = data.frame(
  energy = c(A,B),
  method = rep(c("A","B"), c(length(A), length(B)))
)
```

Source: Rice (2006; page 423)

```r
heat %>% ggplot() +
  aes(x = method, y = energy) +
  geom_boxplot(coef = 10) +
  geom_dotplot(
    stackdir = "center",
    binaxis = "y") +
  labs(
    y = "Heat of fusion (cal/g)",
    x = "Method")
```



Latent heat of fusion

```r
t.test(energy ~ method, data = heat, alternative = "greater")
```

```
##
##     Welch Two Sample t-test
##
## data:  energy by method
## t = 3.2499, df = 12.027, p-value = 0.00347
## alternative hypothesis: true difference in means between group A and group B is greater than 0
## 95 percent confidence interval:
##  0.01897943        Inf
## sample estimates:
## mean in group A mean in group B
##        80.02077        79.97875
```

```r
t0_original = t.test(energy ~ method, data = heat, alternative = "greater")$statistic
t0_original
```

```
##        t
## 3.249867
```

Latent heat of fusion

How many permutations of the class label are there?

```
n = nrow(heat)
n
```

```
## [1] 21
```

```
factorial(n)
```

```
## [1] 5.109094e+19
```

```
english::words(factorial(n))
```

fifty-one quintillion ninety quadrillion nine hundred and forty-two trillion one hundred and seventy-one billion seven hundred and nine million four hundred and forty thousand

Latent heat of fusion

Permutation test p-value

```r
B = 10000 # number of permuted samples we will consider
permuted_heat = heat # make a copy of the data
t_null = vector("numeric", B) # initialise outside loop
for(i in 1:B) {
  permuted_heat$method = sample(heat$method) # this does the permutation
  t_null[i] = t.test(energy ~ method, data = permuted_heat)$statistic
}
mean(t_null>=t0_original)
```

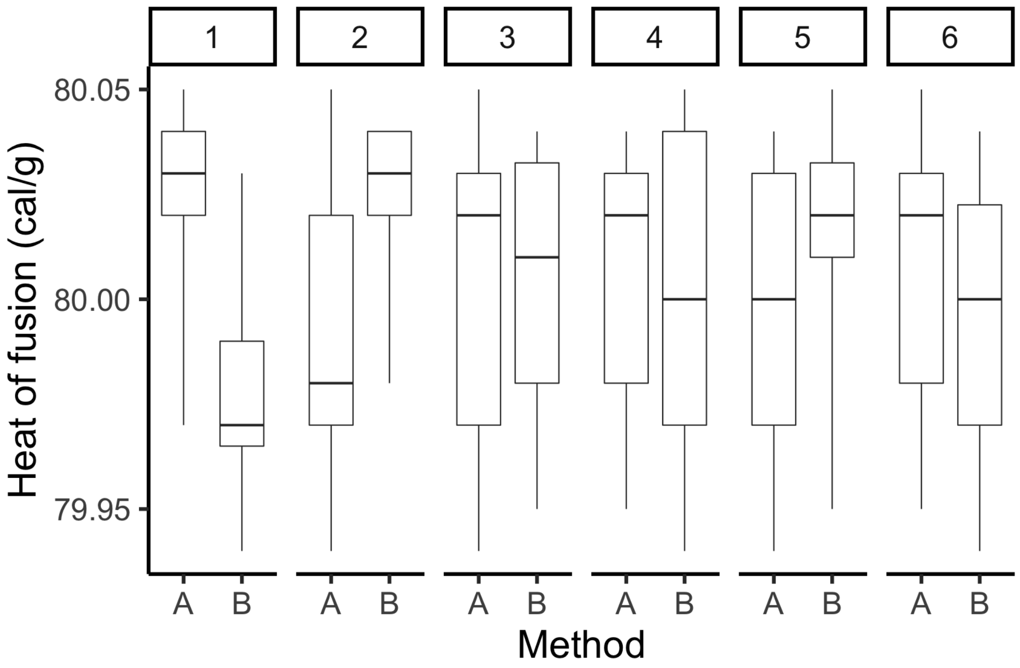
```
## [1] 0.0038
```

Why didn't we need to specify `alternative = "greater"` in the `t.test()` function?

