Writing MSc theses with me

Martin Papenberg

April 14, 2025

1 Writing Master theses (MSc)

In the recent years, my research focus has been on developing and evaluating methods for data analysis. This document is meant as an introduction to topics that one should be (at least partially) familiar with when having me as supervisor for a Master thesis that has a methodological focus.

The following topics can be addressed in a Master thesis project:

1. Method comparison
2. Anything related to the "anticlustering" method (see here)

This document is an outline to "1. Method comparison". Feel free to contact me if you are interested in pursuing either topic. Please do! Even if not everything that is written in this document is clear from the start.

Pursuing either topic will emphasise data analysis and programming with R, and both will probably entail coding a simulation study. This document should provide the required background and references for starting a project. Working through this document and completing the exercises will also be part of the "Modul C" that has to be completed alongside the Master thesis project.

2 Method comparison studies

As scientists we usually have to be experts in a subject domain as well as a range of "methods" for analysing data (i.e., statistics). However, we often lack the proper background for evaluating the usefulness of the methods that we apply. Instead, we have to rely on other people's judgement, or we do "what has always been done". To complicate things, new methods continue to be developed in a fast pace, seemingly outperforming older approaches. Keeping up is challenging. For example, the recent years have seen the rise of Bayes factors as alternatives to traditional null hypothesis significance tests relying on p-values (e.g., Dablander et al., 2022). Usually, authors presenting a new method present their own (usually favourable) evaluation of this method; neutral comparisons are rare (Boulesteix et al., 2017).

Conducting a study to compare competing methods is a possible project for a Master thesis with me (e.g., see Brandenburg & Papenberg, 2024).

References

Boulesteix, A. L., Binder, H., Abrahamowicz, M., & Sauerbrei, W. (2017). On the necessity and design of studies comparing statistical methods. *Biometrical Journal, 60*(1), 216–218.

Brandenburg, N., & Papenberg, M. (2024). Reassessment of innovative methods to determine the number of factors: A simulation-based comparison of exploratory graph analysis and next eigenvalue sufficiency test. Psychological Methods, 29(1), 21–47. https://doi.org/10.1037/met0000527 (Preprint)

Dablander, F., Huth, K., Gronau, Q. F., Etz, A., & Wagenmakers, E. J. (2022). A puzzle of proportions: Two popular Bayesian tests can yield dramatically different conclusions. *Statistics in medicine, 41*(8), 1319-1333.

3 Background: Simulation studies

One way of evaluating the usefulness of a (statistical) method is to conduct a simulation study. In such a simulation, we feed data whose properties are known (which is unlike doing "real" research) into a set of competing methods. Based on some criterion for "success", we then compare how well the competing methods perform. When comparing statistical methods, performance is usually assessed by testing which method best recovers the underlying reality of the data—which we luckily know because it is a simulation.

Thus, simulation studies can answer the following question: If we happen to know how reality works, will my method find out? If a method fails to recover even a known reality, it is highly questionable that it can perform adequately on real data. Conversely, good performance in a simulation does not guarantee desirable properties in the real world. How well results of a simulation can be transferred back to reality is always up to debate.

3.1 Motivating example: Parametric tests are preferred when assumptions are met

The t-test is the go-to statistical method when comparing two group means. However, in statistics class we usually learn that "non parametric" tests may be needed when some of the assumptions of the parametric t-tests are not met. A motivating example shows that when the assumptions are indeed met, the t-test is preferred over its non-parametric pendant: the Wilcoxon Rank Sum test (also known as "Mann-Whitney" test). Note that technically, the Wilcoxon test does not test equality in population means, but it is often used for this purpose (although it probably shouldn't be).

Following up on my "definition" of simulation studies in the previous section, let's consider the "reality" in my little simulation: Here, I assume that two populations truly differ in their means, e.g., a group of patients that receives treatment versus a control group. I want to find out how well the competing methods—the t-test and the Wilcoxon test—reflect this pattern. To this end, I compare their statistical power, i.e., the relative proportion of results that are significant. Because I am interested in statistical power, the simulation always assumes that there is a true effect. A different question would be how well the two methods conform to a pre-set error level α (when there is no difference between population means).

In the remainder of this section, `R` code is provided that implements a small scale comparison of the two tests. It is no problem at all if the code is not clear at first. The exercises in the end will guide you through all elements of the code.

