Priming Mindfulness Project

Comparison & analysis report

Rémi Thériault

2022-05-15

Contents

Introduction

This report describes the results of the first priming mindfulness study, as well as of the two online pilot studies from the Varela project. It was made using Dominique Makowski's Supplementary Materials template (click "visit website" above for info).

Packages & Data

Packages

```
library(rempsyc)
library(dplyr)
library(interactions)
library(performance)
library(see)
library(patchwork)
library(ggplot2)
library(rstatix)
library(forecast)
library(DescTools)
library(report)
library(bestNormalize)

summary(report::report(sessionInfo()))
```

The analysis was done using the R Statistical language (v4.2.0; R Core Team, 2022) on Windows 10 x64, using the packages bestNormalize (v1.8.2), DescTools (v0.99.44), effectsize (v0.6.0.7), ggplot2 (v3.3.5), forecast (v8.16), rmarkdown (v2.13), interactions (v1.1.5), parameters (v0.17.0.5), insight (v0.17.0), performance (v0.9.0), see (v0.7.0), easystats (v0.4.3), correlation (v0.8.0), modelbased (v0.8.0), bayestestR (v0.11.5.2), report (v0.5.1), datawizard (v0.4.0), tidyverse (v1.3.1), dplyr (v1.0.8), forcats (v0.5.1), patchwork (v1.1.1), purrr (v0.3.4), readr (v2.1.2), rempsyc (v0.0.3.3), rstatix (v0.7.0), stringr (v1.4.0), tibble (v3.1.6) and tidyr (v1.2.0).

Data

```
df <- read.csv("data/data.csv")
data <- read.csv("data/fulldataset.csv")

cat(paste("The data consists of",
          report::report_participants(data),
          ". There are no demographics available."))
```

The data consists of 245 participants () . There are no demographics available.

```
# Dummy-code group variable
data <- data %>%
  mutate(condition_dum = ifelse(condition == "Mindfulness", 1, 0),
         condition = as.factor(condition))

# Allocation ratio
cat(paste("The allocation ratio is: ",
          report::report(data$condition)))
```

The allocation ratio is: x: 2 levels, namely Control (n = 128, 52.24%) and Mindfulness (n = 117, 47.76%)

Data Preparation

In this stage, we define a list of our relevant variables and standardize them according to the Median Absolute Deviation (MAD), which is more robust to extreme observations than standardization around the mean.

```r
# Make list of DVs
col.list <- c("blastintensity", "blastduration", "blastintensity.duration",
              "blastintensity.first", "blastduration.first",
              "blastintensity.duration.first", "KIMS", "BSCS", "BAQ",
              "SOPT", "IAT")

# Create new variable blastintensity.duration
data$blastintensity.duration <- (data$blastintensity * data$blastduration)
data$blastintensity.duration.first <- (data$blastintensity.first *
                                         data$blastduration.first)
# Divide by 2? Do some other sort of transformation given I multiplied two scores?
# Should I multiply them after standardization or before?

# Standardize and center main continuous IV variable (based on MAD)
# data <- data %>%
#   mutate(across(all_of(col.list),
#                 ~as.numeric(.x), #scale_mad(.x),
#                 .names = "{col}.mad"))
# We avoid standardizing now because it creates problems with the bestNormalize() function, and the lat

# Rename col.list with the MAD extension
# col.list <- paste0(col.list, ".mad")
```

Basic

Blast Intensity * Duration Why combine the intensity and duration scores? Should we? For a discussion, see:

Elson, M., Mohseni, M. R., Breuer, J., Scharkow, M., & Quandt, T. (2014). Press CRTT to measure aggressive behavior: the unstandardized use of the competitive reaction time task in aggression research. *Psychological assessment*, *26*(2), 419. https://doi.org/10.1037/a0035569

- Bushman and Baumeister (1998) used the sum of volume and duration settings in the first of 25 trials [p. 3]
- Lindsay and Anderson (2000) multiplied volume with log-transformed duration settings. The average over 25 trials of those products was their measure for overall aggression.
- Carnagey and Anderson (2005) averaged the products of volume and the square root of duration to form a single "aggressive energy score" (p. 887). No reason is given for this other than the claim that this single score supposedly is a valid measure and that duration should be square rooted.
- Bartholow, Sestir, and Davis (2005) multiplied the average volume and duration settings to form a composite aggressive behavior score. Although Bartholow, Bushman, and Sestir (2006) also used volume and duration settings, they standardized and summed the two parameters instead of multiplying them.
- Sometimes the option of setting the volume and/or duration to zero as a way to act nonaggressively is provided. Including settings of zero as an option also raises further questions, for example, how to handle trials in which participants set only one of the two intensity parameters to zero. [**Note: we do have zero as option**]
- With regard to the analysis, there is no definitive answer to the question of how to calculate aggression scores, or whether different scores might measure different types of aggression, as long as none of them have been properly validated. As it seems that volume and duration do not measure the exact same construct, it is advisable to consider them as separate measures for related subdimensions of aggression.

First sound blast Why use the first sound blast only instead of the average of all trials? Should we?

According to some, the Taylor Aggression Paradigm is not a measure of aggression per say, but of reactive aggression, because participants react to the other "participant's" aggression. They suggest that for a pure measure of aggression, it is recommended to use only the first sound blast used by the participant before he receives one himself. At this stage, we attempt the analyses with all these different measures of aggression for exploratory purposes. See earlier reference to Elson et al. (2014):

- If researchers are interested in measuring unprovoked aggression, they should also look at the settings in the first trial. Those studying provoked aggression or retaliation, on the other hand, should focus on all trials except the first one.

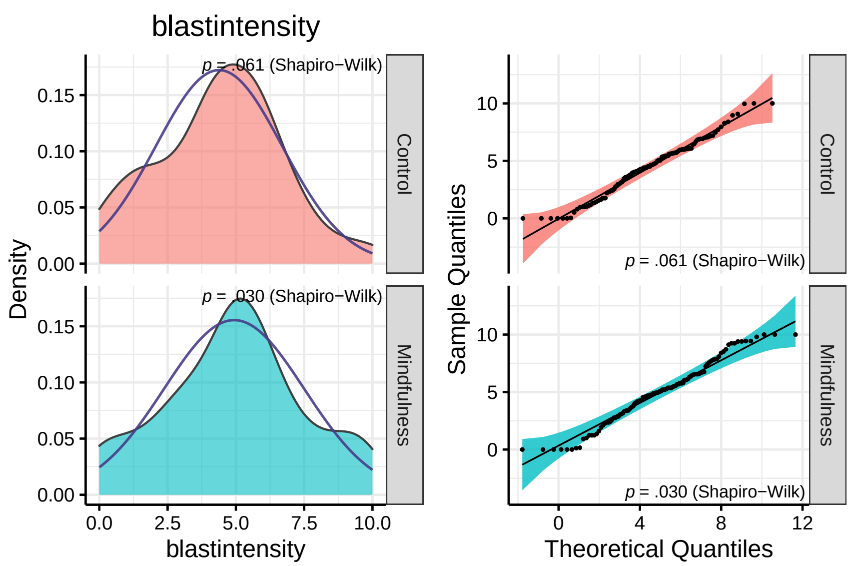
t-tests

In this section, we will: (a) test assumptions of normality, (b) transform variables violating assumptions, (c) test assumptions of homoscedasticity, (d) identify and winsorize outliers, and (e) conduct the t-tests.

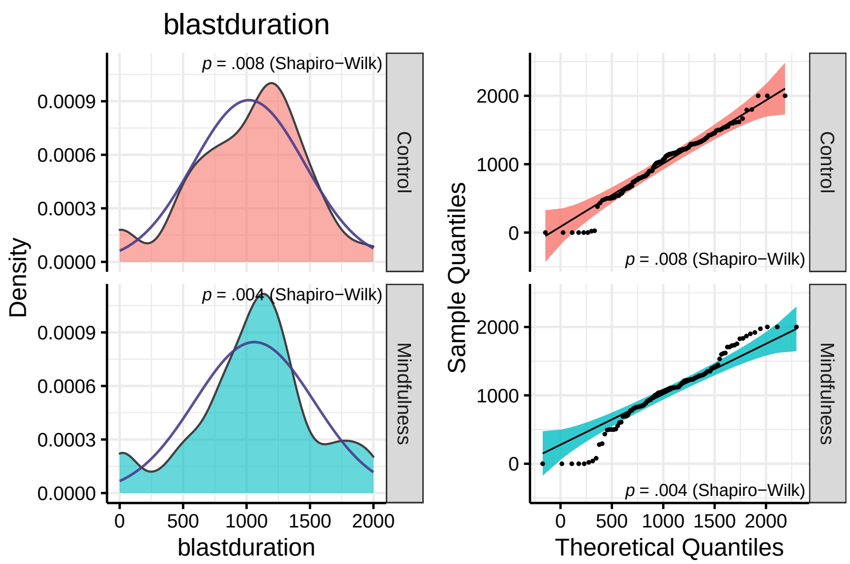
Normality

```
# Group normality
sapply(col.list, function(x)
  nice_normality(data,
                 x,
                 "condition",
                 shapiro = TRUE,
                 title = x),
  USE.NAMES = TRUE,
  simplify = FALSE)
```

