Structure

Random Pattern

Permutation Test

11/18/2022 (Week 13)

Jingjing Yang, PhD

Assistant Professor of Human Genetics

Jingjing.yang@emory.edu

Hypothesis Test

- Testing **Null (H_0)** vs. **Alternative (H_a)** hypothesis
- Choose an appropriate **Test Statistic**
- **P-value**: The probability of getting a test statistic as or more extreme than the one from your current data, when the null hypothesis is true.
- **Option 1**: Calculate from the known test statistic distribution (cdf/pdf) under the null hypothesis, e.g., Standard Normal distribution N(0, 1), Student's t distribution, F distribution, Chi-square distribution
  - For example, two-sample t-test statistic

$$T = \frac{\widehat{\mu_1} - \widehat{\mu_2}}{S/\sqrt{n}}$$

  - Testing if two groups of data have the same mean

$$H_0: \mu_1 = \mu_2 \quad \text{vs. } H_a: \mu_1 \neq \mu_2$$

- **Option 2**: **Permutation**!

Permutation Test

- **Permutation test**: obtaining p-value by permuting the current observed data to simulate null scenarios
  - For example, *wilcox.test()* for the Wilcoxon-Mann-Whitney test or *mantelhaen.test()* for the Cochran-Mantel-Haenszel χ^2 test in **R**

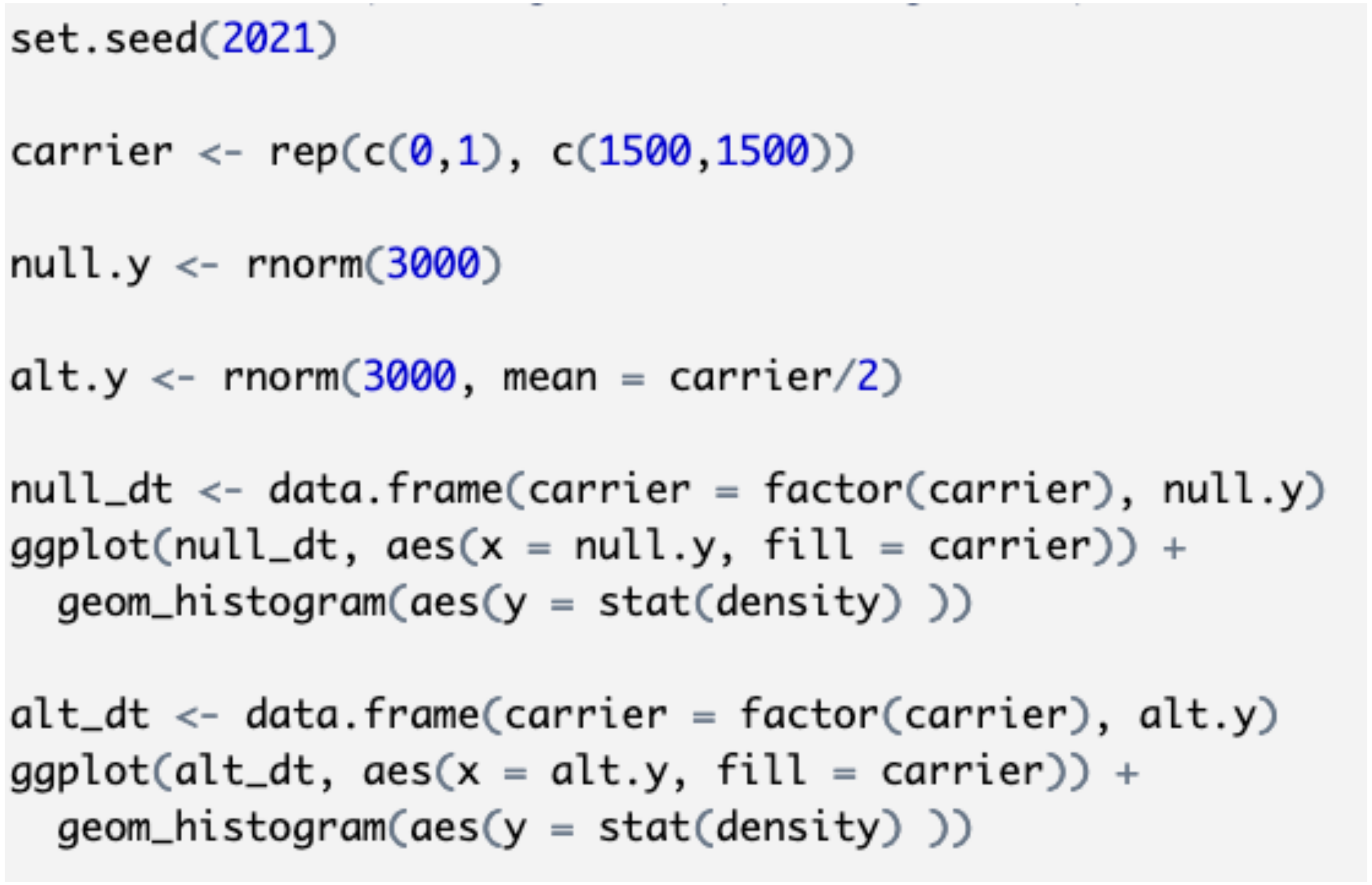
- A permutation test gives a simple way to compute the sampling distribution for any test statistic, **under the null hypothesis**
  - Example NULL: A set of genetic variants has absolutely no effect on the outcome.

Permutation Test Procedure

- Generate a large number of data sets under the null hypothesis by **permuting the observed data**
  - If the null hypothesis is true, changing the exposure would have no effect on the outcome.
  - If the null hypothesis is true, the shuffled data sets should look like the real data, otherwise they should look different from the real data.
  - Permute group labels (e.g., for testing mean difference between two groups) or measurements, and calculate test statistic values with permuted data
- Calculate real test statistic value from observed data
- The ranking of the real test statistic value among the shuffled test statistic values gives a **permutation test p-value**

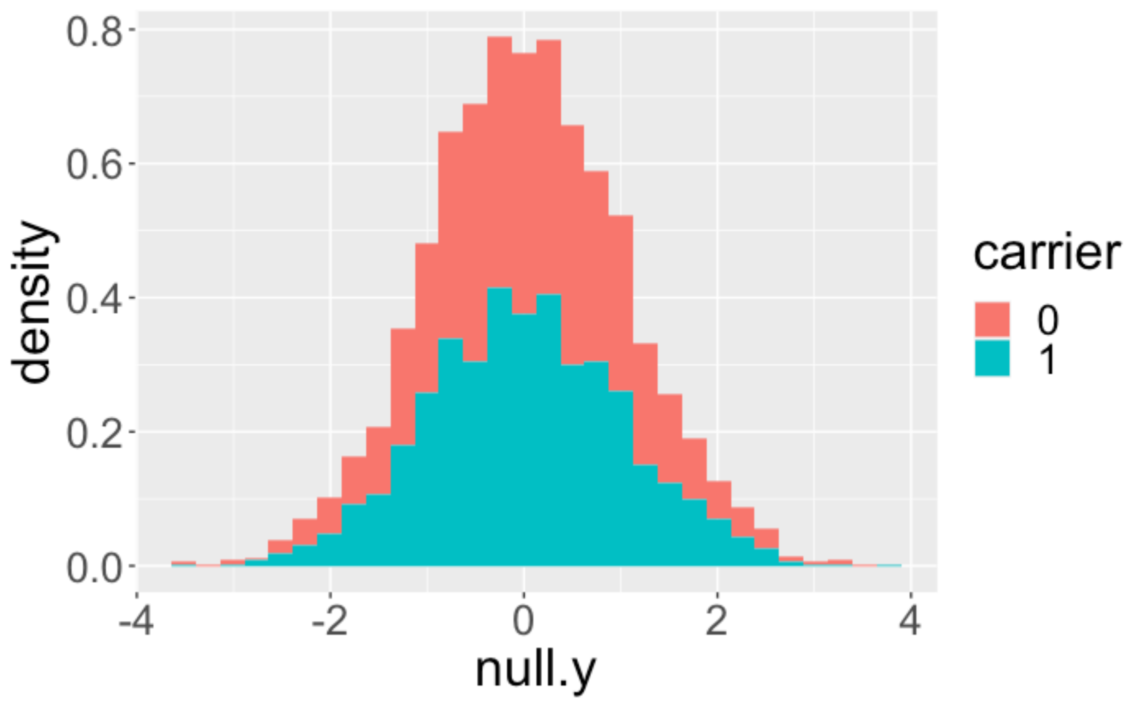
Example 1: test two-group mean difference

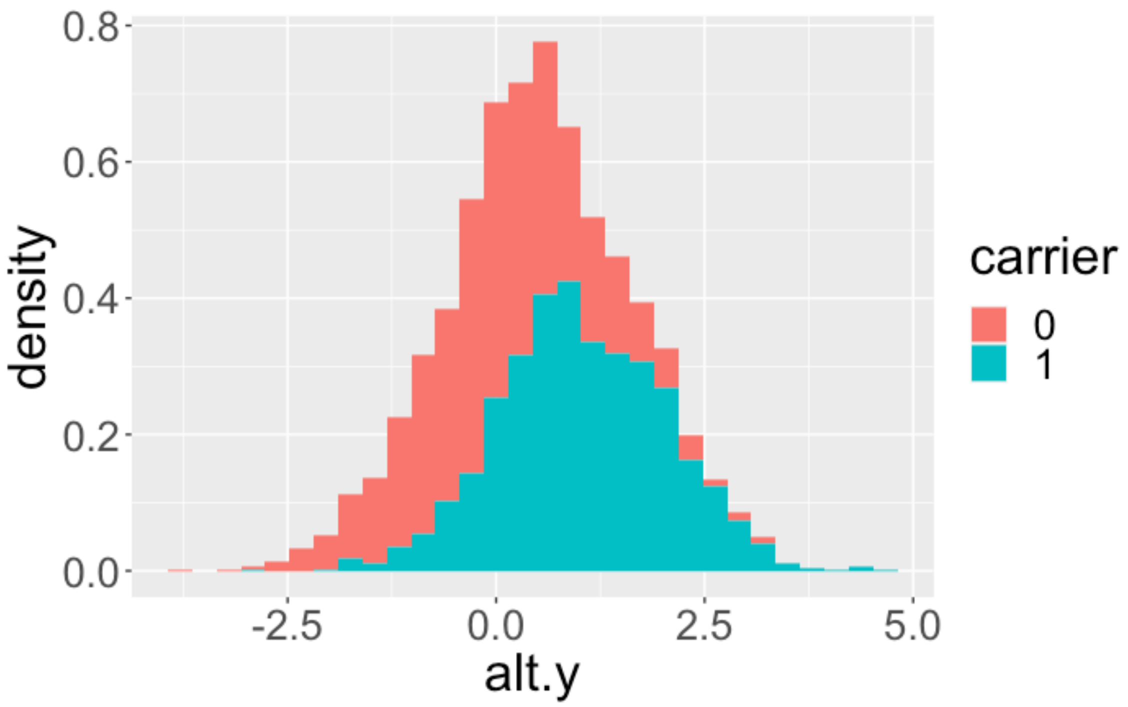
- Simulate 3000 samples with 1500 not carrying a mutation and 1500 carrying a mutation.
- A null phenotype \`*null.y*\` and an alternative phenotype \`*alt.y*\` vectors were simulated from \`*N(0, 1)*\` and \`*N(carrier/2, 1)*\`, respectively.



```r
set.seed(2021)

carrier <- rep(c(0,1), c(1500,1500))

null.y <- rnorm(3000)

alt.y <- rnorm(3000, mean = carrier/2)

null_dt <- data.frame(carrier = factor(carrier), null.y)
ggplot(null_dt, aes(x = null.y, fill = carrier)) +
  geom_histogram(aes(y = stat(density) ))

alt_dt <- data.frame(carrier = factor(carrier), alt.y)
ggplot(alt_dt, aes(x = alt.y, fill = carrier)) +
  geom_histogram(aes(y = stat(density) ))
```

Simulated Null and Alternative Phenotype Vectors





Standard two sample t test results

```r
t.test(null.y ~ carrier, var.equal = TRUE)
```

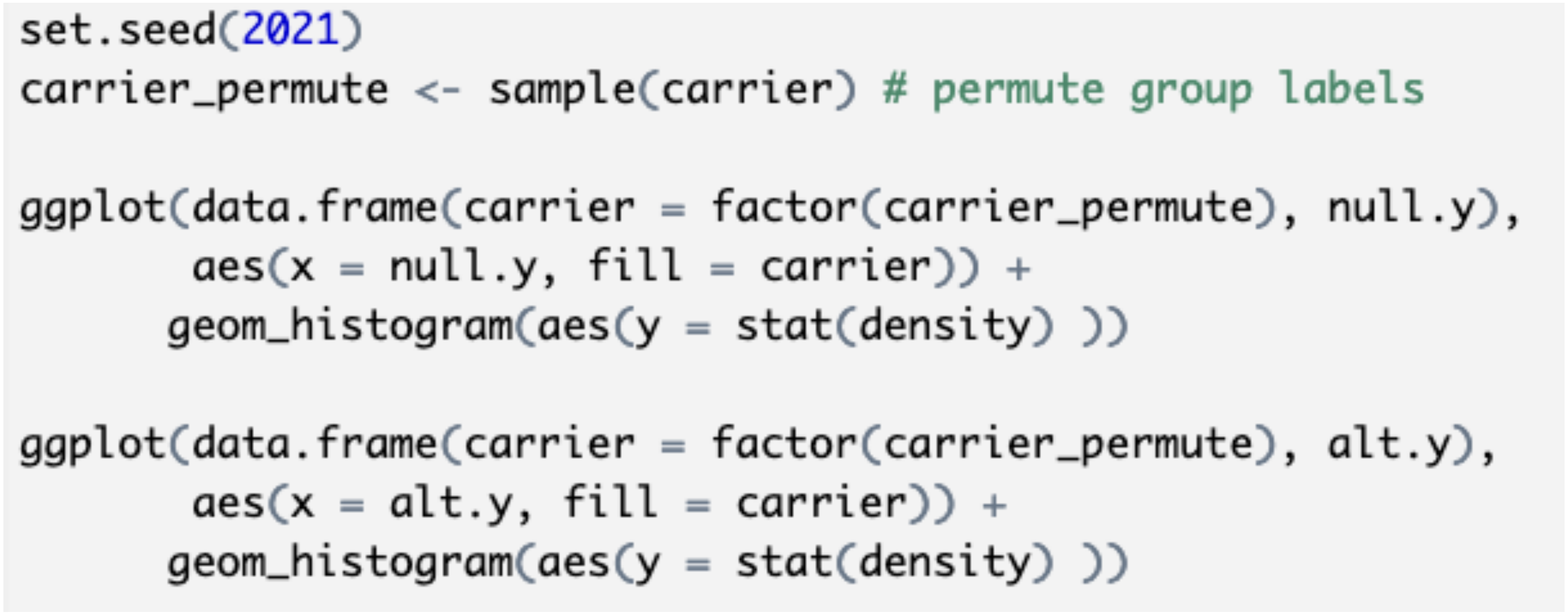
```
##
##  Two Sample t-test
##
## data:  null.y by carrier
## t = -0.43073, df = 2998, p-value = 0.6667
## alternative hypothesis: true difference in means between group 0 and group 1 is not equal
to 0
## 95 percent confidence interval:
##  -0.08886870  0.05685645
## sample estimates:
## mean in group 0 mean in group 1
##     0.005983959     0.021990082
```

