Simulation Study: Negative Correlations and Equality of Means

February 10, 2021

Session Info

Give the session info (reduced).

```
## [1] "R version 3.6.3 (2020-02-29)"
```

```
## [1] "x86_64-pc-linux-gnu"
```

Load Libraries

If the libraries are not installed yet, you need to install them using, for example, the command: install.packages("ggplot2").

```
library(MASS)
library(ggplot2)
library(plyr)
library(GGally)
library(rstatix)
```

Give the package versions.

```
##  rstatix   GGally     plyr  ggplot2     MASS
##  "0.6.0"  "2.0.0"  "1.8.6"  "3.3.3" "7.3-53"
```

Introduction

The purpose of this file is two-fold: a) It discusses under which conditions negative correlations (i.e. trade-offs) entail equi-complexity in the sense of a non-significant difference in the means of two distributions. b) It illustrates the decisions we take to statistically analyse the respective data, i.e. checking for normality of the data, choosing a statistical test for assessing equality in the mean values of two distributions, and getting effect sizes for this test. Both is achieved by using simulated data, i.e. pseudo-complexity measurements for two domains (e.g. syntax and semantics) across different languages.

Theoretical Background

Let us assume we have n languages for which we measure complexities in two domains, e.g. syntax and morphology, such that we get two vectors of measurments $m = (m_1, m_2, \ldots, m_n)$ and $s = (s_1, s_2, \ldots, s_n)$. This situation is illustrated in the table below.

language	m	s
L_1	m_1	s_1
L_2	m_2	s_2
L_3	m_3	s_3
L_4	m_4	s_4
L_5	m_5	s_5
...	...	...
L_n	m_n	s_n

Trade-offs as Negative Correlations

Trade-offs are here conceptualized as negative correlations. We here choose the Pearson (r) and Spearman (ρ) correlation coefficients as examples. While the former measures linear dependence, the latter is a non-parametric rank correlation.

Across the languages L_1 to L_n the Pearson correlation coefficient between the complexity measurements in the two domains (e.g. morphology and syntax) is defined as

$$r_{ms} = \frac{\sum_{i=1}^{n}(m_i - \overline{m})(s_i - \overline{s})}{\sqrt{\sum_{i=1}^{n}(m_i - \overline{m})^2}\sqrt{\sum_{i=1}^{n}(s_i - \overline{s})^2}}, \tag{1}$$

where n is the number of data points in the paired samples, m_i and s_i are individual measurements in the respective domain, and $\overline{m}$ and $\overline{s}$ are the arithmetic means of the columns, i.e. complexity measurements in a certain domain, with

$$\overline{m} = \frac{1}{n}\sum_{i=1}^{n} m_i, \tag{2}$$

and

$$\overline{s} = \frac{1}{n}\sum_{i=1}^{n} s_i. \tag{3}$$

We have a negative correlation $r_{ms} < 0$ iff the numerator is negative, i.e.

$$\sum_{i=1}^{n}(m_i - \overline{m})(s_i - \overline{s}) < 0. \tag{4}$$

Note that the denominator cannot be negative.

The Spearman correlation coefficient, on the other hand, is then defined as

$$\rho_{ms} = 1 - \frac{6\sum_{i=1}^{n}(\mathrm{rank}(m_i) - \mathrm{rank}(s_i))^2}{n(n^2-1)}, \tag{5}$$

where rank() is a function that gives the rank of the respective value when the values are ranked in ascending order (i.e. the smallest receives rank 1, the second smallest rank 2 etc.). Note that this definition only holds for distinct integers being ranked, which we will assume here for simplicity. We get a negative correlation iff

$$\frac{6\sum_{i=1}^{n}(\mathrm{rank}(m_i) - \mathrm{rank}(s_i))^2}{n(n^2-1)} > 1. \tag{6}$$