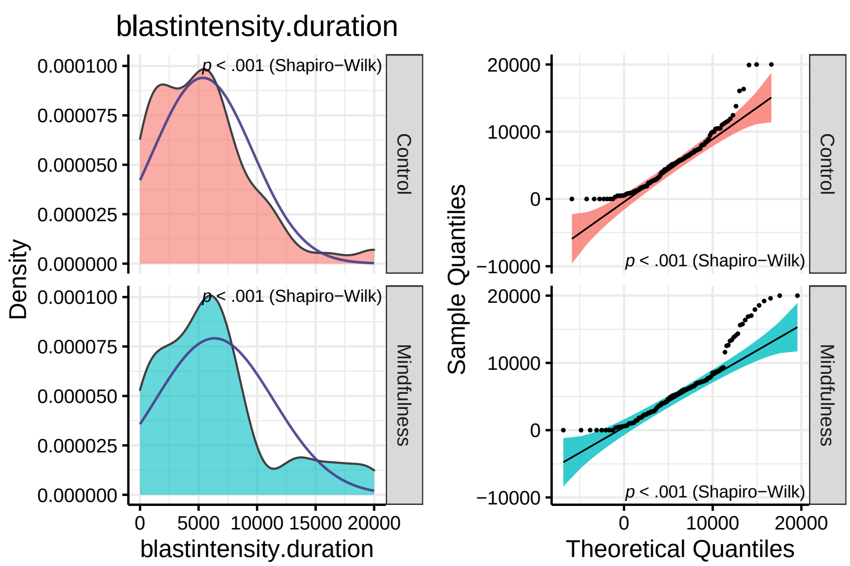
```
> $blastintensity
```



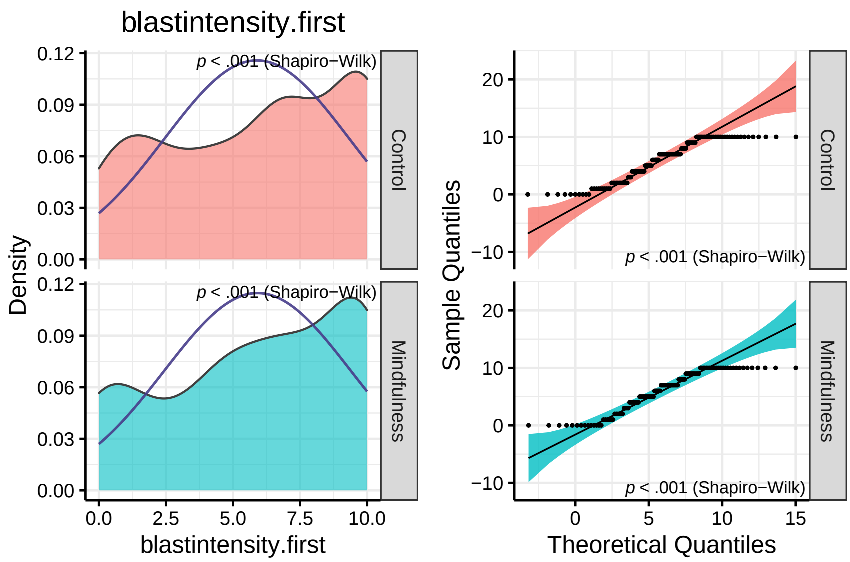

```
>
> $blastduration
```



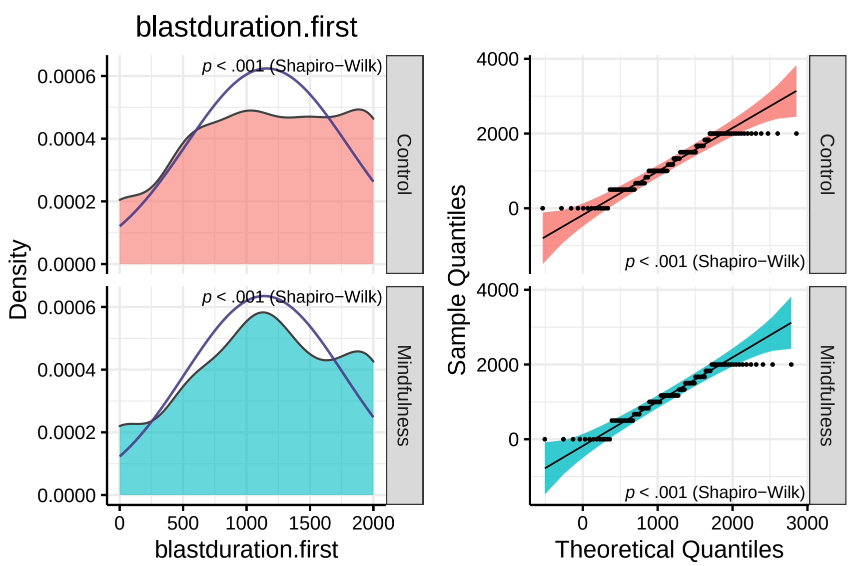

```
>
> $blastintensity.duration
```



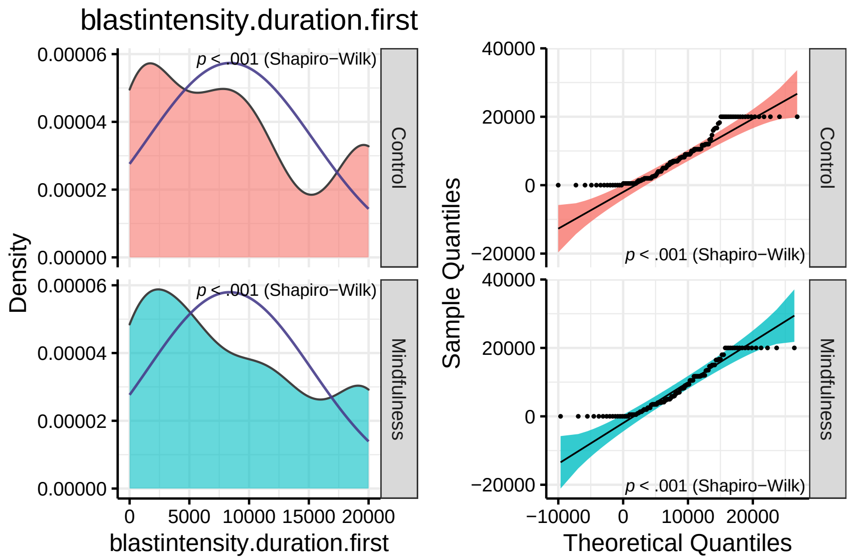

```
>
> $blastintensity.first
```




```
>
> $blastduration.first
```

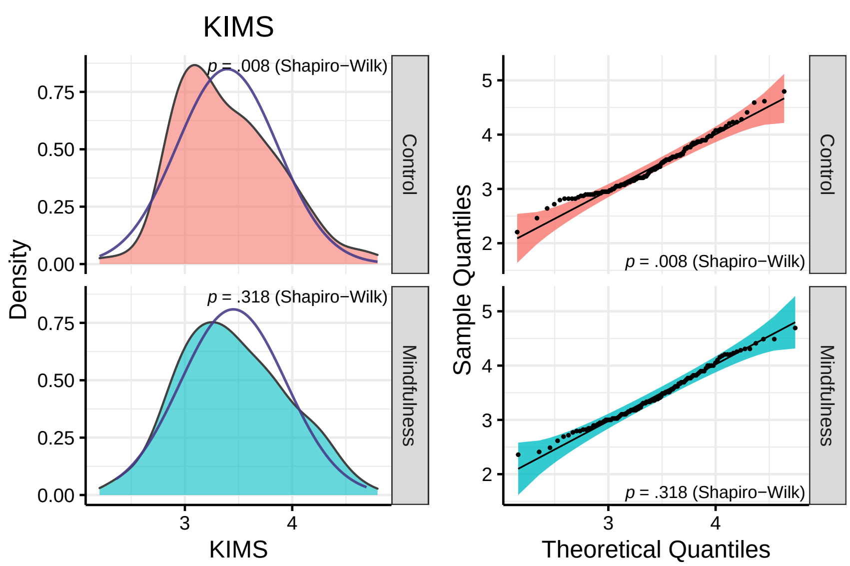



```
>
> $blastintensity.duration.first
```



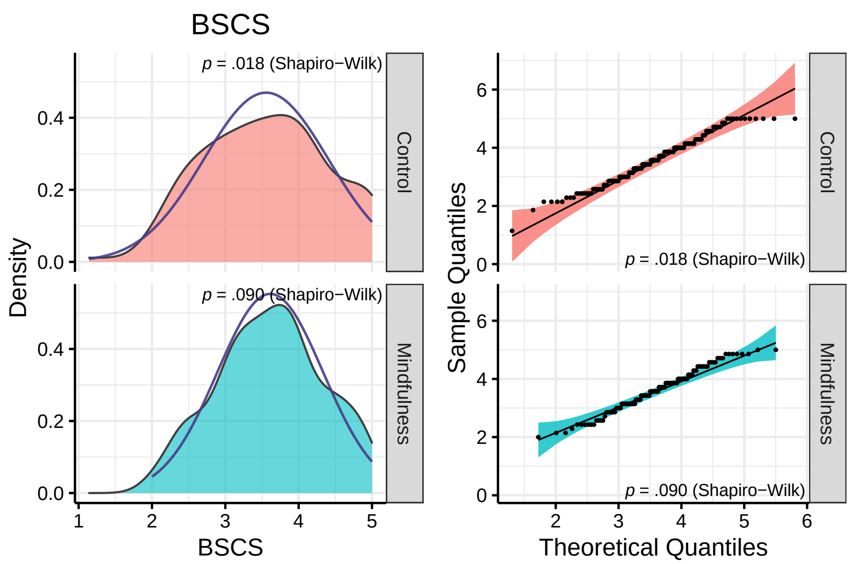
blastintensity.duration.first
p < .001 (Shapiro−Wilk)
Control
Mindfulness
Density
Sample Quantiles
Theoretical Quantiles
blastintensity.duration.first


```
>
> $KIMS
```



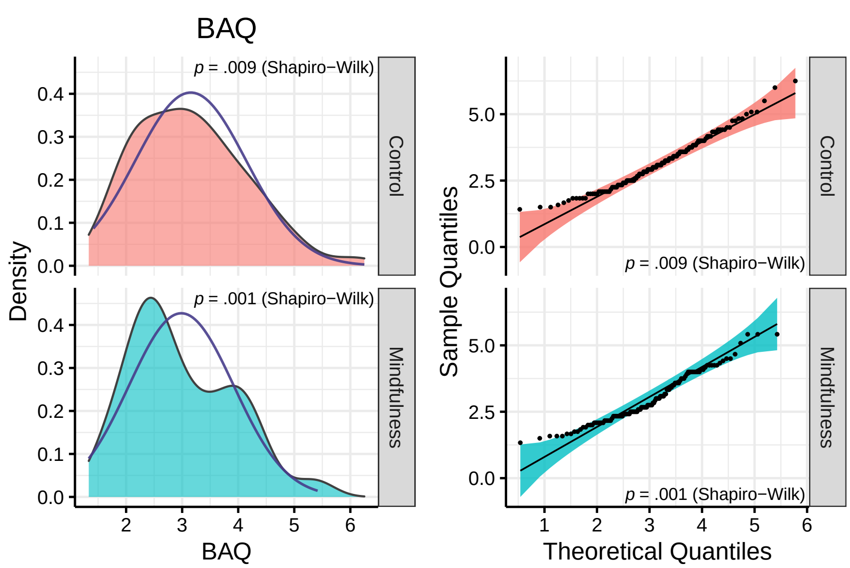
KIMS
p = .008 (Shapiro−Wilk)
p = .318 (Shapiro−Wilk)
Control
Mindfulness
Density
Sample Quantiles
KIMS
Theoretical Quantiles


```
>
> $BSCS
```



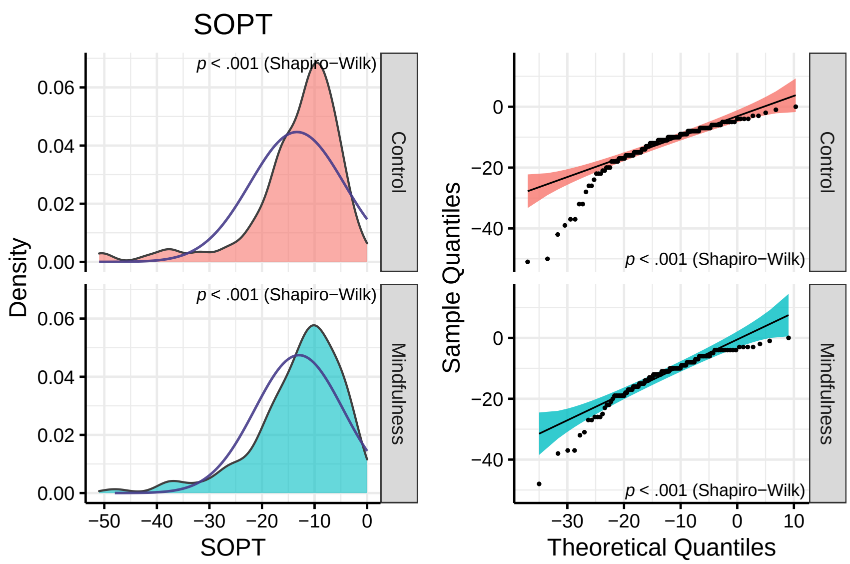
BSCS
p = .018 (Shapiro−Wilk)
p = .090 (Shapiro−Wilk)
Control
Mindfulness
Density
Sample Quantiles
Theoretical Quantiles


```
>
> $BAQ
```

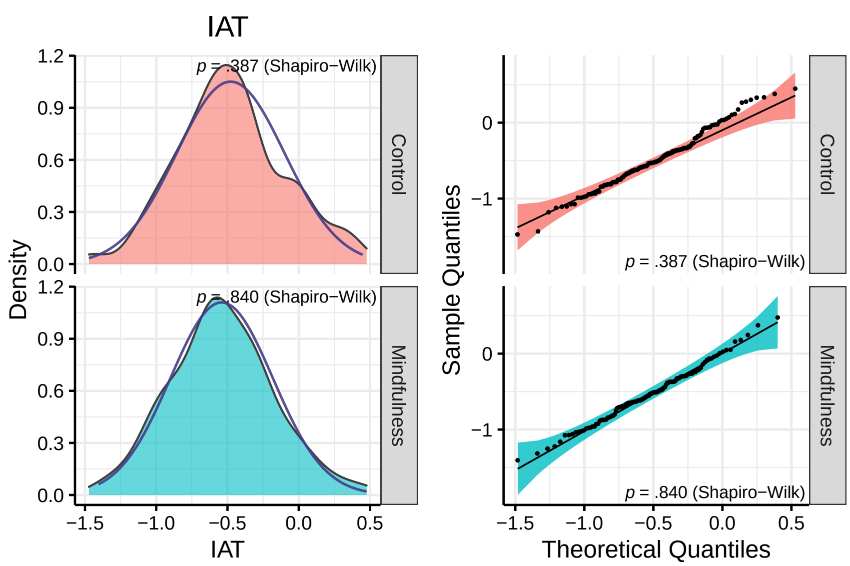


BAQ
p = .009 (Shapiro−Wilk)
p = .001 (Shapiro−Wilk)
Control
Mindfulness
Density
Sample Quantiles
Theoretical Quantiles


```
>
> $SOPT
```




```
>
> $IAT
```



Several variables are clearly skewed. Let's apply transformations.