First, we define two functions that both first generate data from a normal distribution for two groups and then perform a test: `ttest_2groups()` performs the t-test and `wilcox_2groups()` performs the Wilcoxon test.

```
ttest_2groups <- function(X, N, D) {
  data1 <- rnorm(N, D)
  data2 <- rnorm(N)
  t.test(data1, data2, var.equal = TRUE)$p.value
}

wilcox_2groups <- function(X, N, D) {
  data1 <- rnorm(N, D)
  data2 <- rnorm(N)
  wilcox.test(data1, data2)$p.value
}
```

The arguments N is the sample size in each group and the argument D is the true effect size, i.e., the difference in population means. The argument `X` can be ignored at first; it is later used to call the functions repeatedly. Note that according to the function definitions, the assumptions of the t-test are indeed met: The function `rnorm()` generates independent samples from a normal distribution, while the standard deviation of each population is the same: the default value of `rnorm()`'s argument `sd` is 1, which is not changed. So,

when either a function is called, it first generates two samples from a normal distribution whose means differ according to the argument D. Subsequently, a t-test / Wilcoxon test compares the two groups. The functions return only a single number (i.e., a numeric vector of length 1), which is the p-value associated with the comparison.

To repeat the data generation and testing many times in a simulation, we first define a variable that sets the number of simulation runs:

```
nsim <- 1000
```

Thus, we will perform 1000 t-tests and 1000 Wicoxon tests, each time using different randomly generated data.

Next, we define variables for the effect size D and the sample size N (for each group), which we will pass to the functions `ttest_2groups()` and `wilcox_2groups()`:

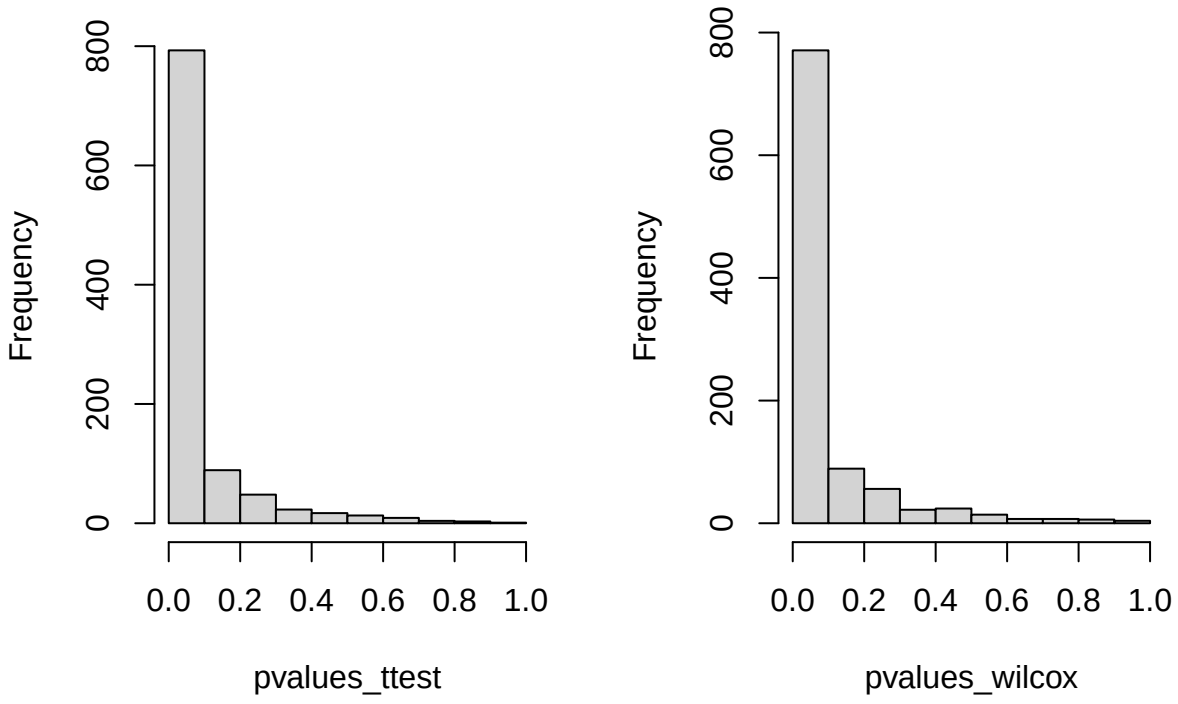
```
N <- 50
D <- 0.5
```

Now, let's start the simulation with the function `sapply()`, which repeatedly calls the two functions:

```
pvalues_ttest <- sapply(1:nsim, ttest_2groups, N = N, D = D)
pvalues_wilcox <- sapply(1:nsim, wilcox_2groups, N = N, D = D)
```