Equi-Complexity as Equality of Means

Furthermore, we conceptualize equi-complexity of languages L_1 to L_n here as equality of arithmetic means. For instance, if language L_1 has a morphological complexity of $m_1 = 5$ and a syntactic complexity of $s_1 = 1$, and language L_2 has $m_2 = 1$ and $s_2 = 5$, then the arithmetic mean complexity is 3 for both. Hence, they are considered overall equally complex. Assume more generally that m_j and s_j as well as m_k and s_k represent complexity measurements for two languages L_j and L_k. We would then consider the languages equi-complex iff

$$\overline{L_j} = \overline{L_k}, \tag{7}$$

where

$$\overline{L_j} = \frac{1}{d}(m_j + s_j), \tag{8}$$

and

$$\overline{L_k} = \frac{1}{d}(m_k + s_k). \tag{9}$$

Here d is the number of different domains for which we measure complexity, i.e. d = 2 in our example of morphology and syntax.

Negative Correlations do not entail Equality of Means

We here proof by counterexample that neither a negative Pearson nor a negative Spearman correlation between two samples of complexity measurements in different domains strictly entail equality of means for the respective languages from which these measurments were taken. In other words, we will disproof by counterexample the claim that

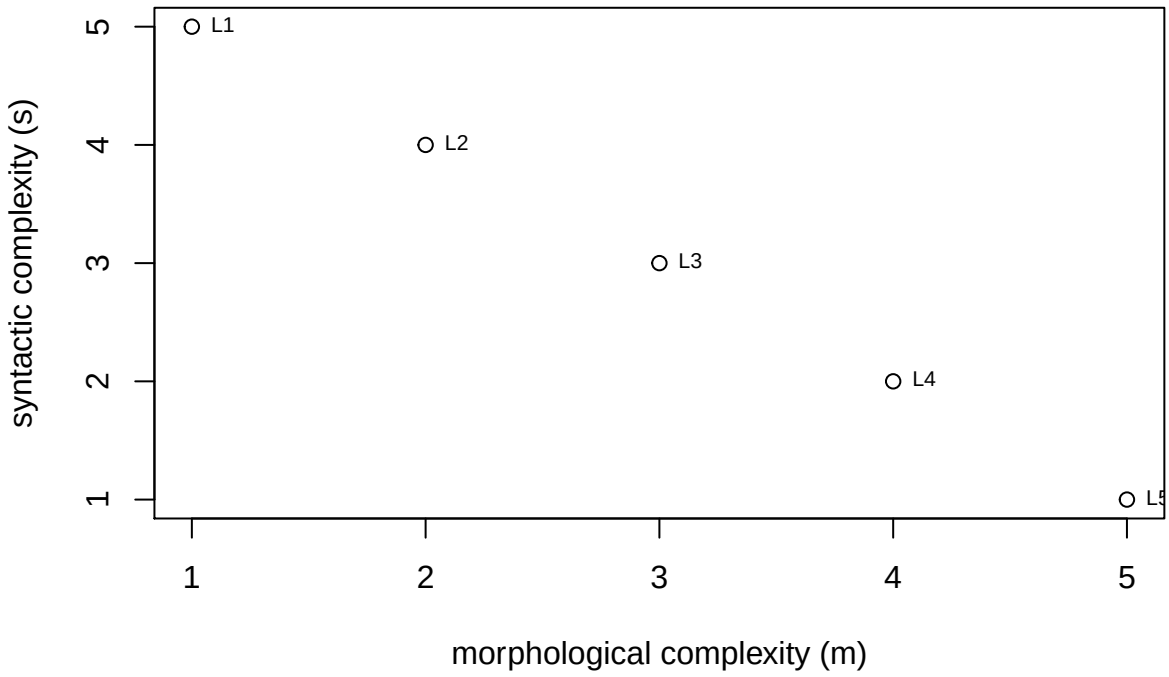
$$(r_{ms} < 0), (\rho_{ms} < 0) \vdash (\overline{L_j} = \overline{L_k}). \tag{10}$$