Transformation

```
# <!-- Normally, the SOPT raw scores represent the number of errors, but I had multiplied it by -1 init
#
# <!-- We also add a constant of 1 to avoid scores of zero which can interfere with the transformation.

# Not necessary anymore since we use the `bestNormalize` package.
```

The command below transforms variables according to the best possible transformation (via the `bestNormalize` package), and also standardize the variables.

```r
predict_bestNormalize <- function(var, print.transform = TRUE) {
  x <- bestNormalize(var, standardize = TRUE)
  if (print.transform == TRUE) {
    print(cur_column())
    print(x$chosen_transform)
  }
  predict(x)
}

set.seed(100)
data <- data %>%
  mutate(across(all_of(col.list),
                predict_bestNormalize,
                .names = "{.col}.BN"))
```

```
> [1] "blastintensity"
> orderNorm Transformation with 245 nonmissing obs and ties
>  - 142 unique values
>  - Original quantiles:
>   0%  25%  50%  75% 100%
>  0.0  3.0  4.8  6.0 10.0
> [1] "blastduration"
> orderNorm Transformation with 245 nonmissing obs and ties
>  - 201 unique values
>  - Original quantiles:
>   0%  25%  50%  75% 100%
>    0  793 1100 1300 2000
> [1] "blastintensity.duration"
> Standardized sqrt(x + a) Transformation with 245 nonmissing obs.:
>  Relevant statistics:
>  - a = 0
>  - mean (before standardization) = 69
>  - sd (before standardization) = 33
> [1] "blastintensity.first"
> orderNorm Transformation with 245 nonmissing obs and ties
>  - 11 unique values
>  - Original quantiles:
>   0%  25%  50%  75% 100%
>    0    3    7    9   10
> [1] "blastduration.first"
> center_scale(x) Transformation with 245 nonmissing obs.
>  Estimated statistics:
>  - mean (before standardization) = 1149
>  - sd (before standardization) = 633
> [1] "blastintensity.duration.first"
> orderNorm Transformation with 245 nonmissing obs and ties
>  - 59 unique values
>  - Original quantiles:
>    0%   25%   50%   75%  100%
>     0  2000  7000 13300 20000
> [1] "KIMS"
> Standardized asinh(x) Transformation with 245 nonmissing obs.:
>  Relevant statistics:
>  - mean (before standardization) = 1.9
```

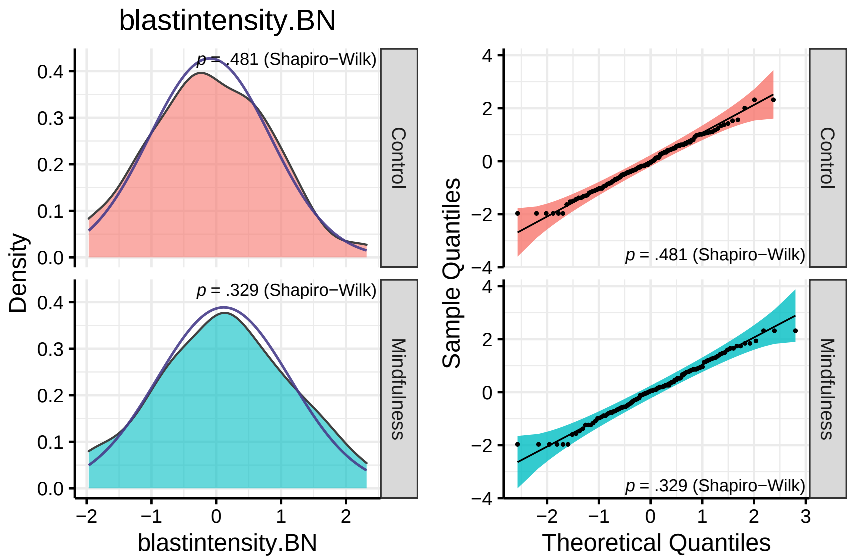
```
>  - sd (before standardization) = 0.13
> [1] "BSCS"
> Standardized asinh(x) Transformation with 245 nonmissing obs.:
>  Relevant statistics:
>  - mean (before standardization) = 2
>  - sd (before standardization) = 0.22
> [1] "BAQ"
> Standardized asinh(x) Transformation with 245 nonmissing obs.:
>  Relevant statistics:
>  - mean (before standardization) = 1.8
>  - sd (before standardization) = 0.3
> [1] "SOPT"
> Standardized asinh(x) Transformation with 245 nonmissing obs.:
>  Relevant statistics:
>  - mean (before standardization) = -3.1
>  - sd (before standardization) = 0.69
> [1] "IAT"
> center_scale(x) Transformation with 245 nonmissing obs.
>  Estimated statistics:
>  - mean (before standardization) = -0.51
>  - sd (before standardization) = 0.37
```

```
col.list <- paste0(col.list, ".BN")
```

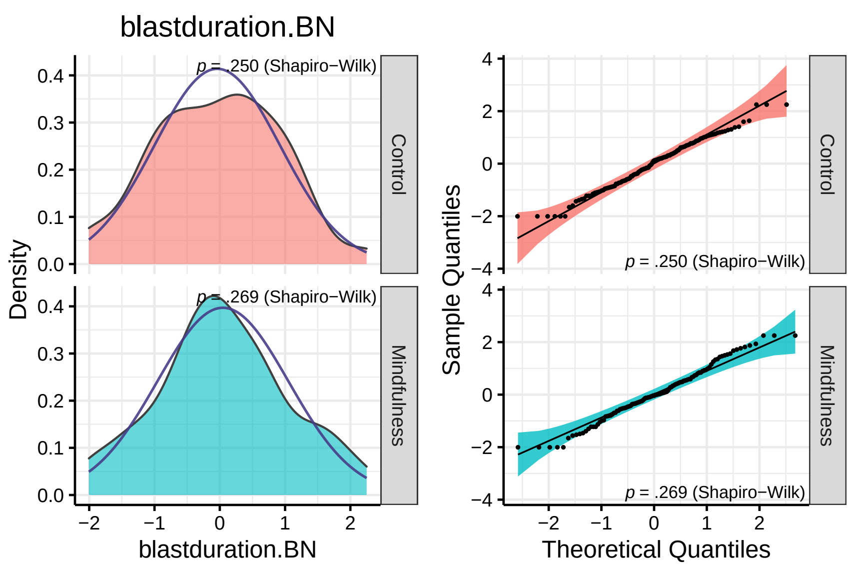
Let's check if normality was corrected.

```
# Group normality
sapply(col.list, function(x)
  nice_normality(data,
                 x,
                 "condition",
                 shapiro = TRUE,
                 title = x),
  USE.NAMES = TRUE,
  simplify = FALSE)
```

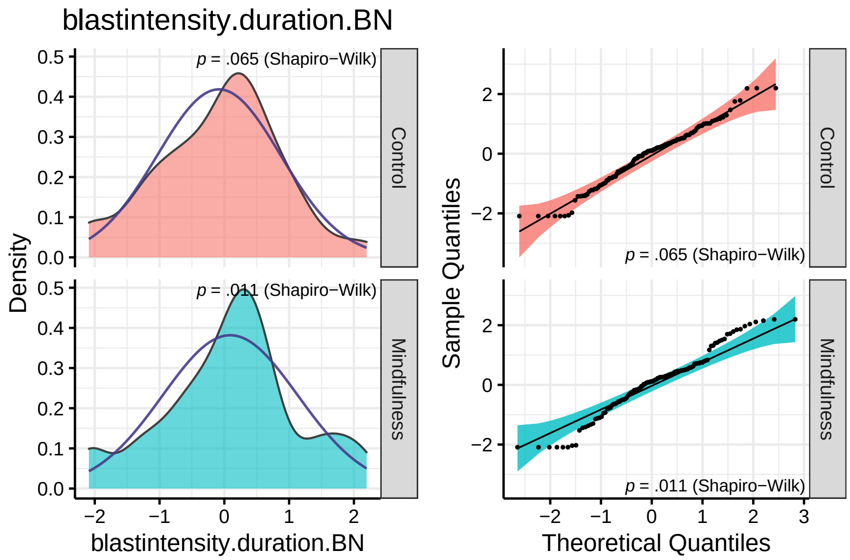
```
> $blastintensity.BN
```



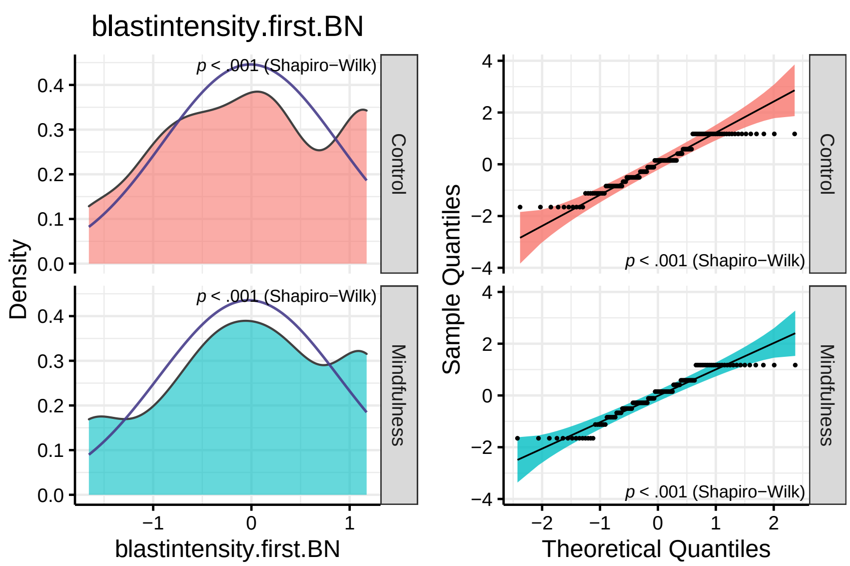

```
>
> $blastduration.BN
```



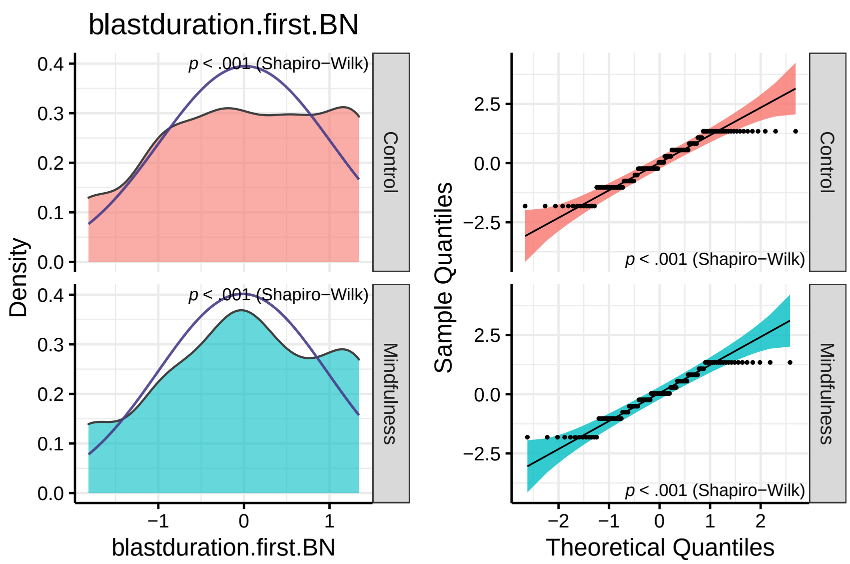

```
>
> $blastintensity.duration.BN
```



