Analysis

Stefan P. Thoma

3/10/2020

Setup

Install / load packages needed:

```
knitr::opts_chunk$set(echo = TRUE)
if (!require("pacman")) install.packages("pacman")
```

```
## Loading required package: pacman
```

```
p_load(tidyverse, lme4, lmerTest, mvoutlier, nlme, multcomp, lsmeans, xtable, jtools, tikzDevice, gmodel
#pacman::p_load_gh("jaredhuling/jcolors")
#jcolors::jcolors("default")
#ggplot <- function(...) ggplot2::ggplot(...) + scale_color_brewer(palette=jcolors::jcolors()) + scale_
```

```
sessionInfo()
```

```
## R version 4.1.2 (2021-11-01)
## Platform: x86_64-apple-darwin17.0 (64-bit)
## Running under: macOS Big Sur 10.16
##
## Matrix products: default
## BLAS:   /Library/Frameworks/R.framework/Versions/4.1/Resources/lib/libRblas.0.dylib
## LAPACK: /Library/Frameworks/R.framework/Versions/4.1/Resources/lib/libRlapack.dylib
##
## locale:
## [1] en_US.UTF-8/en_US.UTF-8/en_US.UTF-8/C/en_US.UTF-8/en_US.UTF-8
##
## attached base packages:
## [1] parallel  stats     graphics  grDevices utils     datasets  methods
## [8] base
##
## other attached packages:
##  [1] tidylog_1.0.2       optimx_2021-10.12   performance_0.8.0
##  [4] gmodels_2.18.1      tikzDevice_0.12.3.1 jtools_2.1.4
##  [7] xtable_1.8-4        lsmeans_2.30-0      emmeans_1.7.1-1
## [10] multcomp_1.4-17     TH.data_1.1-0       MASS_7.3-54
## [13] survival_3.2-13     mvtnorm_1.1-3       nlme_3.1-153
## [16] mvoutlier_2.1.1     sgeostat_1.0-27     lmerTest_3.1-3
## [19] lme4_1.1-27.1       Matrix_1.3-4        forcats_0.5.1
## [22] stringr_1.4.0       dplyr_1.0.7         purrr_0.3.4
## [25] readr_2.0.2         tidyr_1.1.4         tibble_3.1.6
## [28] ggplot2_3.3.5       tidyverse_1.3.1     pacman_0.5.1
##
## loaded via a namespace (and not attached):
```

```
##  [1] fs_1.5.0            lubridate_1.8.0     insight_0.14.5
##  [4] httr_1.4.2          numDeriv_2016.8-1.1 tools_4.1.2
##  [7] backports_1.4.0     utf8_1.2.2          R6_2.5.1
## [10] DBI_1.1.1           colorspace_2.0-2    withr_2.4.3
## [13] tidyselect_1.1.1    compiler_4.1.2      cli_3.1.0
## [16] rvest_1.0.1         xml2_1.3.2          sandwich_3.0-1
## [19] scales_1.1.1        DEoptimR_1.0-9      robustbase_0.93-9
## [22] digest_0.6.29       minqa_1.2.4         rmarkdown_2.11
## [25] pkgconfig_2.0.3     htmltools_0.5.2     dbplyr_2.1.1
## [28] fastmap_1.1.0       rlang_0.4.12        readxl_1.3.1
## [31] rstudioapi_0.13     generics_0.1.1      zoo_1.8-9
## [34] jsonlite_1.7.2      gtools_3.9.2        magrittr_2.0.1
## [37] Rcpp_1.0.7          munsell_0.5.0       fansi_1.0.2
## [40] lifecycle_1.0.1     stringi_1.7.6       yaml_2.2.1
## [43] grid_4.1.2          gdata_2.18.0        crayon_1.4.2
## [46] lattice_0.20-45     haven_2.4.3         splines_4.1.2
## [49] pander_0.6.4        hms_1.1.1           knitr_1.37
## [52] pillar_1.6.4        boot_1.3-28         estimability_1.3
## [55] clisymbols_1.2.0    codetools_0.2-18    reprex_2.0.1
## [58] glue_1.6.0          evaluate_0.14       modelr_0.1.8
## [61] vctrs_0.3.8         nloptr_1.2.2.3      tzdb_0.1.2
## [64] cellranger_1.1.0    gtable_0.3.0        assertthat_0.2.1
## [67] xfun_0.29           broom_0.7.10        coda_0.19-4
## [70] filehash_2.4-2      ellipsis_0.3.2
```

Load Data

```
data <- read_csv("data/cleanData.csv")
```

```
## Rows: 284 Columns: 131
```

```
## -- Column specification --------------------------------------------------------
## Delimiter: ","
## chr    (9): id, sex, condition, Frage1, Frage2, Frage3, Leiter, Anmerkungen,...
## dbl  (121): time, iat, ccs1, ccs2, ccs3, ccs4, ccs5, ccs6, ccs7, ccs8, ccs9,...
## dttm   (1): StartDate
```

```
##
## i Use `spec()` to retrieve the full column specification for this data.
## i Specify the column types or set `show_col_types = FALSE` to quiet this message.
```

```
#data <- data %>% dplyr::select(
#  id, time, iat, ccs, nr, nep, ipq, sod, ses, age, edu, sex, pol, vr_exp, vr_eval1, vr_eval2, vr_eval3
#  vr_eval4, vr_eval5, span, seen, condition, starts_with("Frage"), hr_mean, Leiter, Anmerkungen, Zeit
#)

head(data)
```

```
## # A tibble: 6 x 131
##   id        time    iat StartDate  ccs1  ccs2  ccs3  ccs4  ccs5  ccs6
##   <chr>    <dbl>  <dbl> <dttm>    <dbl> <dbl> <dbl> <dbl> <dbl> <dbl>
## 1 28287312     1  0.179 NA            1     1     2     1     1     1
## 2 28287312     2  0.409 NA            1     1     1     1     1     1
## 3 77995467     1 -0.496 NA            1     1     1     1     1     1
## 4 77995467     2 -0.362 NA            1     1     1     1     1     1
## 5 43795961     1  0.517 NA            2     1     1     1     2     1
```

```
## 6 43795961     2  0.634 NA            2     1     1     1     2     1
## # ... with 121 more variables: ccs7 <dbl>, ccs8 <dbl>, ccs9 <dbl>, ccs10 <dbl>,
## #   ccs11 <dbl>, ccs12 <dbl>, nr1 <dbl>, nr2 <dbl>, nr3 <dbl>, nr4 <dbl>,
## #   nr5 <dbl>, nr6 <dbl>, nr7 <dbl>, nr8 <dbl>, nr9 <dbl>, nr10 <dbl>,
## #   nr11 <dbl>, nr12 <dbl>, nr13 <dbl>, nr14 <dbl>, nr15 <dbl>, nr16 <dbl>,
## #   nr17 <dbl>, nr18 <dbl>, nr19 <dbl>, nr20 <dbl>, nr21 <dbl>, nep1 <dbl>,
## #   nep2 <dbl>, nep3 <dbl>, nep4 <dbl>, nep5 <dbl>, nep6 <dbl>, nep7 <dbl>,
## #   nep8 <dbl>, nep9 <dbl>, nep10 <dbl>, nep11 <dbl>, nep12 <dbl>, ...
```

```
# factor for vr or not
data <- data %>% group_by(id) %>%
  mutate(
    vr = ifelse(condition %in% c("a", "b", "c"), TRUE, FALSE),
    type = factor(ifelse(vr, "vr", "control")),
    condition = factor(condition, levels = c("b", "a", "c", "video", "text.bild", "text"))
  )
```

```
## group_by: one grouping variable (id)

## mutate (grouped): converted 'condition' from character to factor (0 new NA)

##                   new variable 'vr' (logical) with 2 unique values and 0% NA

##                   new variable 'type' (factor) with 2 unique values and 0% NA
```

Keep in mind the conditions coding:

a == abstract

b == realistic

c == realistic but badly so

Descriptives

Reliability

```
vars <- names(data)

ccs.vars <- vars[startsWith(vars, "ccs")]
nr.vars <- vars[startsWith(vars, "nr")]
nep.vars <- vars[startsWith(vars, "nep")]
ipq.vars <- vars[startsWith(vars, "ipq")]
sod.vars <- vars[startsWith(vars, "sod")]

vars.list1 <- list(ccs.vars, nr.vars, nep.vars)
vars.list2 <- list(ipq.vars, sod.vars)

#remove overall score (shortest name)
# this is a bit more robust compared to simply removing the last item
remove_overall <- function(char.vec){
 nm <- char.vec[which.min(nchar(char.vec))]
 char.vec <- char.vec[-which.min(nchar(char.vec))]
 char.vec
}

vars.list1 <- lapply(vars.list1, remove_overall)
```

```r
vars.list2 <- lapply(vars.list2, remove_overall)



reliable <- function(data, vars){
  alph <- psych::alpha(data[vars], title = vars[1])

  #omeg <- psych::omega(data[vars], plot = FALSE)
 df <- data.frame(alpha = alph$total$raw_alpha, ci.low = alph$total$raw_alpha - 1.96 * alph$total$ase, c
 df <- round(df, 3)
 df$var = vars[1]
 df
}


# measures which are measured twice
alpha.1 <- lapply(vars.list1, function(x) reliable(data = data, x))
# measures which are measured only once (sod and ipq)
alpha.2 <- lapply(vars.list2, function(x) reliable(data = data %>% filter(time == 1), x))
```

```
## filter (grouped): removed 142 rows (50%), 142 rows remaining
## filter (grouped): removed 142 rows (50%), 142 rows remaining
```

```r
alphas <- c(alpha.1, alpha.2)

(alpha.df <- do.call("rbind", alphas) %>%
  dplyr::select(var, alpha, ci.low, ci.up))
```

```
##     var alpha ci.low ci.up
## 1  ccs1 0.845  0.819 0.871
## 2   nr1 0.840  0.813 0.866
## 3  nep1 0.686  0.632 0.739
## 4  ipq1 0.826  0.785 0.867
## 5 sod_1 0.805  0.759 0.852
```

cronbach alpha of nep is relatively small. What would mcdonalds omega look like for nep?

```r
psych::omega(data[vars.list1[[3]]], nfactors = 1)
```

```
## Loading required namespace: GPArotation

## Omega_h for 1 factor is not meaningful, just omega_t

## Warning in schmid(m, nfactors, fm, digits, rotate = rotate, n.obs = n.obs, :
## Omega_h and Omega_asymptotic are not meaningful with one factor

## Omega
## Call: omegah(m = m, nfactors = nfactors, fm = fm, key = key, flip = flip,
##     digits = digits, title = title, sl = sl, labels = labels,
##     plot = plot, n.obs = n.obs, rotate = rotate, Phi = Phi, option = option,
##     covar = covar)
## Alpha:                 0.7
## G.6:                   0.73
## Omega Hierarchical:    0.7
## Omega H asymptotic:    0.99
## Omega Total            0.71
##
```

```
## Schmid Leiman Factor loadings greater than  0.2
##          g  F1*   h2   u2 p2
## nep1  0.31      0.10 0.90  1
## nep2  0.38      0.15 0.85  1
## nep3  0.59      0.35 0.65  1
## nep4  0.44      0.20 0.80  1
## nep5  0.55      0.30 0.70  1
## nep6            0.03 0.97  1
## nep7  0.41      0.17 0.83  1
## nep8  0.21      0.04 0.96  1
## nep9            0.01 0.99  1
## nep10 0.46      0.21 0.79  1
## nep11 0.27      0.07 0.93  1
## nep12 0.48      0.23 0.77  1
## nep13 0.29      0.08 0.92  1
## nep14 0.32      0.10 0.90  1
## nep15 0.56      0.31 0.69  1
##
## With eigenvalues of:
##   g F1*
## 2.4 0.0
##
## general/max  1.878568e+16   max/min =   1
## mean percent general =  1    with sd =  0 and cv of  0
## Explained Common Variance of the general factor =  1
##
## The degrees of freedom are 90  and the fit is  0.94
## The number of observations was  284  with Chi Square =  261.23  with prob <  1.3e-18
## The root mean square of the residuals is  0.08
## The df corrected root mean square of the residuals is  0.09
## RMSEA index =  0.082  and the 10 % confidence intervals are  0.071 0.094
## BIC =  -247.18
##
## Compare this with the adequacy of just a general factor and no group factors
## The degrees of freedom for just the general factor are 90  and the fit is  0.94
## The number of observations was  284  with Chi Square =  261.23  with prob <  1.3e-18
## The root mean square of the residuals is  0.08
## The df corrected root mean square of the residuals is  0.09
##
## RMSEA index =  0.082  and the 10 % confidence intervals are  0.071 0.094
## BIC =  -247.18
##
## Measures of factor score adequacy
##                                                  g F1*
## Correlation of scores with factors            0.87   0
## Multiple R square of scores with factors      0.75   0
## Minimum correlation of factor score estimates 0.51  -1
##
##  Total, General and Subset omega for each subset
##                                                  g F1*
## Omega total for total scores and subscales    0.71 0.7
## Omega general for total scores and subscales  0.70 0.7
## Omega group for total scores and subscales    0.00 0.0
```

Omega Total 0.71 seems ok.

