

STT3030 - Labs #1

Arthur Charpentier

Automne 2024

Clean way to work

```
1 > getwd()  
2 [1] "/Users/arthurcharpentier"
```

- ▶ Set up a working directory (using `setwd()`)
- ▶ Make sure your data sets are in your working directory.
- ▶ You might want to have one directory per lab (project)

```
1 > setwd("/Users/arthurcharpentier/stt3030")  
2 > setwd("/Users/arthurcharpentier/stt3030/lab1")
```

R and vectors

- Keep in mind that R is a vector language, so I will use a lot things like

```
1 > resultat = sin(sqrt(1:10))
```

- instead of

```
1 > resultat = vector(mode="numeric", length = 10L)
2 > for (i in 1:length(resultat)) {
3 +   resultat[i] = sin(sqrt(i))
4 + }
```

Help function

Permutation test

A permutation test (also called re-randomization test or shuffle test) is an exact statistical hypothesis test making use of the proof by contradiction. A permutation test involves two or more samples. The null hypothesis is that all samples come from the same distribution $H_0 : F = G$. Under the null hypothesis, the distribution of the test statistic is obtained by calculating all possible values of the test statistic under possible rearrangements of the observed data. Permutation tests are, therefore, a form of resampling. W

See [Fisher \(1936\)](#) and [Pitman \(1937\)](#) for statistical background

- ▶ Suppose we want to run a [permutation test](#)
- ▶ if you know the function to use, that's easy

```
1 > library("exactRankTests")  
2 Package 'exactRankTests' is no longer under development.  
3 Please consider using package 'coin' instead.
```

Help function

```
1 > ?exactRankTests::perm.test  
2 > ?perm.test
```

- ▶ Add ? after a function name to access help
- ▶ Use google (or any web browser) to get help...

R code for a permutation test, with two samples



Tous

Images

Vidéos

Livres

Actualités

Web

Maps

: Plus

Outils



The R Project for Statistical Computing

<https://search.r-project.org/html> · [Traduire cette page](#) ⋮

Two-Sample Permutation Test

This function carries out an hypothesis **test** in which the null hypothesis is that the **two samples** are governed by the same underlying generative probability ...

Help function



R code for a permutation test, with two samples



Tous

Images

Vidéos

Actualités

Carte

Shopping

Aide

Chat



Canada (en) ▼

Filtre parental : modéré ▼

Date indifférente ▼

<https://search.r-project.org> > CRAN > refmans > EnvStats > html > twoSamplePermutationTestLoca...

R: Two-Sample or Paired-Sample Randomization (Permutation) Test...

The **two-sample permutation test** works by simply enumerating all possible **permutations** of group assignments, and for each **permutation** computing the difference between the measures of location for each group (Manly, 2007, p. 113; Efron and Tibshirani, 1993, p. 202). The measure of location for a gro...

twosamples: Fast Permutation B...

Package. Fast Permutation Based Two Sample Tests. 2.0.1 Fast randomization...

R: Two-Sample Permutation Test

Value. A list with three components: the value of the statistic for the original tw...

R: One and Two Sample Permut...

Details. The permutation test is performed for integer valued...

Help with Chat-GPT (don't google it, prompt it)

- ▶ you write what you want to do in plain English and get the R code instantly



ChatGPT



Chat

Help with Chat-GPT (don't google it, prompt it)

what would be the R code for a permutation test, with two samples x and y ?