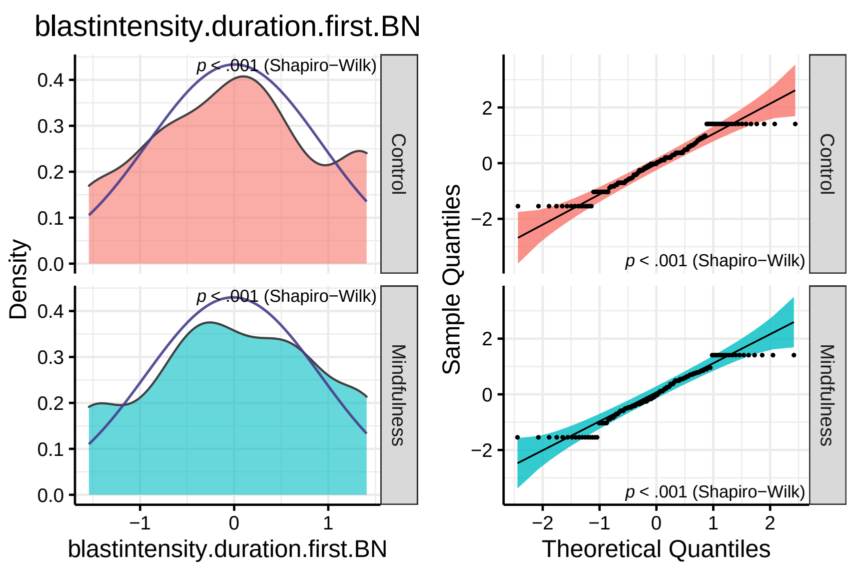

```
>
> $blastintensity.first.BN
```



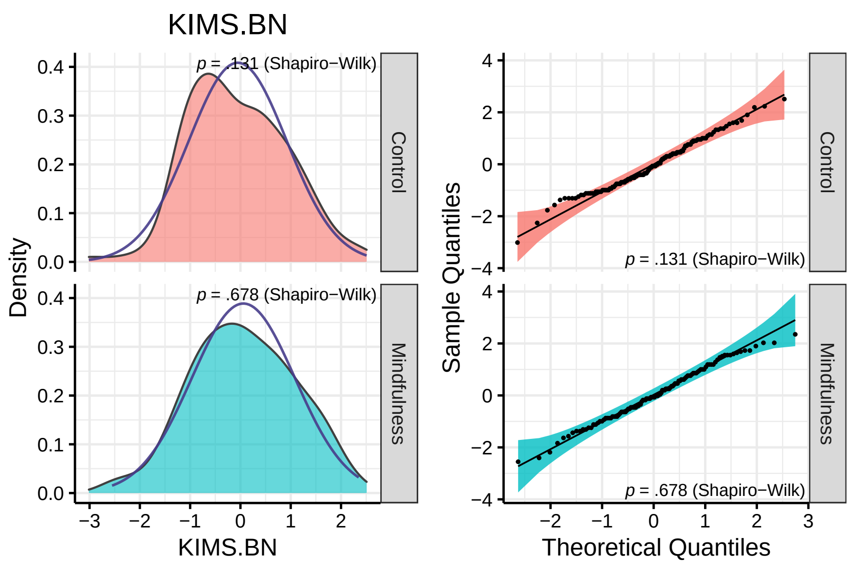

```
>
> $blastduration.first.BN
```



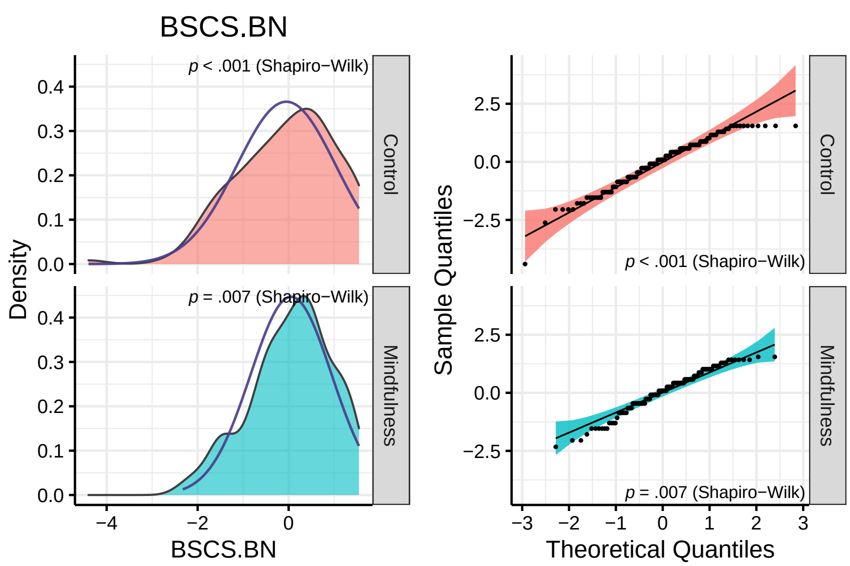

```
>
> $blastintensity.duration.first.BN
```



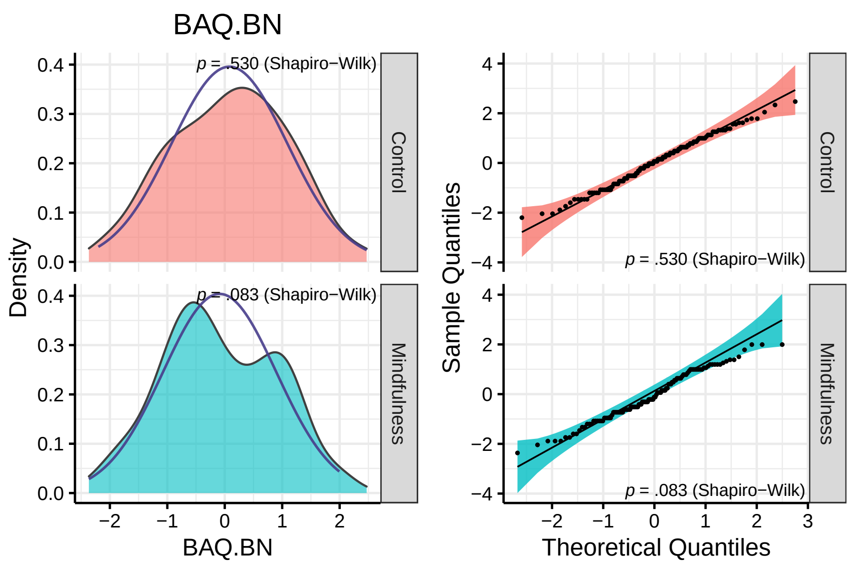

```
>
> $KIMS.BN
```



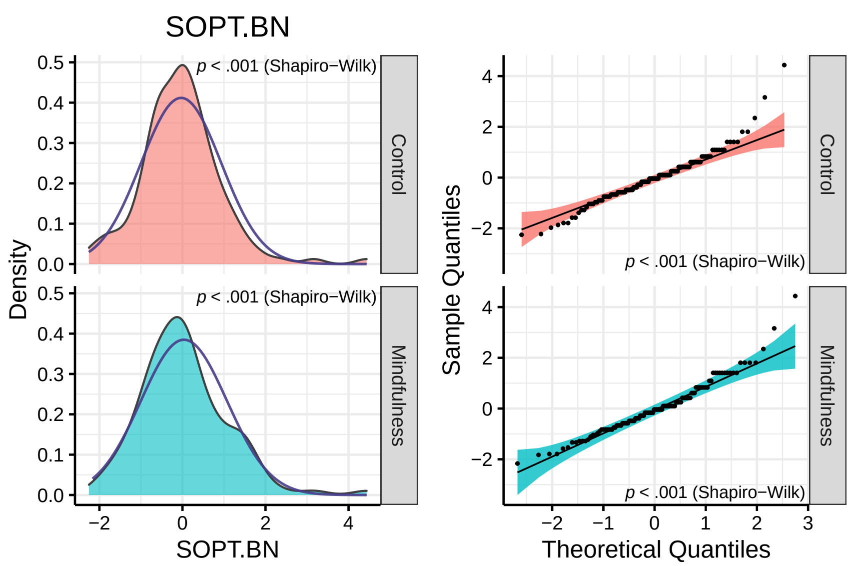

```
>
> $BSCS.BN
```



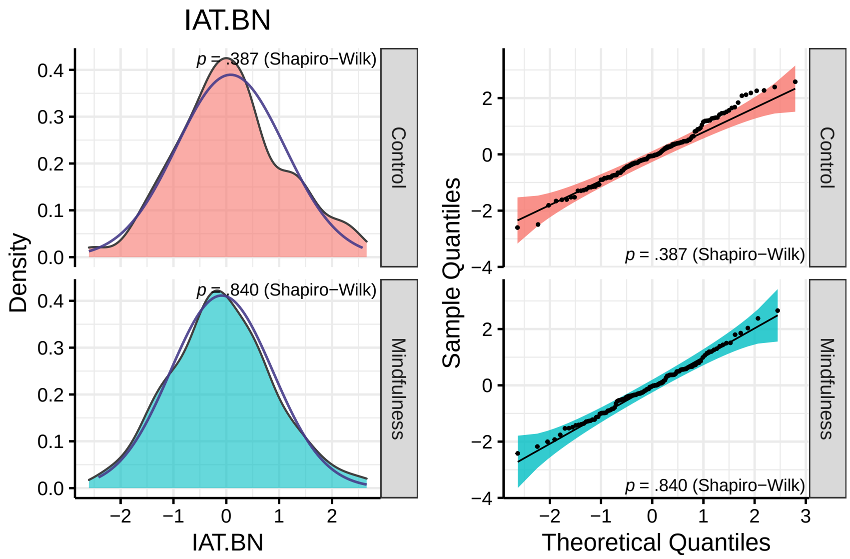

```
>
> $BAQ.BN
```




```
>
> $SOPT.BN
```



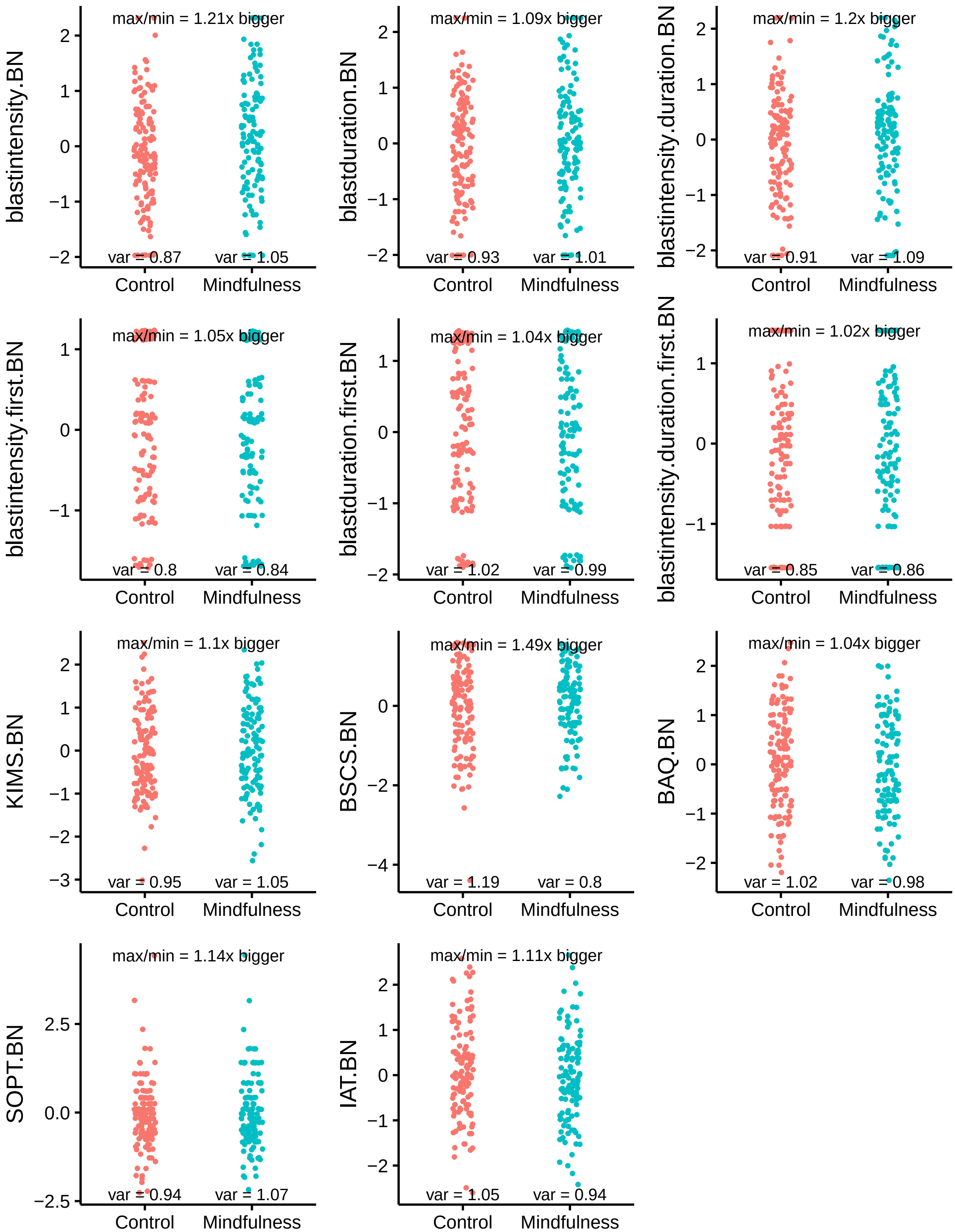

```
>
> $IAT.BN
```



Looks rather reasonable now, though not perfect (fortunately t-tests are quite robust against violations of normality). We can now resume with the next step: checking variance.