```r
B = 6
perm_heat = heat
perm_heat$id = 1
for(i in 2:6){
  temp = heat
  temp$method = sample(temp$method)
  temp$id = i
  perm_heat = rbind(perm_heat, temp)
}
perm_heat %>%
  group_by(id) %>%
  summarise(
    t_stat = t.test(
      energy[method=="A"],
      energy[method=="B"])$statistic
  )
```

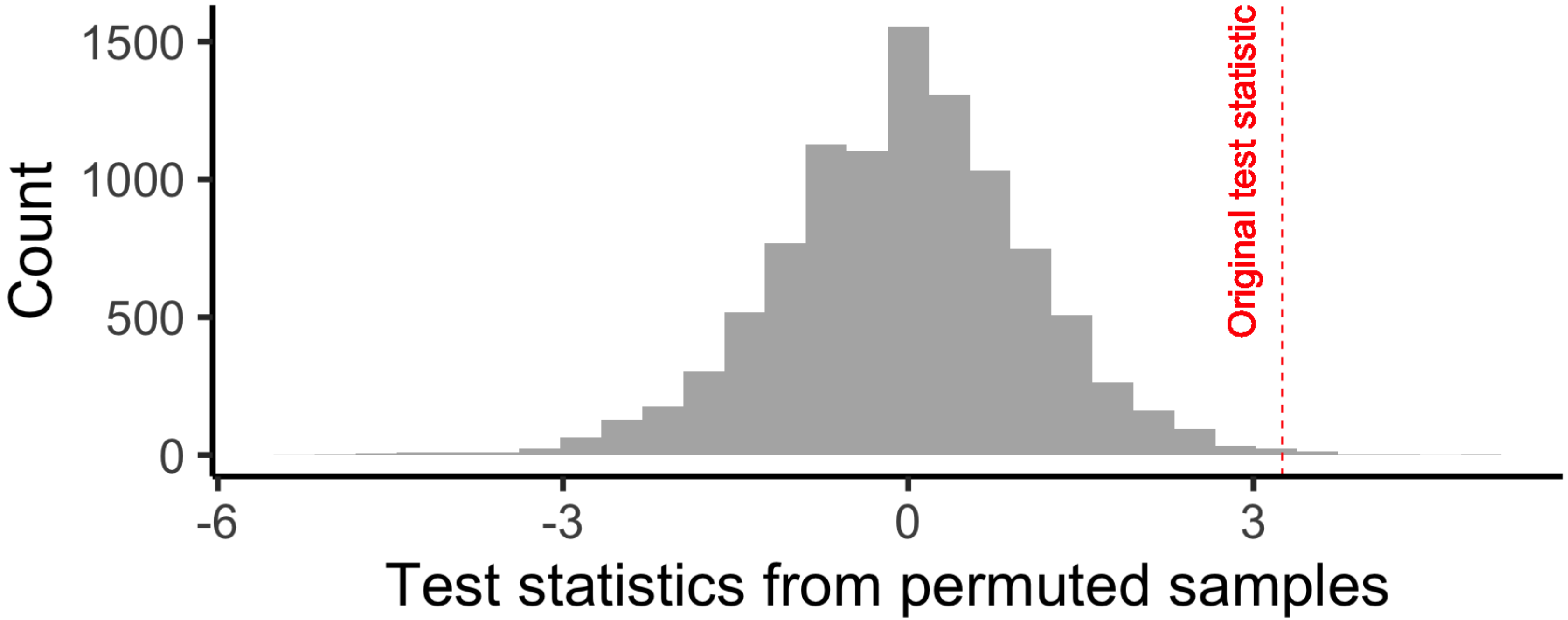
```
## # A tibble: 6 × 2
##      id  t_stat
##   <dbl>   <dbl>
## 1     1  3.25
## 2     2 -2.74
## 3     3 -0.0253
## 4     4  0.333
## 5     5 -0.969
## 6     6  0.885
```

```r
perm_heat %>% ggplot() +
  aes(x = method, y = energy) +
  geom_boxplot(coef = 10) +
  scale_y_continuous(n.breaks = 3) +
  facet_wrap(vars(id), ncol = 6) +
  labs(y = "Heat of fusion (cal/g)",
       x = "Method")
```



Latent heat of fusion

