# 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 to illustrate 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. This also further illustrates the theoretical points made in the file on "theoretical background" with simulated data.

## 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
```

```r
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.

```r
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).

```r
# 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)
```

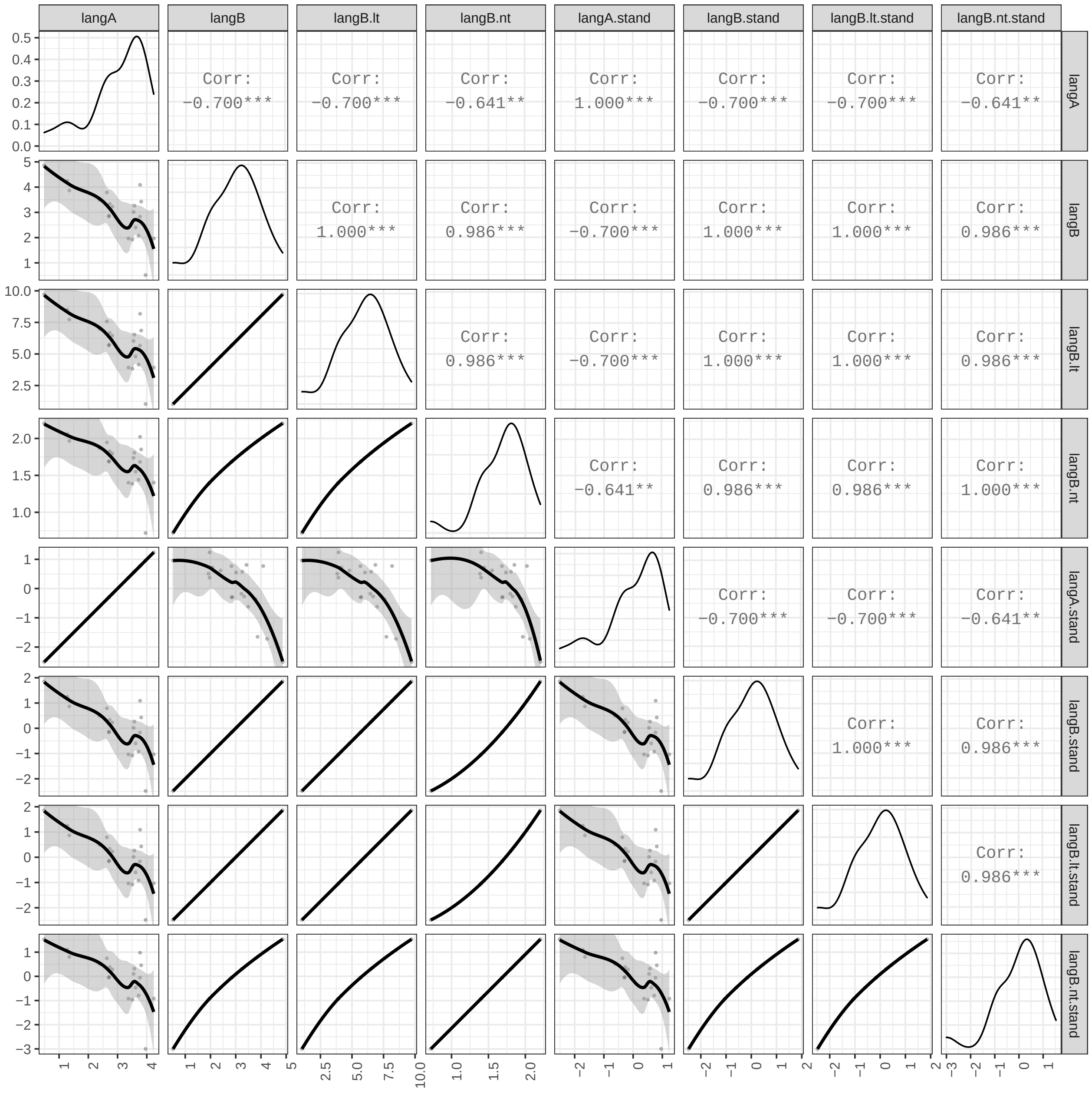
```
##     language measurement    value
## 1 Language A           1 4.238895
## 2 Language A           2 3.767582
## 3 Language A           3 3.507511
## 4 Language A           4 0.490032
## 5 Language A           5 3.549220
## 6 Language A           6 3.374442
```

```r
# 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.