Firstly, assume that we have a perfect negative Pearson and Spearman correlation between two samples, i.e. $r_{ms} = \rho_{ms} = -1$. For example, assume that m = (1, 2, 3, 4, 5) and s = (5, 4, 3, 2, 1) across five different languages. This is illustrated in the plot below

```
language = c("L1", "L2", "L3", "L4", "L5")
m <- c(1, 2, 3, 4, 5)
s <- c(5, 4, 3, 2, 1)
example.df <- data.frame(language, m, s)
print(example.df)
```

```
##   language m s
## 1       L1 1 5
## 2       L2 2 4
## 3       L3 3 3
## 4       L4 4 2
## 5       L5 5 1
```

```
plot(example.df$m, example.df$s, xlab = "morphological complexity (m)",
     ylab = "syntactic complexity (s)")
text(example.df$m, example.df$s, labels = example.df$language, cex = 0.7, pos = 4)
```



For the Pearson correlation we have

Generate Correlated Data

We here generate correlated pseudo-complexity measurements for two languages A and B. We then apply linear and non-linear transformations to illustrate how this impacts the results of correlation and mean value analyses.

```
# set the seed for random number generation in order to
# get the same result when the code is re-run
set.seed(1)
# set parameters
n = 20 # number of datapoints
r = -0.7 # predefined correlation
a = 2 # constant for linear transformation
# generate the data
data <- mvrnorm(n = n, mu = c(3, 3), Sigma = matrix(c(1, r, r, 1), nrow = 2),
                empirical = TRUE)
langA <- data[, 1]
langB <- data[, 2]
# apply linear transformation to language B measures
langB.lt <- langB*a
# apply non-linear transformation to language B measures
langB.nt <- sqrt(langB)
```

Standardize the data by centering and scaling it.

```
langA.stand <- scale(langA)
langB.stand <- scale(langB)
langB.lt.stand <- scale(langB.lt)
langB.nt.stand <- scale(langB.nt)
```

Put into short format (for scatterplots) and long format data frame (density distributions).

```
# short format
simulation.df.short <- data.frame(langA, langB, langB.lt, langB.nt, langA.stand, langB.stand,
                                  langB.lt.stand, langB.nt.stand)
# long format
# non-standardized
value <- c(langA, langB, langB.lt, langB.nt)
measurement <- rep(c(1:n), times = 4)
language <- c(rep("Language A", times = n),
              rep("Language B", times = n),
              rep("Language B (linear trans.)", times = n),
              rep("Language B (non-linear trans.)", times = n)
              )
simulation.df.long <- data.frame(language, measurement, value)
head(simulation.df.long)
```