Check NEP structure

```
psych::principal(r = data[nep.vars[-length(nep.vars)]])
```

```
## Principal Components Analysis
## Call: psych::principal(r = data[nep.vars[-length(nep.vars)]])
## Standardized loadings (pattern matrix) based upon correlation matrix
##        PC1    h2   u2 com
## nep1  0.38 0.146 0.85   1
## nep2  0.46 0.209 0.79   1
## nep3  0.65 0.418 0.58   1
## nep4  0.52 0.273 0.73   1
## nep5  0.60 0.365 0.63   1
## nep6  0.21 0.045 0.95   1
## nep7  0.48 0.230 0.77   1
## nep8  0.27 0.071 0.93   1
## nep9  0.11 0.012 0.99   1
## nep10 0.52 0.275 0.72   1
## nep11 0.34 0.114 0.89   1
## nep12 0.55 0.297 0.70   1
## nep13 0.36 0.128 0.87   1
## nep14 0.39 0.155 0.84   1
## nep15 0.62 0.384 0.62   1
##
##                  PC1
## SS loadings     3.13
## Proportion Var 0.21
##
## Mean item complexity =  1
## Test of the hypothesis that 1 component is sufficient.
##
## The root mean square of the residuals (RMSR) is  0.1
##  with the empirical chi square  583.99  with prob <  2e-73
##
## Fit based upon off diagonal values = 0.68
```

For the following analysis we reduce the dataframe.

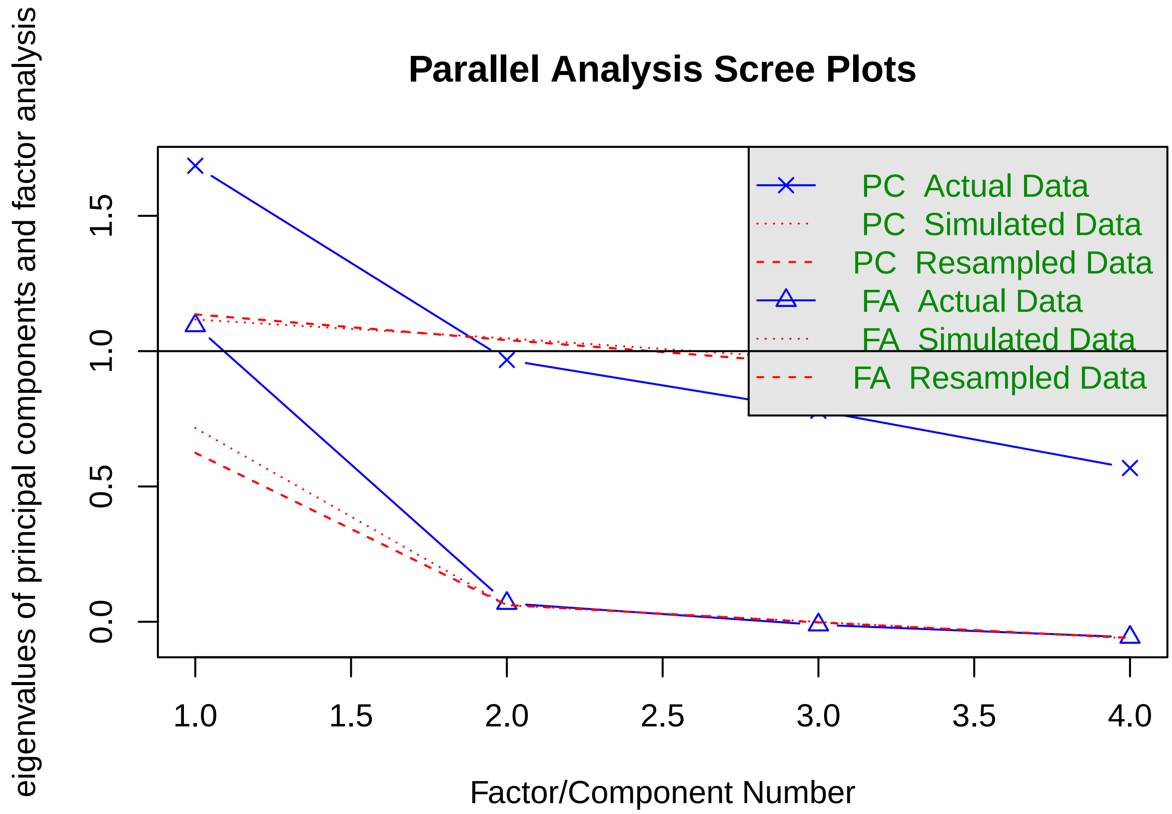
```
data <- data %>% dplyr::select(
  id, time, vr, type, condition, iat, ccs, nr, nep, ipq, sod, ses, age, edu, sex, pol, vr_exp, vr_eval1
)
```

Principal component analysis

We try to find an acceptable model for each DV.

First, I would like to calculate a principal component of all dependent variables (dvs)

```
df_env <- data[c("iat", "ccs", "nr", "nep")]
psych::fa.parallel(df_env)
```




```
## Parallel analysis suggests that the number of factors =  1  and the number of components =  1
```

```
prc_env <- princomp(df_env, cor = TRUE)
# Seems like one factor might just be enough. However, this may be more revealing when done
# on raw data
summary(prc_env)
```

```
## Importance of components:
##                           Comp.1    Comp.2    Comp.3    Comp.4
## Standard deviation     1.2980576 0.9835519 0.8829992 0.7536475
## Proportion of Variance 0.4212384 0.2418436 0.1949219 0.1419962
## Cumulative Proportion  0.4212384 0.6630819 0.8580038 1.0000000
```

```
data$env_pc <- prc_env$scores[,1]
```

Unidimensionality could be assumed. The scores of the first principal component were stored in `data$env_pc`. This vector can now be used as a dependent variable in further exploratory analyses.

This plot is not very useful I guess. Too crowded.

Check Intervention

This section is concerned only with the VR conditions a, b, c. Specifically with the variables vr_eval 1:5 and with the sod and presence scale IPQ.

Check for outliers on these scales:

```
#check.data <- data %>% dplyr::filter((vr == T & time == 1 ) & !is.na(ipq))
check.data <- data %>% dplyr::filter((time == 1 ) & !is.na(ipq))
```

```
#outlier.data <- data %>% ungroup() %>% dplyr::filter(vr == T & time == 1) %>% dplyr::select(starts_wit
# drop_na()
#mvoutlier::chisq.plot(check.data %>% ungroup() %>% dplyr::select(starts_with("vr_eval"), sod, ipq))
# remove: 38  1 63
# which corresponds to the ids:
#remove.ids <- check.data$id[c(38,  1, 63)]
# "44466757" "32504483" "80688810"
```

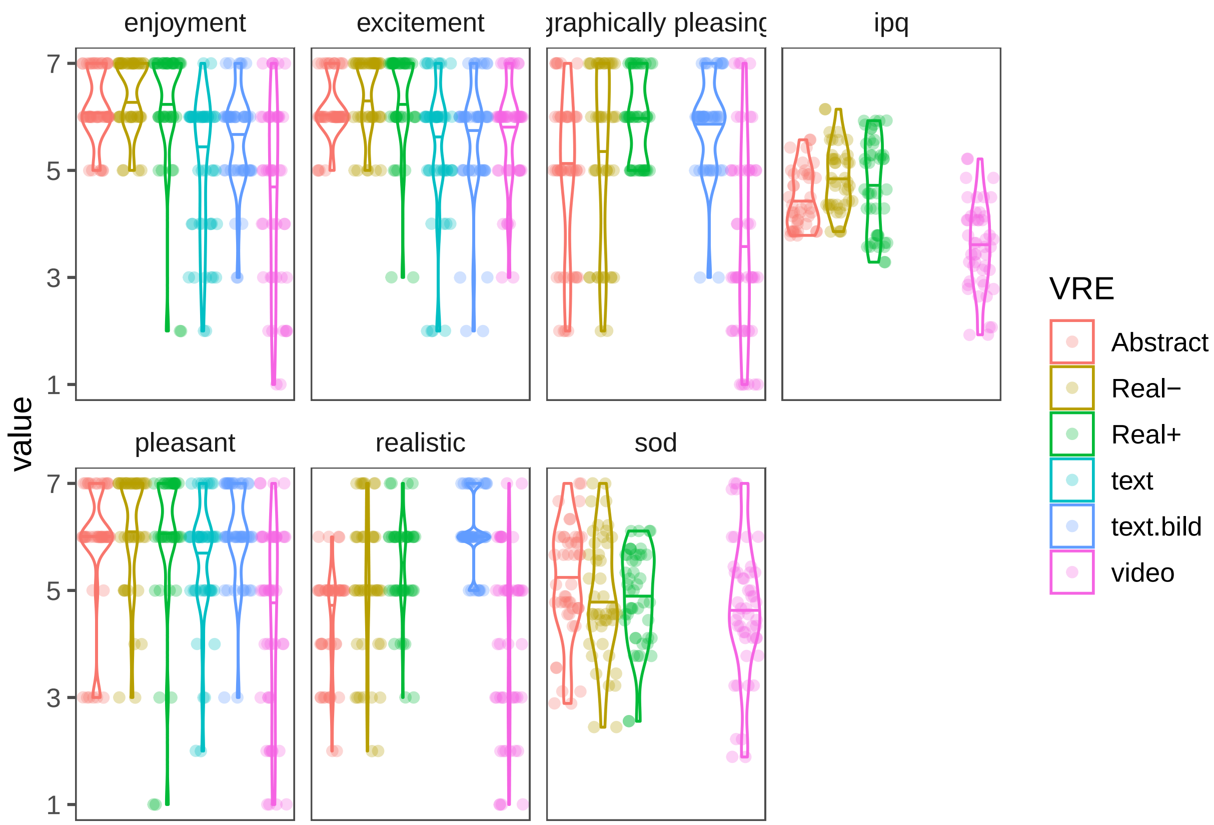
Plot

```
vars <-c("vr_eval1", "vr_eval2", "vr_eval3", "vr_eval4", "vr_eval5", "ipq", "sod")

desc_plot_data <- gather(data, specific, value, vars) %>%
 # filter(!is.na(ipq)) %>%
  arrange(id, specific) %>%
  mutate(specific = ifelse(specific=="vr_eval1", "excitement",
                      ifelse(specific=="vr_eval2", "graphically pleasing",
                             ifelse(specific=="vr_eval3", "pleasant",
                                    ifelse(specific=="vr_eval4", "realistic",
                                           ifelse(specific=="vr_eval5", "enjoyment", specific))))),
         VRE = ifelse(condition == "b", "Real+",
                    ifelse(condition=="c", "Real-",
                           ifelse(condition=="a", "Abstract", as.character(condition))))) %>%
  dplyr::select(specific, value, id, VRE)
```

```
## Note: Using an external vector in selections is ambiguous.
## i Use `all_of(vars)` instead of `vars` to silence this message.
## i See <https://tidyselect.r-lib.org/reference/faq-external-vector.html>.
## This message is displayed once per session.

## gather: reorganized (ipq, sod, vr_eval1, vr_eval2, vr_eval3, ...) into (specific, value) [was 284x32

## mutate: changed 1,420 values (71%) of 'specific' (0 new NA)

##         new variable 'VRE' (character) with 6 unique values and 0% NA
```

```
(vr_eval_plot <- ggplot(data = desc_plot_data, aes(x = VRE, y = value, color = VRE)) +
  facet_wrap( ~specific, nrow = 2)+
  geom_violin(draw_quantiles = .5) + #, position = position_jitterdodge(dodge.width = 0.8, jitter.width
  geom_point(alpha = .3, position = "jitter")+#, position = position_jitterdodge(dodge.width = 0, jitte
  ggthemes::theme_few() +
  ylab("value") +
  xlab("")+
 # labs(title="")+
  theme(axis.title.x=element_blank(),
        axis.text.x=element_blank(),
        axis.ticks.x=element_blank())+

  scale_y_continuous(breaks = c(1,3,5,7)) +
  theme(legend.position = "right") )
```

```
## Warning: Removed 304 rows containing non-finite values (stat_ydensity).

## Warning: Removed 304 rows containing missing values (geom_point).
```



ipq and sod separately:

HLM

So we will create two models for each dependent variables:

First, the model will have the formula:

`dv ~ condition * time + (time | id)`

This will be simplified to the following model if model fit is singular:

`dv ~ condition * time + (1 | id)`

This will estimate a random intercept for each participant. This model will only take as input the vr conditions (`a`, `b` & `c`), or: `vr == TRUE`.

The second model will have the formula:

`dv ~ vr * time + (time | condition) + (1 | id)`

Where a random slope for time is estimated per condition. Further, there is a random intercept per condition, and per id.