Standard two sample t test results

```
t.test(alt.y ~ carrier, var.equal = TRUE)
```

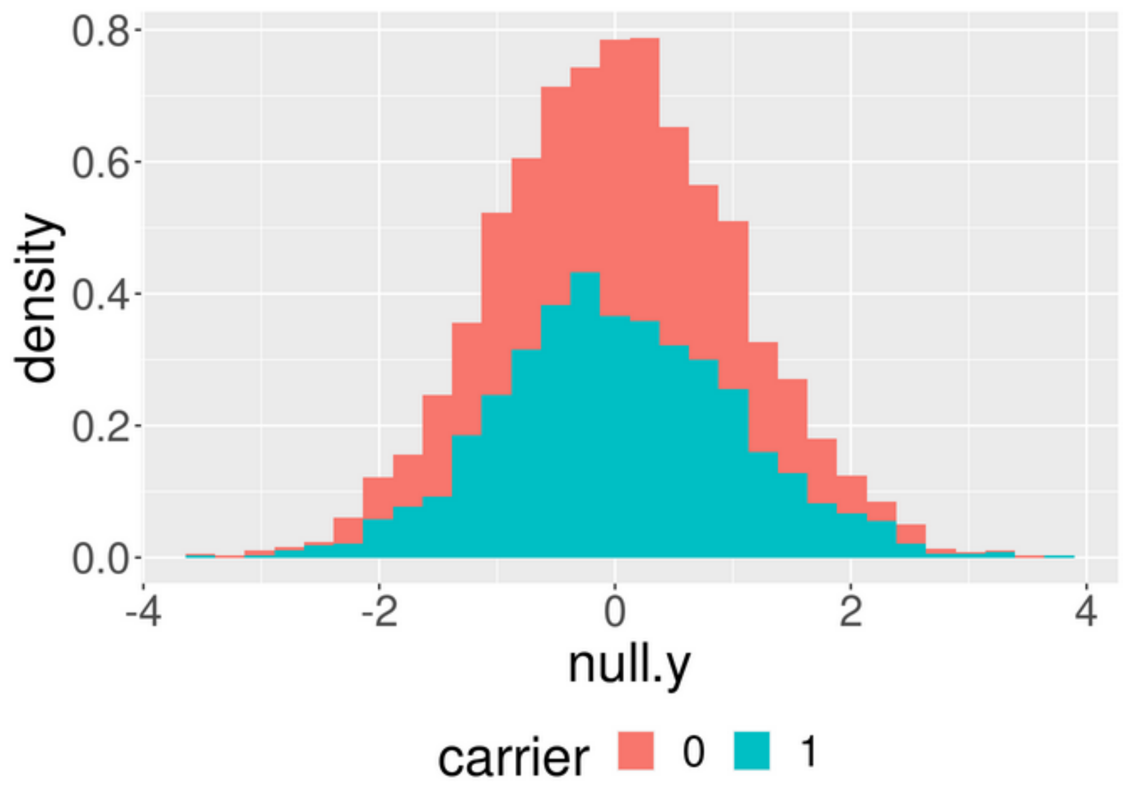
```
##
##  Two Sample t-test
##
## data:  alt.y by carrier
## t = -12.241, df = 2998, p-value < 2.2e-16
## alternative hypothesis: true difference in means between group 0 and group 1 is not equal
to 0
## 95 percent confidence interval:
##  -0.5223674 -0.3781248
## sample estimates:
## mean in group 0 mean in group 1
##      0.05789964      0.50814578
```

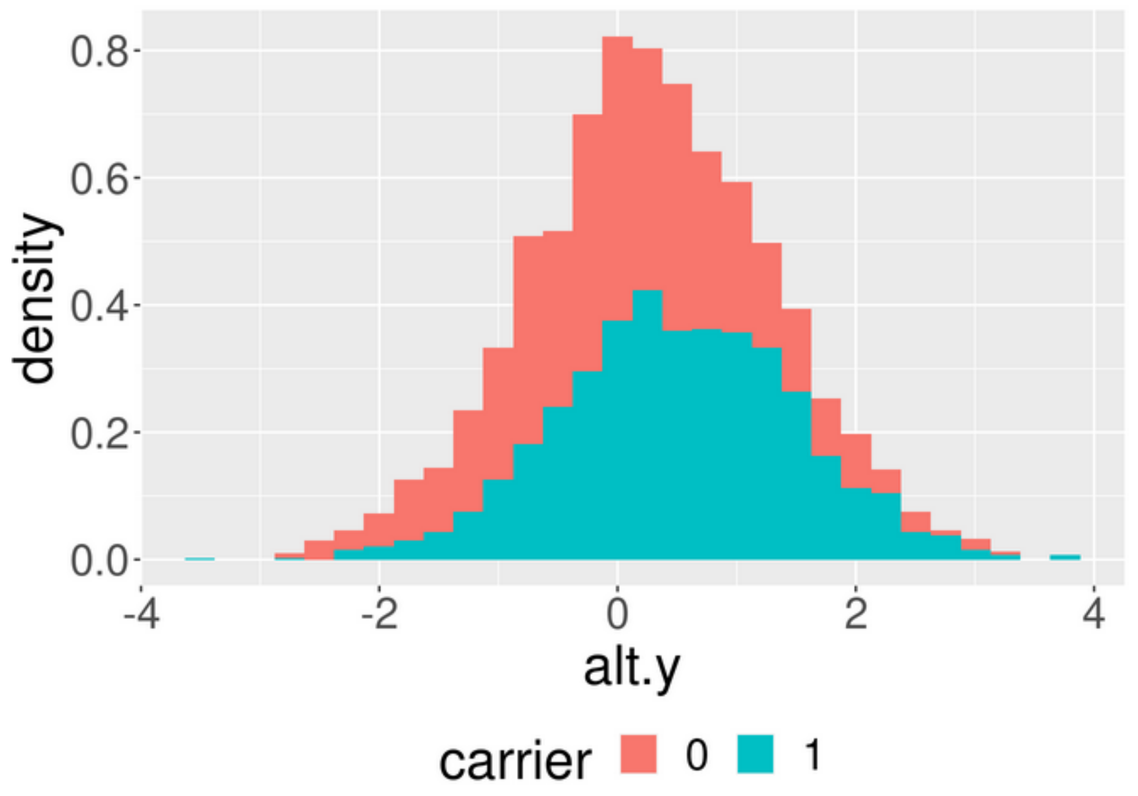
Permute *carrier* group status by *sample()*



```r
set.seed(2021)
carrier_permute <- sample(carrier) # permute group labels

ggplot(data.frame(carrier = factor(carrier_permute), null.y),
       aes(x = null.y, fill = carrier)) +
      geom_histogram(aes(y = stat(density) ))

ggplot(data.frame(carrier = factor(carrier_permute), alt.y),
       aes(x = alt.y, fill = carrier)) +
      geom_histogram(aes(y = stat(density) ))
```

Permute `*carrier*` group status



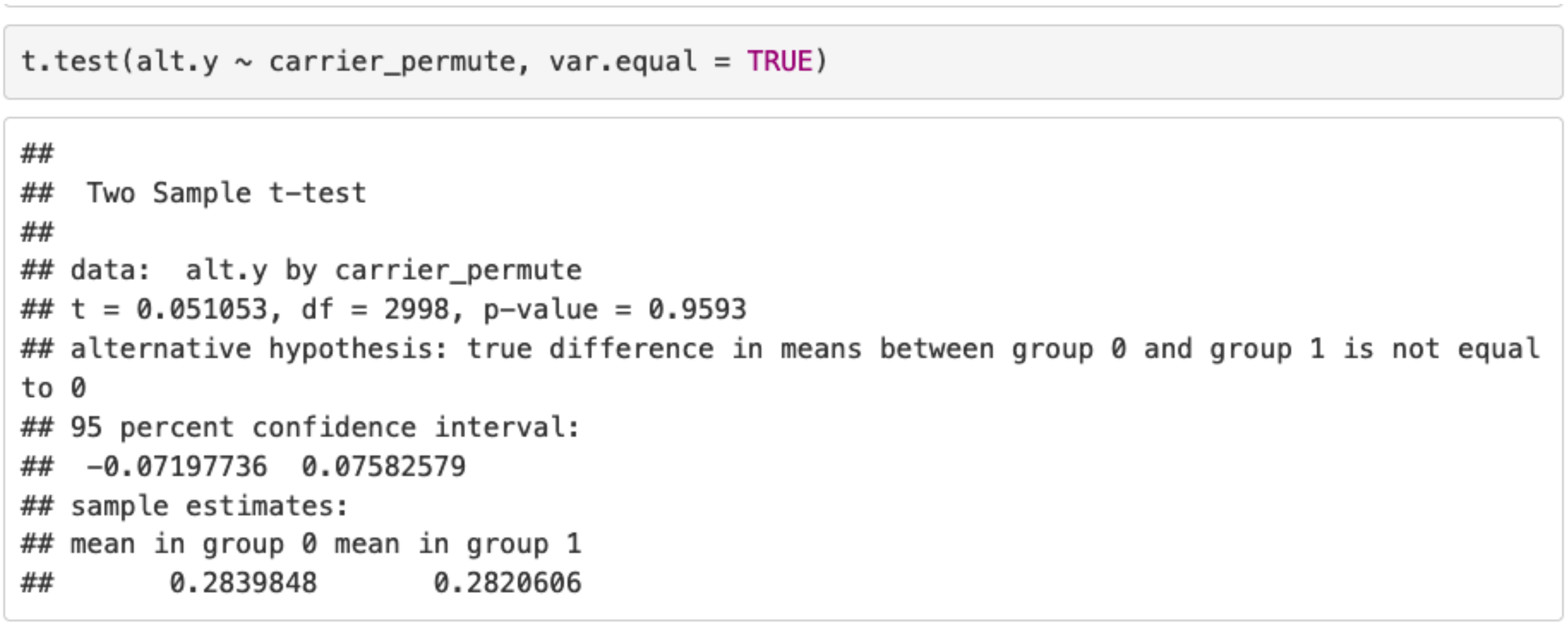


Standard two sample t-test results with permuted group labels

```r
t.test(null.y ~ carrier_permute, var.equal = TRUE)
```

```
##
##  Two Sample t-test
##
## data:  null.y by carrier_permute
## t = -1.6776, df = 2998, p-value = 0.09352
## alternative hypothesis: true difference in means between group 0 and group 1 is not equal
to 0
## 95 percent confidence interval:
##  -0.13514488  0.01051642
## sample estimates:
## mean in group 0 mean in group 1
##     -0.01717009      0.04514414
```

Standard two sample t-test results with permuted group labels



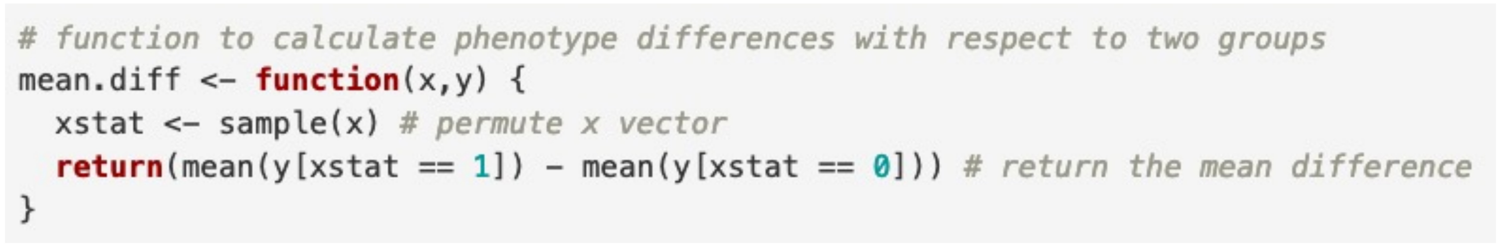
```r
t.test(alt.y ~ carrier_permute, var.equal = TRUE)
```

```
##
##  Two Sample t-test
##
## data:  alt.y by carrier_permute
## t = 0.051053, df = 2998, p-value = 0.9593
## alternative hypothesis: true difference in means between group 0 and group 1 is not equal
to 0
## 95 percent confidence interval:
##  -0.07197736  0.07582579
## sample estimates:
## mean in group 0 mean in group 1
##       0.2839848       0.2820606
```

Permutation test: two-group mean difference

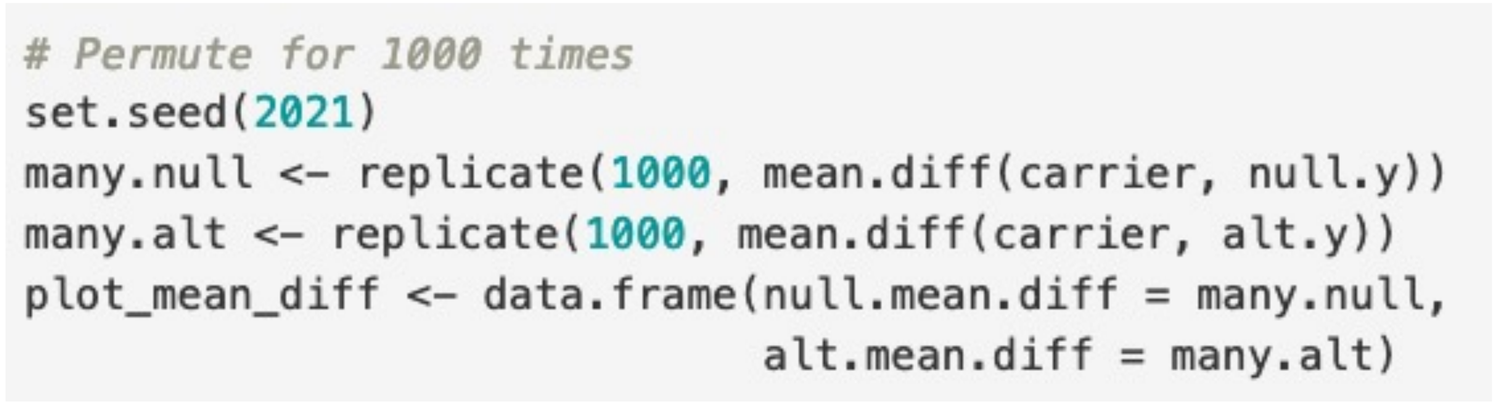
- Consider statistic that is the mean difference between two groups

$$\mu_{diff} = \widehat{\mu_1} - \widehat{\mu_2}$$



```r
# function to calculate phenotype differences with respect to two groups
mean.diff <- function(x,y) {
  xstat <- sample(x) # permute x vector
  return(mean(y[xstat == 1]) - mean(y[xstat == 0])) # return the mean difference
}
```