Certainly! Below is the R code for a permutation test with two samples, x and y. This script calculates the p-value for the hypothesis that the two samples come from the same distribution by comparing the observed difference in means to the distribution of differences obtained by randomly permuting the combined data.

```
1 # Sample data
2 x <- c(1, 2, 3, 4, 5)
3 y <- c(6, 7, 8, 9, 10)
4
5 # Function to perform permutation test
6 permutation_test <- function(x, y, num_permutations = 10000) {
7   # Combine the data
8   combined <- c(x, y)
9   n_x <- length(x)
```


Help with Chat-GPT (don't google it, prompt it)

```
10  n_y <- length(y)
11
12  # Calculate the observed difference in means
13  observed_diff <- mean(x) - mean(y)
14
15  # Initialize a vector to store the permutation differences
16  perm_diffs <- numeric(num_permutations)
17
18  # Perform the permutations
19  for (i in 1:num_permutations) {
20    # Permute the combined data
21    permuted <- sample(combined)
22
23    # Split the permuted data into two new samples
24    perm_x <- permuted[1:n_x]
25    perm_y <- permuted[(n_x + 1):(n_x + n_y)]
26
27    # Calculate the difference in means for the permuted samples
```

Help with Chat-GPT (don't google it, prompt it)

```
28     perm_diffs[i] <- mean(perm_x) - mean(perm_y)
29   }
30
31   # Calculate the p-value
32   p_value <- mean(abs(perm_diffs) >= abs(observed_diff))
33
34   # Return the observed difference and p-value
35   list(observed_diff = observed_diff, p_value = p_value)
36 }
37
38 # Run the permutation test
39 result <- permutation_test(x, y)
40 print(result)
```

Help with Chat-GPT (don't google it, prompt it)

is there a R package to do the same ?

Yes, there are R packages that can perform permutation tests. One commonly used package is `coin`. The `coin` package provides a comprehensive framework for conditional inference procedures, including permutation tests. Below is an example of how to perform a permutation test using the `coin` package.

First, you'll need to install the `coin` package if you haven't already:

```
1 install.packages("coin")
```

Then, you can use the following code to perform a permutation test on two samples `x` and `y`:

Help with Chat-GPT (don't google it, prompt it)

```
1 # Load the coin package
2 library(coin)
3
4 # Sample data
5 x <- c(1, 2, 3, 4, 5)
6 y <- c(6, 7, 8, 9, 10)
7
8 # Combine the data into a data frame
9 data <- data.frame(
10   values = c(x, y),
11   group = factor(rep(c("x", "y"), times = c(length(x), length(y))))
12 )
13
14 # Perform the permutation test
15 result <- oneway_test(values ~ group, data = data, distribution = "
    approximate")
16
17 # Print the result
```

Help with Chat-GPT (don't google it, prompt it)

```
18 print(result)
```

In this code:

- The `coin` package is used to perform the permutation test.
- The data from the two samples `x` and `y` are combined into a single data frame with a grouping factor.
- The `oneway_test` function from the `coin` package is used to perform the permutation test.
- The result of the test is printed.

Help with Chat-GPT (don't google it, prompt it)

can you explain the maths of that test ?

Certainly! The permutation test, also known as a randomization test, is a non-parametric statistical test used to determine if there is a significant difference between two groups. Unlike traditional parametric tests (like the t-test), permutation tests do not make assumptions about the distribution of the data, making them robust and versatile.

Permutation Test: Mathematical Explanation

- ▶ **Null Hypothesis (H_0):** The two samples come from the same distribution.
- ▶ **Alternative Hypothesis (H_1):** The two samples come from different distributions.

Procedure:

Help with Chat-GPT (don't google it, prompt it)

1. Compute the test statistic for the observed data:

$$T_{\text{obs}} = \bar{x} - \bar{y}$$

2. Combine and permute the data.
3. Compute the test statistic for each permutation:

$$T_i = \bar{x}_i - \bar{y}_i$$

4. Generate the permutation distribution.
5. Calculate the p-value:

$$p = \frac{\sum_{i=1}^B I(|T_i| \geq |T_{\text{obs}}|)}{B}$$

Help with Chat-GPT (don't google it, prompt it)

Example: Difference in Means Consider two samples:

$$x = \{1, 2, 3, 4, 5\}, \quad y = \{6, 7, 8, 9, 10\}$$

1. Observed Test Statistic:

$$T_{\text{obs}} = \bar{x} - \bar{y} = 3 - 8 = -5$$

2. Permutation Process:

- Combine x and y :

$$\text{combined} = 1, 2, 3, 4, 5, 6, 7, 8, 9, 10$$

- Shuffle and split into two new groups (of sizes 5 each):
 - Permutation 1: $\{1, 2, 3, 4, 6\}$ and $\{5, 7, 8, 9, 10\}$
 - Permutation 2: $\{2, 3, 4, 5, 7\}$ and $\{1, 6, 8, 9, 10\}$
 - ... and so on.