Should model fit be singular, we would simplify the model to:

`dv ~ vr * time + (1 | condition) + (1 | id)`

If still singular, we would simplify to:

`dv ~ vr * time + (1 | id)`

Helping function

```
fit.lme <- function(form, dat){
  lme4::lmer(formula = form, data = dat)
  #lme4::lmer(formula = form, data = dat,
  #           control = lmerControl(optimizer = "optimx", optCtrl = list(method = "nlminb", starttests =
}
```

```
fit_models <- function(dv, dat){
# this function returns a function which fits a model based on a formula minus the predictors.
# This function can be used in the next function which implements the conditions for reducing model com

  function(predictors){
    form <- formula(paste(dv, predictors, sep = " ~ "))
    print(form)
    fit <- fit.lme(form = form, dat = dat)
  }
}
```

```
predictors.vr <-  c("condition * time + (time | id)", "condition * time + (1 | id)")
predictors.all <- c("time*type + (time | condition) + (1 | id)", "time * type + (-1 + time | condition)
#predictors.all2 <- c("vr * time + (time | condition) + (1 | id)", "vr * time + (1 | condition) + (1 |
#predictors.all3 <- c("vr * time + (time | condition) + (1 | id)", "vr * time + (-1 + time | condition)
# function to fit various models based on different inputs of predictors
fit_many <- function(pred.vector, dat, dv){
  fit_model <- fit_models(dv, dat)

  sing <- TRUE
  i <- 1
  while((sing) & i<=length(pred.vector)){
    model <- try(fit_model(pred.vector[i]))

    if(class(model)!="try-error"){
      sing <- isSingular(model)
    }

    i <- i + 1
  }
  print(paste("is model singular: ", sing))
  model
}
```

Vector containing name of all dv's

```
dvs <-  c("iat", "ccs", "nr", "nep", "env_pc")
```

```
# split data frame:
data.vr <- data %>% dplyr::filter(vr)
```

Contrast: We want the effect of "time" to be the average effect over all conditions. Therefore we set the contrast of the condition variable to `contr.sum` in accordance with https://stats.oarc.ucla.edu/r/library/r-library-contrast-coding-systems-for-categorical-variables/#DEVIATION. This is sometimes called `unweighted effect coding` or `deviation coding`.

```
data.vr$condition <-droplevels(data.vr$condition)
contrasts(data.vr$condition) <- contr.sum(3)
```

```
contrasts(data$type) <- contr.sum(2)
```

Initial model fitting

```
vr.models <- lapply(dvs, FUN = function(dv) fit_many(pred.vector = predictors.vr, dat = data.vr, dv = dv
```

```
## iat ~ condition * time + (time | id)
## <environment: 0x7fe507a06278>
## Error : number of observations (=138) <= number of random effects (=138) for term (time | id); the ra
## iat ~ condition * time + (1 | id)
## <environment: 0x7fe510b92048>
## [1] "is model singular:  FALSE"
## ccs ~ condition * time + (time | id)
## <environment: 0x7fe5039991c8>
## Error : number of observations (=138) <= number of random effects (=138) for term (time | id); the ra
## ccs ~ condition * time + (1 | id)
## <environment: 0x7fe50649a798>
## [1] "is model singular:  FALSE"
## nr ~ condition * time + (time | id)
## <environment: 0x7fe503c98038>
## Error : number of observations (=138) <= number of random effects (=138) for term (time | id); the ra
## nr ~ condition * time + (1 | id)
## <environment: 0x7fe504cb1e08>
## [1] "is model singular:  FALSE"
## nep ~ condition * time + (time | id)
## <environment: 0x7fe5116bfeb0>
## Error : number of observations (=138) <= number of random effects (=138) for term (time | id); the ra
## nep ~ condition * time + (1 | id)
## <environment: 0x7fe5114388b0>
## [1] "is model singular:  FALSE"
## env_pc ~ condition * time + (time | id)
## <environment: 0x7fe500897938>
## Error : number of observations (=138) <= number of random effects (=138) for term (time | id); the ra
## env_pc ~ condition * time + (1 | id)
## <environment: 0x7fe500ac04e0>
## [1] "is model singular:  FALSE"
```

```
all.models  <-  lapply(dvs, FUN = function(dv) fit_many(pred.vector = predictors.all, dat = data, dv = d
```

```
## iat ~ time * type + (time | condition) + (1 | id)
## <environment: 0x7fe5112c1038>
```

```
## boundary (singular) fit: see ?isSingular
```

```
## iat ~ time * type + (-1 + time | condition) + (1 | id)
## <environment: 0x7fe500d87348>
## [1] "is model singular:  FALSE"
## ccs ~ time * type + (time | condition) + (1 | id)
## <environment: 0x7fe4e59a4920>
```

```
## boundary (singular) fit: see ?isSingular
```

```
## ccs ~ time * type + (-1 + time | condition) + (1 | id)
## <environment: 0x7fe4e6aaec58>
```

```
## boundary (singular) fit: see ?isSingular
```

```
## ccs ~ time * type + (1 | id)
## <environment: 0x7fe4f71193c0>
## [1] "is model singular:  FALSE"
## nr ~ time * type + (time | condition) + (1 | id)
## <environment: 0x7fe5279f42b0>
## boundary (singular) fit: see ?isSingular
## nr ~ time * type + (-1 + time | condition) + (1 | id)
## <environment: 0x7fe526bad478>
## [1] "is model singular:  FALSE"
## nep ~ time * type + (time | condition) + (1 | id)
## <environment: 0x7fe521b46ca8>
## boundary (singular) fit: see ?isSingular
## nep ~ time * type + (-1 + time | condition) + (1 | id)
## <environment: 0x7fe517d5f6a8>
## [1] "is model singular:  FALSE"
## env_pc ~ time * type + (time | condition) + (1 | id)
## <environment: 0x7fe52223e9e8>
## boundary (singular) fit: see ?isSingular
## env_pc ~ time * type + (-1 + time | condition) + (1 | id)
## <environment: 0x7fe520a5c7c8>
## boundary (singular) fit: see ?isSingular
## env_pc ~ time * type + (1 | id)
## <environment: 0x7fe5705b6c68>
## [1] "is model singular:  FALSE"
```

```
presence.models <- lapply(dvs, FUN = function(dv) fit_many(pred.vector = "ipq * time + (1 | id)", dat =
```

```
## iat ~ ipq * time + (1 | id)
## <environment: 0x7fe511b07dd8>
## [1] "is model singular:  FALSE"
## ccs ~ ipq * time + (1 | id)
## <environment: 0x7fe51291f778>
## [1] "is model singular:  FALSE"
## nr ~ ipq * time + (1 | id)
## <environment: 0x7fe512f6fa90>
## [1] "is model singular:  FALSE"
## nep ~ ipq * time + (1 | id)
## <environment: 0x7fe4f6d5ff90>
## [1] "is model singular:  FALSE"
## env_pc ~ ipq * time + (1 | id)
## <environment: 0x7fe506d4ad10>
## [1] "is model singular:  FALSE"
```

```
sod.models <- lapply(dvs, FUN = function(dv) fit_many(pred.vector = "sod * time + (1 | id)", dat = data
```

```
## iat ~ sod * time + (1 | id)
## <environment: 0x7fe511e3d278>
## [1] "is model singular:  FALSE"
## ccs ~ sod * time + (1 | id)
## <environment: 0x7fe4f748af98>
## [1] "is model singular:  FALSE"
```

```
## nr ~ sod * time + (1 | id)
## <environment: 0x7fe510393a68>
## [1] "is model singular:  FALSE"
## nep ~ sod * time + (1 | id)
## <environment: 0x7fe506029158>
## [1] "is model singular:  FALSE"
## env_pc ~ sod * time + (1 | id)
## <environment: 0x7fe510e1f830>
## [1] "is model singular:  FALSE"
```

Model diagnostics

I save model diagnostics as pdfs separately, for visibility reasons.

```
plot_diagn <- function(model){

  filename <- paste( model@call$formula[2], sub("\\ .*", "", model@call$formula[3]), sep = "_")
  png(filename = paste("analysisOutputs/diagnostics/", filename, ".png", sep = ""),   # The directory y
    #paper = "a3",
    height = 5900/4,
    width = 4200/4
    )

  print(performance::check_model(model)  )

  dev.off()
}
```

```
lapply(vr.models, FUN = plot_diagn)
```

```
## [[1]]
## pdf
##   2
##
## [[2]]
## pdf
##   2
##
## [[3]]
## pdf
##   2
##
## [[4]]
## pdf
##   2
##
## [[5]]
## pdf
##   2
```

```
lapply(all.models, FUN = plot_diagn)
```

```
## [[1]]
## pdf
##   2
##
```

```
## [[2]]
## pdf
##   2
##
## [[3]]
## pdf
##   2
##
## [[4]]
## pdf
##   2
##
## [[5]]
## pdf
##   2
```

I focus model diagnostic on the vr models. They include all data. Residuals are slightly left skewed. However, this does not yet warrant a transformation of the dv in my opinion.

IAT

\begin{figure}


Linearity
Reference line should be flat and horizontal
Residuals
Fitted values
Homogeneity of Variance
Reference line should be flat and horizontal
√|Std. residuals|
Fitted values
Collinearity
Higher bars (>5) indicate potential collinearity issues
Variance Inflation Factor (VIF)
time
vr
vr:time
low (< 5)
moderate (< 10)
high (>= 10)
Influential Observations
Points should be inside the contour lines
Std. Residuals
Leverage (hii)
0.8
280
105
102
248
279
Normality of Residuals
Dots should fall along the line
Sample Quantiles
Standard Normal Distribution Quantiles
Normality of Residuals
Distribution should be close to the normal curve
Density
Residuals
Normality of Random Effects (id)
Dots should be plotted along the line
RE Quantiles
Theoretical Quantiles
Normality of Random Effects (condition)
Dots should be plotted along the line
RE Quantiles
Theoretical Quantiles

{

}

\caption{iat_vr_diagnostics} \end{figure} Some thoughts: Band of residuals increases as fitted values increase. Homogeneity of variance seems acceptable. Random effects appear normal.

CCS

First some descriptives about ccs.

```
data %>% group_by(as.factor(time)) %>%
  summarise(mean = mean(ccs),
            range = range(ccs),
            median = median(ccs))
```

```
## group_by: one grouping variable (as.factor(time))
```

```
## summarise: now 4 rows and 4 columns, one group variable remaining (as.factor(time))
```

```
## # A tibble: 4 x 4
## # Groups:   as.factor(time) [2]
##   `as.factor(time)`  mean range median
##   <fct>             <dbl> <dbl>  <dbl>
## 1 1                  1.46  1      1.33
## 2 1                  1.46  3.25   1.33
## 3 2                  1.44  1      1.29
## 4 2                  1.44  3.83   1.29
```

\begin{figure}


Linearity
Reference line should be flat and horizontal
Residuals
Fitted values
Homogeneity of Variance
Reference line should be flat and horizontal
√|Std. residuals|
Fitted values
Collinearity
Higher bars (>5) indicate potential collinearity issues
Variance Inflation Factor (VIF)
time
vr
vr:time
low (< 5)
moderate (< 10)
high (>= 10)
Influential Observations
Points should be inside the contour lines
Std. Residuals
Leverage (hii)
124
123
0.8
0.8
Normality of Residuals
Dots should fall along the line
Sample Quantiles
Standard Normal Distribution Quantiles
Normality of Residuals
Distribution should be close to the normal curve
Density
Residuals
Normality of Random Effects (id)
Dots should be plotted along the line
RE Quantiles
Theoretical Quantiles