```
t_null %>% data.frame() %>% ggplot() + aes(x = t_null) +
  geom_histogram(alpha=0.5) +
  geom_vline(xintercept = t0_original, colour = "red", linetype = "dashed") +
  geom_text(aes(x = t0_original, label = "Original test statistic", y = Inf),
            colour = "red", angle = 90, hjust = 1, vjust = -1, size = 7) +
  labs(x = "Test statistics from permuted samples", y = "Count")
```

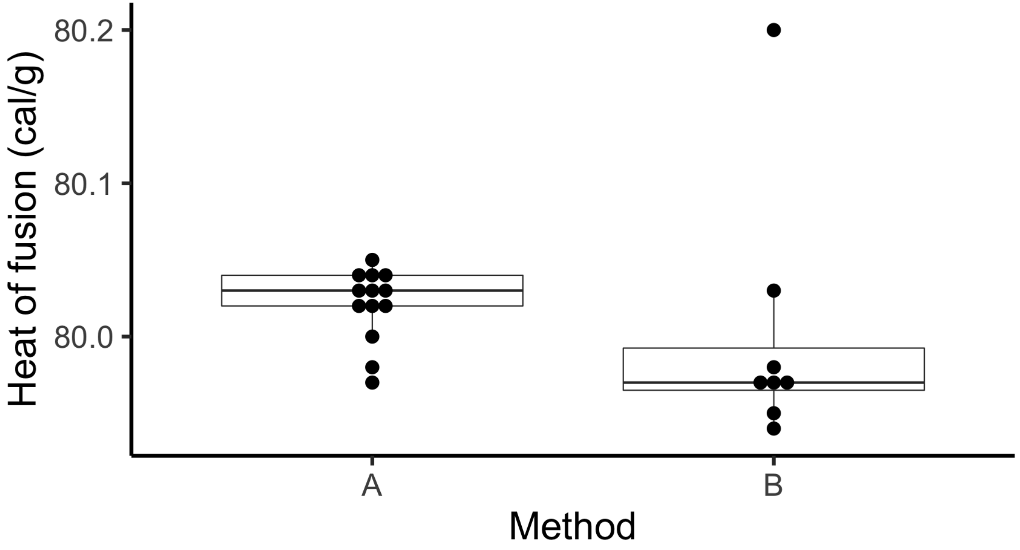


What about outliers?

What happens if there is an outlier in the data?

```r
# change the first value for the B method
heat1 = heat
heat1$energy[14] = 80.20 # instead of 80.02
```

```r
ggplot(heat1, aes(x = method, y = energy)) +
  geom_boxplot() +
  geom_dotplot(stackdir = "center",
               binaxis = "y") +
  labs(y = "Heat of fusion (cal/g)",
       x = "Method")
```



```r
wilcox.test(energy ~ method, data = heat1, alternative = "greater", correct = FALSE)
```

```
## Warning in wilcox.test.default(x = c(79.98, 80.04, 80.02,
## 80.04, 80.03, : cannot compute exact p-value with ties

##
##      Wilcoxon rank sum test
##
## data:  energy by method
## W = 80.5, p-value = 0.01859
## alternative hypothesis: true location shift is greater than 0
```