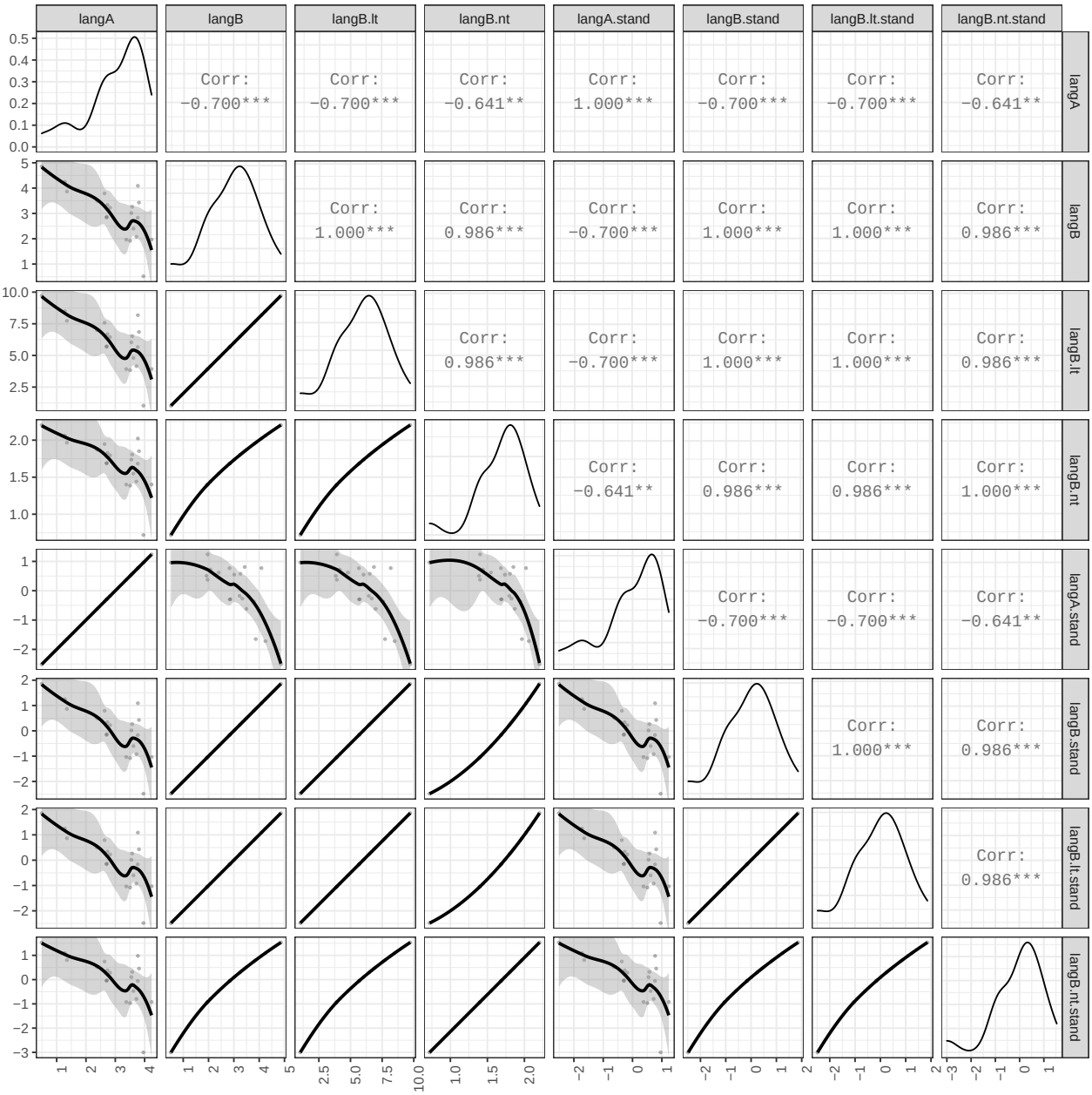
```
##     language measurement    value
## 1 Language A           1 4.238895
## 2 Language A           2 3.767582
## 3 Language A           3 3.507511
## 4 Language A           4 0.490032
## 5 Language A           5 3.549220
## 6 Language A           6 3.374442
```

```
# standardized
value <- c(langA.stand, langB.stand, langB.lt.stand, langB.nt.stand)
measurement <- rep(c(1:n), times = 4)
language <- c(rep("Language A (standardized)", times = n),
              rep("Language B (standardized)", times = n),
              rep("Language B (linear trans., standardized)", times = n),
              rep("Language B (non-linear trans., standardized)", times = n)
              )
simulation.df.long.stand <- data.frame(language, measurement, value)
head(simulation.df.long.stand)
```

```
##                     language measurement      value
## 1 Language A (standardized)           1  1.2388953
## 2 Language A (standardized)           2  0.7675818
## 3 Language A (standardized)           3  0.5075114
## 4 Language A (standardized)           4 -2.5099680
## 5 Language A (standardized)           5  0.5492195
## 6 Language A (standardized)           6  0.3744419
```

Scatterplot with Correlations

```
simulation.scatterplot <- ggpairs(simulation.df.short,
                        lower = list(continuous = wrap("smooth_loess", alpha = 0.3,
                                                       lwd = 0.5, size = 2))) +
                        #upper = list(continuous = wrap('cor', method = "spearman"))) +
                        theme_bw() +
                        theme(axis.text.x = element_text(angle = 90, hjust = 1))
print(simulation.scatterplot)
```



Density Distributions

Plot density distributions of complexity pseudo-measurements by language. Individual values for each complexity pseudo-measurement are plotted as black dots. The central value (0) is indicated by a vertical dotted line for visual reference. The median and mean values of complexity pseudo-measurements per language might also be indicated.