{

}

\caption{ccs_vr_diagnostics} \end{figure}

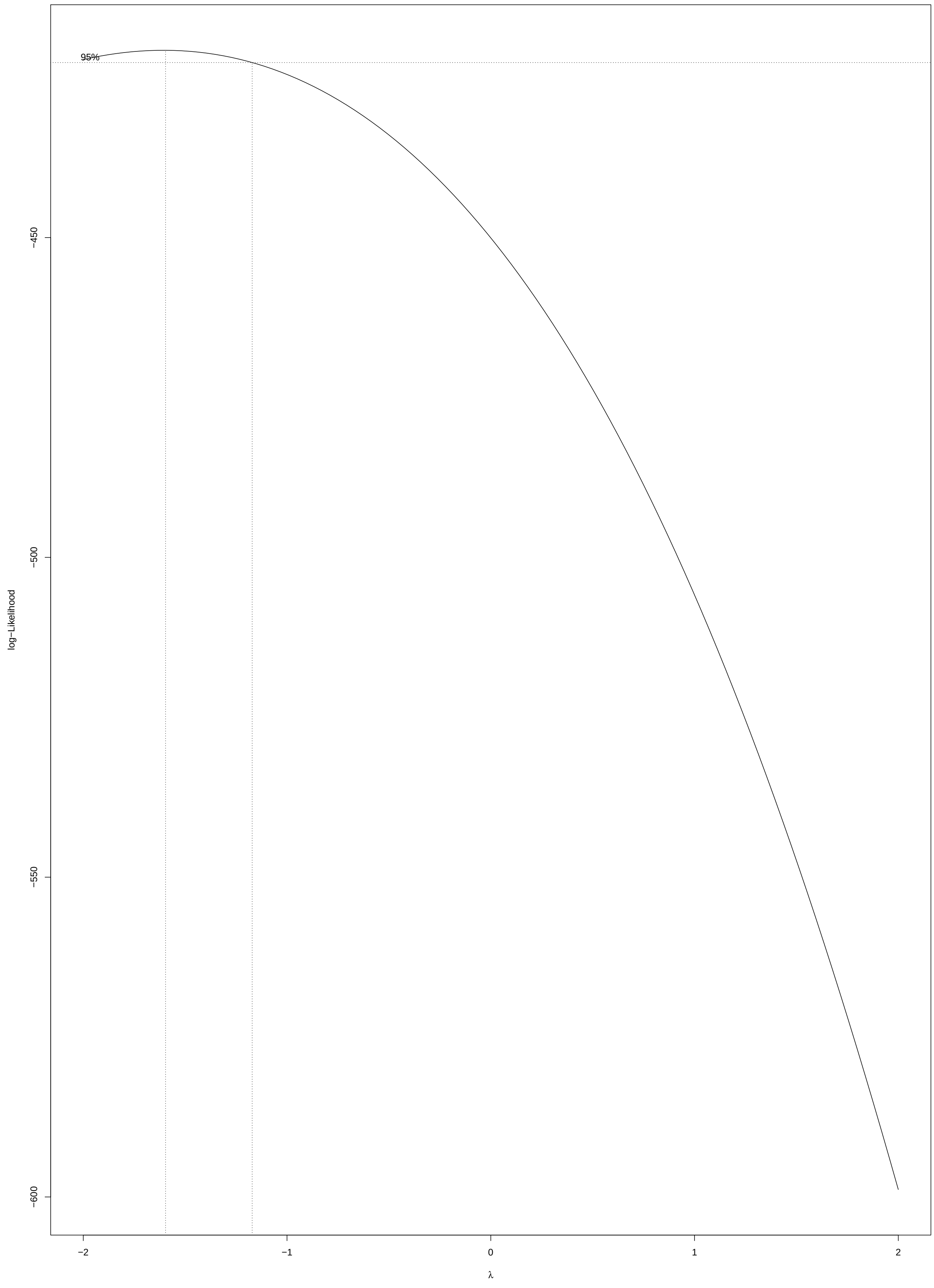
Some thoughts: Homogeneity of variance appears implausible. Residual variance increases with larger fitted values.

Residuals are also not normally distributed. Random effects do not appear normal.

The reason for this unexpected behaviour may well be the floor-effect of the dv ccs. There was generally a very low ccs score for participants. This is due to the relatively extreme nature of climate change scepticism, especially in a relatively well educated sample.

Maybe a boxcox transformation may help:

```
#estimate lambda of the boxcox transformation
bc <- boxcox(ccs ~ vr * time, data = data)
```


95%
−450
−500
−550
−600
log−Likelihood
−2
−1
0
1
2
λ

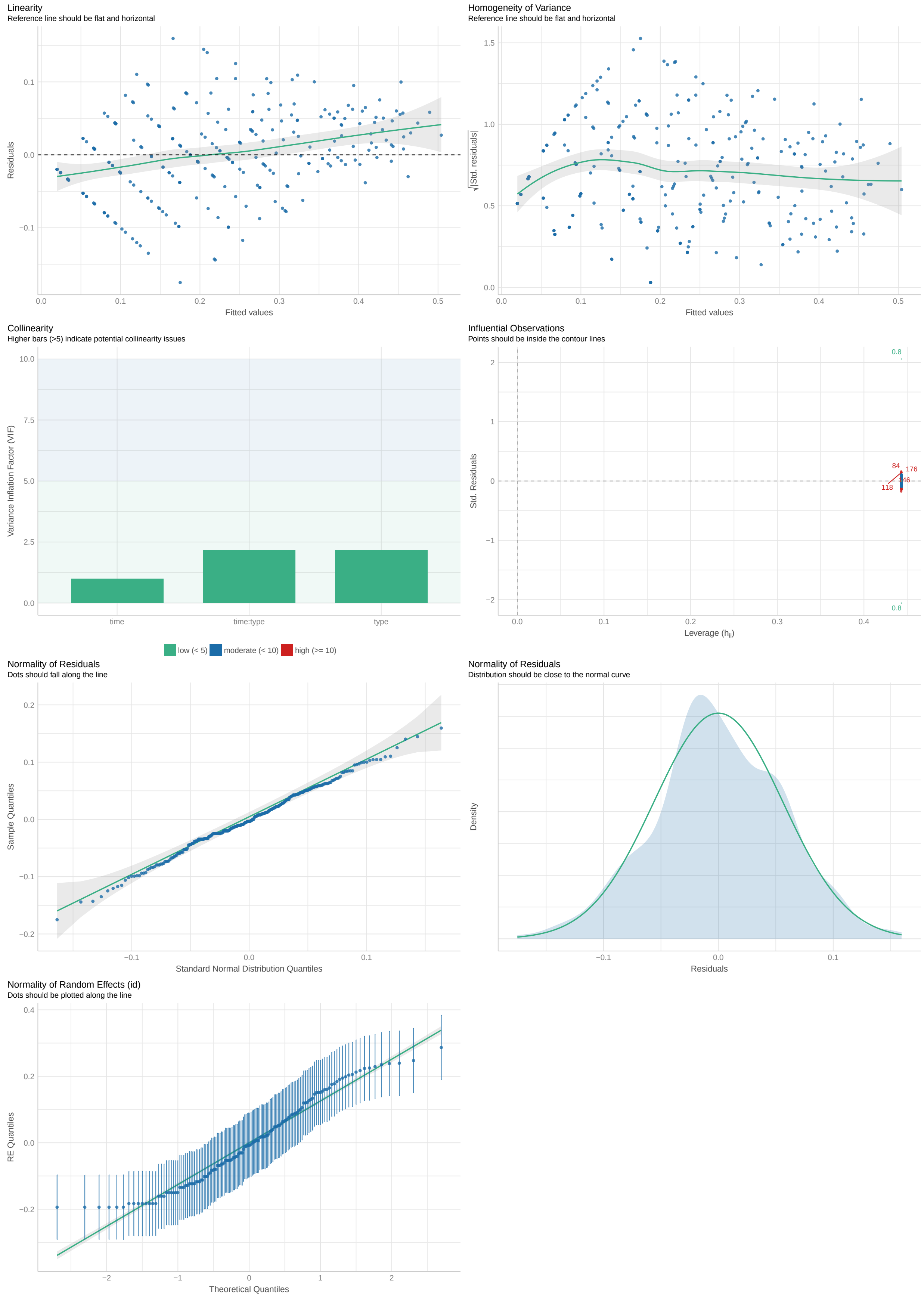
```
lambda_ccs <- bc$x[which.max(bc$y)]

# transform data according to the transformation
data <- data %>%
  mutate(ccs_bc = (ccs^lambda_ccs-1)/lambda_ccs)
```

```
## mutate: new variable 'ccs_bc' (double) with 28 unique values and 0% NA
```

```
# refit the model
all.ccs2 <- fit_many(pred.vector = predictors.all, dat = data, dv = "ccs_bc")
```

```
## ccs_bc ~ time * type + (time | condition) + (1 | id)
## <environment: 0x7fe511f58558>

## boundary (singular) fit: see ?isSingular

## ccs_bc ~ time * type + (-1 + time | condition) + (1 | id)
## <environment: 0x7fe506088be0>

## boundary (singular) fit: see ?isSingular

## ccs_bc ~ time * type + (1 | id)
## <environment: 0x7fe5065131e0>
## [1] "is model singular:  FALSE"
```

```
performance::check_model(all.ccs2)
```

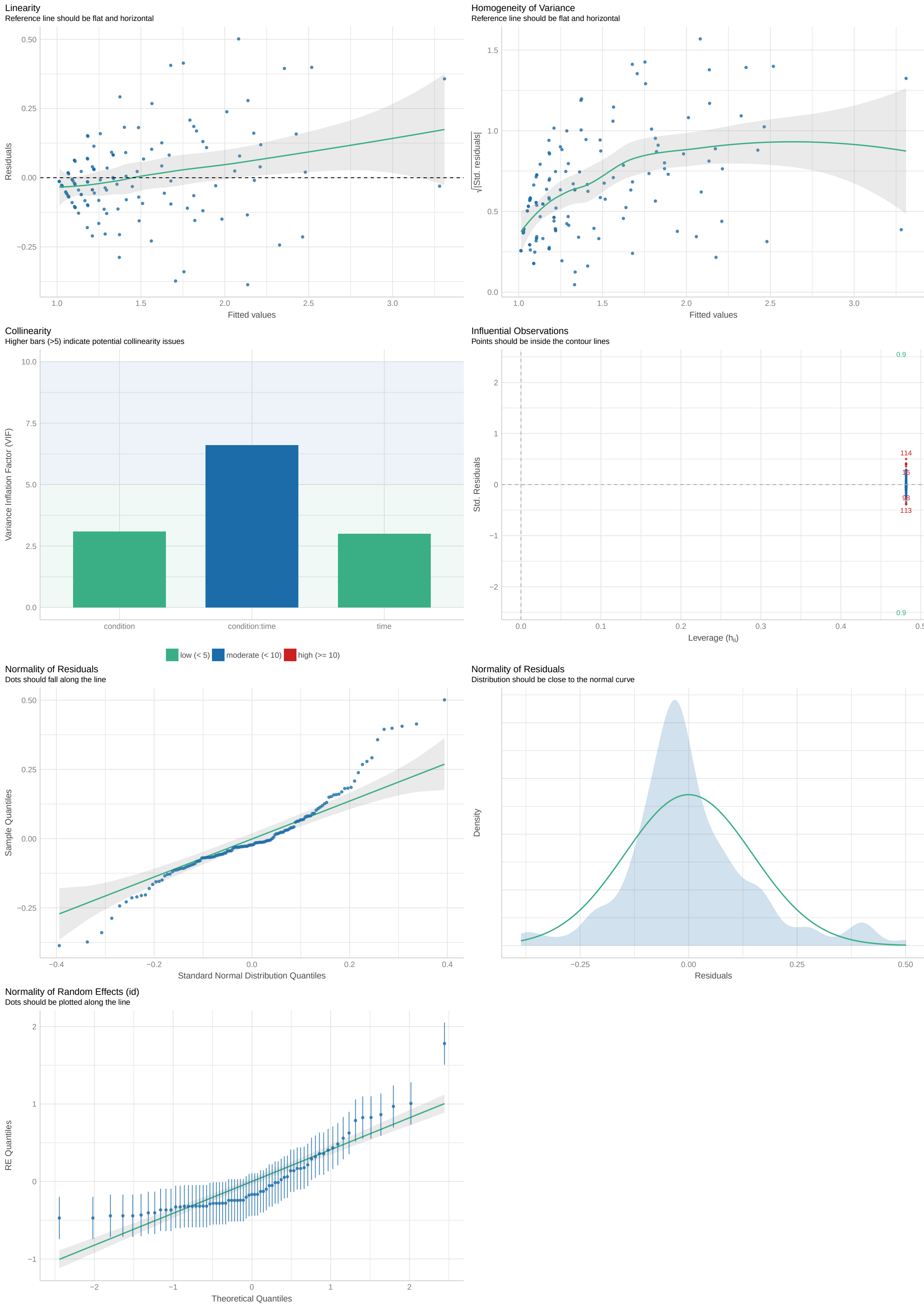

Linearity
Reference line should be flat and horizontal
Residuals
Fitted values
Homogeneity of Variance
Reference line should be flat and horizontal
√|Std. residuals|
Fitted values
Collinearity
Higher bars (>5) indicate potential collinearity issues
Variance Inflation Factor (VIF)
time
time:type
type
low (< 5)
moderate (< 10)
high (>= 10)
Influential Observations
Points should be inside the contour lines
Std. Residuals
Leverage (h_{ii})
0.8
84
176
46
118
Normality of Residuals
Dots should fall along the line
Sample Quantiles
Standard Normal Distribution Quantiles
Normality of Residuals
Distribution should be close to the normal curve
Density
Residuals
Normality of Random Effects (id)
Dots should be plotted along the line
RE Quantiles
Theoretical Quantiles

The situation has improved! All model assumptions appear plausible.

```
all.models[[2]] <- all.ccs2
```