```r
t.test(energy ~ method, data = heat1, alternative = "greater")
```

```
##
##      Welch Two Sample t-test
##
## data:  energy by method
## t = 0.63712, df = 7.6977, p-value = 0.2713
## alternative hypothesis: true difference in means between group A and group B is greater than 0
## 95 percent confidence interval:
##  -0.03774242         Inf
## sample estimates:
## mean in group A mean in group B
##        80.02077        80.00125
```

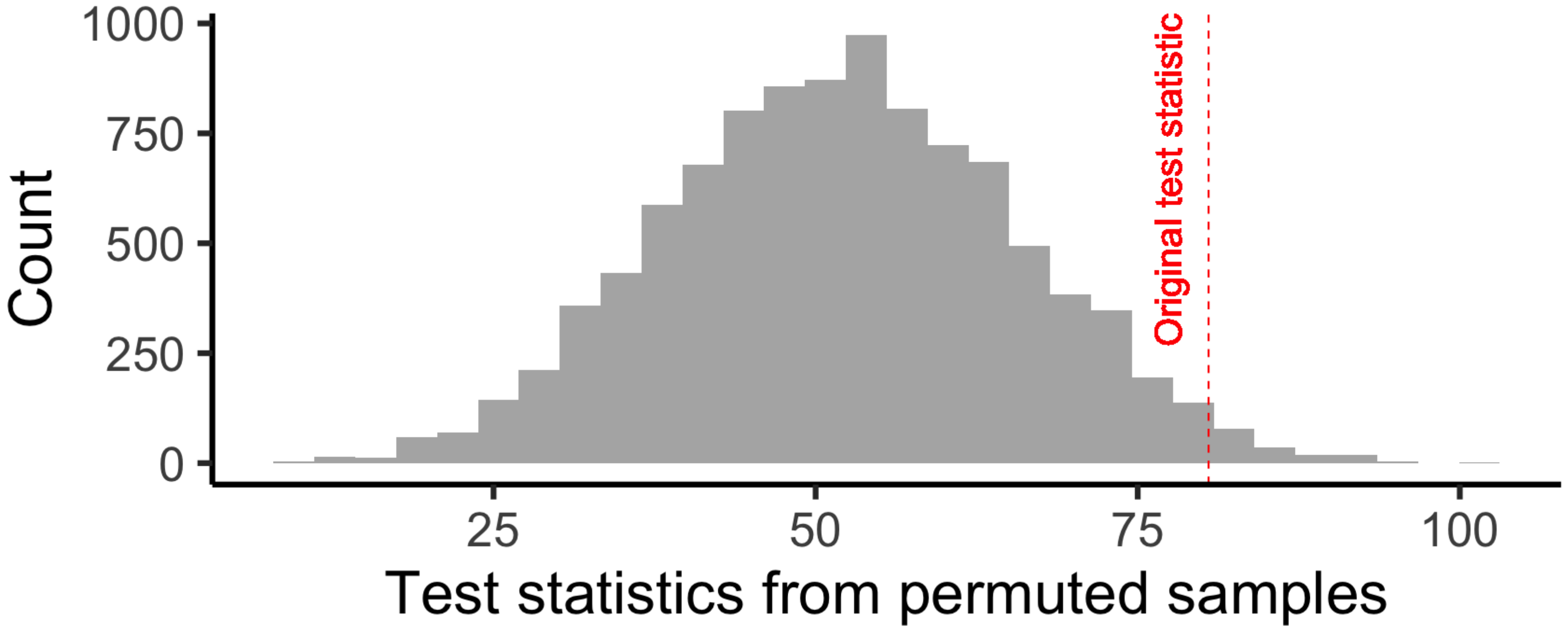
Permutation test using the Wilcoxon rank-sum test statistic

```
t0_original = wilcox.test(energy ~ method, data = heat1)$statistic
set.seed(1234)
B = 10000
permuted_heat1 = heat1
t_null = vector("numeric", B)
for(i in 1:B){
  permuted_heat1$method = sample(heat1$method)
  t_null[i] = wilcox.test(energy ~ method, data = permuted_heat1)$statistic
}
mean(t_null >= t0_original)
```

```
## [1] 0.0192
```

Permutation test using the Wilcoxon rank-sum test statistic