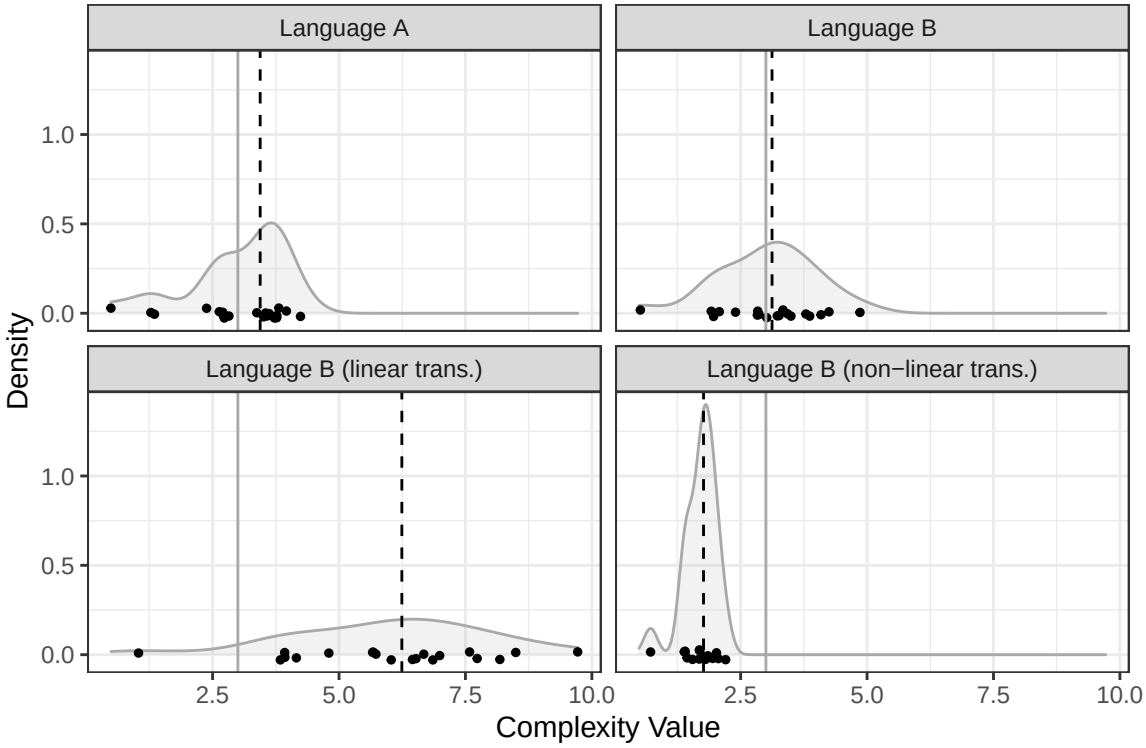
Non-Standardized Vectors

Get mean, median, and standard deviation values.

```
# get mean values for each language
mu <- ddply(simulation.df.long, "language", summarise, grp.mean = mean(value, na.rm = T))
# get median values for each language
med <- ddply(simulation.df.long, "language", summarise, grp.median = median(value, na.rm = T))
# get standard deviation values for each language
sdev <- ddply(simulation.df.long, "language", summarise, grp.sd = sd(value, na.rm = T))
```