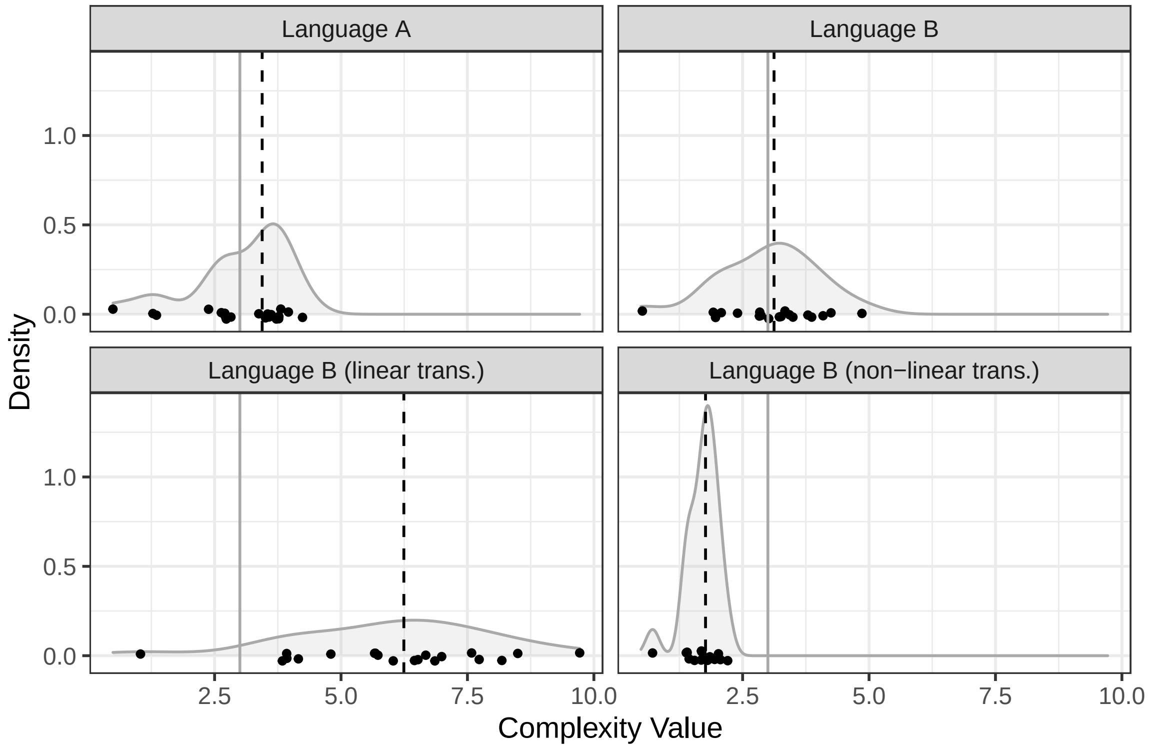
### Non-Standardized Vectors

Get mean, median, and standard deviation values.

```r
# 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.

```r
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

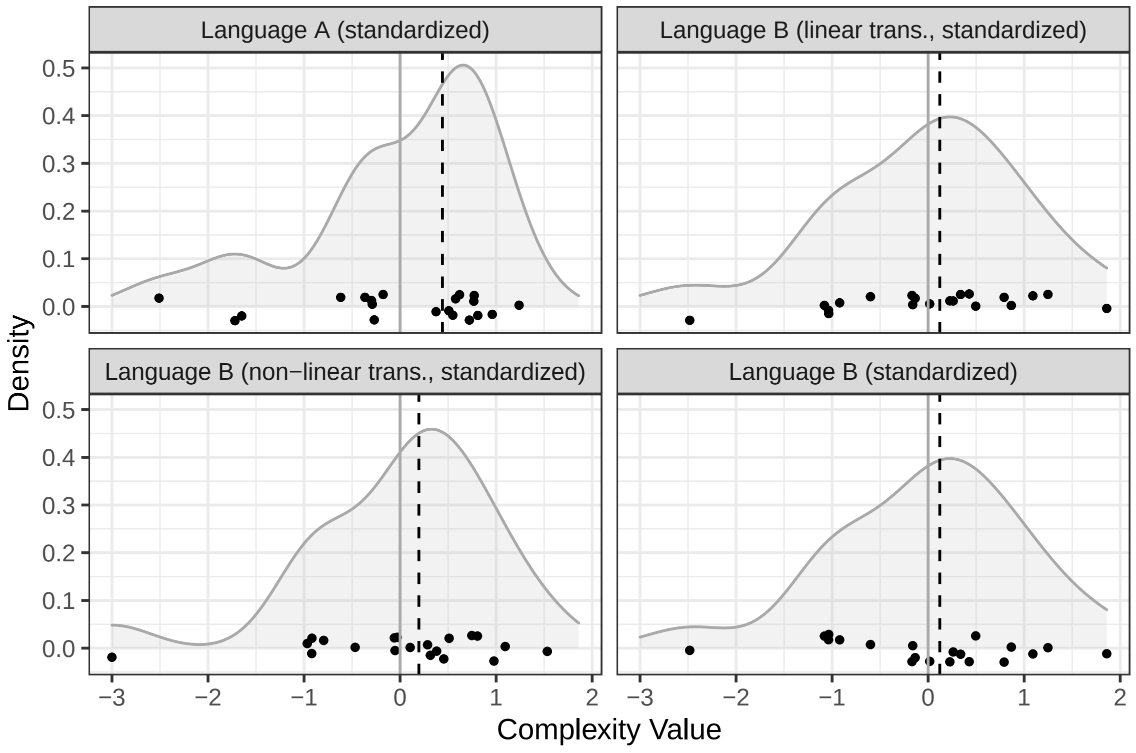
Get mean, median, and standard deviation values.

```r
# 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)
```