The variables `pvalues_ttest` and `pvalues_wilcox` now each contain a vector of length `nsim` $= 1000$, storing the p-values that were obtained during each simulation run. For example, we might display the distribution of these p-values in two histograms:

```
par(mfrow = c(1, 2))
hist(pvalues_ttest, main = "")
hist(pvalues_wilcox, main ="")
```



Because there is a true effect in the population (i.e., $D = 0.5$), the histograms are heavily skewed, favoring small p-values. Since we are interested in statistical power, we won't be working with the raw p-values and instead determine how often the p-values were lower than the α level of .05:

```
mean(pvalues_ttest <= .05) # proportion of significant p-values for the t-test
```

```
## [1] 0.673
```

```
mean(pvalues_wilcox <= .05) # proportion of significant p-values for the Wilcoxon test
```

```
## [1] 0.656
```

The number of significant p-values is higher for the t-test—but only very slightly! Thus, the t-test has better statistical power, at least for $N = 50$, Cohen's $d = 0.5$—and if you believe the observed difference is not just spurious due to sampling error. If the difference between t-test and Wilcoxon test does not seem convincing to you, we might need to test the difference in power for statistical significance, which will be done in the next section.

3.2 Statistical inference on data from a simulation study

The quantity that we are interested in in our little simulation is the proportion of significant p-values for the two competing methods (i.e., the statistical power of the tests). Hence, we categorise the p-values as significant vs. non-significant and accept the loss of information by the categorisation. Let's display these results in a contingency table:

```
test_significant <- cbind(
  ttest = table(pvalues_ttest <= .05),
  wilcox = table(pvalues_wilcox <= .05)
)
```

```
test_significant
```

```
##       ttest wilcox
## FALSE   327    344
## TRUE    673    656
```

We observed a higher number of significant results for the t-test. However, as in any study, we cannot really be certain if the observed difference reflects a true difference in performance of if it is due to mere sampling error. Using χ^2 test on the contingency table, we might test the observed difference in proportions of significant results itself on significance:

```
chisq.test(test_significant)
```

```
##
##  Pearson's Chi-squared test with Yates' continuity correction
##
## data:  test_significant
## X-squared = 0.57415, df = 1, p-value = 0.4486
```

The difference is not significant!

A non-significant result can be observed because (a) the power is really not different between the tests (thus, the non-significant result is "correct") or (b) the method comparison itself lacks power.[1] One way of increasing power is collecting more data. A good thing in simulation studies is that we can increase the number of simulation runs rather easily. Let's just repeat the same process with more data sets:

```
nsim <- 10000
pvalues_ttest <- sapply(1:nsim, ttest_2groups, N = N, D = D)
mean(pvalues_ttest <= .05) # proportion of significant p-values for the t-test
```

```
## [1] 0.6992
```

```
pvalues_wilcox <- sapply(1:nsim, wilcox_2groups, N = N, D = D)
mean(pvalues_wilcox <= .05) # proportion of significant p-values for the Wilcoxon test
```

```
## [1] 0.6771
```

```
test_significant <- cbind(
  ttest = table(pvalues_ttest <= .05),
  wilcox = table(pvalues_wilcox <= .05)
)
chisq.test(test_significant)
```

```
##
##  Pearson's Chi-squared test with Yates' continuity correction
##
## data:  test_significant
## X-squared = 11.277, df = 1, p-value = 0.0007848
```

The pattern holds with more repetitions, and now the statistical power differs significantly between methods. Because simulation studies can often easily increase the number of data sets and usually employ a rather large number of simulation runs, statistical tests are often omitted when reporting the results of a simulation study. This may not always be a good thing.

As we have seen, one way of making the simulation study more informative is to use more repetitions: generate more data sets / do more "simulation runs". Another approach of improving the statistical power in our

[1] Note that we are talking on a meta level here: Our simulation investigates the statistical power of two methods, the t-test and the Wilcoxon test. And for our simulation study, we can ask the question to most efficiently compare these two methods.

simulation is to apply the t-tests and the Wilcoxon tests on the same data set rather than independent data sets. Then, we can compare *dependent* proportions between these methods; the χ^2-test I performed above (rightly) assumed independence between observations.