Plot density distributions with indication of median (mean) values.

```
density.plot <- ggplot(simulation.df.long, aes(x = value)) +
 geom_density(alpha = .2, fill = "grey", color = "darkgrey") +
 geom_jitter(data = simulation.df.long, aes(x = value, y = 0),
             size = 1, height = 0.03, width = 0) + # add some jitter to prevent overplotting
  geom_vline(aes(xintercept = 3), color = "darkgrey") +
  geom_vline(data = med, aes(xintercept = grp.median), linetype = "dashed") +
  facet_wrap(~ language) +
  #xlim(-3, 3) +
  labs(x = "Complexity Value", y = "Density") +
  theme_bw() +
  theme(legend.position = "none")
print(density.plot)
```



Standardized Vectors

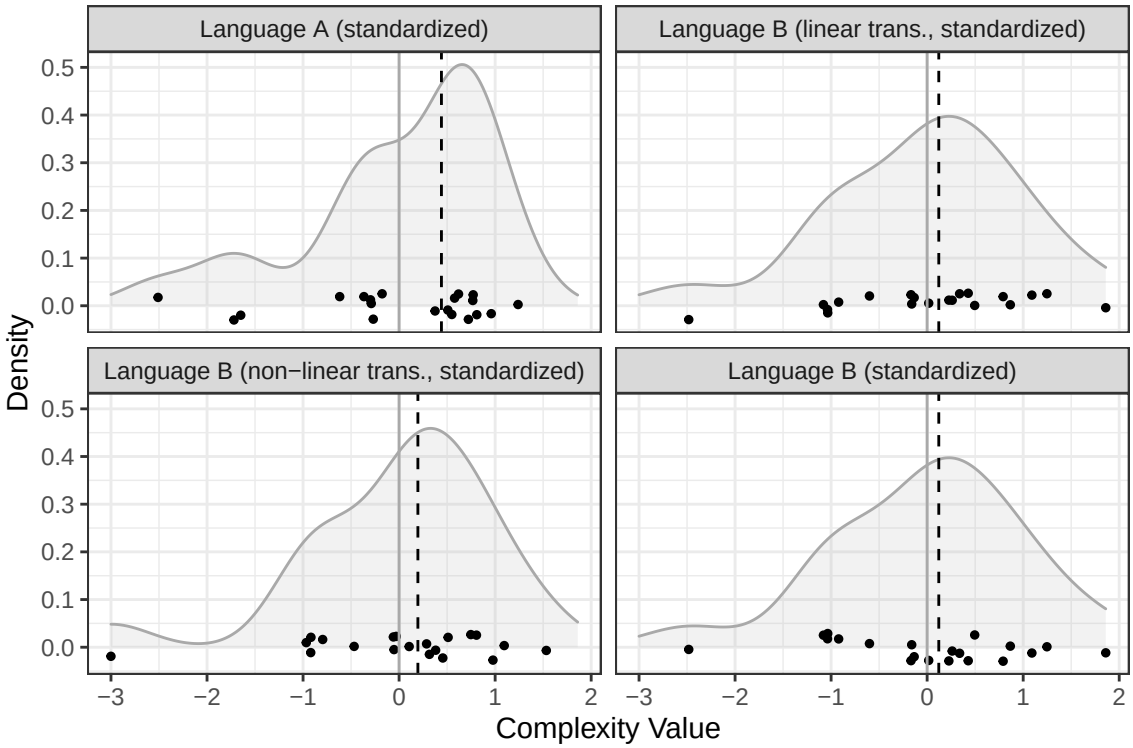
Get mean, median, and standard deviation values.

```
# get mean values for each language
mu <- ddply(simulation.df.long.stand, "language", summarise, grp.mean = mean(value, na.rm = T))
# get median values for each language
med <- ddply(simulation.df.long.stand, "language", summarise, grp.median = median(value, na.rm = T))
```

```r
# get standard deviation values for each language
sdev <- ddply(simulation.df.long.stand, "language", summarise, grp.sd = sd(value, na.rm = T))
```