Homoscedasticity

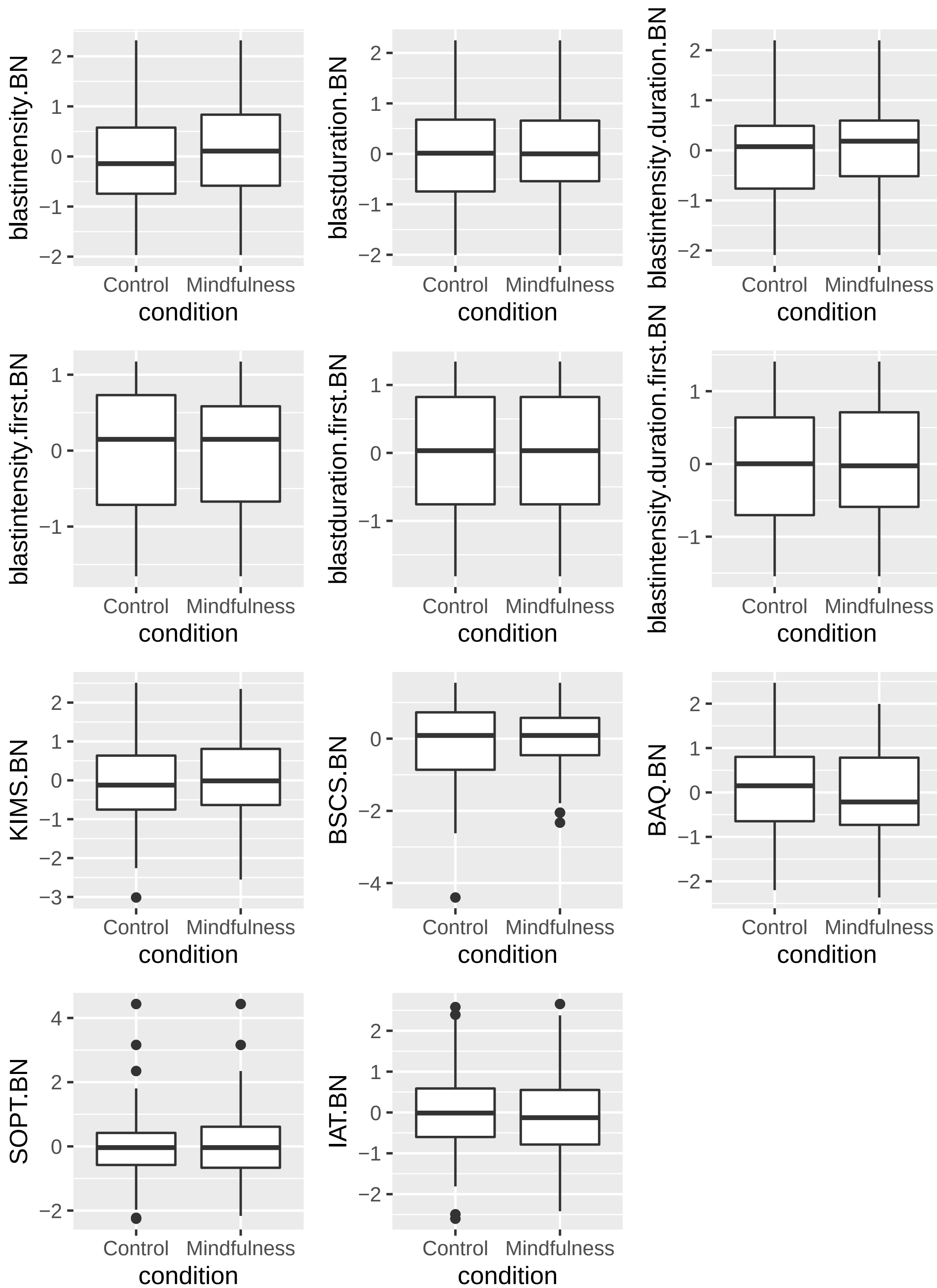
```
# Plotting variance
plots(lapply(col.list, function(x) {
  nice_varplot(data, x, group = "condition")
  }),
  n_columns = 3)
```



Variance looks good. No group has four times the variance of any other group. We can now resume with checking outliers.

Outliers

```
# Using boxplots
plots(lapply(col.list, function(x) {
  ggplot(data, aes(condition, !!sym(x))) +
  geom_boxplot()
  }),
  n_columns = 3)
```



There are some outliers, but nothing unreasonable. Let's still check with the 3 median absolute deviations (MAD) method.

```
find_mad(data, col.list, criteria = 3)
```

```
> 6 outlier(s) based on 3 median absolute deviations for variable(s):
```

```
>  blastintensity.BN blastduration.BN blastintensity.duration.BN blastintensity.first.BN blastduration.
>
> Outliers per variable:

> $BSCS.BN
>   Row BSCS.BN
> 1 242    -4.4
>
> $SOPT.BN
>   Row SOPT.BN
> 1  61     4.4
> 2  84     3.2
> 3  90     3.2
> 4 218     4.4
>
> $IAT.BN
>   Row IAT.BN
> 1  54    2.7
```

```
# 6 people after our transformations
```

Winsorization

Visual assessment and the MAD method confirm we have some outlier values. We could ignore them but because they could have disproportionate influence on the models, one recommendation is to winsorize them by bringing the values at 3 SD. Instead of using the standard deviation around the mean, however, we use the absolute deviation around the median, as it is more robust to extreme observations. For a discussion, see:

Leys, C., Klein, O., Bernard, P., & Licata, L. (2013). Detecting outliers: Do not use standard deviation around the mean, use absolute deviation around the median. *Journal of Experimental Social Psychology, 49*(4), 764–766. https://doi.org/10.1016/j.jesp.2013.03.013

```
# Winsorize variables of interest with MAD
data <- data %>%
  mutate(across(all_of(col.list),
                winsorize_mad,
                .names = "{.col}.w"))

# Update col.list
col.list <- paste0(col.list, ".w")
```

Outliers are still present but were brought back within reasonable limits, where applicable. We are now ready to compare the group condition (Control vs. Mindfulness Priming) across our different variables with the t-tests.

t-tests

```
nice_t_test(data,
            response = col.list,
            group = "condition") %>%
  nice_table(highlight = 0.10)
```

```
> Using Welch t-test (base R's default; cf. https://doi.org/10.5334/irsp.82).
> For the Student t-test, use `var.equal = TRUE`.
>
```

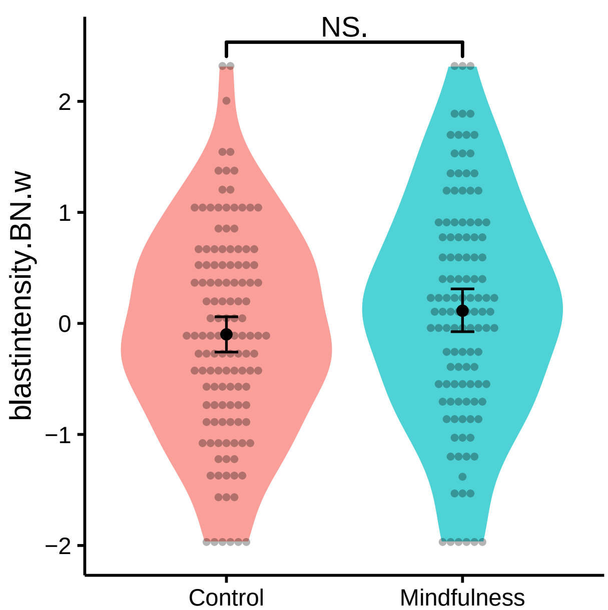
>

Dependent Variable	*t*	*df*	*p*	*d*	95% CI
blastintensity.BN.w	**-1.69**	**235.03**	**.092**	**-0.22**	**[-0.47, 0.03]**
blastduration.BN.w	-0.69	238.85	.491	-0.09	[-0.34, 0.16]
blastintensity.duration.BN.w	-1.35	235.28	.177	-0.17	[-0.42, 0.08]
blastintensity.first.BN.w	0.15	239.90	.882	0.02	[-0.23, 0.27]
blastduration.first.BN.w	0.25	241.71	.800	0.03	[-0.22, 0.28]
blastintensity.duration.first.BN.w	0.14	240.64	.893	0.02	[-0.23, 0.27]
KIMS.BN.w	-0.87	238.36	.383	-0.11	[-0.36, 0.14]
BSCS.BN.w	-0.74	241.85	.459	-0.09	[-0.34, 0.16]
BAQ.BN.w	1.34	241.81	.182	0.17	[-0.08, 0.42]
SOPT.BN.w	-0.55	236.69	.583	-0.07	[-0.32, 0.18]
IAT.BN.w	1.33	242.70	.184	0.17	[-0.08, 0.42]

Interpretation: There is no clear group effect from our experimental condition on our different variables. However, there is a marginal effect of condition on blast intensity, whereas the mindfulness group has slightly higher blast intensity than the control group. Let's visualize this effect.

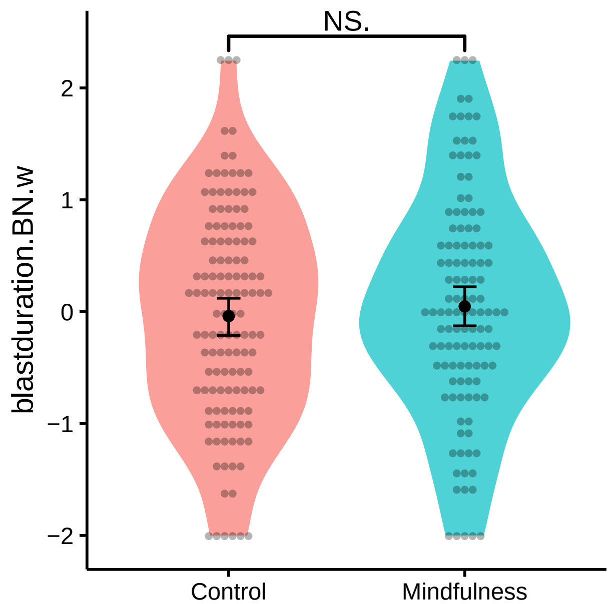
Violin plots

```
nice_violin(data,
            group = "condition",
            response = "blastintensity.BN.w",
            comp1 = 1,
            comp2 = 2,
            obs = TRUE)
```



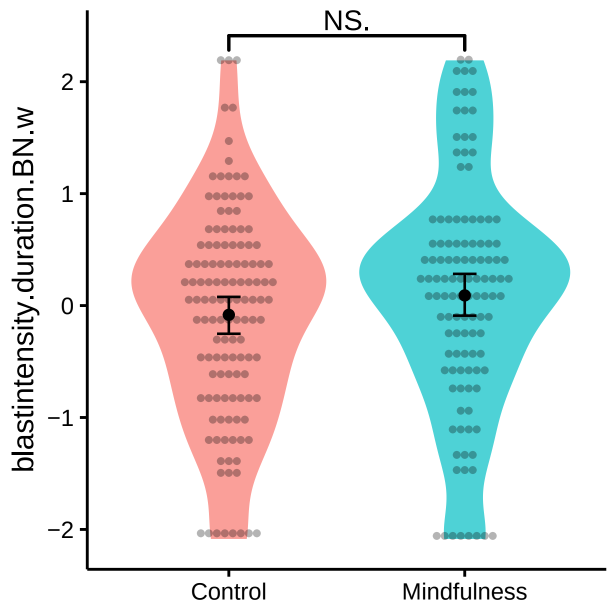
Blast Intensity

```
nice_violin(data,
            group = "condition",
            response = "blastduration.BN.w",
            comp1 = 1,
            comp2 = 2,
            obs = TRUE)
```



Blast Duration

```
nice_violin(data,
            group = "condition",
            response = "blastintensity.duration.BN.w",
            comp1 = 1,
            comp2 = 2,
            obs = TRUE)
```



Blast Intensity * Duration

Means, SD

Let's extract the means and standard deviations for journal reporting.

```
data %>%
    group_by(condition) %>%
    summarize(M = mean(blastintensity),
              SD = sd(blastintensity),
              N = n()) %>%
  nice_table(width = 0.40)
```

Blast Intensity

condition	*M*	*SD*	*N*
Control	4.38	2.32	128
Mindfulness	4.93	2.57	117

```
data %>%
    group_by(condition) %>%
    summarize(M = mean(blastduration),
              SD = sd(blastduration),
              N = n()) %>%
  nice_table(width = 0.40)
```

Blast Duration

condition	*M*	*SD*	*N*
Control	1,019.01	440.26	128
Mindfulness	1,061.38	471.63	117

condition	*M*	*SD*	*N*

```r
data %>%
    group_by(condition) %>%
    summarize(M = mean(blastintensity.duration),
              SD = sd(blastintensity.duration),
              N = n()) %>%
  nice_table(width = 0.40)
```

Blast Intensity * Duration

condition	*M*	*SD*	*N*
Control	5,368.70	4,245.46	128
Mindfulness	6,356.67	5,041.47	117