```
t_null %>% data.frame() %>% ggplot() + aes(x = t_null) +
  geom_histogram(alpha=0.5) +
  geom_vline(xintercept = t0_original, colour = "red", linetype = "dashed") +
  geom_text(aes(x = t0_original, label = "Original test statistic", y = Inf),
            colour = "red", angle = 90, hjust = 1, vjust = -1, size = 7) +
  labs(x = "Test statistics from permuted samples", y = "Count")
```



Are there other test statistics we can use?

Can you think of another test statistic that would be robust to outliers?

Difference in medians

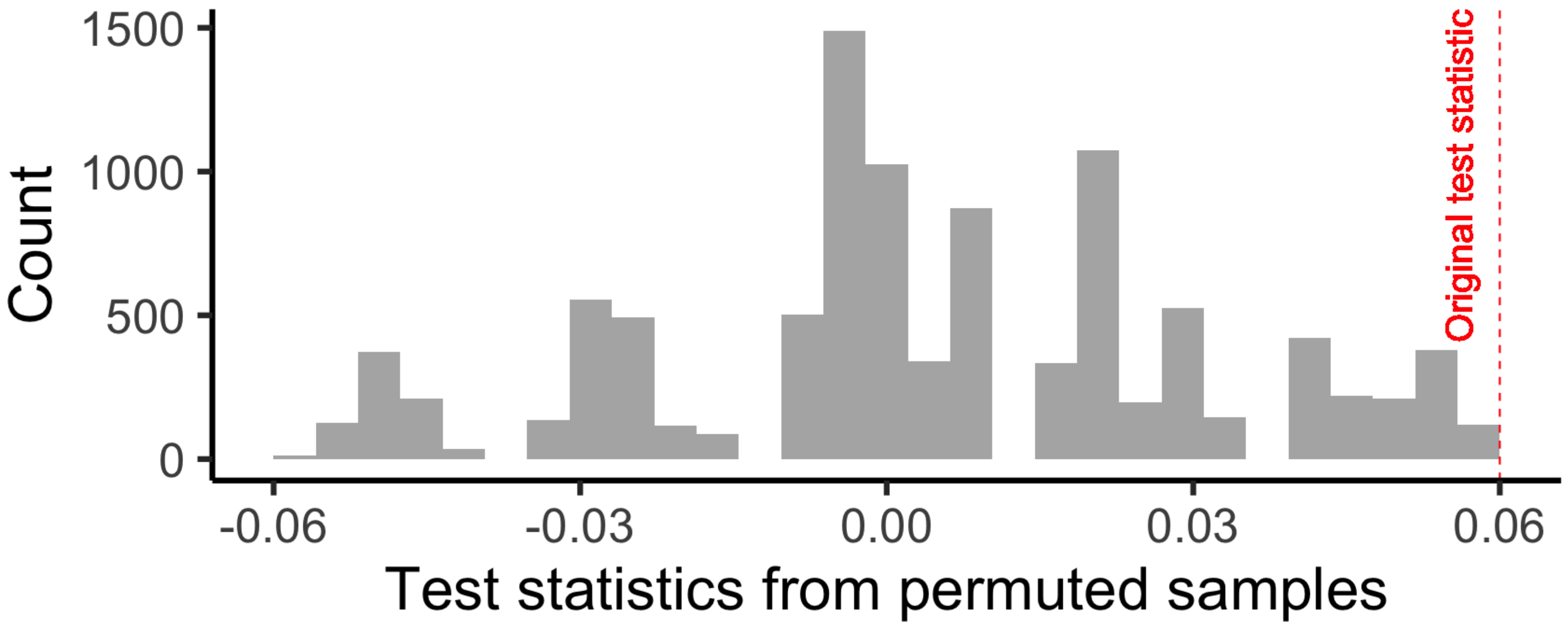
$$T = \tilde{x} - \tilde{y}$$

```
t0_original = median(heat1$energy[heat1$method=="A"]) - median(heat1$energy[heat1$method=="B"])
B = 10000
t_null = vector("numeric", B)
for(i in 1:B){
  permuted_heat1$method = sample(heat1$method)
  median_a =  median(permuted_heat1$energy[permuted_heat1$method=="A"])
  median_b =  median(permuted_heat1$energy[permuted_heat1$method=="B"])
  t_null[i] = median_a - median_b
}
mean(t_null >= t0_original)
```