- Calculate the statistic value for each permuted data set

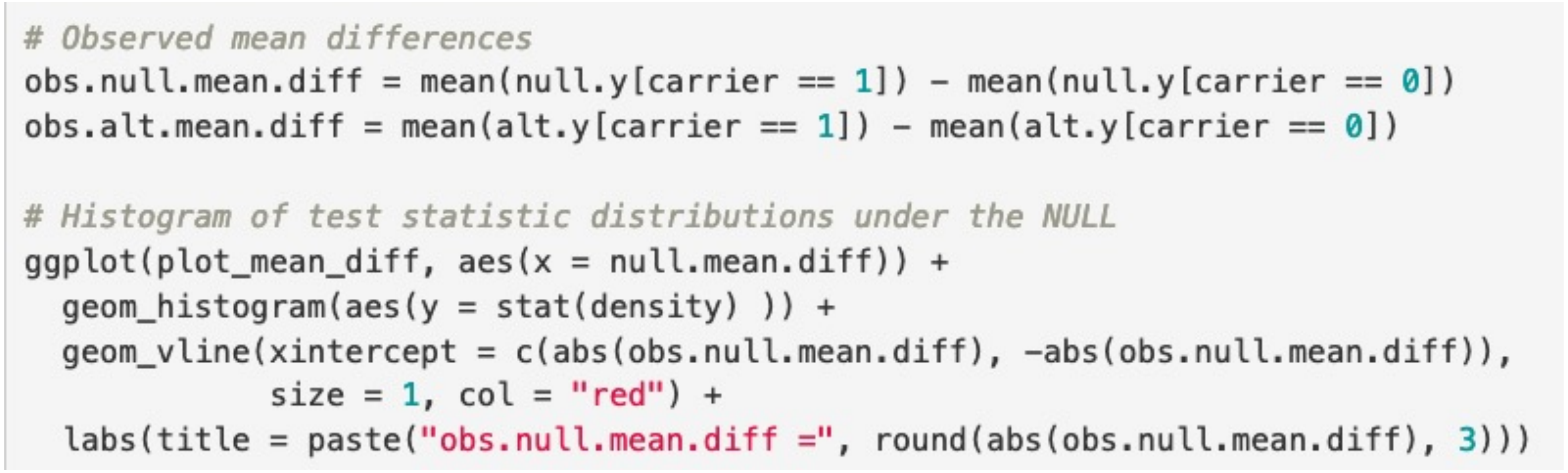


```r
# Permute for 1000 times
set.seed(2021)
many.null <- replicate(1000, mean.diff(carrier, null.y))
many.alt <- replicate(1000, mean.diff(carrier, alt.y))
plot_mean_diff <- data.frame(null.mean.diff = many.null,
                             alt.mean.diff = many.alt)
```

- Permute group labels for 1000 times

Permutation test: two-group mean difference

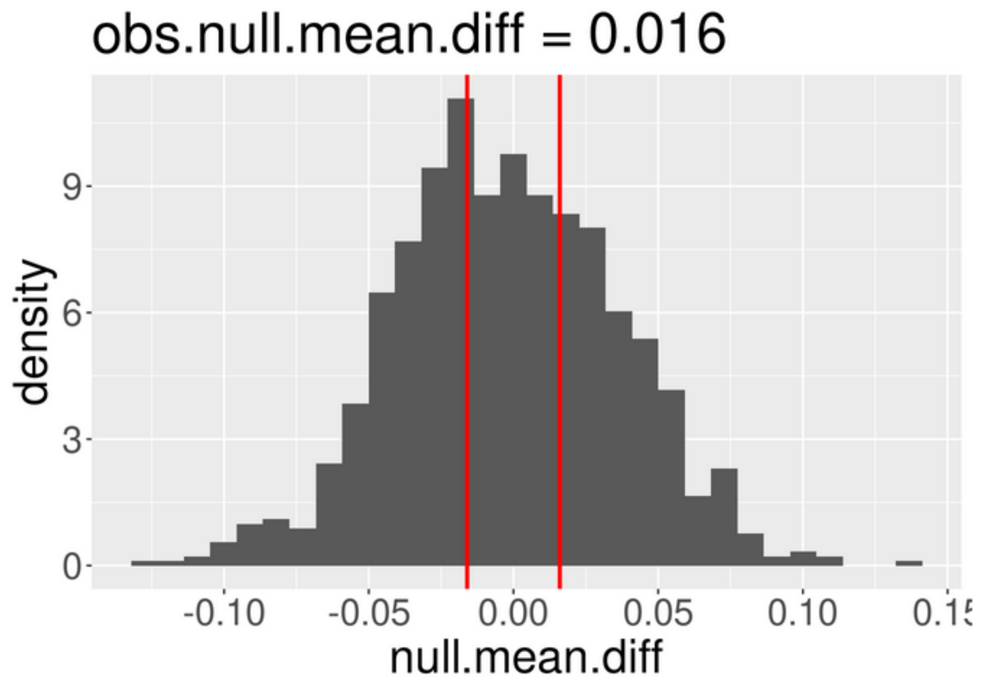
Construct the numeric distribution of μ_{diff}

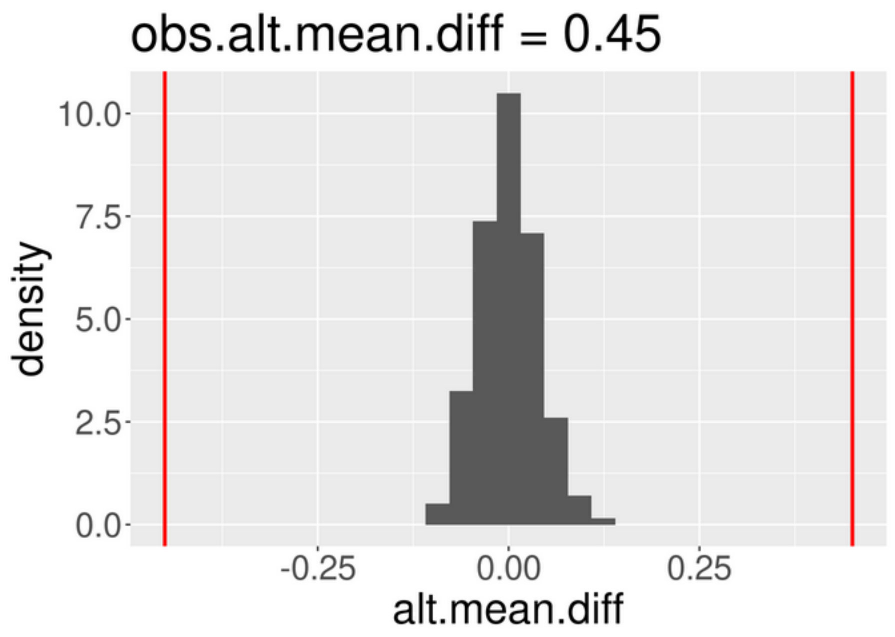


```r
# Observed mean differences
obs.null.mean.diff = mean(null.y[carrier == 1]) - mean(null.y[carrier == 0])
obs.alt.mean.diff = mean(alt.y[carrier == 1]) - mean(alt.y[carrier == 0])

# Histogram of test statistic distributions under the NULL
ggplot(plot_mean_diff, aes(x = null.mean.diff)) +
  geom_histogram(aes(y = stat(density) )) +
  geom_vline(xintercept = c(abs(obs.null.mean.diff), -abs(obs.null.mean.diff)),
             size = 1, col = "red") +
  labs(title = paste("obs.null.mean.diff =", round(abs(obs.null.mean.diff), 3)))
```

Permutation test: two-group mean difference

- Calculate the p-value from the numeric distribution: two-side

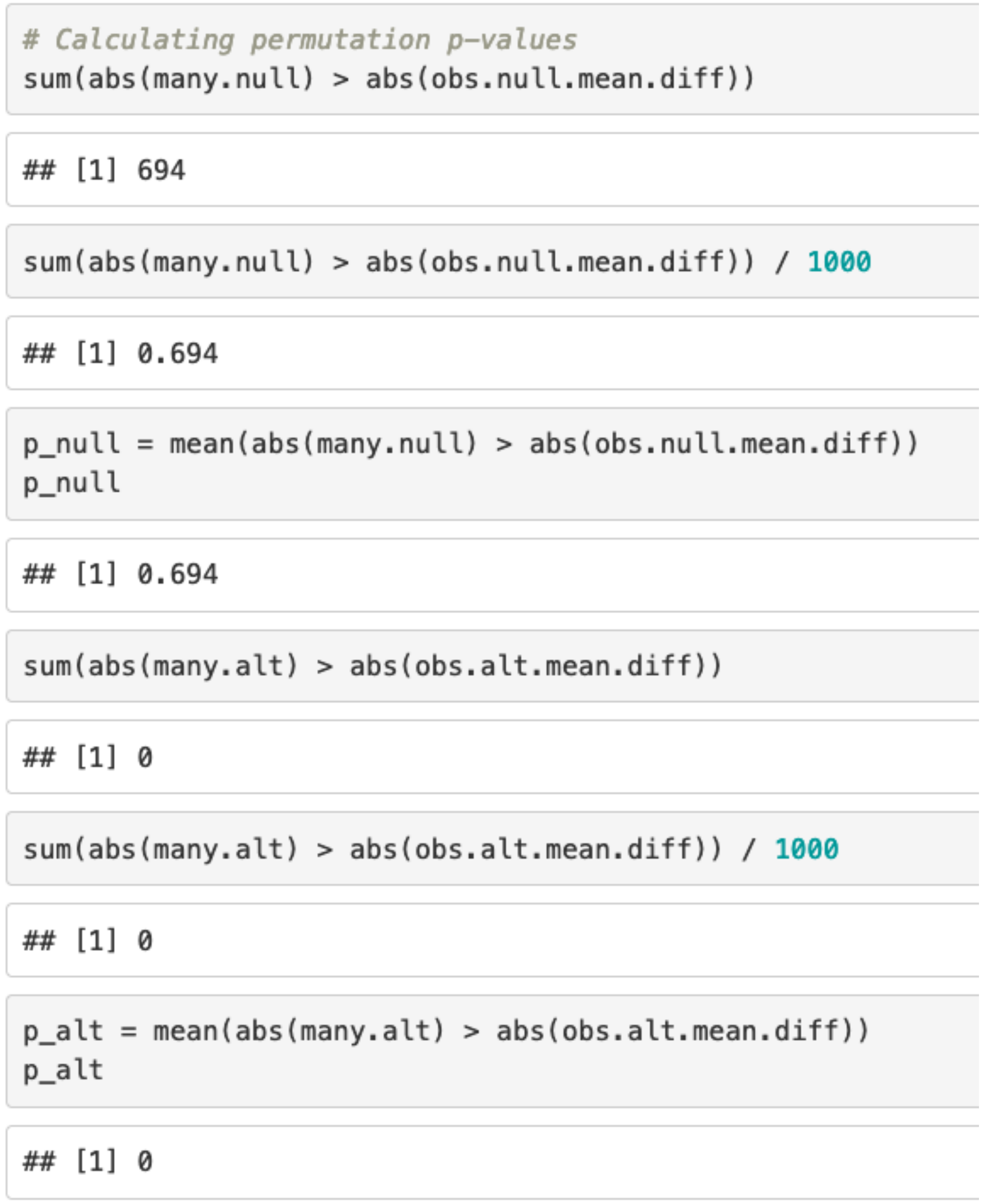




Read line: ± absolute value of observed mean difference

Permutation test: two-group mean difference

- Calculate the p-value from the numeric distribution
    - Count the total number of permutations that result in two-group mean differences more extreme than the observed mean difference, and then divide by the total number of permutations



```
# Calculating permutation p-values
sum(abs(many.null) > abs(obs.null.mean.diff))
```

```
## [1] 694
```

```
sum(abs(many.null) > abs(obs.null.mean.diff)) / 1000
```

```
## [1] 0.694
```

```
p_null = mean(abs(many.null) > abs(obs.null.mean.diff))
p_null
```

```
## [1] 0.694
```

```
sum(abs(many.alt) > abs(obs.alt.mean.diff))
```

```
## [1] 0
```

```
sum(abs(many.alt) > abs(obs.alt.mean.diff)) / 1000
```

```
## [1] 0
```

```
p_alt = mean(abs(many.alt) > abs(obs.alt.mean.diff))
p_alt
```

```
## [1] 0
```

Permutation p-value

Now consider a permutation test that randomly permutes the data B times (instead of all $\binom{N}{n}$ times). A permutation test approximates a randomization test. In fact, the permutation test can be analyzed using the following binomial random variable:

$$X_P = \#\text{ permutations out of B that give a more extreme value than the observed test statistic}$$
$$X_P \sim Bin(p_R, B)$$
$$SE(X_P) = \sqrt{\frac{p_R(1-p_R)}{B}} \approx \sqrt{\frac{\hat{p}_P(1-\hat{p}_P)}{B}}$$

Consider a situation where interest is in a small effect, say p-value≈ 0.01. The SE should be less than 0.001.