Plot density distributions with indication of median (mean) values.

```r
density.plot.stand <- ggplot(simulation.df.long.stand, aes(x = value)) +
 geom_density(alpha = .2, fill = "grey", color = "darkgrey") +
 geom_jitter(data = simulation.df.long.stand, aes(x = value, y = 0),
             size = 1, height = 0.03, width = 0) + # add some jitter to prevent overplotting
  geom_vline(aes(xintercept = 0), color = "darkgrey") +
  geom_vline(data = med, aes(xintercept = grp.median), linetype = "dashed") +
  facet_wrap(~ language) +
  #xlim(-3, 3) +
  labs(x = "Complexity Value", y = "Density") +
  theme_bw() +
  theme(legend.position = "none")
print(density.plot.stand)
```



Save figure to file.

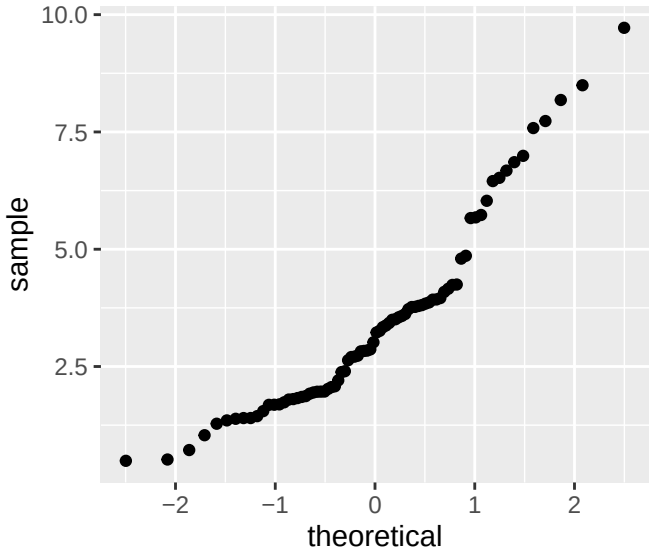
```r
#ggsave("Figures/Simulation/scatterplot.pdf", density.plot, dpi = 300, scale = 1,
#       device = cairo_pdf)
```

Normality

The assumption that the tested data stems from a normally distributed population is often necessary for the mathematical proofs underlying standard statistical techniques. We might apply normality tests to check for this assumption (e.g. Baayen 2008, p. 73), but some statisticians advise against such pre-tests, since they are often too sensitive (MacDonald 2014, p. 133-136, Rasch et al. (2020), p. 67). In fact, Rasch et al. (2020, p. xi) argue based on earlier simulation studies that almost all standard statistical tests are fairly robust

against deviations from normality. However, it is still advisable to check for gross deviations from normality in the data. One common way of doing this is quantile-quantile plots. The points should here roughly follow a straight line (Crawley 2007, p. 281).

```
ggplot(simulation.df.long, aes(sample = value)) +
  stat_qq()
```



Choose statistical tests

Select a statistical test: Standard t-tests can be used to assess significant differences in the means of the pseudo-complexity distributions, if we assume that the underlying population distributions are normal. Wilcoxon tests are a non-parametric alternative, i.e. they do not make assumptions about the underlying population distribution, e.g. normality (Crawley 2007, p. 283; Baayen 2008, p. 77).

Run pairwise Wilcoxon tests.

```
# we add some random noise here to the value vector with
# the function jitter(), since we otherwise get warnings due to ties in the data
p.values <- pairwise.wilcox.test(jitter(simulation.df.long$value), simulation.df.long$language,
                     paired = F, p.adjust.method = "holm")
p.values$p.value
```

```
##                                 Language A   Language B
## Language B                     8.830570e-01           NA
## Language B (linear trans.)     8.445624e-07 2.471386e-06
## Language B (non-linear trans.) 1.041642e-04 2.774694e-06
##                                Language B (linear trans.)
## Language B                                             NA
## Language B (linear trans.)                             NA
## Language B (non-linear trans.)                1.816803e-07
```