For within the VE:

\begin{figure}


Linearity
Reference line should be flat and horizontal
Residuals
Fitted values
Homogeneity of Variance
Reference line should be flat and horizontal
√|Std. residuals|
Fitted values
Collinearity
Higher bars (>5) indicate potential collinearity issues
Variance Inflation Factor (VIF)
condition
condition:time
time
low (< 5)
moderate (< 10)
high (>= 10)
Influential Observations
Points should be inside the contour lines
Std. Residuals
Leverage (h_{ii})
0.9
114
113
Normality of Residuals
Dots should fall along the line
Sample Quantiles
Standard Normal Distribution Quantiles
Normality of Residuals
Distribution should be close to the normal curve
Density
Residuals
Normality of Random Effects (id)
Dots should be plotted along the line
RE Quantiles
Theoretical Quantiles

{

}

\caption{ccs_condition_diagnostics} \end{figure}

And based on the transformed ccs:

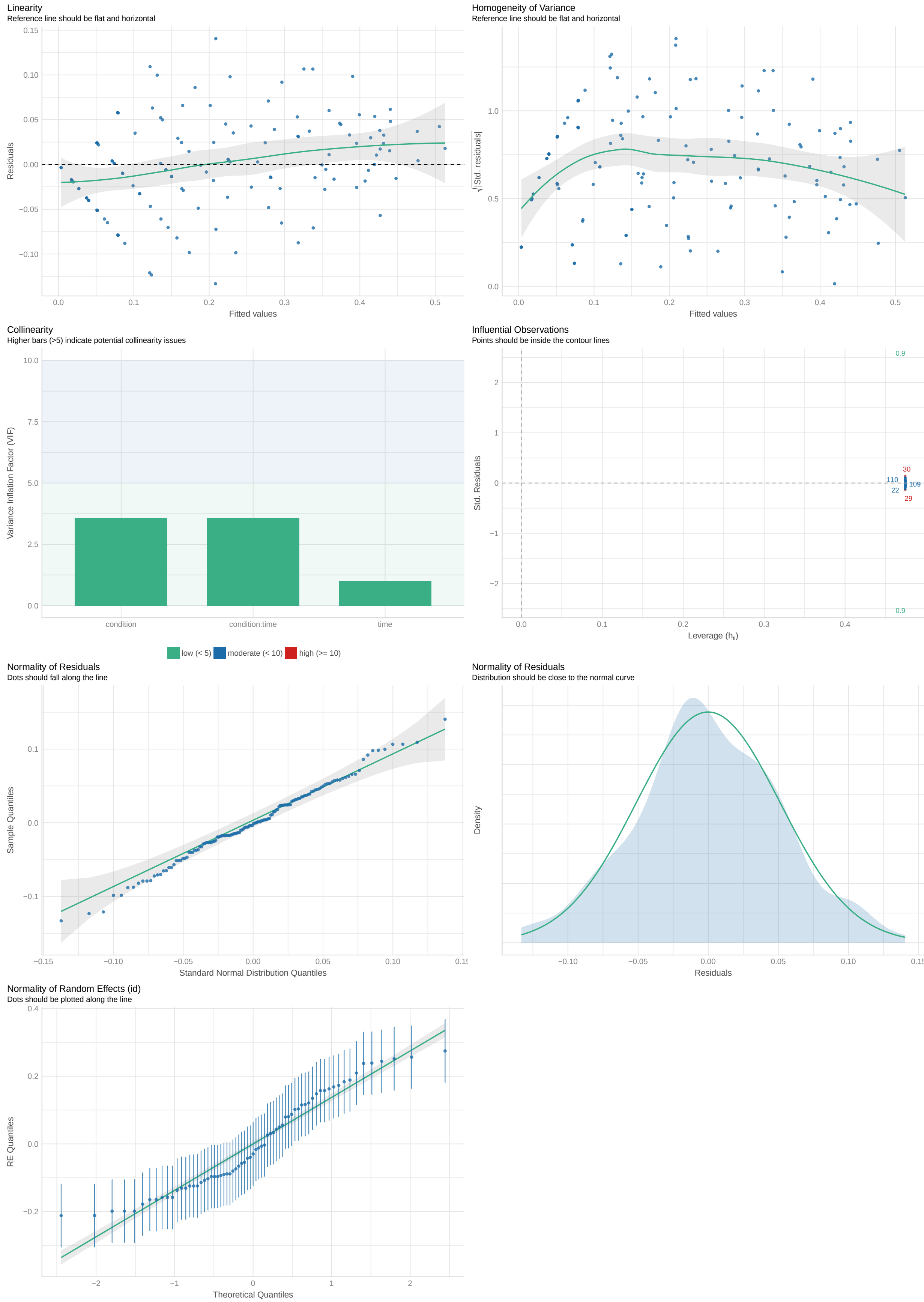
```
data.vr <- data.vr %>%
  mutate(ccs_bc = (ccs^lambda_ccs-1)/lambda_ccs)
```

```
## mutate: new variable 'ccs_bc' (double) with 24 unique values and 0% NA
```

```
vr.ccs2 <- fit_many(pred.vector = predictors.vr, dat = data.vr, dv = "ccs_bc")
```

```
## ccs_bc ~ condition * time + (time | id)
## <environment: 0x7fe504b9f3f8>
## Error : number of observations (=138) <= number of random effects (=138) for term (time | id); the ra
## ccs_bc ~ condition * time + (1 | id)
## <environment: 0x7fe503c89e40>
## [1] "is model singular:  FALSE"
```

```
performance::check_model(vr.ccs2)
```


Linearity
Reference line should be flat and horizontal
Residuals
Fitted values
Homogeneity of Variance
Reference line should be flat and horizontal
√|Std. residuals|
Fitted values
Collinearity
Higher bars (>5) indicate potential collinearity issues
Variance Inflation Factor (VIF)
condition
condition:time
time
low (< 5)
moderate (< 10)
high (>= 10)
Influential Observations
Points should be inside the contour lines
Std. Residuals
Leverage (h_{ii})
0.9
30
110
109
22
29
Normality of Residuals
Dots should fall along the line
Sample Quantiles
Standard Normal Distribution Quantiles
Normality of Residuals
Distribution should be close to the normal curve
Density
Residuals
Normality of Random Effects (id)
Dots should be plotted along the line
RE Quantiles
Theoretical Quantiles

```
vr.models[[2]] <- vr.ccs2
```

NR

\begin{figure}


Linearity
Reference line should be flat and horizontal
Residuals
Fitted values
Homogeneity of Variance
Reference line should be flat and horizontal
√|Std. residuals|
Fitted values
Collinearity
Higher bars (>5) indicate potential collinearity issues
Variance Inflation Factor (VIF)
time
vr
vr:time
low (< 5)
moderate (< 10)
high (>= 10)
Influential Observations
Points should be inside the contour lines
Std. Residuals
Leverage (h_ii)
0.8
280
105
102
248
279
Normality of Residuals
Dots should fall along the line
Sample Quantiles
Standard Normal Distribution Quantiles
Normality of Residuals
Distribution should be close to the normal curve
Density
Residuals
Normality of Random Effects (id)
Dots should be plotted along the line
RE Quantiles
Theoretical Quantiles
Normality of Random Effects (condition)
Dots should be plotted along the line
RE Quantiles
Theoretical Quantiles

{

}

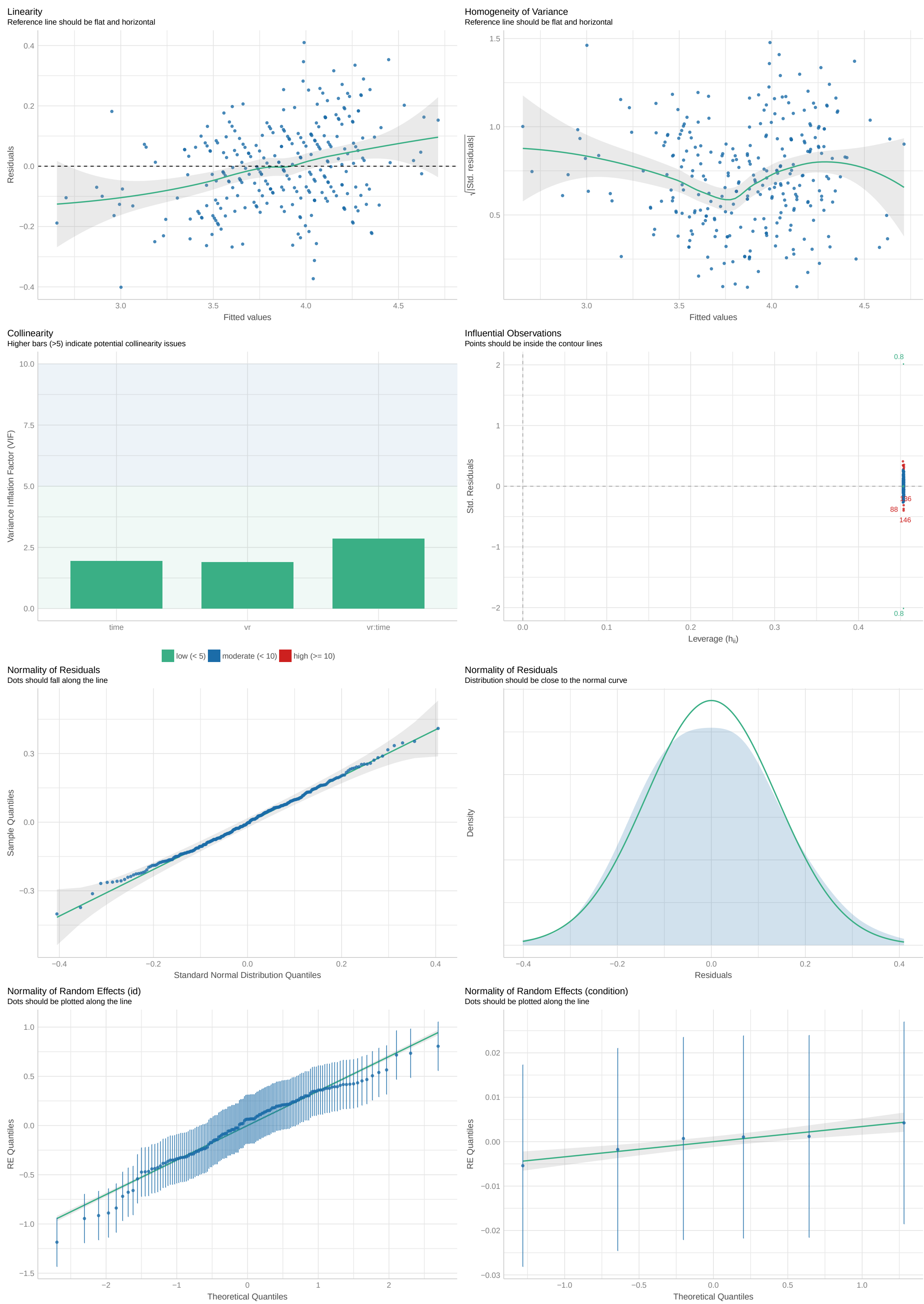
\caption{nr_vr_diagnostics} \end{figure}

Model assumptions are not too far off: Slight slope in the "fitted vs residuals". Homogeneity assumption is appropriate. Residual distribution is slightly skewed with a heavy left tail. ID intercept distribution is not quite normal, but not far from it. Slightly skewed as well.

Overall assumptions seem acceptable and warrant no further action.