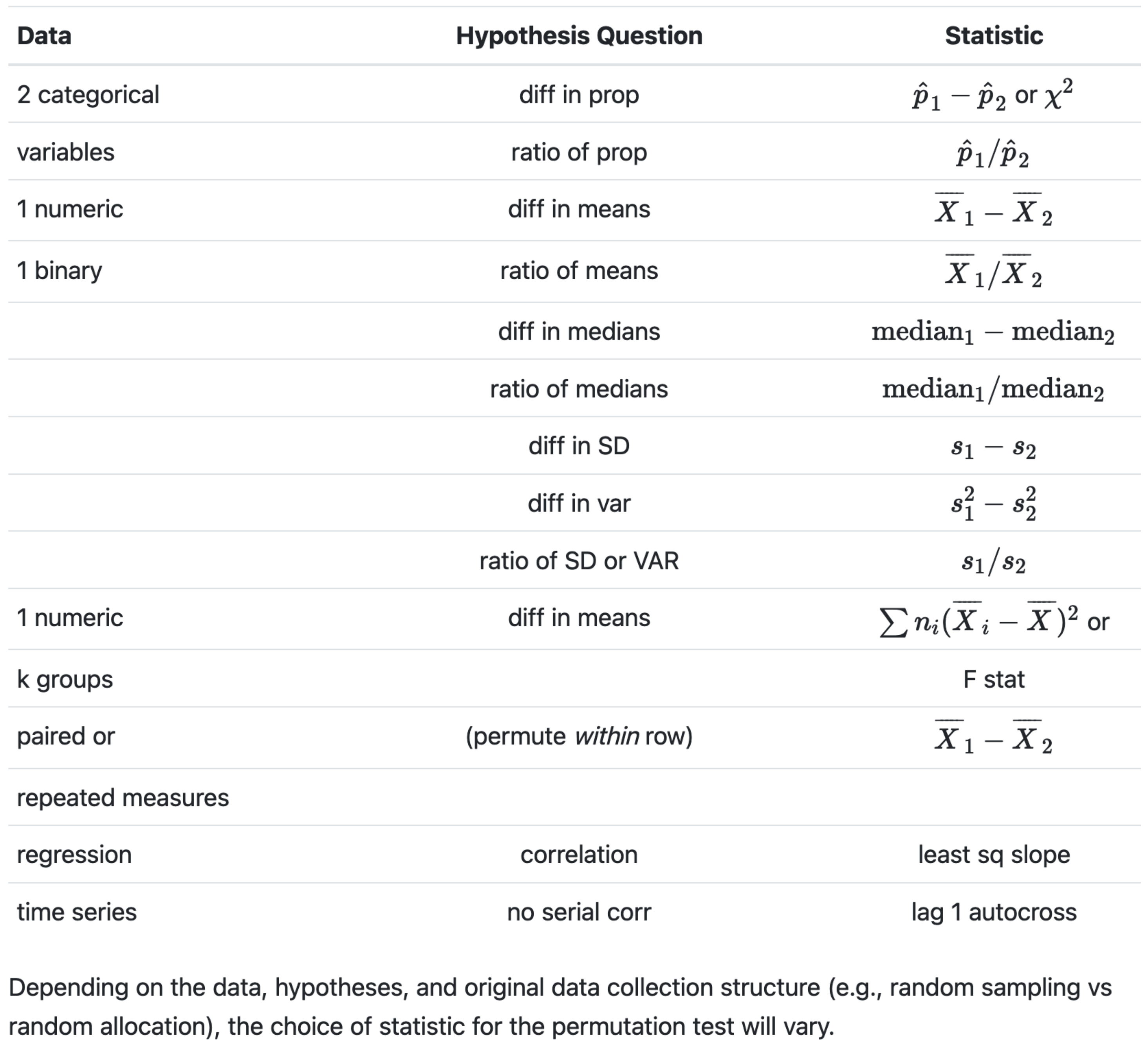
$$\begin{aligned} 0.001 &= \sqrt{(0.01)\cdot(0.99)/B} \\ B &= (0.01)\cdot(0.99)/(0.001)^2 \\ &= 9900 \end{aligned}$$

Another way to look at the same problem is to use the estimated p-value = $\hat{p}_P = \frac{X_P}{B}$ to come up with a confidence interval for p_R.

CI for $p_R \approx \hat{p}_P \pm 1.96\sqrt{\frac{\hat{p}_P(1-\hat{p}_P)}{B}}$

How many permutations are needed if using significance threshold 0.001?

Example test statistics



Data	Hypothesis Question	Statistic
2 categorical	diff in prop	$\hat{p}_1 - \hat{p}_2$ or χ^2
variables	ratio of prop	$\hat{p}_1 / \hat{p}_2$
1 numeric	diff in means	$\overline{X}_1 - \overline{X}_2$
1 binary	ratio of means	$\overline{X}_1 / \overline{X}_2$
	diff in medians	$\text{median}_1 - \text{median}_2$
	ratio of medians	$\text{median}_1 / \text{median}_2$
	diff in SD	$s_1 - s_2$
	diff in var	$s_1^2 - s_2^2$
	ratio of SD or VAR	s_1 / s_2
1 numeric	diff in means	$\sum n_i(\overline{X}_i - \overline{X})^2$ or
k groups		F stat
paired or	(permute *within* row)	$\overline{X}_1 - \overline{X}_2$
repeated measures		
regression	correlation	least sq slope
time series	no serial corr	lag 1 autocross

Depending on the data, hypotheses, and original data collection structure (e.g., random sampling vs random allocation), the choice of statistic for the permutation test will vary.

Pros and Cons of Permutation test

Pros	Cons
Require no known test statistic distribution under the null hypothesis	Give the same p-value as t-test for testing mean difference of two-group samples
No distribution assumptions for the observed data	Computationally expensive (could consider a flexible number of permutations per test or parallel computing)
Useful when there is no known distribution, aka, analytical pdf/cdf formula, for the test statistic under the null hypothesis	Require a large number of permutations when significance level is small to account for multiple testing

When permutation test is useful

- Suppose we test additive effects of 8 SNPs, one at a time, and we want to know if the most significant association is real.

- For each SNP, the Z-statistic from a logistic regression model has a Normal distribution.

- We need to know the distribution of the most extreme of eight Z-statistics.

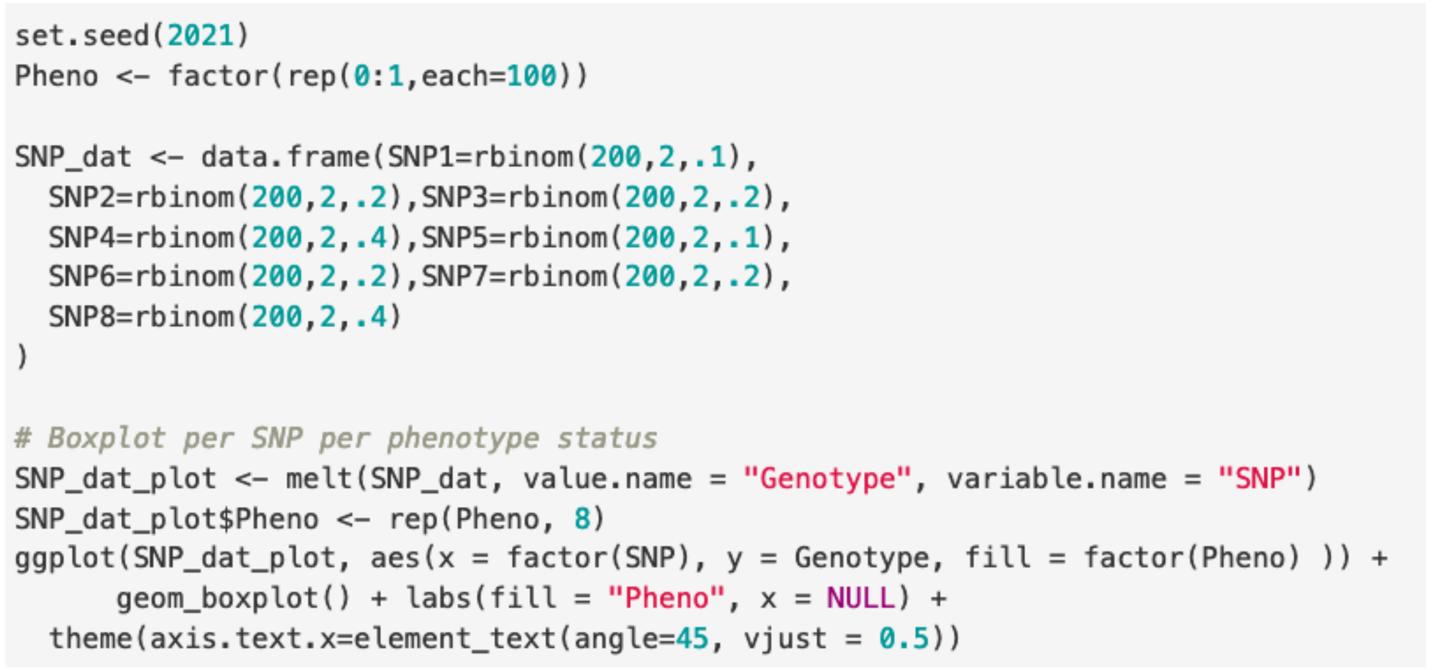
- This is not a standard distribution, but a permutation test is still straightforward. How?

Permutation for minimum order statistic

- Consider a binary disease phenotype with values "*0*" for control and values "*1*" for disease
- Test the association between a gene and the binary disease phenotype
- A total of eight SNPs within the test gene
- Test statistic: **minimum p-value** of single variant tests across all eight SNPs, i.e., **maximum abs(Z-score)** of single variant tests across all eight SNPs
- Analytical distribution for this minimum order statistic is not trivial, as **maximum abs(Z-score)** no longer follows a N(0, 1) distribution under the null

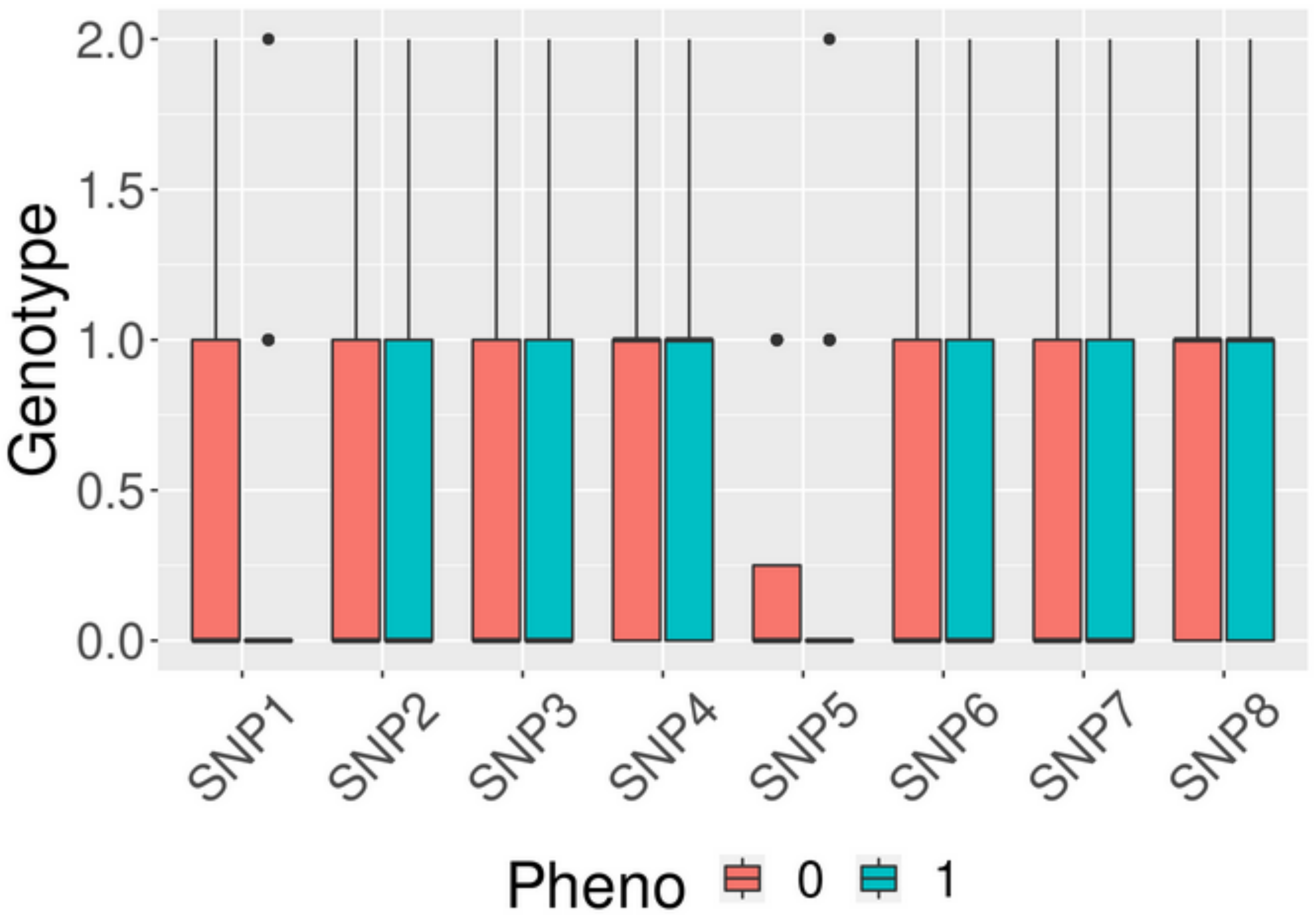
Simulate a binary disease phenotype and genotype data of eight SNPs

- Simulate 200 samples with 100 controls and 100 cases
- Simulate genotype data of eight SNPs with minor allele frequency (0.1, 0.2, 0.2, 0.4, 0.1, 0.2, 0.2, 0.4)



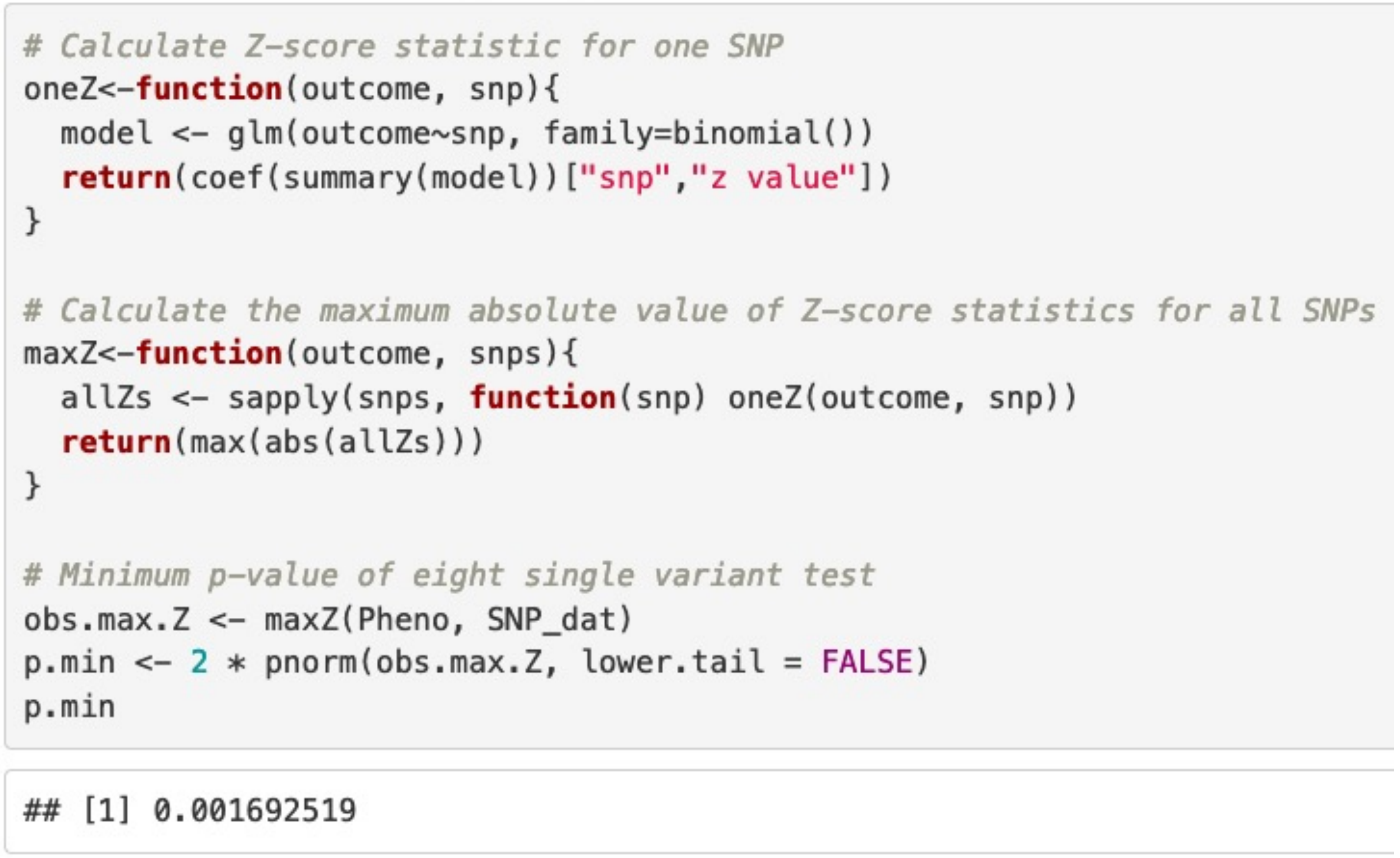
```r
set.seed(2021)
Pheno <- factor(rep(0:1,each=100))

SNP_dat <- data.frame(SNP1=rbinom(200,2,.1),
  SNP2=rbinom(200,2,.2),SNP3=rbinom(200,2,.2),
  SNP4=rbinom(200,2,.4),SNP5=rbinom(200,2,.1),
  SNP6=rbinom(200,2,.2),SNP7=rbinom(200,2,.2),
  SNP8=rbinom(200,2,.4)
)

# Boxplot per SNP per phenotype status
SNP_dat_plot <- melt(SNP_dat, value.name = "Genotype", variable.name = "SNP")
SNP_dat_plot$Pheno <- rep(Pheno, 8)
ggplot(SNP_dat_plot, aes(x = factor(SNP), y = Genotype, fill = factor(Pheno) )) +
      geom_boxplot() + labs(fill = "Pheno", x = NULL) +
  theme(axis.text.x=element_text(angle=45, vjust = 0.5))
```

Simulate a binary disease phenotype and genotype data of eight SNPs



Is this a null scenario?