3. Compute Test Statistic for Each Permutation:

Help with Chat-GPT (don't google it, prompt it)

- For Permutation 1:

$$T_1 = (\bar{x} - \bar{y}) = \frac{16}{5} - \frac{39}{5} = -4.6$$

- Repeat this process for all permutations.

4. P-value Calculation: Suppose after 10,000 permutations, 200 permutations resulted in a test statistic as extreme or more extreme than $|-5|$:

$$p = \frac{200}{10000} = 0.02$$

If the p-value is less than the chosen significance level (e.g., 0.05), you reject the null hypothesis, suggesting that there is a significant difference between the two samples. The permutation test involves calculating the observed test statistic, repeatedly shuffling and splitting the combined data to form new samples, calculating the test statistic for each permutation, and then comparing the observed statistic to the distribution of permuted statistics to determine the p-value. This method is powerful

Help with Chat-GPT (don't google it, prompt it)

because it relies on the data itself rather than assumptions about the underlying distributions.

```
1 > result
2
3   Approximative Two-Sample Fisher-Pitman
4   Permutation Test
5
6 data:  values by group (x, y)
7 Z = -2.6112, p-value = 0.0094
8 alternative hypothesis: true mu is not equal to 0
```

Help with Chat-GPT (don't google it, prompt it)

why T_{obs} is not the same as $Z = -2.6112$ returned by the R function ?

Great question! The difference between T_{obs} and the Z -value returned by the coin package can be attributed to the different test statistics used by each method. Let me explain in detail.

T_{obs} in Manual Permutation Test

In the manual permutation test example provided, T_{obs} is the observed difference in means between the two samples. This is a straightforward calculation:

$$T_{\text{obs}} = \bar{x} - \bar{y}$$

where \bar{x} and \bar{y} are the sample means of the two groups x and y , respectively.

Z -value in the coin Package

Help with Chat-GPT (don't google it, prompt it)

The coin package in R, when performing permutation tests, often standardizes the test statistic. Specifically, the `oneway_test` function computes a standardized statistic under the null hypothesis, which is typically a Z -value in the context of asymptotic approximations.

Example Calculation:

Here's the detailed step-by-step process:

Observed Test Statistic:

In the manual method, the observed test statistic T_{obs} is calculated as the difference in means:

$$T_{\text{obs}} = \bar{x} - \bar{y} = 3 - 8 = -5$$

Standardization in the coin Package:

The coin package uses a different approach to calculate the test statistic. It computes a standardized version of the test statistic, which is often akin to the Z -score. This standardization takes into account the variance and sample sizes.

Help with Chat-GPT (don't google it, prompt it)

The Z-score is typically calculated as:

$$Z = \frac{(\bar{x} - \bar{y}) - \text{Expected Difference under } H_0}{\text{Standard Error}}$$

However, the coin package's `oneway_test` function may use a different form of the statistic that is robust and appropriate for the permutation context. This involves more complex adjustments and standardizations that reflect the permutation distribution.

```
1 > x = rnorm(10)
2 > y = rnorm(10)
3 > data <- data.frame(
4 +   values = c(x, y),
5 +   group = factor(rep(c("x", "y"), times = c(length(x), length(y)))
6 + )
```

Help with Chat-GPT (don't google it, prompt it)

```
7 > result <- oneway_test(values ~ group, data = data, distribution = "  
  approximate")  
8 > result  
9  
10 Approximative Two-Sample Fisher-Pitman  
11 Permutation Test  
12  
13 data: values by group (x, y)  
14 Z = -0.83915, p-value = 0.4184  
15 alternative hypothesis: true mu is not equal to 0
```

- ▶ Include details in your query to get more relevant answers
e.g. "Comment the code liberally to explain what each piece does and why it's written that way"
- ▶ Specify the steps required to complete a task
e.g.