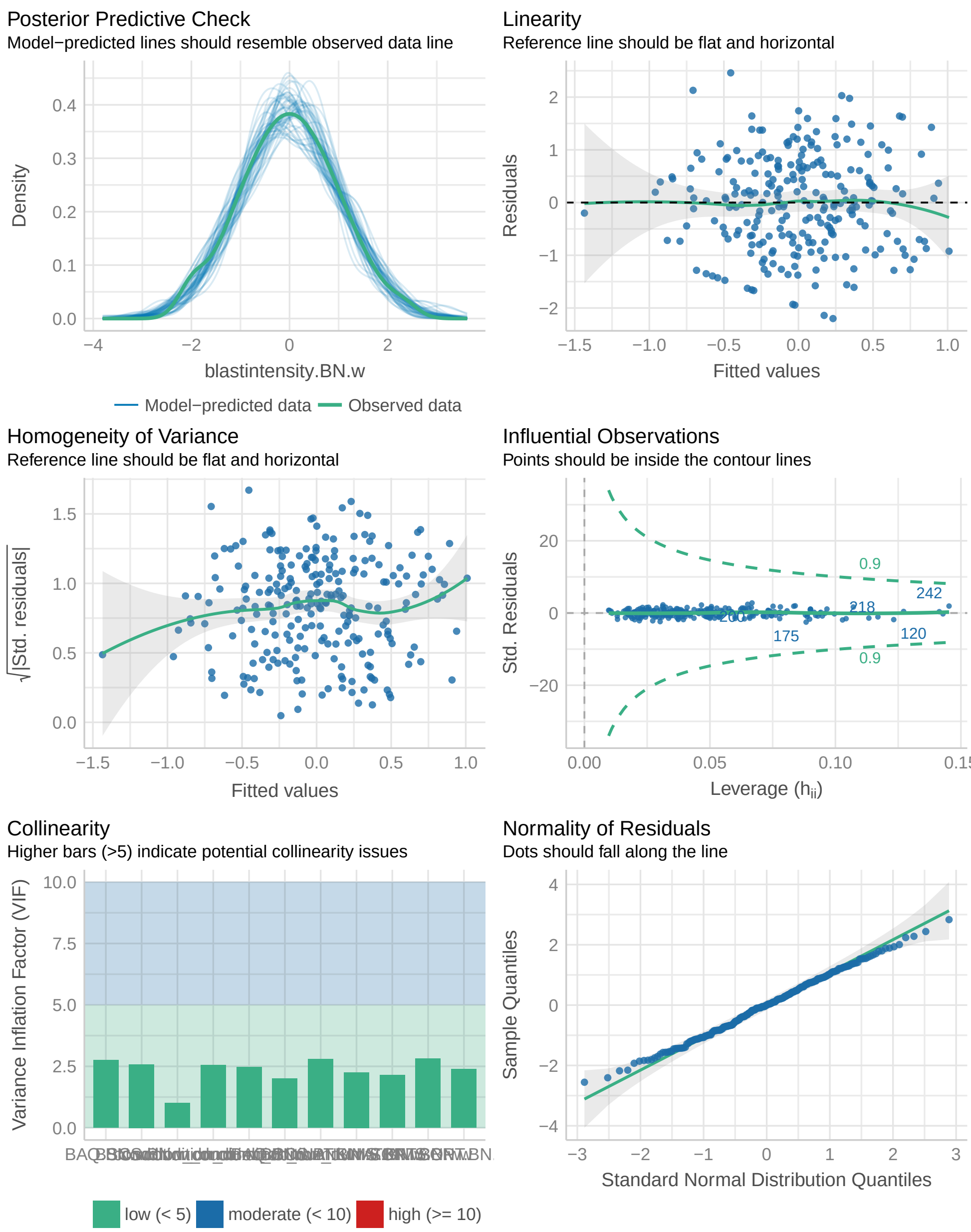
Moderations

Let's see if our variables don't interact together with our experimental condition. But first, let's test the models assumptions.

Assumptions

Blast Intensity

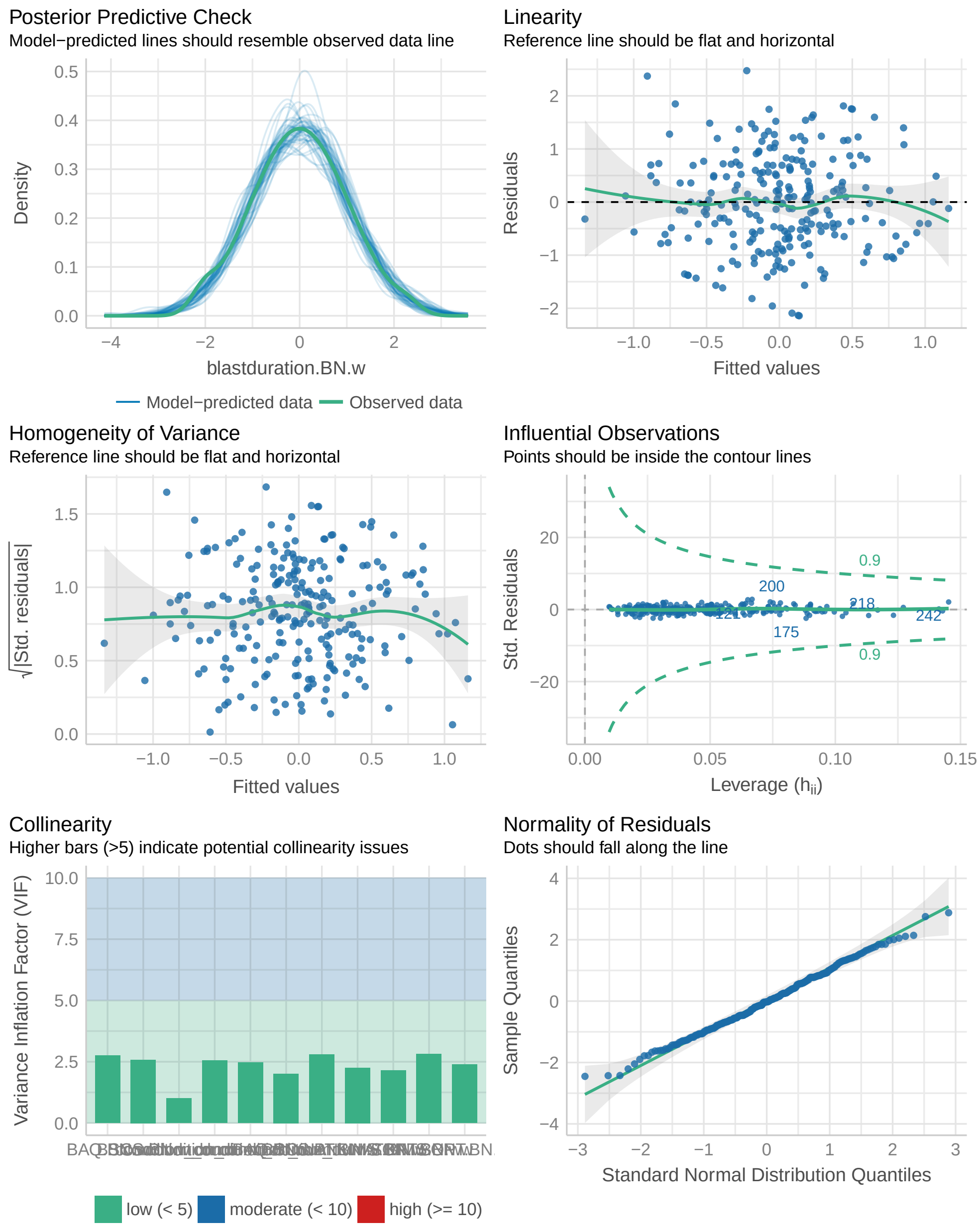
```r
big.mod1 <- lm(blastintensity.BN.w ~ condition_dum*KIMS.BN.w +
                 condition_dum*BSCS.BN.w + condition_dum*BAQ.BN.w +
                 condition_dum*SOPT.BN.w + condition_dum*IAT.BN.w,
               data = data, na.action="na.exclude")
check_model(big.mod1)
```



The model assumptions look really good actually, even with all these variables [less true now with the standardization and winsorization!]. Let's look at the results.

Blast Duration

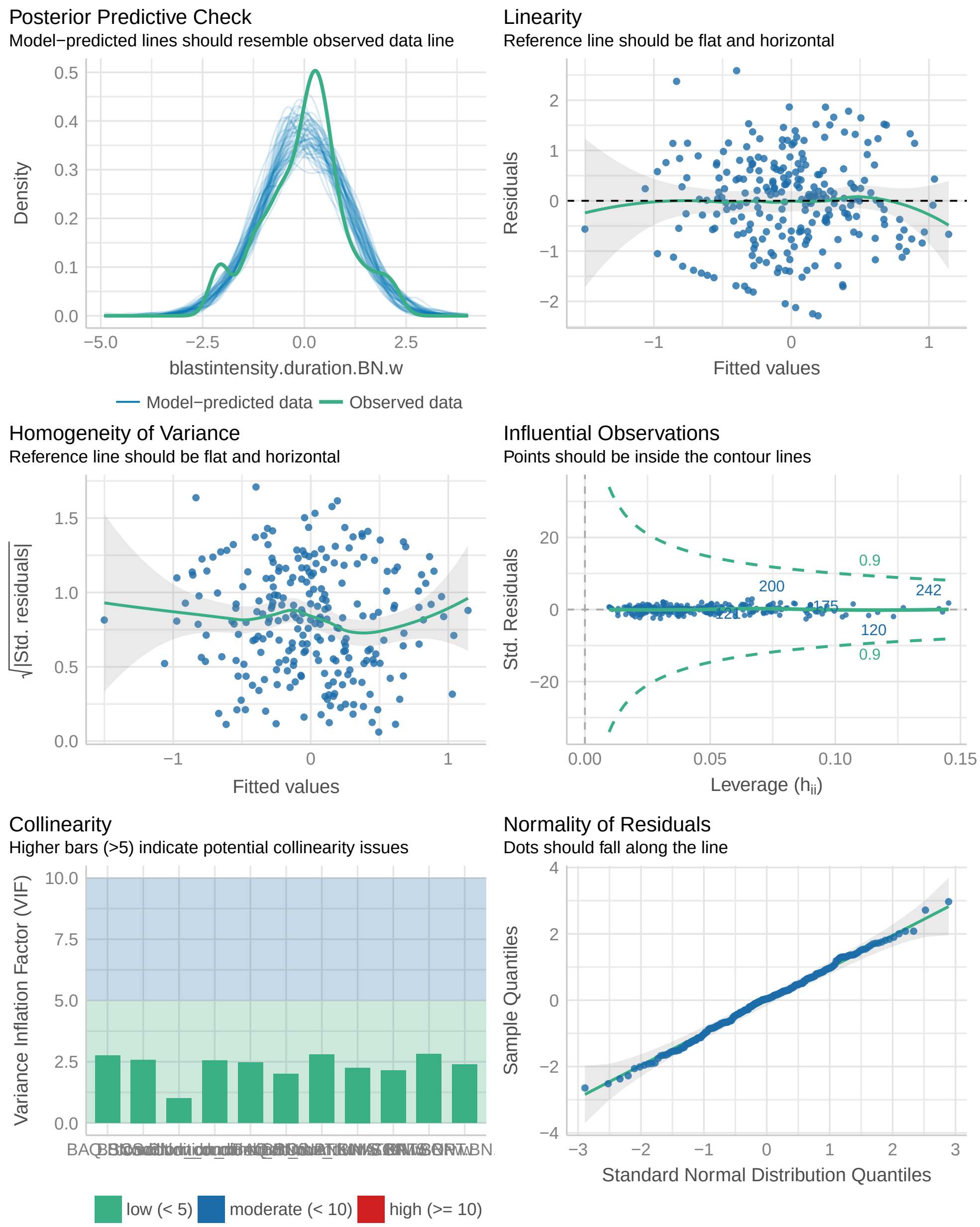
```
big.mod2 <- lm(blastduration.BN.w ~ condition_dum*KIMS.BN.w +
                 condition_dum*BSCS.BN.w + condition_dum*BAQ.BN.w +
                 condition_dum*SOPT.BN.w + condition_dum*IAT.BN.w,
               data = data, na.action="na.exclude")
check_model(big.mod2)
```



The model assumptions look really good actually, even with all these variables [less true now with the standardization and winsorization!]. Let's look at the results.

Blast Intensity * Duration

```
big.mod3 <- lm(blastintensity.duration.BN.w ~ condition_dum*KIMS.BN.w +
                 condition_dum*BSCS.BN.w + condition_dum*BAQ.BN.w +
                 condition_dum*SOPT.BN.w + condition_dum*IAT.BN.w,
               data = data, na.action="na.exclude")
check_model(big.mod3)
```



The model assumptions look really good actually, even with all these variables [less true now with the standardization and winsorization!]. Let's look at the results.