Observed minimum p-value

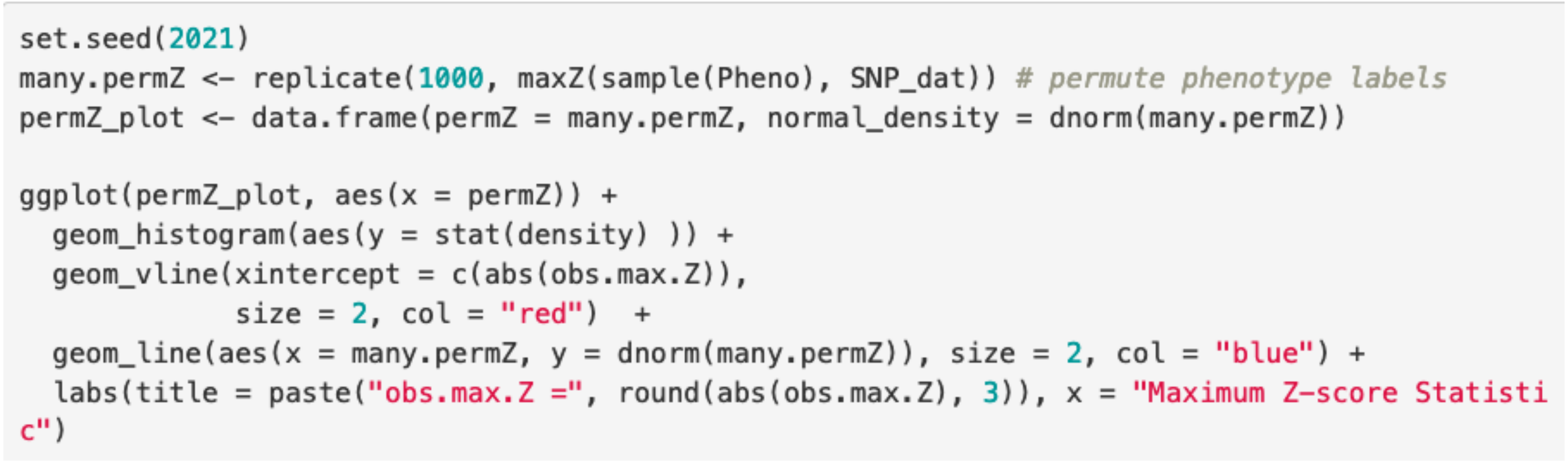


```r
# Calculate Z-score statistic for one SNP
oneZ<-function(outcome, snp){
  model <- glm(outcome~snp, family=binomial())
  return(coef(summary(model))["snp","z value"])
}

# Calculate the maximum absolute value of Z-score statistics for all SNPs
maxZ<-function(outcome, snps){
  allZs <- sapply(snps, function(snp) oneZ(outcome, snp))
  return(max(abs(allZs)))
}

# Minimum p-value of eight single variant test
obs.max.Z <- maxZ(Pheno, SNP_dat)
p.min <- 2 * pnorm(obs.max.Z, lower.tail = FALSE)
p.min
```

```
## [1] 0.001692519
```

Permutation for minimum order statistic

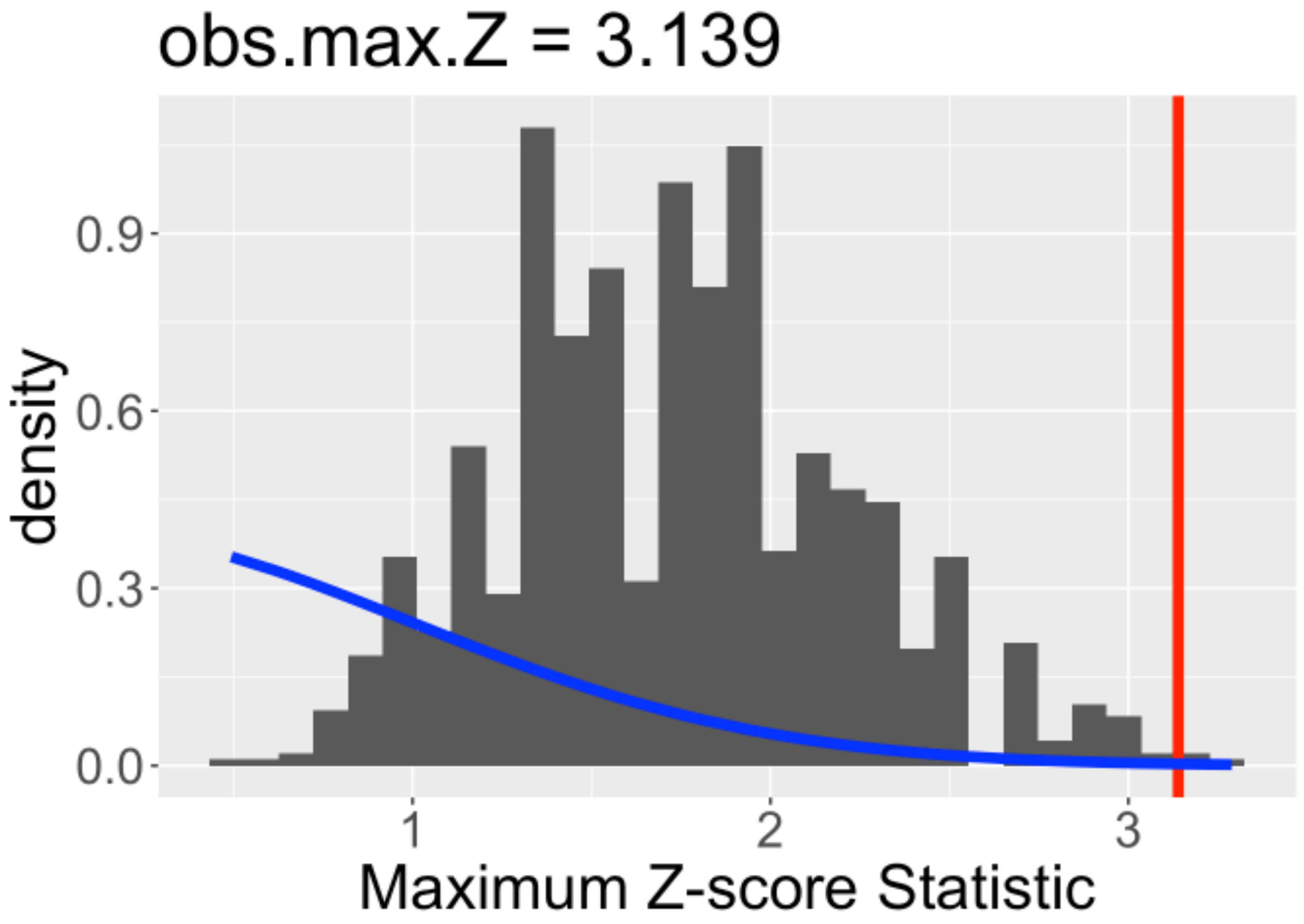
- Generating data sets under null hypothesis by permuting the disease phenotype
  - No association between the test disease phenotype and gene
  - AKA, disease phenotype is independent of all eight SNPs in the gene



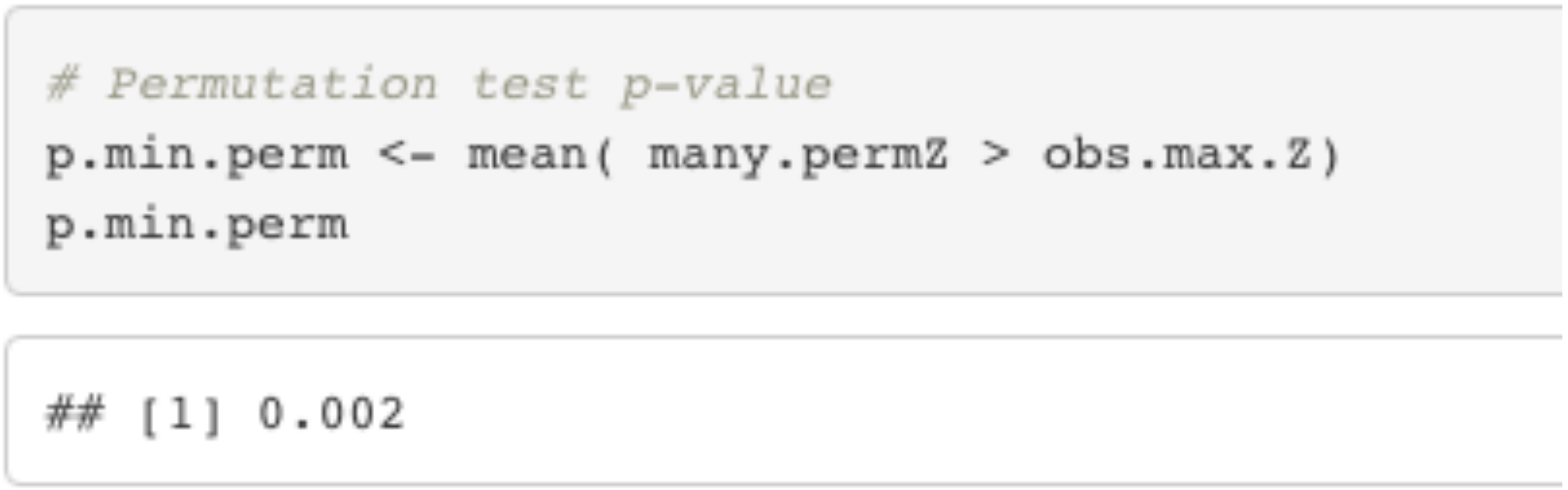
```r
set.seed(2021)
many.permZ <- replicate(1000, maxZ(sample(Pheno), SNP_dat)) # permute phenotype labels
permZ_plot <- data.frame(permZ = many.permZ, normal_density = dnorm(many.permZ))

ggplot(permZ_plot, aes(x = permZ)) +
  geom_histogram(aes(y = stat(density) )) +
  geom_vline(xintercept = c(abs(obs.max.Z)),
             size = 2, col = "red")  +
  geom_line(aes(x = many.permZ, y = dnorm(many.permZ)), size = 2, col = "blue") +
  labs(title = paste("obs.max.Z =", round(abs(obs.max.Z), 3)), x = "Maximum Z-score Statisti
c")
```

Distribution of permuted maximum Z-score statistic

- Blue line: N(0, 1) density function
- Red line: Observed maximum Z-score statistic value



Permutation test p-value for the maximum Z-score statistic



```r
# Permutation test p-value
p.min.perm <- mean( many.permZ > obs.max.Z)
p.min.perm
```

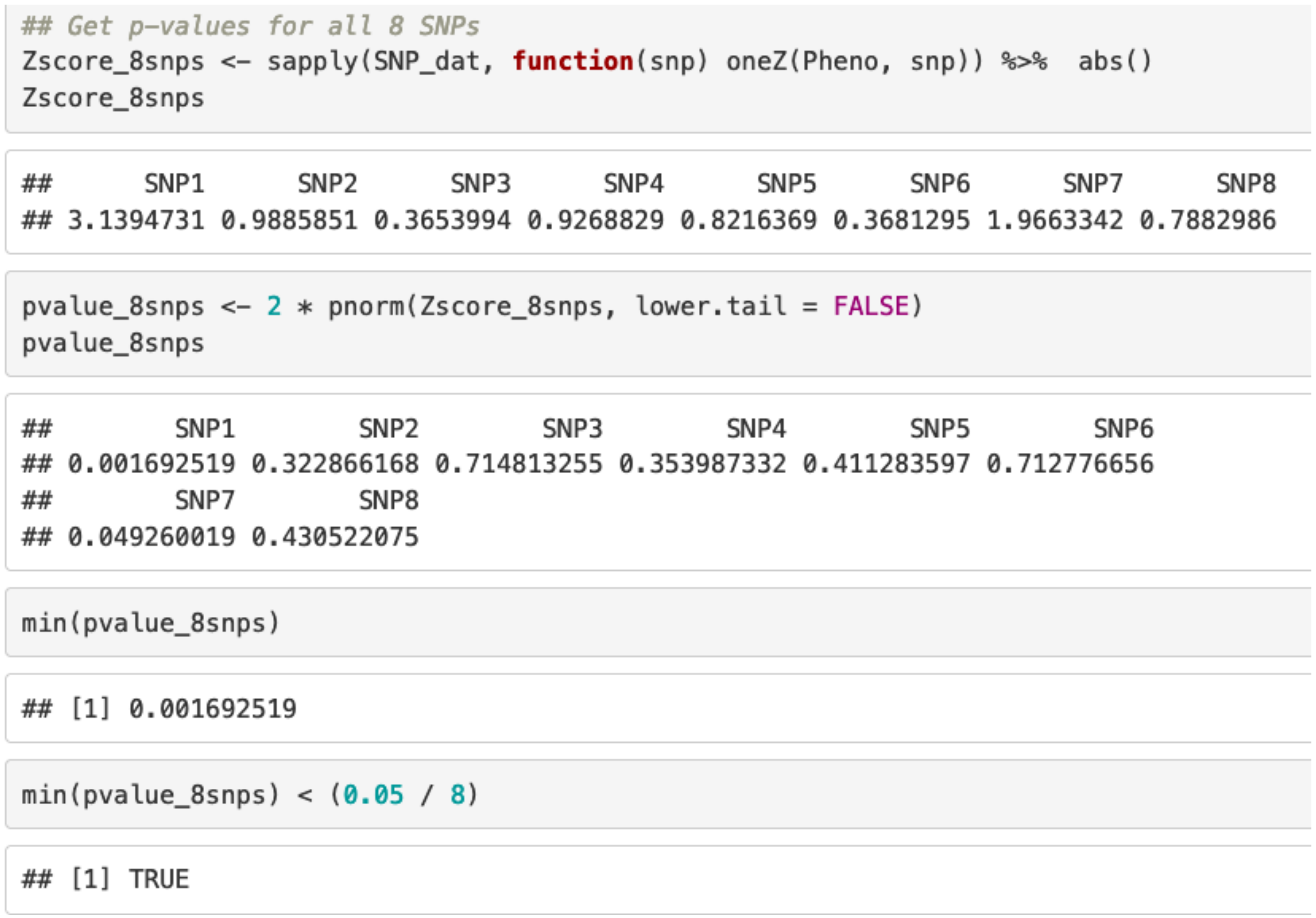
```
## [1] 0.002
```

You might get p-value < 0.05 for a set of simulated data under the null hypothesis!