NEP

\begin{figure}


Linearity
Reference line should be flat and horizontal
Residuals
Fitted values
Homogeneity of Variance
Reference line should be flat and horizontal
√|Std. residuals|
Fitted values
Collinearity
Higher bars (>5) indicate potential collinearity issues
Variance Inflation Factor (VIF)
time
vr
vr:time
low (< 5)
moderate (< 10)
high (>= 10)
Influential Observations
Points should be inside the contour lines
Std. Residuals
Leverage (hii)
0.8
88
146
Normality of Residuals
Dots should fall along the line
Sample Quantiles
Standard Normal Distribution Quantiles
Normality of Residuals
Distribution should be close to the normal curve
Density
Residuals
Normality of Random Effects (id)
Dots should be plotted along the line
RE Quantiles
Theoretical Quantiles
Normality of Random Effects (condition)
Dots should be plotted along the line
RE Quantiles
Theoretical Quantiles

{

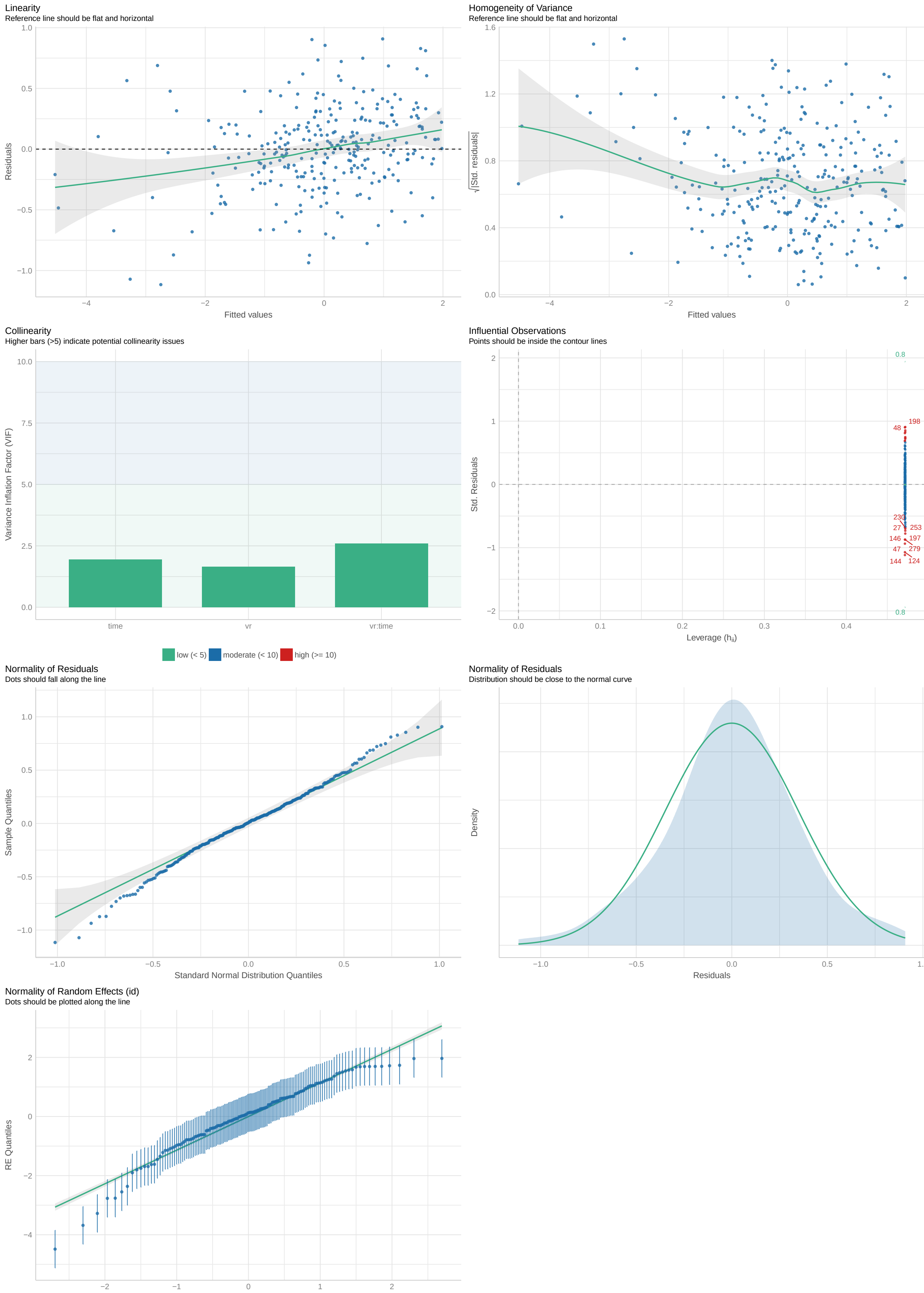
}

\caption{nep_vr_diagnostics} \end{figure} Model assumptions are not too far off: Slight slope in the "fitted vs residuals". Residuals appear normally distributed, slightly skewed to the right.

Overall assumptions seem acceptable and warrant no further action.

Principal component

\begin{figure}


Linearity
Reference line should be flat and horizontal
Residuals
Fitted values
Homogeneity of Variance
Reference line should be flat and horizontal
√|Std. residuals|
Fitted values
Collinearity
Higher bars (>5) indicate potential collinearity issues
Variance Inflation Factor (VIF)
time
vr
vr:time
low (< 5)
moderate (< 10)
high (>= 10)
Influential Observations
Points should be inside the contour lines
Std. Residuals
Leverage (h_{ii})
Normality of Residuals
Dots should fall along the line
Sample Quantiles
Standard Normal Distribution Quantiles
Normality of Residuals
Distribution should be close to the normal curve
Density
Residuals
Normality of Random Effects (id)
Dots should be plotted along the line
RE Quantiles
Theoretical Quantiles

{

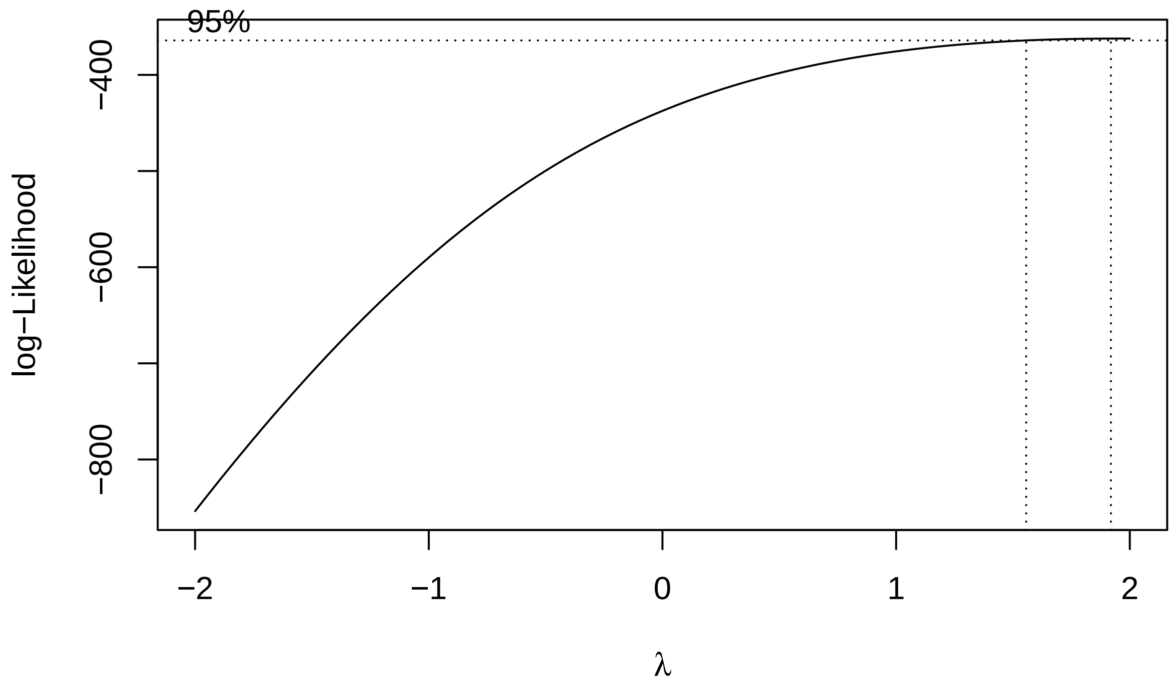
}

\caption{env_pc_vr_diagnostics} \end{figure}

```
min(data$env_pc)
```

```
## [1] -4.954711
```

```
data$env_pc2 <- data$env_pc + 6

#estimate lambda of the boxcox transformation
bc <- boxcox(env_pc2 ~ vr * time, data = data)
```



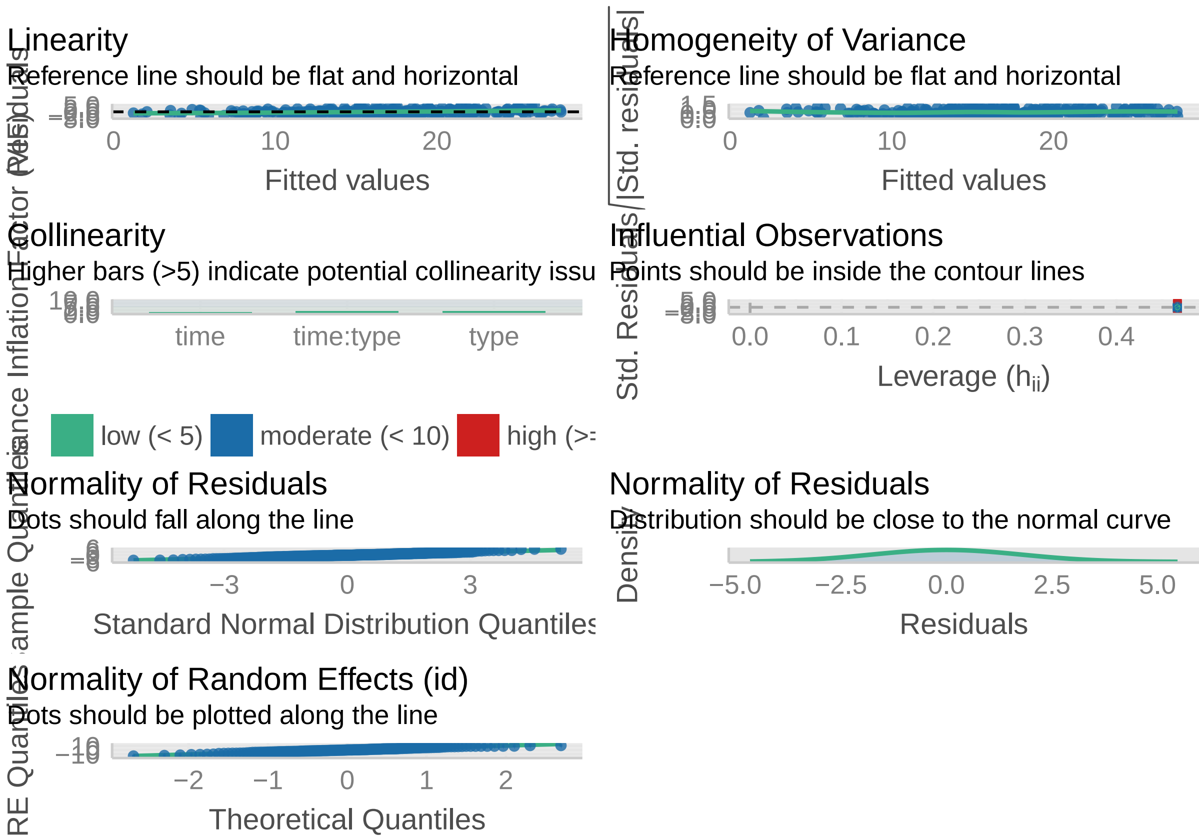

```
lambda_pc <- bc$x[which.max(bc$y)]

# transform data according to the transformation
data <- data %>%
  mutate(env_pc_bc = (env_pc2^lambda_pc-1)/lambda_pc)
```

```
## mutate: new variable 'env_pc_bc' (double) with 284 unique values and 0% NA
```

```
# refit the model
all.env_pc2 <- fit_many(pred.vector = predictors.all, dat = data, dv = "env_pc_bc")
```

```
## env_pc_bc ~ time * type + (time | condition) + (1 | id)
## <environment: 0x7fe5172611e0>

## boundary (singular) fit: see ?isSingular

## env_pc_bc ~ time * type + (-1 + time | condition) + (1 | id)
## <environment: 0x7fe506bc7240>

## boundary (singular) fit: see ?isSingular

## env_pc_bc ~ time * type + (1 | id)
## <environment: 0x7fe521710ea0>
## [1] "is model singular:  FALSE"
```

```r
performance::check_model(all.env_pc2)
```



Transformation does not help much here. Lets not do it.