Effect Size

Statistical significance is only one part of the story. For instance, a difference in complexity values might be statistically significant, but so small that it is negligible for any theorizing. In fact, it is sometimes argued

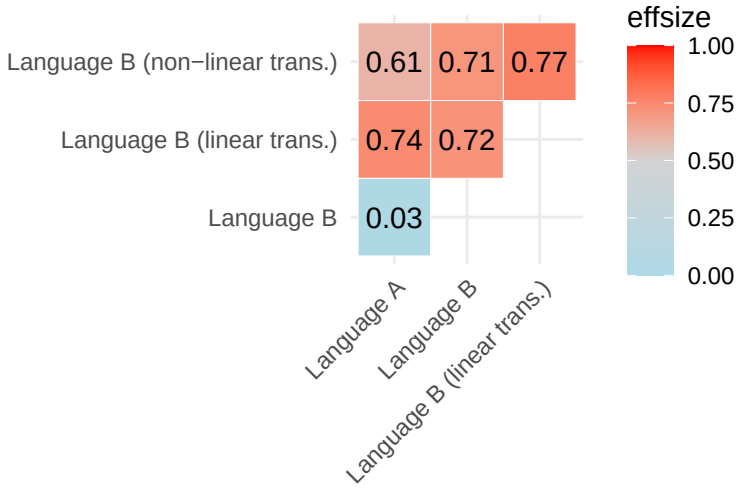
that effect sizes - rather than p-values - should be the aim of statistical inquiry (Cahusac 2020, p. 12-15). An overview of effect size measures per statistical test is given in Patil (2020). In conjunction with the Wilcoxon signed rank test we here use the statistic r (i.e. function wilcox_effsize() of the "rstatix" package).

```
# non-standardized
effect.sizes <- wilcox_effsize(simulation.df.long, value ~ language, paired = F)
# standardized
effect.sizes.stand <- wilcox_effsize(simulation.df.long.stand, value ~ language, paired = F)
#print(effect.sizes)
```

Effect Size Heatmap

Plot a heatmap with effect sizes to get a better overview.

```
# non-standardized
effect.sizes.plot <- ggplot(as.data.frame(effect.sizes), aes(group1, group2)) +
  geom_tile(aes(fill = effsize), color = "white") +
  scale_fill_gradient2(low = "light blue", mid = "light grey", high = "red",
                       midpoint = 0.5, limit = c(0,1)) +
  geom_text(aes(label = round(effsize, 2))) +
  labs(x = "", y = "") +
  theme_minimal() +
  theme(axis.text.x = element_text(angle = 45, hjust = 1))
effect.sizes.plot
```



For the standardized vectors.

```
effect.sizes.plot.stand <- ggplot(as.data.frame(effect.sizes.stand), aes(group1, group2)) +
  geom_tile(aes(fill = effsize), color = "white") +
  scale_fill_gradient2(low = "light blue", mid = "light grey", high = "red",
                       midpoint = 0.5, limit = c(0,1)) +
  geom_text(aes(label = round(effsize, 2))) +
  labs(x = "", y = "") +
  theme_minimal() +
  theme(axis.text.x = element_text(angle = 45, hjust = 1))
effect.sizes.plot.stand
```

References

Baayen, R. H. (2008). Analyzing linguistic data: A practical introduction using statistics in R. Cambridge University Press.

Cahusac, P. M. B. (2021). Evidence-based statistics. John Wiley & Sons.

Crawley, M. J. (2007). The R book. John Wiley & Sons Ltd.

McDonald, J.H. (2014). Handbook of Biological Statistics (3rd ed.). Sparky House Publishing, Baltimore, Maryland. online at http://www.biostathandbook.com

Patil, I. (2020). Test and effect size details. online at https://cran.r-project.org/web/packages/statsExpressions/vignettes/stats_details.html.

Rasch, D., Verdooren, R., and J. Pilz (2020). Applied statistics. Theory and problem solutions with R. John Wiley & Sons Ltd.