What dose this mean?

Consider 8 individual tests

- Adjusting for multiple testing by Bonferroni correction, with significance level $\alpha = 0.05/8 = 0.00625$



```
## Get p-values for all 8 SNPs
Zscore_8snps <- sapply(SNP_dat, function(snp) oneZ(Pheno, snp)) %>%  abs()
Zscore_8snps
```

```
##      SNP1      SNP2      SNP3      SNP4      SNP5      SNP6      SNP7      SNP8
## 3.1394731 0.9885851 0.3653994 0.9268829 0.8216369 0.3681295 1.9663342 0.7882986
```

```
pvalue_8snps <- 2 * pnorm(Zscore_8snps, lower.tail = FALSE)
pvalue_8snps
```

```
##        SNP1        SNP2        SNP3        SNP4        SNP5        SNP6
## 0.001692519 0.322866168 0.714813255 0.353987332 0.411283597 0.712776656
##        SNP7        SNP8
## 0.049260019 0.430522075
```

```
min(pvalue_8snps)
```

```
## [1] 0.001692519
```

```
min(pvalue_8snps) < (0.05 / 8)
```

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

In-Class Exercise 1 (Assignment 8)

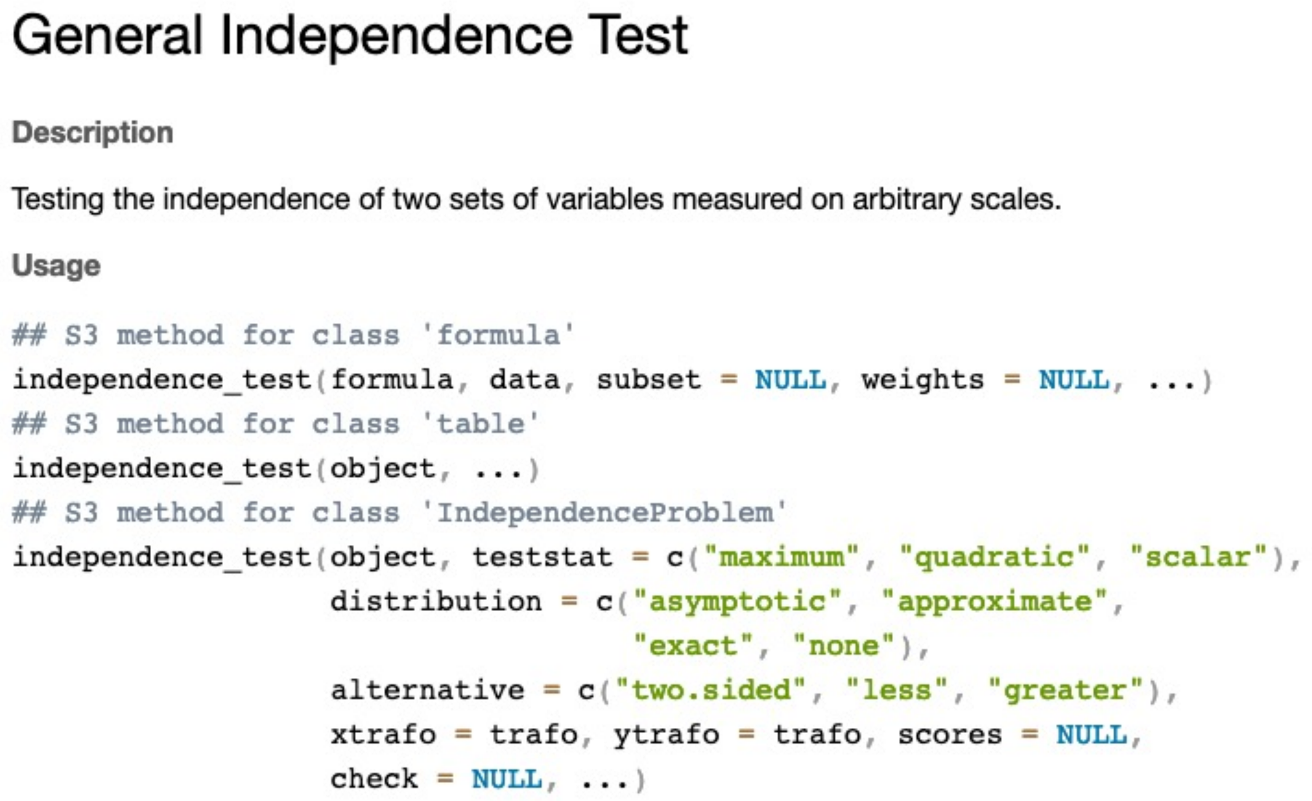
Write your own function for permutation test.

R package **coin** (**co**nditional **in**ference)

- The ***coin*** package implements a unified approach to **permutation tests** providing a huge class of independence tests for nominal, ordered, numeric, and censored data as well as multivariate data at mixed scales.
- Provide conditional versions (permutation tests) of classical tests, such as tests for location and scale problems in two or more samples, independence in two- or three-way contingency tables, or association problems for censored, ordered categorical or multivariate data.
- Approximations of the exact null distribution via the **limiting distribution (\`asymptotic\`)** or **conditional Monte Carlo resampling (\`approximate\`)** are available for every test procedure.
- The **exact** null distribution is currently available for univariate two-sample problems only.

R library *coin*

- Framework was developed by Strasser and Weber (1999): theoretical insights of a unified treatment of a huge class of permutation tests
- Generic functions for obtaining statistics, conditional expectation and covariance matrices as well as p-value, distribution, density and quantile functions for the reference distribution
  - Help to extract information from these objects
  - Conveniently interfaced in the function ***independence_test()***



General Independence Test

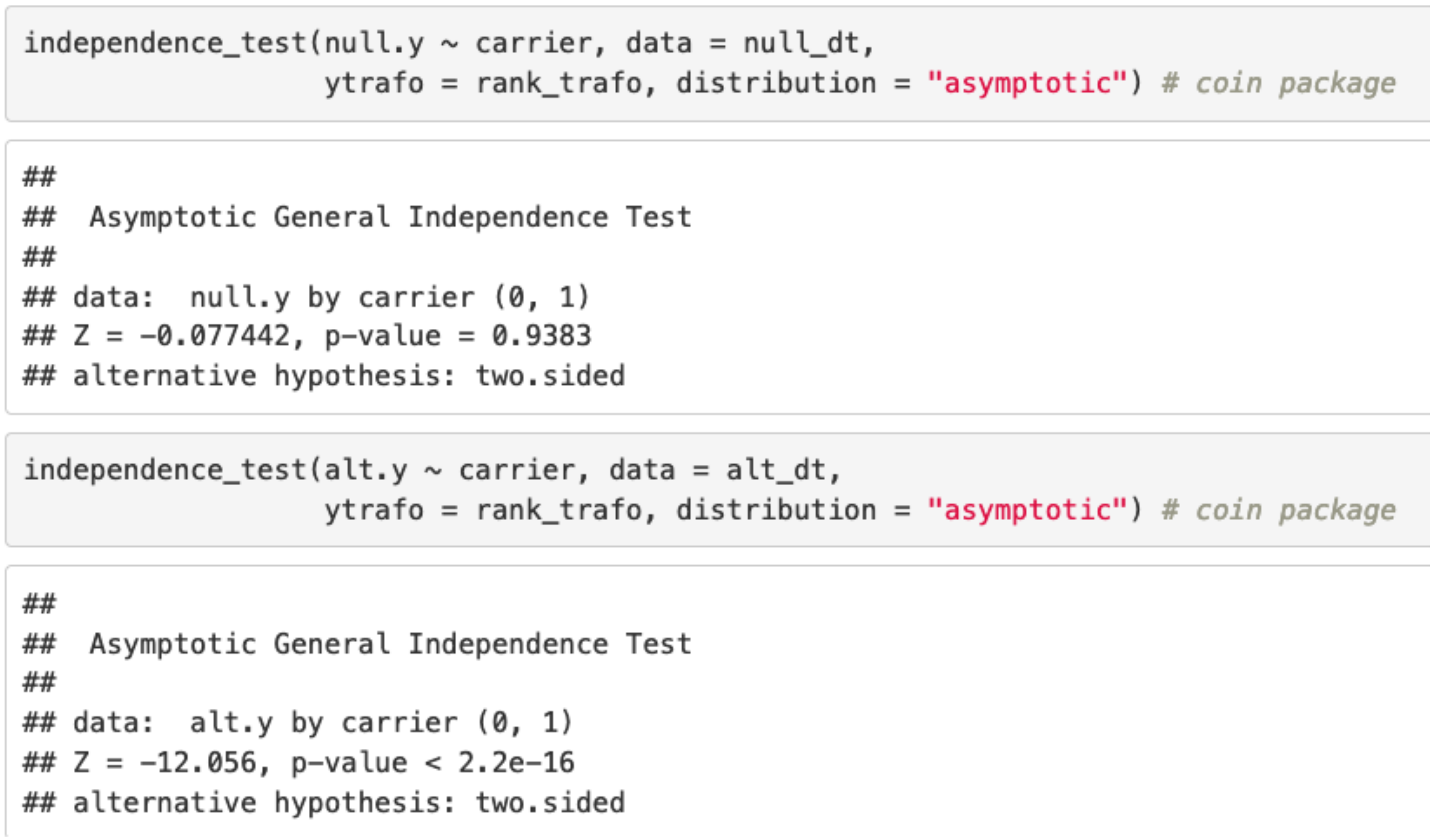
Description

Testing the independence of two sets of variables measured on arbitrary scales.

Usage

```
## S3 method for class 'formula'
independence_test(formula, data, subset = NULL, weights = NULL, ...)
## S3 method for class 'table'
independence_test(object, ...)
## S3 method for class 'IndependenceProblem'
independence_test(object, teststat = c("maximum", "quadratic", "scalar"),
                  distribution = c("asymptotic", "approximate",
                                   "exact", "none"),
                  alternative = c("two.sided", "less", "greater"),
                  xtrafo = trafo, ytrafo = trafo, scores = NULL,
                  check = NULL, ...)
```

Apply ` *independence_test()*` to Example 1



```r
independence_test(null.y ~ carrier, data = null_dt,
                  ytrafo = rank_trafo, distribution = "asymptotic") # coin package
```

```
##
##  Asymptotic General Independence Test
##
## data:  null.y by carrier (0, 1)
## Z = -0.077442, p-value = 0.9383
## alternative hypothesis: two.sided
```

```r
independence_test(alt.y ~ carrier, data = alt_dt,
                  ytrafo = rank_trafo, distribution = "asymptotic") # coin package
```

```
##
##  Asymptotic General Independence Test
##
## data:  alt.y by carrier (0, 1)
## Z = -12.056, p-value < 2.2e-16
## alternative hypothesis: two.sided
```

How dose *independence_test()* work?

- The data are pre-processed along with their transformations
- Deviations from independence are captured by a (possibly multivariate) linear statistic
- Standardized by conditional expectation and variance, and aggregated to a final test statistic
- Consider testing the independence between two variables Y and X, under a certain block structure of the observations (B) — for example, study centers in a multi-center randomized clinical trial where only a re-randomization of observations within blocks is admissible.

$$H_0 : D(\mathbf{Y}|\mathbf{X}, B) = D(\mathbf{Y}|B)$$

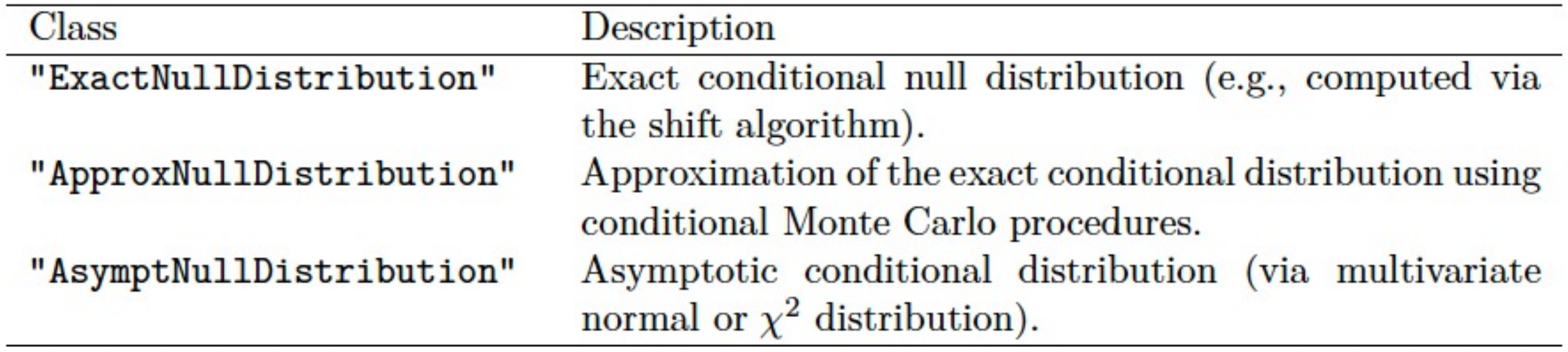
$$\mathbf{T} = \sum_{j=1}^{k} \mathbf{T}_j \in \mathbb{R}^{pq} \qquad \mathbf{T}_j = \operatorname{vec}\left(\sum_{i=1}^{n} I(b_i = j) w_i g(\mathbf{X}_i) h(\mathbf{Y}_i)^\top\right) \in \mathbb{R}^{pq}.$$

$$\mathbf{z} = \boldsymbol{\sigma}^{-1/2}(\mathrm{t} - \boldsymbol{\mu}).$$

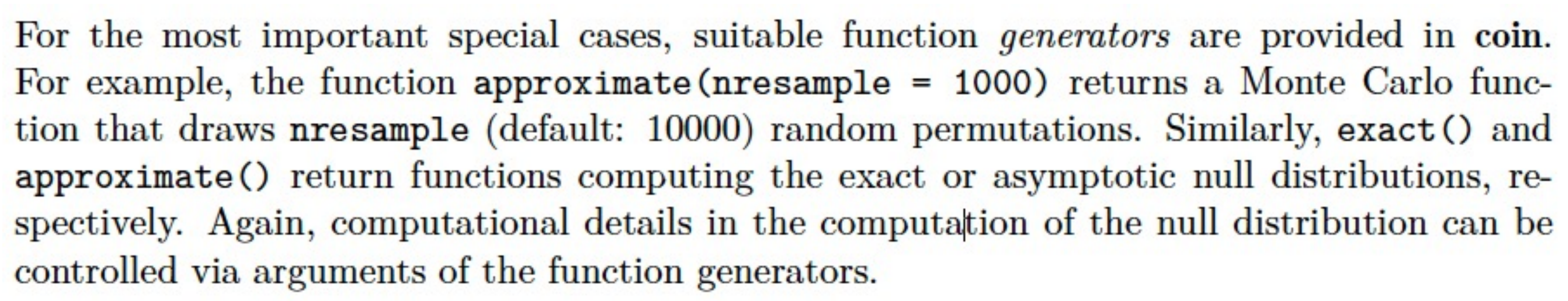
μ =Conditional expectation of T under Null

σ =Conditional variance of T under Null

Derive conditional null distribution



Class	Description
`"ExactNullDistribution"`	Exact conditional null distribution (e.g., computed via the shift algorithm).
`"ApproxNullDistribution"`	Approximation of the exact conditional distribution using conditional Monte Carlo procedures.
`"AsymptNullDistribution"`	Asymptotic conditional distribution (via multivariate normal or χ^2 distribution).