But let us save the final model diagnostics plots as well.

```r
lapply(vr.models, FUN = plot_diagn)
```

```
## [[1]]
## pdf
##   2
##
## [[2]]
## pdf
##   2
##
## [[3]]
## pdf
##   2
##
## [[4]]
## pdf
##   2
##
## [[5]]
## pdf
##   2
```

```r
lapply(all.models, FUN = plot_diagn)
```

```
## [[1]]
## pdf
##   2
##
## [[2]]
## pdf
##   2
##
## [[3]]
## pdf
##   2
##
## [[4]]
## pdf
##   2
##
## [[5]]
## pdf
##   2
```

Inference

vr vs control

```r
lapply(all.models, FUN = lmerTest:::summary.lmerModLmerTest)
```

```
## Coercing object to class 'lmerModLmerTest'
## Coercing object to class 'lmerModLmerTest'
## Coercing object to class 'lmerModLmerTest'
## Coercing object to class 'lmerModLmerTest'
## Coercing object to class 'lmerModLmerTest'
```

```
## [[1]]
## Linear mixed model fit by REML. t-tests use Satterthwaite's method [
## lmerModLmerTest]
## Formula: iat ~ time * type + (-1 + time | condition) + (1 | id)
##    Data: dat
##
## REML criterion at convergence: 315.2
##
## Scaled residuals:
##      Min       1Q   Median       3Q      Max
## -3.00917 -0.60925  0.02082  0.59421  2.05824
##
## Random effects:
##  Groups    Name        Variance Std.Dev.
##  id        (Intercept) 0.098991 0.31463
##  condition time        0.002013 0.04486
##  Residual              0.094707 0.30775
## Number of obs: 284, groups:  id, 142; condition, 6
##
## Fixed effects:
##               Estimate Std. Error         df t value Pr(>|t|)
```

```
## (Intercept)   0.230413   0.063522 225.155717   3.627 0.000354 ***
## time         -0.009332   0.040871  20.397083  -0.228 0.821660
## type1        -0.152101   0.063522 225.155717  -2.394 0.017465 *
## time:type1    0.074652   0.040871  20.397083   1.827 0.082445 .
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Correlation of Fixed Effects:
##            (Intr) time   type1
## time       -0.771
## type1       0.028 -0.022
## time:type1 -0.022  0.022 -0.771
##
## [[2]]
## Linear mixed model fit by REML. t-tests use Satterthwaite's method [
## lmerModLmerTest]
## Formula: ccs_bc ~ time * type + (1 | id)
##    Data: dat
##
## REML criterion at convergence: -349
##
## Scaled residuals:
##      Min       1Q   Median       3Q      Max
## -2.33016 -0.44377 -0.04596  0.56783  2.12335
##
## Random effects:
##  Groups   Name        Variance Std.Dev.
##  id       (Intercept) 0.018962 0.13770
##  Residual             0.005645 0.07513
## Number of obs: 284, groups:  id, 142
##
## Fixed effects:
##               Estimate Std. Error         df t value Pr(>|t|)
## (Intercept)   0.234121   0.018236 278.363626  12.838   <2e-16 ***
## time         -0.011764   0.008920 140.000003  -1.319    0.189
## type1        -0.002096   0.018236 278.363626  -0.115    0.909
## time:type1   -0.002811   0.008920 140.000003  -0.315    0.753
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Correlation of Fixed Effects:
##            (Intr) time   type1
## time       -0.734
## type1       0.028 -0.021
## time:type1 -0.021  0.028 -0.734
##
## [[3]]
## Linear mixed model fit by REML. t-tests use Satterthwaite's method [
## lmerModLmerTest]
## Formula: nr ~ time * type + (-1 + time | condition) + (1 | id)
##    Data: dat
##
## REML criterion at convergence: 265.4
##
```

```
## Scaled residuals:
##      Min       1Q   Median       3Q      Max
## -2.59554 -0.44842  0.02171  0.48568  1.76425
##
## Random effects:
##  Groups    Name        Variance  Std.Dev.
##  id        (Intercept) 0.2263312 0.47574
##  condition time        0.0003591 0.01895
##  Residual              0.0400008 0.20000
## Number of obs: 284, groups:  id, 142; condition, 6
##
## Fixed effects:
##              Estimate Std. Error        df t value Pr(>|t|)
## (Intercept)   3.81760    0.05482 268.09998  69.644   <2e-16 ***
## time          0.03769    0.02497   9.57301   1.509    0.164
## type1        -0.01084    0.05482 268.09998  -0.198    0.843
## time:type1    0.03891    0.02497   9.57301   1.558    0.152
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Correlation of Fixed Effects:
##            (Intr) time   type1
## time       -0.618
## type1       0.028 -0.017
## time:type1 -0.017  0.025 -0.618
##
## [[4]]
## Linear mixed model fit by REML. t-tests use Satterthwaite's method [
## lmerModLmerTest]
## Formula: nep ~ time * type + (-1 + time | condition) + (1 | id)
##    Data: dat
##
## REML criterion at convergence: 187.5
##
## Scaled residuals:
##      Min       1Q   Median       3Q      Max
## -2.13567 -0.52383 -0.01767  0.49178  2.18200
##
## Random effects:
##  Groups    Name        Variance  Std.Dev.
##  id        (Intercept) 0.1427876 0.37787
##  condition time        0.0001482 0.01217
##  Residual              0.0352898 0.18786
## Number of obs: 284, groups:  id, 142; condition, 6
##
## Fixed effects:
##              Estimate Std. Error        df t value Pr(>|t|)
## (Intercept)   3.75582    0.04743 276.21351  79.181  < 2e-16 ***
## time          0.07381    0.02285  14.88848   3.230  0.00565 **
## type1        -0.03505    0.04743 276.21351  -0.739  0.46061
## time:type1    0.01894    0.02285  14.88848   0.829  0.42021
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
```

```
## Correlation of Fixed Effects:
##            (Intr) time   type1
## time       -0.688
## type1       0.028 -0.019
## time:type1 -0.019  0.027 -0.688
##
## [[5]]
## Linear mixed model fit by REML. t-tests use Satterthwaite's method [
## lmerModLmerTest]
## Formula: env_pc ~ time * type + (1 | id)
##    Data: dat
##
## REML criterion at convergence: 769
##
## Scaled residuals:
##      Min       1Q   Median       3Q      Max
## -2.33852 -0.41115  0.01461  0.44995  1.90160
##
## Random effects:
##  Groups   Name        Variance Std.Dev.
##  id       (Intercept) 1.4722   1.2133
##  Residual             0.2278   0.4773
## Number of obs: 284, groups:  id, 142
##
## Fixed effects:
##              Estimate Std. Error        df t value Pr(>|t|)
## (Intercept)  -0.23777    0.13566 267.65387  -1.753  0.08081 .
## time          0.15874    0.05667 140.00000   2.801  0.00581 **
## type1        -0.14234    0.13566 267.65387  -1.049  0.29502
## time:type1    0.10297    0.05667 140.00000   1.817  0.07134 .
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Correlation of Fixed Effects:
##            (Intr) time   type1
## time       -0.627
## type1       0.028 -0.018
## time:type1 -0.018  0.028 -0.627
```

Follow up tests:

```
# iat
contrast( emmeans(all.models[[1]],  ~ time | type))
```

```
## type = vr:
##  contrast estimate     SE   df t.ratio p.value
##  1 effect  -0.0327 0.0292 23.0  -1.118  0.2753
##  2 effect   0.0327 0.0292 23.0   1.118  0.2753
##
## type = control:
##  contrast estimate     SE   df t.ratio p.value
##  1 effect   0.0420 0.0286 21.2   1.470  0.1563
##  2 effect  -0.0420 0.0286 21.2  -1.470  0.1563
##
## Degrees-of-freedom method: kenward-roger
```

```
## P value adjustment: fdr method for 2 tests
```

```r
# ccs
contrast( emmeans(all.models[[2]],  ~ time | type))
```

```
## type = vr:
##  contrast estimate      SE  df t.ratio p.value
##  1 effect  0.00729 0.00640 140   1.139  0.2565
##  2 effect -0.00729 0.00640 140  -1.139  0.2565
##
## type = control:
##  contrast estimate      SE  df t.ratio p.value
##  1 effect  0.00448 0.00622 140   0.720  0.4728
##  2 effect -0.00448 0.00622 140  -0.720  0.4728
##
## Degrees-of-freedom method: kenward-roger
## P value adjustment: fdr method for 2 tests
```

```r
# nr
contrast( emmeans(all.models[[3]],  ~ time | type))
```

```
## type = vr:
##  contrast  estimate     SE    df t.ratio p.value
##  1 effect -0.038302 0.0179 10.66  -2.142  0.0562
##  2 effect  0.038302 0.0179 10.66   2.142  0.0562
##
## type = control:
##  contrast  estimate     SE    df t.ratio p.value
##  1 effect  0.000613 0.0174  9.67   0.035  0.9727
##  2 effect -0.000613 0.0174  9.67  -0.035  0.9727
##
## Degrees-of-freedom method: kenward-roger
## P value adjustment: fdr method for 2 tests
```

```r
# nep
contrast( emmeans(all.models[[4]],  ~ time | type))
```

```
## type = vr:
##  contrast estimate     SE   df t.ratio p.value
##  1 effect  -0.0464 0.0164 14.4  -2.833  0.0130
##  2 effect   0.0464 0.0164 14.4   2.833  0.0130
##
## type = control:
##  contrast estimate     SE   df t.ratio p.value
##  1 effect  -0.0274 0.0159 13.0  -1.721  0.1090
##  2 effect   0.0274 0.0159 13.0   1.721  0.1090
##
## Degrees-of-freedom method: kenward-roger
## P value adjustment: fdr method for 2 tests
```

Within the VEs (condition)

```r
lapply(vr.models, FUN = lmerTest:::summary.lmerModLmerTest)
```

```
## Coercing object to class 'lmerModLmerTest'
## Coercing object to class 'lmerModLmerTest'
```

```
## Coercing object to class 'lmerModLmerTest'
## Coercing object to class 'lmerModLmerTest'
## Coercing object to class 'lmerModLmerTest'
## [[1]]
## Linear mixed model fit by REML. t-tests use Satterthwaite's method [
## lmerModLmerTest]
## Formula: iat ~ condition * time + (1 | id)
##    Data: dat
##
## REML criterion at convergence: 177.6
##
## Scaled residuals:
##      Min       1Q   Median       3Q      Max
## -2.60384 -0.59622 -0.01456  0.55523  1.93986
##
## Random effects:
##  Groups   Name        Variance Std.Dev.
##  id       (Intercept) 0.1132   0.3365
##  Residual             0.1068   0.3268
## Number of obs: 138, groups:  id, 69
##
## Fixed effects:
##                  Estimate Std. Error        df t value Pr(>|t|)
## (Intercept)       0.07831    0.09684 106.85910   0.809    0.421
## condition1        0.09886    0.13695 106.85910   0.722    0.472
## condition2       -0.22578    0.13695 106.85910  -1.649    0.102
## time              0.06532    0.05563  66.00000   1.174    0.245
## condition1:time  -0.07723    0.07868  66.00000  -0.982    0.330
## condition2:time   0.03911    0.07868  66.00000   0.497    0.621
##
## Correlation of Fixed Effects:
##             (Intr) cndtn1 cndtn2 time   cndt1:
## condition1   0.000
## condition2   0.000 -0.500
## time        -0.862  0.000  0.000
## conditn1:tm  0.000 -0.862  0.431  0.000
## conditn2:tm  0.000  0.431 -0.862  0.000 -0.500
##
## [[2]]
## Linear mixed model fit by REML. t-tests use Satterthwaite's method [
## lmerModLmerTest]
## Formula: ccs_bc ~ condition * time + (1 | id)
##    Data: dat
##
## REML criterion at convergence: -149.5
##
## Scaled residuals:
##      Min       1Q   Median       3Q      Max
## -1.88848 -0.39218 -0.03104  0.48900  1.99181
##
## Random effects:
##  Groups   Name        Variance Std.Dev.
##  id       (Intercept) 0.022653 0.15051
```

```
##  Residual             0.004976 0.07054
## Number of obs: 138, groups:  id, 69
##
## Fixed effects:
##                  Estimate Std. Error         df t value Pr(>|t|)
## (Intercept)       0.232025   0.026247 131.560393   8.840 5.38e-15 ***
## condition1       -0.057282   0.037119 131.560393  -1.543    0.125
## condition2        0.014219   0.037119 131.560393   0.383    0.702
## time             -0.014575   0.012010  66.000002  -1.214    0.229
## condition1:time   0.015010   0.016985  66.000001   0.884    0.380
## condition2:time   0.006983   0.016985  66.000001   0.411    0.682
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Correlation of Fixed Effects:
##             (Intr) cndtn1 cndtn2 time   cndt1:
## condition1   0.000
## condition2   0.000 -0.500
## time        -0.686  0.000  0.000
## conditn1:tm  0.000 -0.686  0.343  0.000
## conditn2:tm  0.000  0.343 -0.686  0.000 -0.500
##
## [[3]]
## Linear mixed model fit by REML. t-tests use Satterthwaite's method [
## lmerModLmerTest]
## Formula: nr ~ condition * time + (1 | id)
##    Data: dat
##
## REML criterion at convergence: 102.9
##
## Scaled residuals:
##      Min       1Q   Median       3Q      Max
## -1.74225 -0.50265 -0.03626  0.54439  1.71416
##
## Random effects:
##  Groups   Name        Variance Std.Dev.
##  id       (Intercept) 0.11846  0.3442
##  Residual             0.04125  0.2031
## Number of obs: 138, groups:  id, 69
##
## Fixed effects:
##                 Estimate Std. Error        df t value Pr(>|t|)
## (Intercept)       3.80676    0.06860 129.34427  55.494  < 2e-16 ***
## condition1        0.09593    0.09701 129.34427   0.989  0.32460
## condition2       -0.28916    0.09701 129.34427  -2.981  0.00344 **
## time              0.07660    0.03458  66.00000   2.215  0.03018 *
## condition1:time   0.01449    0.04890  66.00000   0.296  0.76787
## condition2:time   0.03727    0.04890  66.00000   0.762  0.44870
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Correlation of Fixed Effects:
##             (Intr) cndtn1 cndtn2 time   cndt1:
## condition1   0.000
```

```
## condition2   0.000 -0.500
## time        -0.756  0.000  0.000
## conditn1:tm  0.000 -0.756  0.378  0.000
## conditn2:tm  0.000  0.378 -0.756  0.000 -0.500
##
## [[4]]
## Linear mixed model fit by REML. t-tests use Satterthwaite's method [
## lmerModLmerTest]
## Formula: nep ~ condition * time + (1 | id)
##    Data: dat
##
## REML criterion at convergence: 98.3
##
## Scaled residuals:
##      Min       1Q   Median       3Q      Max
## -1.32935 -0.59590 -0.04369  0.49713  1.85313
##
## Random effects:
##  Groups   Name        Variance Std.Dev.
##  id       (Intercept) 0.13842  0.3721
##  Residual             0.03437  0.1854
## Number of obs: 138, groups:  id, 69
##
## Fixed effects:
##                 Estimate Std. Error        df t value Pr(>|t|)
## (Intercept)      3.72077    0.06706 131.99874  55.488  < 2e-16 ***
## condition1      -0.04541    0.09483 131.99874  -0.479  0.63283
## condition2      -0.05990    0.09483 131.99874  -0.632  0.52869
## time             0.09275    0.03156  66.00000   2.939  0.00454 **
## condition1:time  0.02319    0.04463  66.00000   0.520  0.60514
## condition2:time  0.03188    0.04463  66.00000   0.714  0.47754
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Correlation of Fixed Effects:
##             (Intr) cndtn1 cndtn2 time   cndt1:
## condition1   0.000
## condition2   0.000 -0.500
## time        -0.706  0.000  0.000
## conditn1:tm  0.000 -0.706  0.353  0.000
## conditn2:tm  0.000  0.353 -0.706  0.000 -0.500
##
## [[5]]
## Linear mixed model fit by REML. t-tests use Satterthwaite's method [
## lmerModLmerTest]
## Formula: env_pc ~ condition * time + (1 | id)
##    Data: dat
##
## REML criterion at convergence: 372.1
##
## Scaled residuals:
##      Min       1Q   Median       3Q      Max
## -1.61435 -0.42168  0.01546  0.43108  1.86555
##
```

```
## Random effects:
##  Groups   Name        Variance Std.Dev.
##  id       (Intercept) 1.2457   1.1161
##  Residual             0.2474   0.4974
## Number of obs: 138, groups:  id, 69
##
## Fixed effects:
##                   Estimate Std. Error         df t value Pr(>|t|)
## (Intercept)      -0.380108   0.189692 130.611470  -2.004  0.04716 *
## condition1        0.214738   0.268266 130.611470   0.800  0.42489
## condition2       -0.532468   0.268266 130.611470  -1.985  0.04926 *
## time              0.261713   0.084687  66.000001   3.090  0.00293 **
## condition1:time  -0.009723   0.119765  66.000001  -0.081  0.93554
## condition2:time   0.070200   0.119765  66.000001   0.586  0.55978
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Correlation of Fixed Effects:
##              (Intr) cndtn1 cndtn2 time   cndt1:
## condition1    0.000
## condition2    0.000 -0.500
## time         -0.670  0.000  0.000
## conditn1:tm  0.000 -0.670  0.335  0.000
## conditn2:tm  0.000  0.335 -0.670  0.000 -0.500
```