Help with Chat-GPT (don't google it, prompt it)

- ▶ See more on prompt engineering, prompts for data science, or prompt engineering guide

using osmdata R library, write R codes to find locations of all Jean Coutu stores in Montréal, Qc. Then use mapview to visualize those stores on a map

Certainly! Below is an R script that uses the osmdata and mapview packages to find and visualize all Jean Coutu stores in Montréal, Quebec.

```
1 # Install and load necessary packages
2 if (!requireNamespace("osmdata", quietly = TRUE)) {
3   install.packages("osmdata")
4 }
5 if (!requireNamespace("sf", quietly = TRUE)) {
6   install.packages("sf")
7 }
```

Help with Chat-GPT (don't google it, prompt it)

```
8 if (!requireNamespace("mapview", quietly = TRUE)) {  
9   install.packages("mapview")  
10 }  
11  
12 library(osmdata)  
13 library(sf)  
14 library(mapview)  
15  
16 # Define the bounding box for Montr al , QC  
17 montr al _bbox <- getbb("Montr al , Qu bec")  
18  
19 # Query OSM for all Jean Coutu locations in Montr al  
20 jean_coutu_query <- opq(bbox = montr al _bbox) %>%  
21   add_osm_feature(key = "name", value = "Jean Coutu")  
22  
23 # Extract data  
24 jean_coutu_data <- osmdata_sf(jean_coutu_query)  
25
```


Help with Chat-GPT (don't google it, prompt it)

```
26 # Check the structure of the returned data
27 print(names(jean_coutu_data))
28
29 # Extract the points data (store locations)
30 jean_coutu_locations <- jean_coutu_data$osm_points
31
32 # Visualize the locations on a map using mapview
33 mapview(jean_coutu_locations)
```

Explanation:

Install and load packages: The script checks if the required packages (`osmdata`, `sf`, and `mapview`) are installed. If not, it installs them. Then, it loads these packages into the R session.

Define bounding box: The `getbb` function is used to get the bounding box for Montréal, Quebec, which serves as the spatial extent for our query.

Help with Chat-GPT (don't google it, prompt it)

Query OSM data: We use the `opq` function to initiate an Overpass API query with the defined bounding box. We then add a feature using `add_osm_feature` to filter for locations with the name "Jean Coutu".

Extract data: The `osmdata_sf` function is used to retrieve the data as an `sf` object (simple features). This object contains multiple layers, including points, lines, polygons, etc.

Extract store locations: We specifically extract the point data from the retrieved OSM data, which represents the locations of the stores.

Visualize: Finally, we use `mapview` to visualize these store locations on an interactive map.

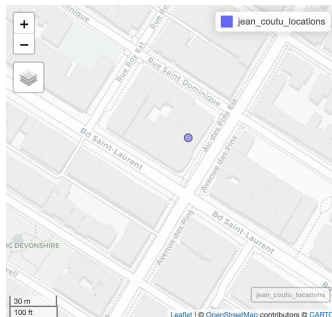
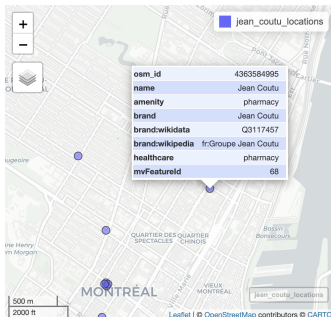
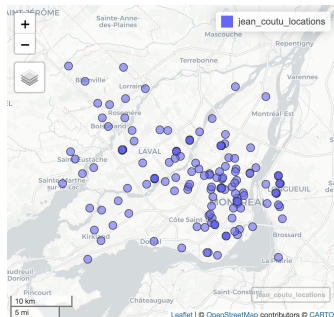
Running the Script:

Make sure you have an active internet connection since the script relies on querying the OpenStreetMap database through the Overpass API.