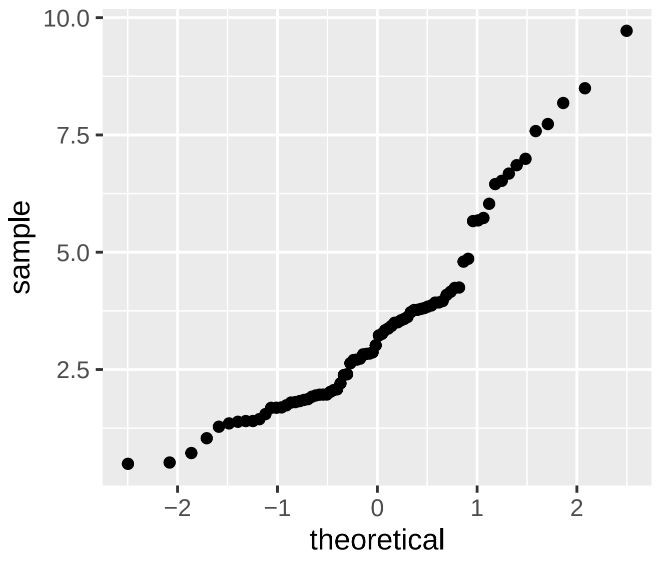
Save figure to file.

```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).

```r
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.

```r
# 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

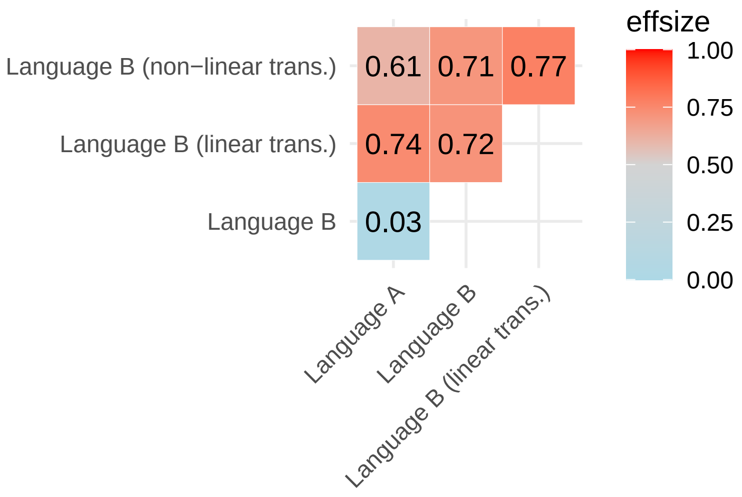
Statistical significance is only one part of the story. For instance, a difference in complexity values might be statistically significant, but so small that it is negligible for any theorizing. In fact, it is sometimes argued that effect sizes - rather than p-values - should be the aim of statistical 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

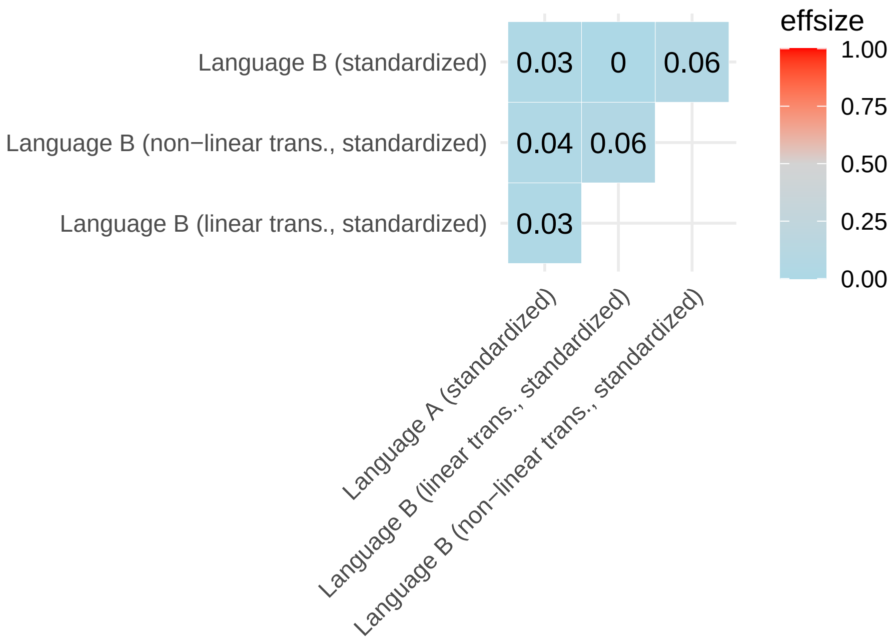
Plot a heatmap with effect sizes to get a better overview.

```
# 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.