Next, we will do just that and define a new function `compare_2groups()`, which calls both tests on the same data; it then returns both p-values obtained from the tests:

```
# Define a functions which generates data from a normal distribution for two groups
# and then perform a (a) t-test and a (b) Wilcoxon test on the same data
compare_2groups <- function(X, N, D) {
  data1 <- rnorm(N, D)
  data2 <- rnorm(N)
  c(ttest = t.test(data1, data2)$p.value,
    wilcox = wilcox.test(data1, data2)$p.value)
}
```

Let's repeat the simulation using the new function:

```
nsim <- 1000
pvalues <- t(sapply(1:nsim, compare_2groups, N = N, D = D))
```

Because the function `compare_2groups()` returns two values and not just one, the variable `pvalues` is a matrix with two columns. The two columns[2] represent to the two methods and each row is a data set that was processed in the simulation:

```
head(pvalues)
```

```
##           ttest      wilcox
## [1,] 0.06897176 0.166906959
## [2,] 0.29328776 0.191423749
## [3,] 0.02199559 0.025283429
## [4,] 0.01213966 0.003999898
## [5,] 0.08225180 0.205865611
## [6,] 0.26874168 0.135564973
```

Let's recode this matrix into categorising if the results were significant:

```
significant <- pvalues <= .05
head(significant)
```

```
##      ttest wilcox
## [1,] FALSE  FALSE
## [2,] FALSE  FALSE
## [3,]  TRUE   TRUE
## [4,]  TRUE   TRUE
## [5,] FALSE  FALSE
## [6,] FALSE  FALSE
```

How often was each test significant:

```
colMeans(significant)
```

```
##  ttest wilcox
##  0.702  0.666
```

Again, the t-test has more significant results. We can now investigate some interesting questions, i.e., how often did the methods differ in their significance assessment.

[2]Had I not used the function `t()` on the return value of `sapply()`, rows and columns would have been reversed, which I think is slightly less convenient for visual inspection.

```r
sum(significant[, "ttest"] != significant[, "wilcox"])
```

```
## [1] 66
```

```r
sum(significant[, "ttest"] > significant[, "wilcox"])
```

```
## [1] 51
```

```r
sum(significant[, "ttest"] < significant[, "wilcox"])
```

```
## [1] 15
```

The tests only disagreed in 6.6% of all data sets. In the vast majority of these cases, the t-test was significant and the Wilcox test was not. Is this difference significant? Because the tests were applied to the same data sets, I can make use of the Mc-Nemar test, which compares dependent proportions:

```r
mcnemar.test(significant[, "ttest"], significant[, "wilcox"])
```

```
##
##  McNemar's Chi-squared test with continuity correction
##
## data:  significant[, "ttest"] and significant[, "wilcox"]
## McNemar's chi-squared = 18.561, df = 1, p-value = 1.646e-05
```

The comparison is highly significant, even when only using 1000 simulation runs, which did not yield conclusive results when the two tests were applied to different data sets.

3.3 Follow-up questions

This was already a small-scale simulation, which yielded informative results! A real simulation study, however, is usually characterized by varying the input parameters, for example by not using only one value for N and the effect size. In order to make any generalization statements from a simulation study, varying the input parameters is necessary. Which parameters should vary (and in what range) has to be chosen carefully. The data generated during a simulation study can become quite extensive and it is necessary to consider an appropriate experimental design, i.e., which conditions vary, how they are combined, and how many data sets are generated for each combination of parameters. Data analysis on such data can also be challenging.

3.4 Exercises

- Understand the simulation code
  - Learn about defining your own functions: How do you define input arguments and return values?
  - What does the function `rnorm()` do?
  - What is the output of the function `t.test()` and how do we assess the p-value?
  - What does the function `sapply()` do? What is the difference in the return value of `sapply()` when passing (a) `ttest_2groups()` / `wilcox_2groups()` or (b) `compare_2groups()`? Why is the function `t()` applied to the output of `sapply()` when using `compare_2groups()`?
  - How do the functions `chisq.test()` and `mcnemar.test()`? What is the the similarity and difference between the two tests?
- Vary N and D; what happens?
- Does the advantage of the t-test hold for other parameters (effect size D, sample size N, ...)
- Can you find data distribution where the Mann Whitney test performs better than the t-test (either in terms of power or alpha error control)