When you run this script, a map will appear in the RStudio Viewer or your default browser, showing the locations of Jean Coutu stores across Montréal.

Help with Chat-GPT (don't google it, prompt it)

If the map doesn't display any stores, it might be due to a lack of precise tagging in OSM. You can try refining the query by including different filters or tags.



Dealing with objects

```
1 > str(result)
2 Formal class 'ScalarIndependenceTest' [package "coin"] with 7 slots
3 ..@ parameter      : chr "mu"
4 ..@ nullvalue      : num 0
5 ..@ distribution:Formal class 'ApproxNullDistribution' [package "
  coin"] with 12 slots
6 .. .. ..@ seed      : int [1:626] 10403 202 1544423891
  -753397414 -138716415 524441244 1725742425 -816412468 1172112304
  -939627685 ...
7 .. .. ..@ nresample : int 10000
8
9 ..@ statistic      :Formal class 'ScalarIndependenceTestStatistic' [
  package "coin"] with 15 slots
10 .. .. ..@ alternative      : chr "two.sided"
11 .. .. ..@ paired          : logi FALSE
12 .. .. ..@ teststatistic    : num -0.839
13 .. .. ..@ standardizedlinearstatistic: num -0.839
14 .. .. ..@ linearstatistic  : num [1, 1] -5.54
```

Dealing with objects

- ▶ We can get only the numerical value of the test statistic Z

```
1 > result@statistic@teststatistic  
2 [1] -0.8391545
```

- ▶ S3 class, call elements using `$`
- ▶ S4 class, call elements (slots) using `@`

Dealing with objects

► S3 class

```
1 > regression = lm(dist ~ speed, data = cars)
2 > str(regression)
3 List of 12
4 $ coefficients : Named num [1:2] -17.58 3.93
5 ..- attr(*, "names")= chr [1:2] "(Intercept)" "speed"
6 $ residuals : Named num [1:50] 3.85 11.85 -5.95 12.05 2.12 ...
7 ..- attr(*, "names")= chr [1:50] "1" "2" "3" "4" ...
8 $ effects : Named num [1:50] -303.914 145.552 -8.115 9.885 0.194
9 ...
10 ..- attr(*, "names")= chr [1:50] "(Intercept)" "speed" "" "" ...
11 $ rank : int 2
12 $ fitted.values: Named num [1:50] -1.85 -1.85 9.95 9.95 13.88 ...
13 ..- attr(*, "names")= chr [1:50] "1" "2" "3" "4" ...
14 $ assign : int [1:2] 0 1
15 $ qr :List of 5
16 ..$ qr : num [1:50, 1:2] -7.071 0.141 0.141 0.141 0.141 ...
```

Dealing with objects

If we want σ , standard deviation of residuals $\hat{\varepsilon}$, in a regression

```
1 > summary(regression)$sigma
2 [1] 15.37959
3 > sqrt(sum(regression$residuals^2)/regression$df.residual)
4 [1] 15.37959
```

Reports

- To produce a nice report, use R-Markdown documents

```
1 ---
2 title: "Regression Analysis Report"
3 author: "Your Name"
4 date: "'r Sys.Date()'"
5 output: html_document
6 ---
7
8 '{r}
9 regression = lm(dist ~ speed, data = cars)
10 sigma = summary(regression)$sigma
11 ''
12
13 Standard deviation is '{r} round(sigma,2)'.
```


- ▶ The outcome will be

Regression Analysis Report

Your Name

```
regression = lm(dist ~ speed, data = cars)
sigma = summary(regression)$sigma
```

Standard deviation is 15.38.

