Exercise Stat-o-Sphere R-Basics Tutorial (@JNOS)

Link  to R-Basics article: <https://doi.org/10.56776/abbd964d.665f7de5>

# The below was translated from German to English via the help of ChatGPT.

**CONTEXT:**

A t-test should be used to check whether a blood pressure-lowering drug makes a significant difference by comparing systolic blood pressure (sysRR) before and after administration. (Keep in mind that this is a highly simplified practice exercise with a manageable dataset, so only sysRR is included, etc.)

Read each point completely before starting the solution.

**Instructions:**

**a)** Create a new script (File -> New File -> New Script). First, import the dataset using read.csv("file\_path\_and\_name"), and choose an object name (essentially a variable/reference name for the imported table). **Do NOT open the CSV in Excel beforehand** (as it may add extra quotation marks to the table). If you encounter errors during import, simply download the CSV file again. If an additional extension (.csv.xls) was added after downloading, remove it by renaming the file (so that only .csv remains). (see [Chapter 6](https://journal.medicine.berlinexchange.de/pub/4wdsmd0f/#import-and-export-of-csv-files-understanding-delimiters-and-packages) on Import/Export.)  
Check whether the data is normally distributed using shapiro.test() and plot(density())—for the sysRR values at time points t1 and t2. If you get an error, inspect the sysRR column in the console to diagnose the issue (then proceed to **b**) to resolve it).

**b)** For participant 5, the sysRR value at t1 is missing. Enter the value **"122 mmHg"** in your dataset, as you found it in your notes but forgot to include it in your Excel table... A Shapiro-Wilk test and t-test cannot be performed while this NA entry is present. Remember how to access specific positions in a dataset to modify entries—this was covered when discussing the c() function. (See [Chapter 5](https://journal.medicine.berlinexchange.de/pub/4wdsmd0f/#classes-of-objects-vectors-matrices-arrays-lists-and-more) or check the corresponding section in the R Basic Code Script.)

**c)** It turns out that the data is normally distributed (otherwise, we would have needed to use a Wilcoxon signed-rank test). Now, what type of t-test is required? Perform a t-test using the t.test() function. Research how this function is used online, or check the documentation with ?t.test(). Also, use the **effsize** package and the cohen.d() function!  
When analyzing the results, consider **means, mean difference, and standard deviation**. Imagine a scenario where the standard deviation is -0.1 mmHg—then an effect size of 2 standard deviations (SD) would be minimal or irrelevant. (Thus, classifications such as "large" given by the cohen.d() function should not be taken at face value—reflect on them yourself, as Cohen's scale has its limitations!)  
Different types of standard deviations exist for different forms of t-tests. Remember that if you assign a name to the output of t.test(), you can extract individual values like the **mean difference** (called estimate in the output) or **SD** from the resulting list.

**d)** Normally, a power analysis is conducted before data collection and analysis. What **sample size** would you typically need for this study, given a power of **80%**, a **95% confidence interval**, and a **hypothetical effect size of 0.5**? Use the pwr.t.test() function from the **pwr** package.

**e)** Clean up the column "fam" (family history). Replace all NA values with "not specified" (no information). Use the functions is.na(), which(), and unique(). (See [Chapter 8.6](https://journal.medicine.berlinexchange.de/pub/4wdsmd0f/#example-vi-adjusting-character-string-entries))

**f)** Create a **pie chart** using pie(table(dataset\_name$fam)). First, use table() to create a frequency table from the "fam" column, then generate the pie chart. Consider the structure of your frequency table—are there redundancies?

**g)** Once all columns are cleaned up, **export the dataset** as a .csv file using write.csv(your\_table\_name, "your\_file\_path\_and\_name.csv"). (See [Chapter 6](https://journal.medicine.berlinexchange.de/pub/4wdsmd0f/#import-and-export-of-csv-files-understanding-delimiters-and-packages) on Import/Export.)

**Bonus Exercise**

**a)** Create a **small synthetic dataset** to test the na.omit() function or another method for handling missing values. For example, make two columns and three rows, with each column containing one NA. Use the functions c(), cbind(), and possibly as.data.frame(). (See Chapter [Chapter 8.5.1](https://journal.medicine.berlinexchange.de/pub/4wdsmd0f/#representation-of-solutionalgorithm-ii-from-example-ii-via-tidyverse-and-the-dangers-of-the-naomit-function))

**b)** **Reformat the table** so that sysRR values for **t1 and t2** appear side by side in a single table. This should eliminate redundancies in the **patient\_id** and "fam" column. Use the cbind() function. You may also need colnames(). (See [Chapter 5](https://journal.medicine.berlinexchange.de/pub/4wdsmd0f/#classes-of-objects-vectors-matrices-arrays-lists-and-more) and [Chapter 8.13](https://journal.medicine.berlinexchange.de/pub/4wdsmd0f/#example-xiii-converting-wide-into-long-format))

**c)** Add a new column to the dataset and **compute the sysRR difference algorithmically** in R. Use cbind() and c(0) to initialize an empty column filled with zeros. (See [Chapter 8.9](https://journal.medicine.berlinexchange.de/pub/4wdsmd0f/#example-ix-working-with-redundant-columns)))

**d)** **Determine how many participants** had a **sysRR difference of ≥10**. The solution approach is found only in the **solutions or online**, not in the slides. Use the >= operator and the which() function (so far, we've only used ==).

**e) (Optional)** Create a nicely formatted table in R using the following code:

# install.packages("gt")

library(gt)

gt(reformatierte\_tabelle)