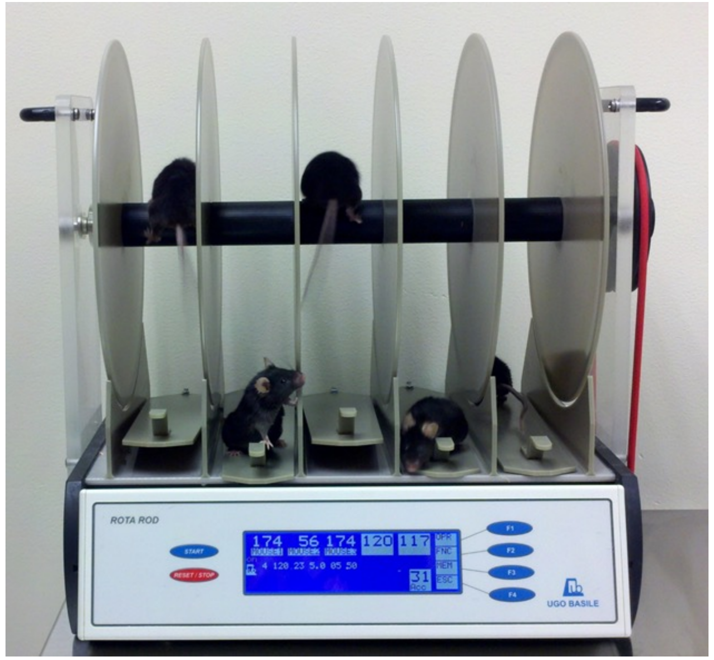
For the most important special cases, suitable function *generators* are provided in **coin**. For example, the function `approximate(nresample = 1000)` returns a Monte Carlo function that draws `nresample` (default: 10000) random permutations. Similarly, `exact()` and `approximate()` return functions computing the exact or asymptotic null distributions, respectively. Again, computational details in the computation of the null distribution can be controlled via arguments of the function generators.

Example: two-sample test (location)

group	time											
control	300	300	300	300	300	300	300	300	300	300	300	300
treatment	18	22	75	163	271	300	300	300	300	300	300	300

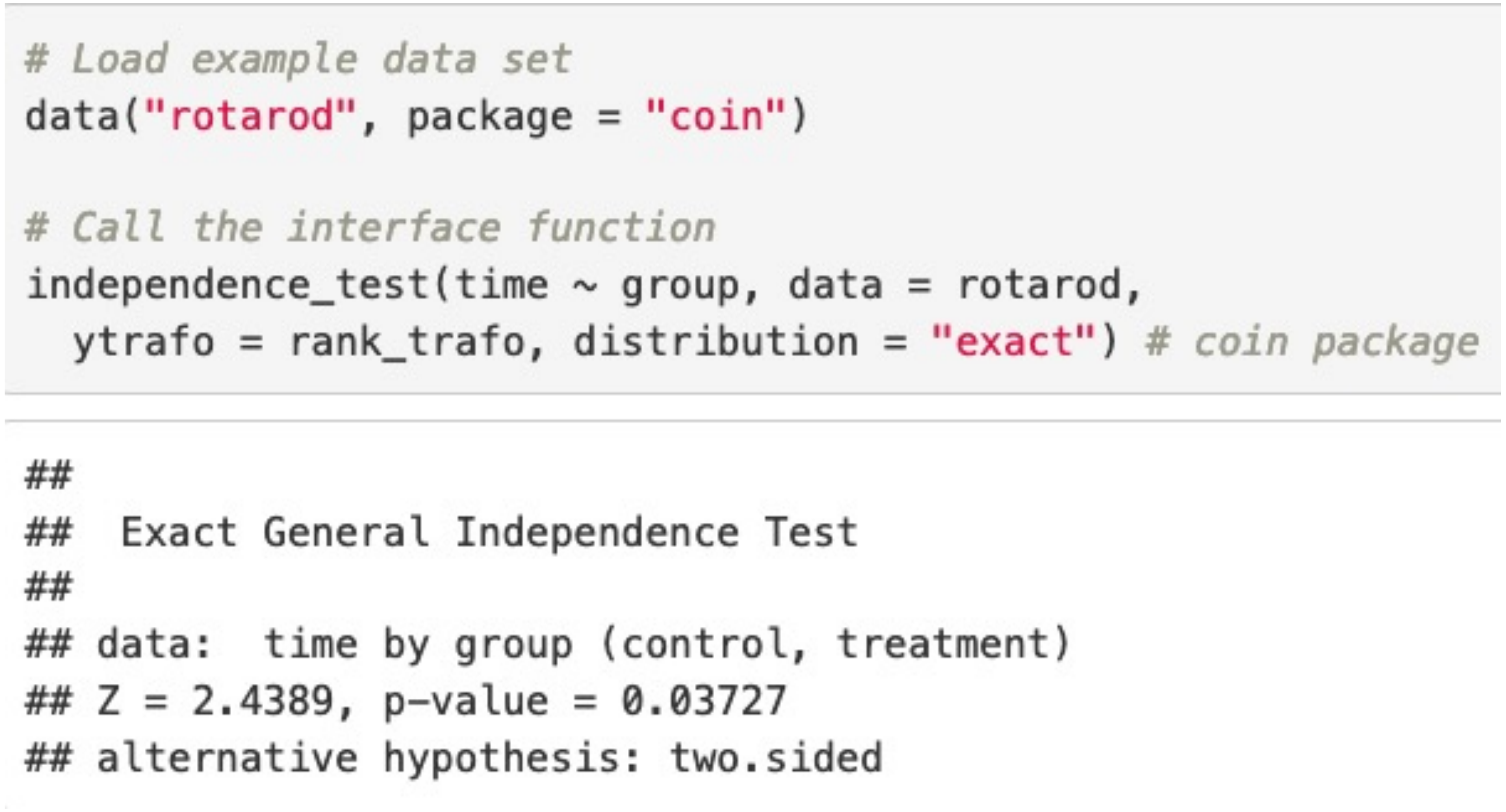
Table 1: The rotarod data: length of time on rotating cylinder by group.

- n = 24 rats received a fixed oral dose of a centrally acting muscle relaxant as active treatment or a saline solvent as control.
- The animals were placed on a rotating cylinder and the length of time each rat remained on the cylinder was measured, up to a maximum of 300 seconds.
- The rats were randomly assigned to the control and treatment groups.
- A permutation test is the appropriate way to investigate if the response is independent of the group assignment.



Example: two-sample test (location)

The data are particularly challenging because of the many ties in the (right-censored) response (19 observations take the maximal value 300) and the quasi-complete separation (smaller values of time are only observed in the treatment group).



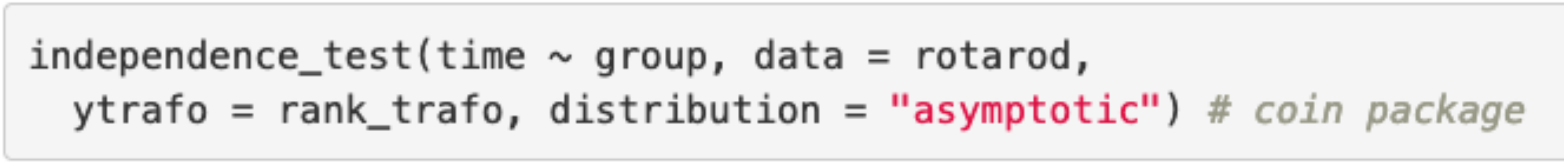
```r
# Load example data set
data("rotarod", package = "coin")

# Call the interface function
independence_test(time ~ group, data = rotarod,
  ytrafo = rank_trafo, distribution = "exact") # coin package
```

```
##
##  Exact General Independence Test
##
## data:  time by group (control, treatment)
## Z = 2.4389, p-value = 0.03727
## alternative hypothesis: two.sided
```

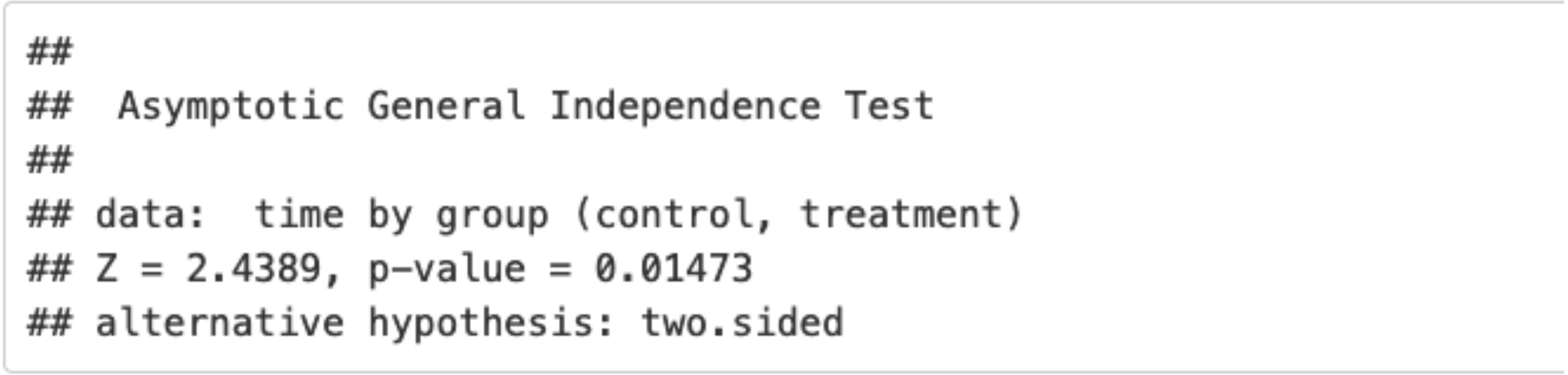
Here, the conditional **Wilcoxon-Mann-Whitney test** was performed via a **rank transformation of the response**, employing the **exact distribution** for obtaining the p-value.

Example: two-sample test (location)

- Input argument of *distribution = “asymptotic”* denotes the conditional null distribution of the test statistic can be approximated by its asymptotic distribution (default).



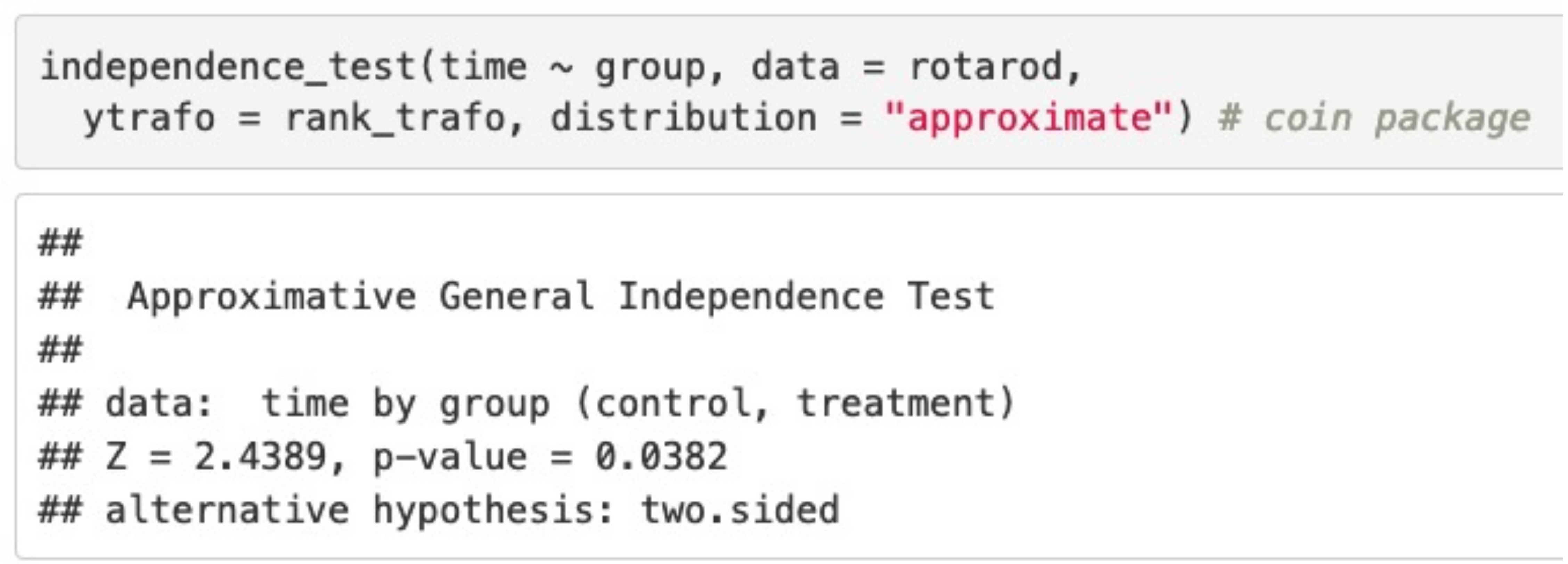
```
independence_test(time ~ group, data = rotarod,
  ytrafo = rank_trafo, distribution = "asymptotic") # coin package
```



```
##
##  Asymptotic General Independence Test
##
## data:  time by group (control, treatment)
## Z = 2.4389, p-value = 0.01473
## alternative hypothesis: two.sided
```