```
## [1] 0.012
```

Difference in medians

```r
t_null %>% data.frame() %>% ggplot() + aes(x = t_null) +
  geom_histogram(alpha=0.5) +
  geom_vline(xintercept = t0_original, colour = "red", linetype = "dashed") +
  geom_text(aes(x = t0_original, label = "Original test statistic", y = Inf),
            colour = "red", angle = 90, hjust = 1, vjust = -1, size = 7) +
  labs(x = "Test statistics from permuted samples", y = "Count")
```



Robustly standardised difference in medians

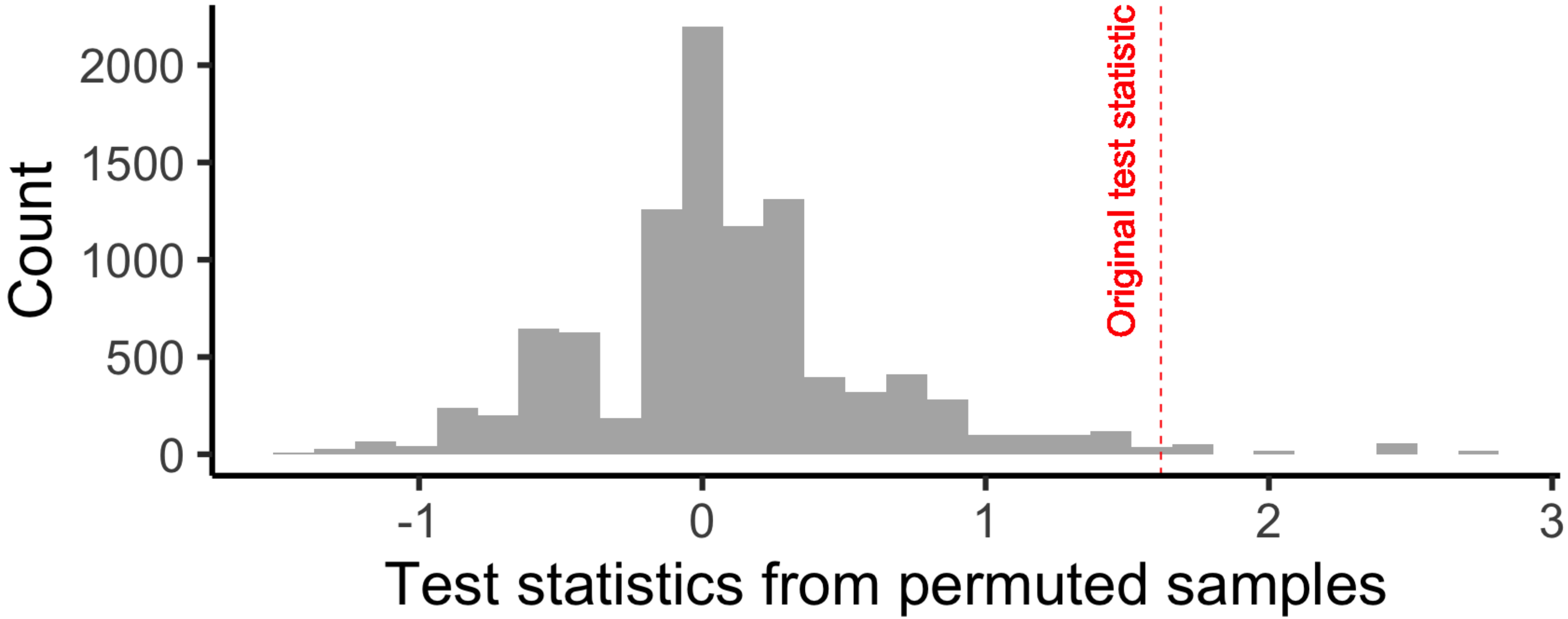
$$T = \frac{\tilde{x} - \tilde{y}}{\text{MAD}(x) + \text{MAD}(y)}$$

```r
median_a = median(heat1$energy[heat1$method=="A"])
median_b = median(heat1$energy[heat1$method=="B"])
mad_a = mad(heat1$energy[heat1$method=="A"])
mad_b = mad(heat1$energy[heat1$method=="B"])
t0_original = (median_a - median_b)/(mad_a + mad_b)

B = 10000
t_null = vector("numeric", B)
for(i in 1:B){
  permuted_heat1$method = sample(heat1$method)
  median_a =  median(permuted_heat1$energy[permuted_heat1$method=="A"])
  median_b =  median(permuted_heat1$energy[permuted_heat1$method=="B"])
  mad_a = mad(permuted_heat1$energy[permuted_heat1$method=="A"])
  mad_b = mad(permuted_heat1$energy[permuted_heat1$method=="B"])
  t_null[i] = (median_a - median_b)/(mad_a + mad_b)
}
mean(t_null >= t0_original)
```