Moderations

Blast Intensity

```
big.mod1 %>%
  nice_lm() %>%
  nice_table(highlight = TRUE)
```

Dependent Variable	Predictor	*df*	*b*	*t*	*p*	*sr*^{2}
blastintensity.BN.w	**condition_dum**	**233**	**0.25**	**2.10**	**.036**	**.02**

Dependent Variable	Predictor	*df*	*b*	*t*	*p*	sr^2
blastintensity.BN.w	KIMS.BN.w	233	0.15	1.52	.130	.01
blastintensity.BN.w	BSCS.BN.w	233	-0.17	-1.79	.075	.01
blastintensity.BN.w	BAQ.BN.w	233	0.08	0.82	.414	.00
blastintensity.BN.w	**SOPT.BN.w**	**233**	**-0.27**	**-2.71**	**.007**	**.03**
blastintensity.BN.w	**IAT.BN.w**	**233**	**0.17**	**1.99**	**.048**	**.01**
blastintensity.BN.w	condition_dum:KIMS.BN.w	233	-0.20	-1.44	.151	.01
blastintensity.BN.w	**condition_dum:BSCS.BN.w**	**233**	**0.51**	**3.39**	**.001**	**.04**
blastintensity.BN.w	condition_dum:BAQ.BN.w	233	0.05	0.37	.708	.00
blastintensity.BN.w	condition_dum:SOPT.BN.w	233	-0.10	-0.76	.451	.00
blastintensity.BN.w	condition_dum:IAT.BN.w	233	-0.16	-1.26	.210	.01

Interpretation: There is one significant interaction: condition by trait self-control (brief self-control scale, BSCS). However, there are marginally significant interactions, with BAQ and IAT (as expected), but not with SOPT.

Interpretation: Again, there seems to be an interaction between condition and self-control. However, there are also effects of the IAT, of SOPT, and of condition. [Note: it seems that after standardizing and winsorizing, the effects of condition and IAT go from significant to marginally significant only.]

Blast Duration

```
big.mod2 %>%
  nice_lm() %>%
  nice_table(highlight = TRUE)
```

Dependent Variable	Predictor	*df*	*b*	*t*	*p*	sr^2
blastduration.BN.w	condition_dum	233	0.13	1.11	.270	.00
blastduration.BN.w	KIMS.BN.w	233	0.11	1.13	.260	.00
blastduration.BN.w	BSCS.BN.w	233	-0.14	-1.50	.134	.01
blastduration.BN.w	BAQ.BN.w	233	0.03	0.31	.754	.00
blastduration.BN.w	**SOPT.BN.w**	**233**	**-0.32**	**-3.33**	**.001**	**.04**
blastduration.BN.w	**IAT.BN.w**	**233**	**0.22**	**2.62**	**.009**	**.02**
blastduration.BN.w	condition_dum:KIMS.BN.w	233	-0.18	-1.35	.179	.01
blastduration.BN.w	**condition_dum:BSCS.BN.w**	**233**	**0.52**	**3.52**	**.001**	**.04**
blastduration.BN.w	condition_dum:BAQ.BN.w	233	0.09	0.69	.489	.00
blastduration.BN.w	condition_dum:SOPT.BN.w	233	-0.07	-0.50	.618	.00
blastduration.BN.w	condition_dum:IAT.BN.w	233	-0.16	-1.31	.192	.01

Blast Intensity * Duration

```r
big.mod3 %>%
  nice_lm() %>%
  nice_table(highlight = TRUE)
```

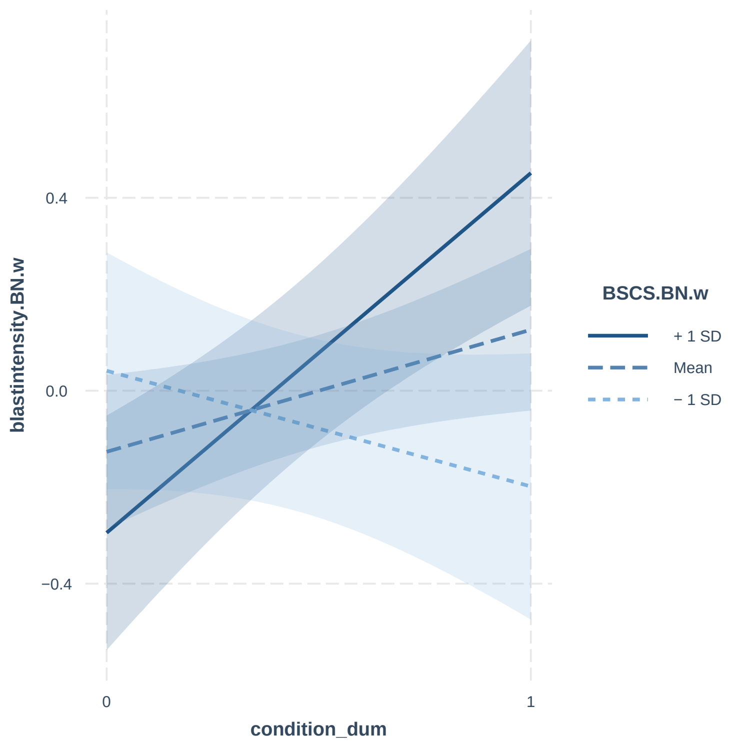
Dependent Variable	Predictor	*df*	*b*	*t*	*p*	sr^2
blastintensity.duration.BN.w	condition_dum	233	0.21	1.82	.070	.01
blastintensity.duration.BN.w	KIMS.BN.w	233	0.15	1.48	.141	.01
blastintensity.duration.BN.w	BSCS.BN.w	233	-0.17	-1.76	.080	.01
blastintensity.duration.BN.w	BAQ.BN.w	233	0.07	0.71	.475	.00
blastintensity.duration.BN.w	**SOPT.BN.w**	**233**	**-0.31**	**-3.11**	**.002**	**.03**
blastintensity.duration.BN.w	**IAT.BN.w**	**233**	**0.20**	**2.33**	**.021**	**.02**
blastintensity.duration.BN.w	condition_dum:KIMS.BN.w	233	-0.22	-1.63	.105	.01
blastintensity.duration.BN.w	**condition_dum:BSCS.BN.w**	**233**	**0.56**	**3.75**	**< .001**	**.05**
blastintensity.duration.BN.w	condition_dum:BAQ.BN.w	233	0.08	0.61	.541	.00
blastintensity.duration.BN.w	condition_dum:SOPT.BN.w	233	-0.08	-0.61	.540	.00
blastintensity.duration.BN.w	condition_dum:IAT.BN.w	233	-0.16	-1.33	.186	.01

Interaction plots

Let's plot the main significant interaction(s).

Blast Intensity

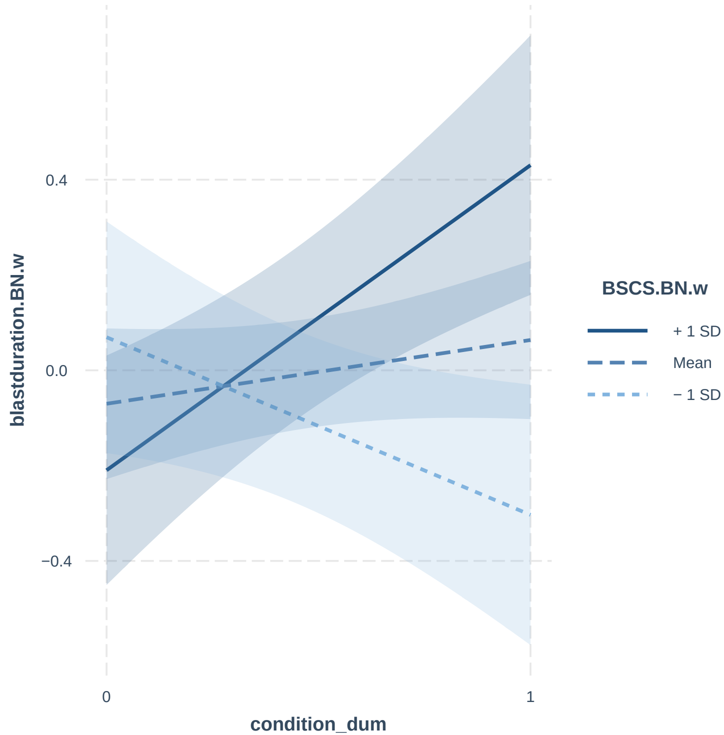
```r
# Plot
interact_plot(big.mod1, pred = "condition_dum", modx = "BSCS.BN.w",
              modxvals = NULL, interval = TRUE)
```



Interpretation: For people with low self-control, the priming mindfulness condition (cond = 1) seems to have little effect on blast intensity relative to the control condition (cond = 0). In contrast, for people with high self-control, the priming mindfulness condition relates to higher blast intensity.

Blast Duration

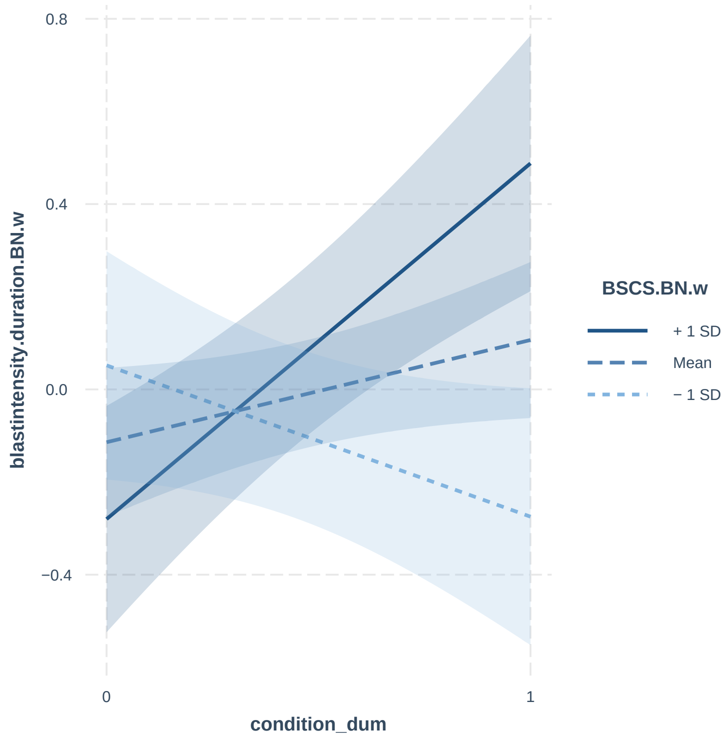
```
# Plot
interact_plot(big.mod2, pred = "condition_dum", modx = "BSCS.BN.w",
              modxvals = NULL, interval = TRUE)
```



Interpretation: For people with low self-control, the priming mindfulness condition (cond = 1) seems to have little effect on blast intensity relative to the control condition (cond = 0). In contrast, for people with high self-control, the priming mindfulness condition relates to higher blast intensity.

Blast Intensity * Duration

```
# Plot
interact_plot(big.mod3, pred = "condition_dum", modx = "BSCS.BN.w",
              modxvals = NULL, interval = TRUE)
```



Interpretation: For people with low self-control, the priming mindfulness condition (cond = 1) seems to have little effect on blast intensity relative to the control condition (cond = 0). In contrast, for people with high self-control, the priming mindfulness condition relates to higher blast intensity.

Simple slopes

Let's look at the simple slopes now (only for the significant interaction).

Blast Intensity

```
big.mod1 %>%
  nice_lm_slopes(predictor = "condition_dum",
                 moderator = "BSCS.BN.w") %>%
  nice_table(highlight = TRUE)
```

Dependent Variable	Predictor (+/-1 SD)	*df*	*b*	*t*	*p*	sr^2
blastintensity.BN.w	condition_dum (LOW-BSCS.BN.w)	233	-0.25	-1.30	.193	.01
blastintensity.BN.w	**condition_dum (MEAN-BSCS.BN.w)**	**233**	**0.25**	**2.10**	**.036**	**.02**
blastintensity.BN.w	**condition_dum (HIGH-BSCS.BN.w)**	**233**	**0.74**	**3.99**	**< .001**	**.06**

Interpretation: The effect of priming mindfulness on blast intensity is only significant for people with a high self-control.