Help with Chat-GPT (2)

- ▶ you write ask to get simple functions

Import a CSV file called "data.csv" into a data frame called "df"

To import a CSV file called "data.csv" into a data frame called "df" in R, you can use the `read.csv()` function from base R. Here is how you can do it:

```
1 df <- read.csv("data.csv")
```

Filter the rows in the data frame "df" where the variable "gender" is equal to "female".

To filter the rows in the data frame `df` where the variable `gender` is equal to "female", you can use the `subset()` function or the `dplyr` package for a more `tidyverse`-friendly approach.

Help with Chat-GPT (2)

Here are both methods:

Using Base R

```
1 df_female <- subset(df, gender == "female")
```

Using dplyr Package

First, install and load the **dplyr** package if you haven't already. Then, filter the data frame::

```
1 install.packages("dplyr")
2 library(dplyr)
3 df_female <- df %>% filter(gender == "female")
```

Help with Chat-GPT (3)

- ▶ generate $X \sim \sum_{i=1}^N Y_i$ (compound Poisson)

can you write a R function that generate compound poisson variables, with gamma jumps ?

Certainly! In R, you can generate compound Poisson variables with gamma-distributed jumps using the following function. The function will take the parameters for the Poisson distribution (lambda) and the gamma distribution (shape and scale) as inputs. Here's a sample implementation:

Help with Chat-GPT (3)

```
1 # Load necessary library
2 install.packages("MASS")
3
4 generate_compound_poisson <- function(n, lambda, shape, scale) {
5   # n: number of compound Poisson variables to generate
6   # lambda: rate parameter for the Poisson distribution
7   # shape: shape parameter for the gamma distribution
8   # scale: scale parameter for the gamma distribution
9
10  # Generate Poisson random variables
11  num_jumps <- rpois(n, lambda)
12
13  # Initialize vector to store compound Poisson variables
14  compound_poisson_vars <- numeric(n)
15
16  # Generate compound Poisson variables
17  for (i in 1:n) {
18    if (num_jumps[i] > 0) {
```

Help with Chat-GPT (3)

```
19     # Generate gamma jumps
20     jumps <- rgamma(num_jumps[i], shape, scale)
21     # Sum the jumps to get the compound Poisson variable
22     compound_poisson_vars[i] <- sum(jumps)
23   } else {
24     # If there are no jumps, the value is 0
25     compound_poisson_vars[i] <- 0
26   }
27 }
28
29 return(compound_poisson_vars)
30 }
31
32 # Example usage
33 n <- 1000      # Number of variables to generate
34 lambda <- 2    # Poisson rate parameter
35 shape <- 2     # Gamma shape parameter
36 scale <- 1     # Gamma scale parameter
```

Help with Chat-GPT (3)

```
37  
38 compound_poisson_samples <- generate_compound_poisson(n, lambda, shape  
    , scale)
```

The function `generate_compound_poisson` generates `n` compound Poisson random variables.

It first generates the number of jumps from a Poisson distribution with parameter `lambda`.

For each generated Poisson variable, if the number of jumps is greater than zero, it generates gamma-distributed jumps using the `rgamma` function and sums them up to get the compound Poisson variable.

If there are no jumps, the value is set to zero.

Replication

► Replication of random samples ?

```
1 > ?set.seed
2 > set.seed(12345)
3 > (x=rnorm(1))
4 [1] 0.5855288
5 > (x=rnorm(1))
6 [1] 0.709466
7 > (x=rnorm(1))
8 [1] -0.1093033
9 > set.seed(12345)
10 > (x=rnorm(1))
11 [1] 0.5855288
```


References

- Fisher, R. (1936). The use of multiple measurements in taxonomic problems. *Annals of eugenics*, 7(2):179–188.
- Gallic, E. (2020). *Notes de cours de R*. <https://egallic.fr/Enseignement/R/Book/>.
- Pitman, E. J. (1937). Significance tests which may be applied to samples from any populations. *Journal of the Royal Statistical Society*, 4(1):119–130.