```
## [1] 0.0182
```

Robustly standardised difference in medians

```r
t_null %>% data.frame() %>% ggplot() + aes(x = t_null) +
  geom_histogram(alpha=0.5) +
  geom_vline(xintercept = t0_original, colour = "red", linetype = "dashed") +
  geom_text(aes(x = t0_original, label = "Original test statistic", y = Inf),
            colour = "red", angle = 90, hjust = 1, vjust = -1, size = 7) +
  labs(x = "Test statistics from permuted samples", y = "Count")
```



Paired sample tests?

Can we use permutation tests if we are testing for a shift in location by sampling from one population?

- For paired tests we think about the differences, $d_i = x_i - y_i$.
- For the Wilcoxon signed-rank test we had a test statistic involving

$$\sum_{i:\, d_i>0} r_i \times \operatorname{sign}(d_i)$$

- We could also think of a statistic where we used the values of the differences,

$$\sum_{i=1}^{n} |d_i| \times \operatorname{sign}(d_i)$$

- For a permutation test permute all possible $\operatorname{sign}(d_i)$.

Smoking

Blood samples from 11 individuals before and after they smoked a cigarette are used to measure aggregation of blood platelets.

Is the aggregation affected by smoking?

```
before = c(25, 25, 27, 44, 30, 67, 53, 53, 52, 60, 28)
after =  c(27, 29, 37, 36, 46, 82, 57, 80, 61, 59, 43)
d = after - before
t.test(d)
```

```
## 
##      One Sample t-test
## 
## data:  d
## t = 2.9065, df = 10, p-value = 0.01566
## alternative hypothesis: true mean is not equal to 0
## 95 percent confidence interval:
##   1.97332 14.93577
## sample estimates:
## mean of x 
##  8.454545
```

There are $2^{11} = 2048$ ways to permutations of sign of 11 differences.

```
sign_permute = permutations(c(-1,1), 11, replace = TRUE)
dim(sign_permute)
```

```
## [1] 2048   11
```

```
head(sign_permute)
```

```
##      [,1] [,2] [,3] [,4] [,5] [,6] [,7] [,8] [,9] [,10] [,11]
## [1,]   -1   -1   -1   -1   -1   -1   -1   -1   -1    -1    -1
## [2,]   -1   -1   -1   -1   -1   -1   -1   -1   -1    -1     1
## [3,]   -1   -1   -1   -1   -1   -1   -1   -1   -1     1    -1
## [4,]   -1   -1   -1   -1   -1   -1   -1   -1   -1     1     1
## [5,]   -1   -1   -1   -1   -1   -1   -1   -1    1    -1    -1
## [6,]   -1   -1   -1   -1   -1   -1   -1   -1    1    -1     1
```