Blast Duration

```r
big.mod2 %>%
  nice_lm_slopes(predictor = "condition_dum",
                 moderator = "BSCS.BN.w") %>%
  nice_table(highlight = TRUE)
```

Dependent Variable	Predictor (+/-1 SD)	*df*	*b*	*t*	*p*	*sr*^{2}
blastduration.BN.w	**condition_dum (LOW-BSCS.BN.w)**	**233**	**-0.38**	**-2.03**	**.044**	**.01**
blastduration.BN.w	condition_dum (MEAN-BSCS.BN.w)	233	0.13	1.11	.270	.00
blastduration.BN.w	**condition_dum (HIGH-BSCS.BN.w)**	**233**	**0.64**	**3.46**	**.001**	**.04**

Blast Intensity * Duration

```r
big.mod3 %>%
  nice_lm_slopes(predictor = "condition_dum",
                 moderator = "BSCS.BN.w") %>%
  nice_table(highlight = TRUE)
```

Dependent Variable	Predictor (+/-1 SD)	*df*	*b*	*t*	*p*	*sr*^{2}
blastintensity.duration.BN.w	condition_dum (LOW-BSCS.BN.w)	233	-0.33	-1.76	.080	.01
blastintensity.duration.BN.w	condition_dum (MEAN-BSCS.BN.w)	233	0.21	1.82	.070	.01
blastintensity.duration.BN.w	**condition_dum (HIGH-BSCS.BN.w)**	**233**	**0.76**	**4.09**	**< .001**	**.06**

Conclusions

Based on the results, it seems that the interaction came up for all three of blast itensity, duration, and the combination of the two. Therefore, perhaps it does make sense to use the combination in future studies.

Full Code

The full script of executive code contained in this document is reproduced here.

```r
# Set up the environment (or use local alternative `source("utils/config.R")`)
source("utils/config.R")

fast <- FALSE  # Make this true to skip the chunks
# This chunk is a bit complex so don't worry about it: it's made to add badges to the HTML versions
# NOTE: You have to replace the links accordingly to have working "buttons" on the HTML versions
if (!knitr::is_latex_output() && knitr::is_html_output()) {
  cat("[![Package](https://www.repostatus.org/badges/latest/active.svg)](https://github.com/rempsyc/remp
      [![Template Website](https://img.shields.io/badge/visit-website-E91E63)](https://realitybending.gi
      [![Website](https://img.shields.io/badge/download-.docx-FF5722)](https://remi-theriault.com/word_a
      [![Website](https://img.shields.io/badge/see-.pdf-FF9800)](https://remi-theriault.com/word_and_pdf
}
library(rempsyc)
library(dplyr)
library(interactions)
library(performance)
```

```r
library(see)
library(patchwork)
library(ggplot2)
library(rstatix)
library(forecast)
library(DescTools)
library(report)
library(bestNormalize)

summary(report::report(sessionInfo()))
df <- read.csv("data/data.csv")
data <- read.csv("data/fulldataset.csv")

cat(paste("The data consists of",
          report::report_participants(data),
          ". There are no demographics available."))

# Dummy-code group variable
data <- data %>%
  mutate(condition_dum = ifelse(condition == "Mindfulness", 1, 0),
         condition = as.factor(condition))

# Allocation ratio
cat(paste("The allocation ratio is: ",
          report::report(data$condition)))

# Make list of DVs
col.list <- c("blastintensity", "blastduration", "blastintensity.duration",
              "blastintensity.first", "blastduration.first",
              "blastintensity.duration.first", "KIMS", "BSCS", "BAQ",
              "SOPT", "IAT")

# Create new variable blastintensity.duration
data$blastintensity.duration <- (data$blastintensity * data$blastduration)
data$blastintensity.duration.first <- (data$blastintensity.first *
                                         data$blastduration.first)
# Divide by 2? Do some other sort of transformation given I multiplied two scores?
# Should I multiply them after standardization or before?

# Standardize and center main continuous IV variable (based on MAD)
# data <- data %>%
#   mutate(across(all_of(col.list),
#                 ~as.numeric(.x), #scale_mad(.x),
#                 .names = "{col}.mad"))
# We avoid standardizing now because it creates problems with the bestNormalize() function, and the lat

# Rename col.list with the MAD extension
# col.list <- paste0(col.list, ".mad")

report::cite_packages(sessionInfo())
# Group normality
sapply(col.list, function(x)
  nice_normality(data,
```

```
                 x,
                 "condition",
                 shapiro = TRUE,
                 title = x),
  USE.NAMES = TRUE,
  simplify = FALSE)

# <!-- Normally, the SOPT raw scores represent the number of errors, but I had multiplied it by -1 init
#
# <!-- We also add a constant of 1 to avoid scores of zero which can interfere with the transformation.

# Not necessary anymore since we use the `bestNormalize` package.

predict_bestNormalize <- function(var, print.transform = TRUE) {
  x <- bestNormalize(var, standardize = TRUE)
  if (print.transform == TRUE) {
    print(cur_column())
    print(x$chosen_transform)
  }
  predict(x)
}

set.seed(100)
data <- data %>%
  mutate(across(all_of(col.list),
                predict_bestNormalize,
                .names = "{.col}.BN"))
col.list <- paste0(col.list, ".BN")
# Group normality
sapply(col.list, function(x)
  nice_normality(data,
                 x,
                 "condition",
                 shapiro = TRUE,
                 title = x),
  USE.NAMES = TRUE,
  simplify = FALSE)
# Plotting variance
plots(lapply(col.list, function(x) {
  nice_varplot(data, x, group = "condition")
  }),
  n_columns = 3)


# Using boxplots
plots(lapply(col.list, function(x) {
  ggplot(data, aes(condition, !!sym(x))) +
  geom_boxplot()
  }),
  n_columns = 3)


find_mad(data, col.list, criteria = 3)
# 6 people after our transformations
```

```r
# Winsorize variables of interest with MAD
data <- data %>%
  mutate(across(all_of(col.list),
                winsorize_mad,
                .names = "{.col}.w"))

# Update col.list
col.list <- paste0(col.list, ".w")


nice_t_test(data,
            response = col.list,
            group = "condition") %>%
  nice_table(highlight = 0.10)

nice_violin(data,
            group = "condition",
            response = "blastintensity.BN.w",
            comp1 = 1,
            comp2 = 2,
            obs = TRUE)
nice_violin(data,
            group = "condition",
            response = "blastduration.BN.w",
            comp1 = 1,
            comp2 = 2,
            obs = TRUE)
nice_violin(data,
            group = "condition",
            response = "blastintensity.duration.BN.w",
            comp1 = 1,
            comp2 = 2,
            obs = TRUE)
data %>%
    group_by(condition) %>%
    summarize(M = mean(blastintensity),
              SD = sd(blastintensity),
              N = n()) %>%
  nice_table(width = 0.40)

data %>%
    group_by(condition) %>%
    summarize(M = mean(blastduration),
              SD = sd(blastduration),
              N = n()) %>%
  nice_table(width = 0.40)
data %>%
    group_by(condition) %>%
    summarize(M = mean(blastintensity.duration),
              SD = sd(blastintensity.duration),
              N = n()) %>%
  nice_table(width = 0.40)
```

```r
big.mod1 <- lm(blastintensity.BN.w ~ condition_dum*KIMS.BN.w +
                 condition_dum*BSCS.BN.w + condition_dum*BAQ.BN.w +
                 condition_dum*SOPT.BN.w + condition_dum*IAT.BN.w,
               data = data, na.action="na.exclude")
check_model(big.mod1)


big.mod2 <- lm(blastduration.BN.w ~ condition_dum*KIMS.BN.w +
                 condition_dum*BSCS.BN.w + condition_dum*BAQ.BN.w +
                 condition_dum*SOPT.BN.w + condition_dum*IAT.BN.w,
               data = data, na.action="na.exclude")
check_model(big.mod2)


big.mod3 <- lm(blastintensity.duration.BN.w ~ condition_dum*KIMS.BN.w +
                 condition_dum*BSCS.BN.w + condition_dum*BAQ.BN.w +
                 condition_dum*SOPT.BN.w + condition_dum*IAT.BN.w,
               data = data, na.action="na.exclude")
check_model(big.mod3)

big.mod1 %>%
  nice_lm() %>%
  nice_table(highlight = TRUE)

big.mod2 %>%
  nice_lm() %>%
  nice_table(highlight = TRUE)

big.mod3 %>%
  nice_lm() %>%
  nice_table(highlight = TRUE)

# Plot
interact_plot(big.mod1, pred = "condition_dum", modx = "BSCS.BN.w",
              modxvals = NULL, interval = TRUE)

# Plot
interact_plot(big.mod2, pred = "condition_dum", modx = "BSCS.BN.w",
              modxvals = NULL, interval = TRUE)

# Plot
interact_plot(big.mod3, pred = "condition_dum", modx = "BSCS.BN.w",
              modxvals = NULL, interval = TRUE)


big.mod1 %>%
  nice_lm_slopes(predictor = "condition_dum",
                 moderator = "BSCS.BN.w") %>%
  nice_table(highlight = TRUE)


big.mod2 %>%
  nice_lm_slopes(predictor = "condition_dum",
```

```
                  moderator = "BSCS.BN.w") %>%
  nice_table(highlight = TRUE)


big.mod3 %>%
  nice_lm_slopes(predictor = "condition_dum",
                  moderator = "BSCS.BN.w") %>%
  nice_table(highlight = TRUE)
```

Package References

```
report::cite_packages(sessionInfo())
```


- (1) Peterson, R. A. (2021). Finding Optimal Normalizing Transformations via bestNormalize. The R Journal, 13:1, 310-329, DOI:10.32614/RJ-2021-041
- Andri Signorell et mult. al. (2021). DescTools: Tools for descriptive statistics. R package version 0.99.44.
- Ben-Shachar M, Lüdecke D, Makowski D (2020). effectsize: Estimation of Effect Size Indices and Standardized Parameters. Journal of Open Source Software, 5(56), 2815. doi: 10.21105/joss.02815
- H. Wickham. ggplot2: Elegant Graphics for Data Analysis. Springer-Verlag New York, 2016.
- Hyndman R, Athanasopoulos G, Bergmeir C, Caceres G, Chhay L, O'Hara-Wild M, Petropoulos F, Razbash S, Wang E, Yasmeen F (2022). *forecast: Forecasting functions for time series and linear models*. R package version 8.16, https://pkg.robjhyndman.com/forecast/.
- JJ Allaire and Yihui Xie and Jonathan McPherson and Javier Luraschi and Kevin Ushey and Aron Atkins and Hadley Wickham and Joe Cheng and Winston Chang and Richard Iannone (2022). rmarkdown: Dynamic Documents for R. R package version 2.13. URL https://rmarkdown.rstudio.com.
- Long JA (2019). *interactions: Comprehensive, User-Friendly Toolkit for Probing Interactions*. R package version 1.1.0, https://cran.r-project.org/package=interactions.
- Lüdecke D, Ben-Shachar M, Patil I, Makowski D (2020). "Extracting, Computing and Exploring the Parameters of Statistical Models using R." *Journal of Open Source Software*, *5*(53), 2445. doi: 10.21105/joss.02445 https://doi.org/10.21105/joss.02445.
- Lüdecke D, Waggoner P, Makowski D (2019). "insight: A Unified Interface to Access Information from Model Objects in R." *Journal of Open Source Software*, *4*(38), 1412. doi:10.21105/joss.01412 https://doi.org/10.21105/joss.01412.
- Lüdecke et al., (2021). performance: An R Package for Assessment, Comparison and Testing of Statistical Models. Journal of Open Source Software, 6(60), 3139. https://doi.org/10.21105/joss.03139
- Lüdecke et al., (2021). see: An R Package for Visualizing Statistical Models. Journal of Open Source Software, 6(64), 3393. https://doi.org/10.21105/joss.03393
- Makowski, D., Ben-Shachar, M. S. & Lüdecke, D. (2020). *The {easystats} collection of R packages*. GitHub.
- Makowski, D., Ben-Shachar, M. S., Patil, I., & Lüdecke, D. (2019). Methods and Algorithms for Correlation Analysis in R. Journal of Open Source Software, 5(51), 2306. doi:10.21105/joss.02306
- Makowski, D., Ben-Shachar, M. S., Patil, I., & Lüdecke, D. (2020). Estimation of Model-Based Predictions, Contrasts and Means. CRAN.

- Makowski, D., Ben-Shachar, M., & Lüdecke, D. (2019). bayestestR: Describing Effects and their Uncertainty, Existence and Significance within the Bayesian Framework. Journal of Open Source Software, 4(40), 1541. doi:10.21105/joss.01541
- Makowski, D., Ben-Shachar, M.S., Patil, I. & Lüdecke, D. (2020). Automated Results Reporting as a Practical Tool to Improve Reproducibility and Methodological Best Practices Adoption. CRAN. Available from https://github.com/easystats/report. doi: .
- Makowski, Lüdecke, Patil, Ben-Shachar, & Wiernik (2021). datawizard: Easy Data Wrangling. CRAN. Available from https://easystats.github.io/datawizard/
- R Core Team (2022). R: A language and environment for statistical computing. R Foundation for Statistical Computing, Vienna, Austria. URL https://www.R-project.org/.
- Wickham et al., (2019). Welcome to the tidyverse. Journal of Open Source Software, 4(43), 1686, https://doi.org/10.21105/joss.01686
- NA
- NA
- NA
- NA
- NA
- NA
- NA
- NA
- NA
- NA

References