Example: two-sample test (location)

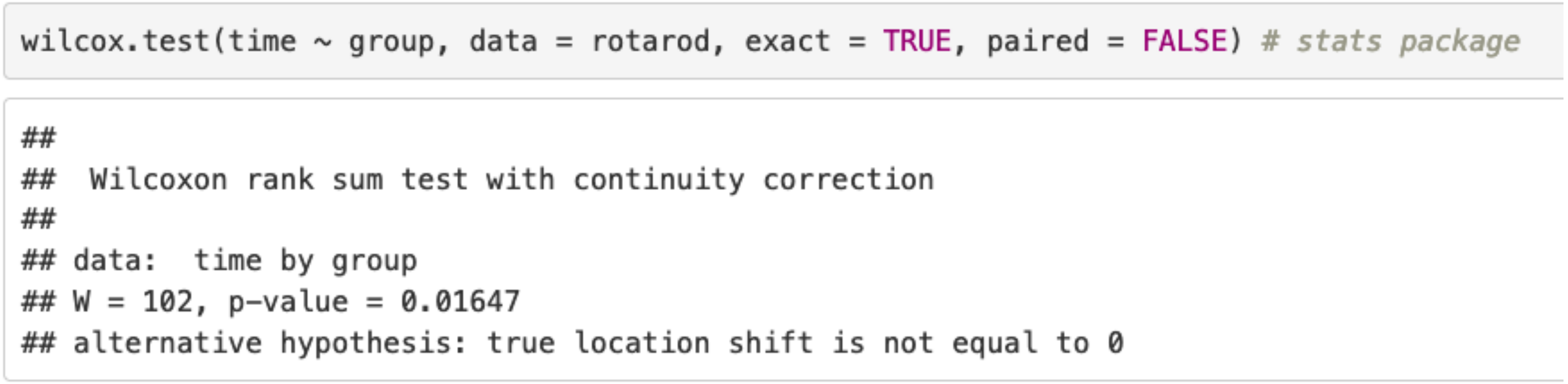
- Conditional null distribution of the test statistic can also be approximated via Monte Carlo resampling (*distribution = "approximate"*).



```r
independence_test(time ~ group, data = rotarod,
  ytrafo = rank_trafo, distribution = "approximate") # coin package
```

```
##
##  Approximative General Independence Test
##
## data:  time by group (control, treatment)
## Z = 2.4389, p-value = 0.0382
## alternative hypothesis: two.sided
```

Compare with *wilcox.test()* from *stats* package



```r
wilcox.test(time ~ group, data = rotarod, exact = TRUE, paired = FALSE) # stats package
```

```
##
##  Wilcoxon rank sum test with continuity correction
##
## data:  time by group
## W = 102, p-value = 0.01647
## alternative hypothesis: true location shift is not equal to 0
```

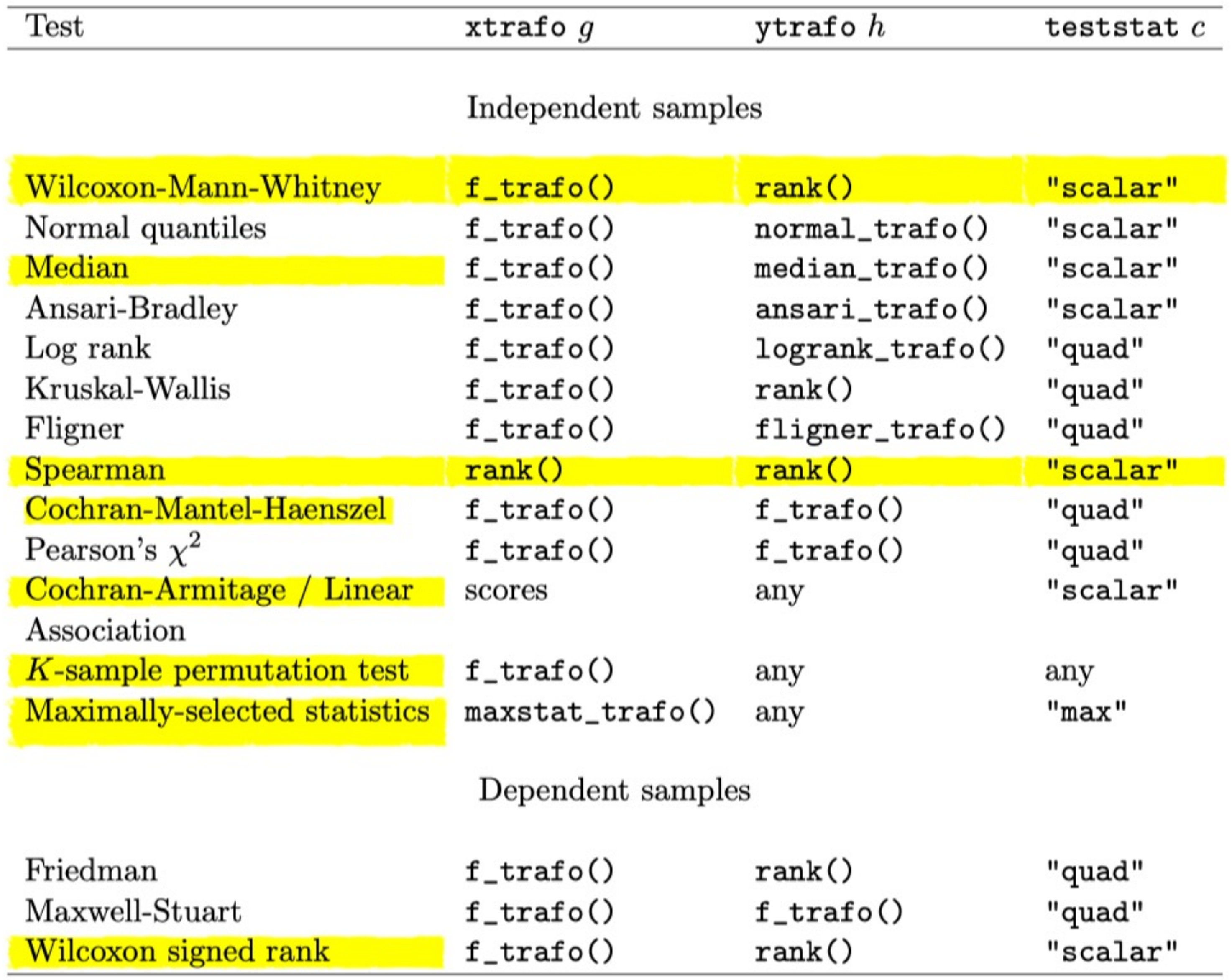
Compare with *t.test()* from *stats* package

```
t.test(time ~ group, data = rotarod) # stats package
```

```
##
##  Welch Two Sample t-test
##
## data:  time by group
## t = 2.3379, df = 11, p-value = 0.03932
## alternative hypothesis: true difference in means between group control and group treatmen
t is not equal to 0
## 95 percent confidence interval:
##    4.641392 153.858608
## sample estimates:
##   mean in group control mean in group treatment
##                  300.00                  220.75
```

t.test() is not appropriate for this dataset.

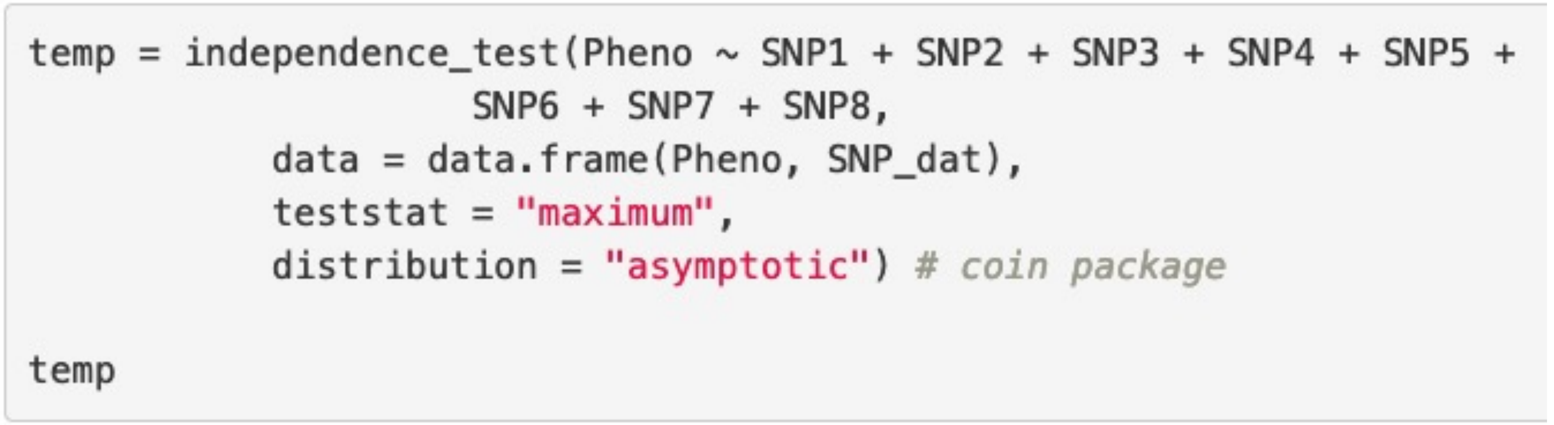
R library *coin*



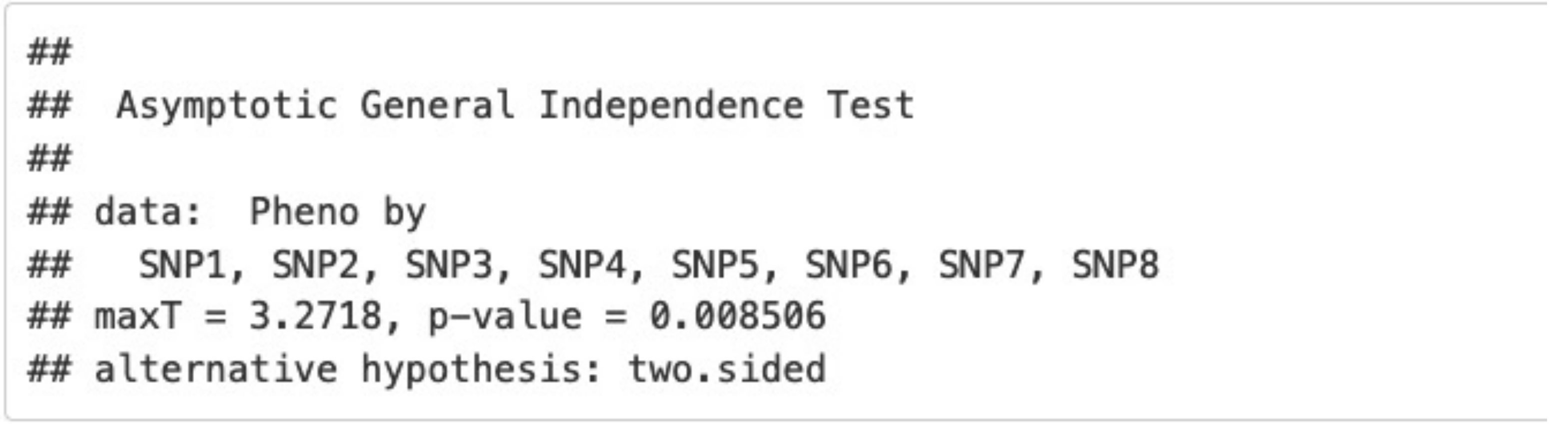
Test	xtrafo g	ytrafo h	teststat c
	Independent samples		
Wilcoxon-Mann-Whitney	`f_trafo()`	`rank()`	`"scalar"`
Normal quantiles	`f_trafo()`	`normal_trafo()`	`"scalar"`
Median	`f_trafo()`	`median_trafo()`	`"scalar"`
Ansari-Bradley	`f_trafo()`	`ansari_trafo()`	`"scalar"`
Log rank	`f_trafo()`	`logrank_trafo()`	`"quad"`
Kruskal-Wallis	`f_trafo()`	`rank()`	`"quad"`
Fligner	`f_trafo()`	`fligner_trafo()`	`"quad"`
Spearman	`rank()`	`rank()`	`"scalar"`
Cochran-Mantel-Haenszel	`f_trafo()`	`f_trafo()`	`"quad"`
Pearson's χ^2	`f_trafo()`	`f_trafo()`	`"quad"`
Cochran-Armitage / Linear Association	scores	any	`"scalar"`
K-sample permutation test	`f_trafo()`	any	any
Maximally-selected statistics	`maxstat_trafo()`	any	`"max"`
	Dependent samples		
Friedman	`f_trafo()`	`rank()`	`"quad"`
Maxwell-Stuart	`f_trafo()`	`f_trafo()`	`"quad"`
Wilcoxon signed rank	`f_trafo()`	`rank()`	`"scalar"`

Table 4: Representations of the conditional counterparts of important classical tests in **coin**.

Apply `*independence_test()*` to Example 2



```r
temp = independence_test(Pheno ~ SNP1 + SNP2 + SNP3 + SNP4 + SNP5 +
                    SNP6 + SNP7 + SNP8,
          data = data.frame(Pheno, SNP_dat),
          teststat = "maximum",
          distribution = "asymptotic") # coin package

temp
```



```
##
##  Asymptotic General Independence Test
##
## data:  Pheno by
##   SNP1, SNP2, SNP3, SNP4, SNP5, SNP6, SNP7, SNP8
## maxT = 3.2718, p-value = 0.008506
## alternative hypothesis: two.sided
```

R library *coin*

- Framework was developed by Strasser and Weber (1999)
  - Theoretical insights of a unified treatment of a huge class of permutation tests
- Salient parts of the Strasser-Weber framework are elucidated by Hothorn et al. (2006)
  - Introduce the package and illustrate the transition from theory to practice
- A thorough description of the software implementation is given by Hothorn et al. (2008)

References

- Strasser, H. and Weber, C. (1999). On the asymptotic theory of permutation statistics. Mathematical Methods of Statistics 8(2), 220–250.
- Hothorn, T., Hornik, K., van de Wiel, M. A. and Zeileis, A. (2006). A Lego system for conditional inference. The American Statistician 60(3), 257–263. doi: 10.1198/000313006X118430
- Hothorn, T., Hornik, K., van de Wiel, M. A. and Zeileis, A. (2008). Implementing a class of permutation tests: The coin package. Journal of Statistical Software 28(8), 1–23. doi: 10.18637/jss.v028.i08
- Permutation Tests in Computational Statistics
  - https://st47s.com/Math154/Notes/permschp.html

Next Lecture and Homeworks

- Homework 8 distributed and due 12/01
- Week 14 (12/02): Machine learning (Jingjing)
  - Homework 9 distributed and due 12/08
- Grading for homework 8 & 9
  - Submitted and show your work for all tasks: 10
  - Missed: 0
  - Feedbacks will still be provided

In-Class Exercise 2 (Assignment 8)

Use the R library ”coin”