```r
(t0_original = mean(d)/sd(d)*sqrt(length(d)))
```

```
## [1] 2.906534
```

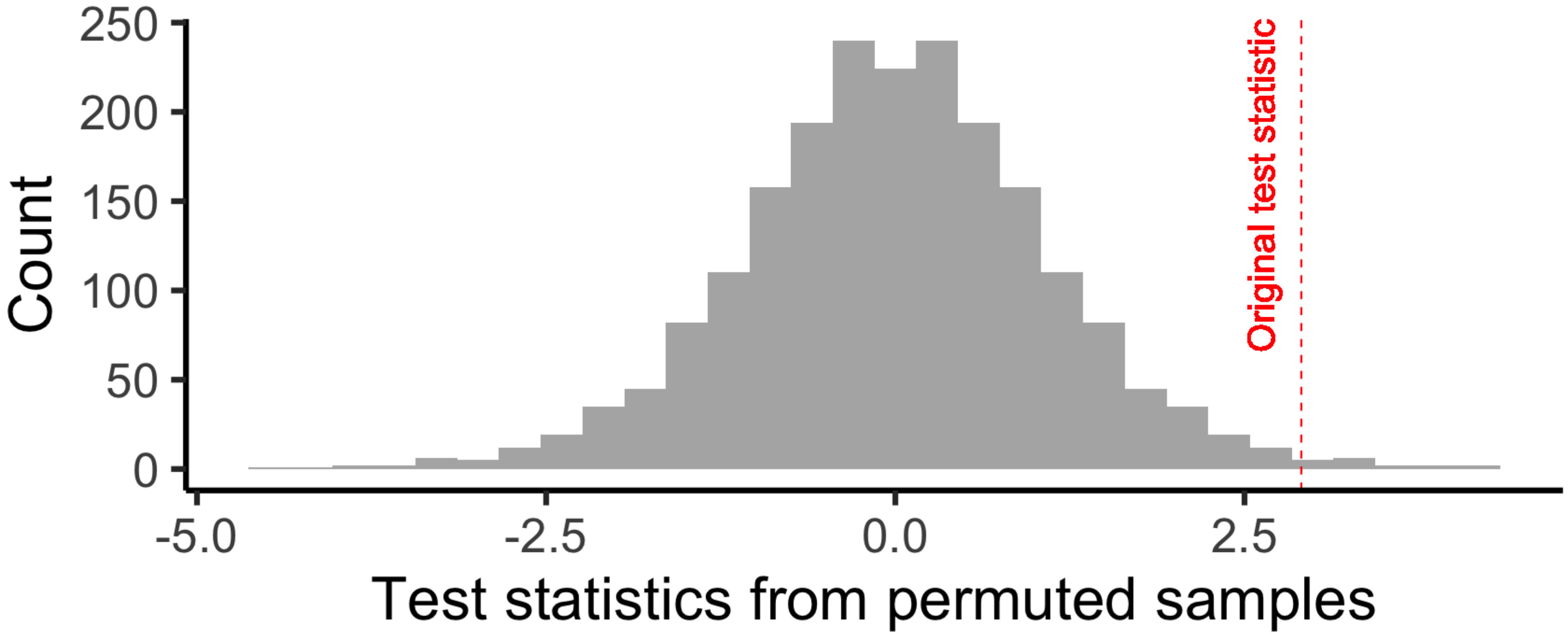
```r
n = length(d)
B = nrow(sign_permute)
t_null = vector("numeric", B)

for(i in 1:nrow(sign_permute)){
  d_permute = d*sign_permute[i,]
  t_null[i] = mean(d_permute)/sd(d_permute)*sqrt(n)
}

mean(abs(t_null) >= abs(t0_original))
```

```
## [1] 0.01660156
```

```r
t.test(d)$p.value
```

```
## [1] 0.01565739
```

```r
t_null %>% data.frame() %>% ggplot() + aes(x = t_null) +
  geom_histogram(alpha=0.5) +
  geom_vline(xintercept = t0_original, colour = "red", linetype = "dashed") +
  geom_text(aes(x = t0_original, label = "Original test statistic", y = Inf),
            colour = "red", angle = 90, hjust = 1, vjust = -1, size = 7) +
  labs(x = "Test statistics from permuted samples", y = "Count")
```



Further reading

- Holmes and Wolfgang (2018; Chapter 6) has a good recap of many of the topics we've discussed 🔗
- Wilber (2019) Visual introduction to permutation testing with alpacas

References

Dobson, A. J. (1983). *An introduction to statistical modelling*. London: Chapman & Hall.

Holmes, S. and H. Wolfgang (2018). *Modern Statistics for Modern Biology*. Stanford University. URL: http://web.stanford.edu/class/bios221/book/.

Lai, R. (2020). *arrangements: Fast Generators and Iterators for Permutations, Combinations, Integer Partitions and Compositions*. R package version 1.1.9. URL: https://CRAN.R-project.org/package=arrangements.

Natrella, M. G. (1963). *Experimental Statistics*. Vol. 91. National Bureau of Standards Handbook. United States Department of Commerce.

Rice, J. (2006). *Mathematical Statistics and Data Analysis*. Advanced series. Cengage Learning. ISBN: 9780534399429.

Wilber, J. (2019). *The Permutation Test: A Visual Explanation of Statistical Testing*. Accessed September 27, 2020. URL: https://www.jwilber.me/permutationtest/.