contrast analysis:

```
# iat
contrast( emmeans(vr.models[[1]],  ~ time | condition))
```

```
## condition = b:
##  contrast estimate     SE df t.ratio p.value
##  1 effect  0.00595 0.0482 66   0.124  0.9020
##  2 effect -0.00595 0.0482 66  -0.124  0.9020
##
## condition = a:
##  contrast estimate     SE df t.ratio p.value
##  1 effect -0.05221 0.0482 66  -1.084  0.2824
##  2 effect  0.05221 0.0482 66   1.084  0.2824
##
## condition = c:
##  contrast estimate     SE df t.ratio p.value
##  1 effect -0.05172 0.0482 66  -1.073  0.2870
##  2 effect  0.05172 0.0482 66   1.073  0.2870
##
## Degrees-of-freedom method: kenward-roger
## P value adjustment: fdr method for 2 tests
```

```
# ccs
contrast( emmeans(vr.models[[2]],  ~ time | condition))
```

```
## condition = b:
##  contrast  estimate     SE df t.ratio p.value
##  1 effect -0.000218 0.0104 66  -0.021  0.9834
##  2 effect  0.000218 0.0104 66   0.021  0.9834
##
```

```
## condition = a:
##  contrast  estimate     SE df t.ratio p.value
##  1 effect  0.003796 0.0104 66   0.365  0.7163
##  2 effect -0.003796 0.0104 66  -0.365  0.7163
##
## condition = c:
##  contrast  estimate     SE df t.ratio p.value
##  1 effect  0.018284 0.0104 66   1.758  0.0834
##  2 effect -0.018284 0.0104 66  -1.758  0.0834
##
## Degrees-of-freedom method: kenward-roger
## P value adjustment: fdr method for 2 tests
```

```
# nr
contrast( emmeans(vr.models[[3]],  ~ time | condition))
```

```
## condition = b:
##  contrast estimate     SE df t.ratio p.value
##  1 effect  -0.0455 0.0299 66  -1.521  0.1330
##  2 effect   0.0455 0.0299 66   1.521  0.1330
##
## condition = a:
##  contrast estimate     SE df t.ratio p.value
##  1 effect  -0.0569 0.0299 66  -1.901  0.0616
##  2 effect   0.0569 0.0299 66   1.901  0.0616
##
## condition = c:
##  contrast estimate     SE df t.ratio p.value
##  1 effect  -0.0124 0.0299 66  -0.415  0.6796
##  2 effect   0.0124 0.0299 66   0.415  0.6796
##
## Degrees-of-freedom method: kenward-roger
## P value adjustment: fdr method for 2 tests
```

```
# nep
contrast( emmeans(vr.models[[4]],  ~ time | condition))
```

```
## condition = b:
##  contrast estimate     SE df t.ratio p.value
##  1 effect  -0.0580 0.0273 66  -2.121  0.0377
##  2 effect   0.0580 0.0273 66   2.121  0.0377
##
## condition = a:
##  contrast estimate     SE df t.ratio p.value
##  1 effect  -0.0623 0.0273 66  -2.280  0.0258
##  2 effect   0.0623 0.0273 66   2.280  0.0258
##
## condition = c:
##  contrast estimate     SE df t.ratio p.value
##  1 effect  -0.0188 0.0273 66  -0.689  0.4931
##  2 effect   0.0188 0.0273 66   0.689  0.4931
##
## Degrees-of-freedom method: kenward-roger
## P value adjustment: fdr method for 2 tests
```

Model comparisons For the three VE conditions we need ot test the predictor `condition` with model comparisons.

```
compare.models <- function(model){
  model0 <- update(model, .~. - time:condition)
  anova(model0, model)
}

lapply(vr.models, compare.models)
```

```
## refitting model(s) with ML (instead of REML)
## refitting model(s) with ML (instead of REML)
## refitting model(s) with ML (instead of REML)
## refitting model(s) with ML (instead of REML)
## refitting model(s) with ML (instead of REML)
```

```
## [[1]]
## Data: dat
## Models:
## model0: iat ~ condition + time + (1 | id)
## model: iat ~ condition * time + (1 | id)
##        npar    AIC    BIC  logLik deviance  Chisq Df Pr(>Chisq)
## model0    6 168.31 185.87 -78.155   156.31
## model     8 171.31 194.73 -77.655   155.31 1.0001  2     0.6065
##
## [[2]]
## Data: dat
## Models:
## model0: ccs_bc ~ condition + time + (1 | id)
## model: ccs_bc ~ condition * time + (1 | id)
##        npar     AIC     BIC logLik deviance  Chisq Df Pr(>Chisq)
## model0    6 -172.93 -155.37 92.464  -184.93
## model     8 -170.74 -147.32 93.368  -186.74 1.8068  2     0.4052
##
## [[3]]
## Data: dat
## Models:
## model0: nr ~ condition + time + (1 | id)
## model: nr ~ condition * time + (1 | id)
##        npar    AIC    BIC  logLik deviance  Chisq Df Pr(>Chisq)
## model0    6 90.456 108.02 -39.228   78.456
## model     8 93.220 116.64 -38.610   77.220 1.2358  2     0.5391
##
## [[4]]
## Data: dat
## Models:
## model0: nep ~ condition + time + (1 | id)
## model: nep ~ condition * time + (1 | id)
##        npar    AIC    BIC  logLik deviance  Chisq Df Pr(>Chisq)
## model0    6 85.962 103.53 -36.981   73.962
## model     8 88.376 111.79 -36.188   72.376 1.5864  2     0.4524
##
## [[5]]
## Data: dat
## Models:
```

```
## model0: env_pc ~ condition + time + (1 | id)
## model: env_pc ~ condition * time + (1 | id)
##        npar    AIC    BIC  logLik deviance  Chisq Df Pr(>Chisq)
## model0    6 371.07 388.63 -179.53   359.07
## model     8 374.64 398.06 -179.32   358.64 0.4205  2     0.8104
```

The condition:time interaction did not significantly add to the explained variance.

Presence & SOD

```
lapply(presence.models, FUN = lmerTest:::summary.lmerModLmerTest)
```

```
## Coercing object to class 'lmerModLmerTest'
## Coercing object to class 'lmerModLmerTest'
## Coercing object to class 'lmerModLmerTest'
## Coercing object to class 'lmerModLmerTest'
## Coercing object to class 'lmerModLmerTest'
```

```
## [[1]]
## Linear mixed model fit by REML. t-tests use Satterthwaite's method [
## lmerModLmerTest]
## Formula: iat ~ ipq * time + (1 | id)
##    Data: dat
##
## REML criterion at convergence: 163.6
##
## Scaled residuals:
##      Min       1Q   Median       3Q      Max
## -2.69457 -0.54823 -0.02948  0.53493  2.01632
##
## Random effects:
##  Groups   Name        Variance Std.Dev.
##  id       (Intercept) 0.1131   0.3363
##  Residual             0.1080   0.3286
## Number of obs: 130, groups:  id, 65
##
## Fixed effects:
##              Estimate Std. Error        df t value Pr(>|t|)
## (Intercept)  -0.58774    0.66781 101.74722  -0.880    0.381
## ipq           0.14043    0.14134 101.74722   0.994    0.323
## time         -0.09185    0.38404  63.00001  -0.239    0.812
## ipq:time      0.03446    0.08128  63.00001   0.424    0.673
##
## Correlation of Fixed Effects:
##          (Intr) ipq    time
## ipq      -0.989
## time     -0.863  0.853
## ipq:time  0.853 -0.863 -0.989
##
## [[2]]
## Linear mixed model fit by REML. t-tests use Satterthwaite's method [
## lmerModLmerTest]
## Formula: ccs ~ ipq * time + (1 | id)
##    Data: dat
##
```

```
## REML criterion at convergence: 120.7
##
## Scaled residuals:
##     Min      1Q  Median      3Q     Max
## -1.7904 -0.3820 -0.1145  0.3312  2.1549
##
## Random effects:
##  Groups   Name        Variance Std.Dev.
##  id       (Intercept) 0.22587  0.4753
##  Residual             0.03731  0.1931
## Number of obs: 130, groups:  id, 65
##
## Fixed effects:
##             Estimate Std. Error        df t value Pr(>|t|)
## (Intercept)   2.11894    0.53069 121.79391   3.993 0.000112 ***
## ipq          -0.14045    0.11232 121.79391  -1.251 0.213506
## time         -0.10019    0.22573  63.00000  -0.444 0.658668
## ipq:time      0.01788    0.04777  63.00000   0.374 0.709468
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Correlation of Fixed Effects:
##          (Intr) ipq    time
## ipq      -0.989
## time     -0.638  0.631
## ipq:time  0.631 -0.638 -0.989
##
## [[3]]
## Linear mixed model fit by REML. t-tests use Satterthwaite's method [
## lmerModLmerTest]
## Formula: nr ~ ipq * time + (1 | id)
##    Data: dat
##
## REML criterion at convergence: 104
##
## Scaled residuals:
##      Min       1Q   Median       3Q      Max
## -1.95321 -0.55273 -0.02821  0.50567  1.89383
##
## Random effects:
##  Groups   Name        Variance Std.Dev.
##  id       (Intercept) 0.14358  0.3789
##  Residual             0.04248  0.2061
## Number of obs: 130, groups:  id, 65
##
## Fixed effects:
##             Estimate Std. Error        df t value Pr(>|t|)
## (Intercept)   3.63662    0.49306 125.31336   7.376 1.96e-11 ***
## ipq           0.03827    0.10435 125.31336   0.367    0.714
## time         -0.08722    0.24088  63.00001  -0.362    0.718
## ipq:time      0.03467    0.05098  63.00001   0.680    0.499
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
```

```
## Correlation of Fixed Effects:
##          (Intr) ipq    time
## ipq      -0.989
## time     -0.733  0.725
## ipq:time  0.725 -0.733 -0.989
##
## [[4]]
## Linear mixed model fit by REML. t-tests use Satterthwaite's method [
## lmerModLmerTest]
## Formula: nep ~ ipq * time + (1 | id)
##    Data: dat
##
## REML criterion at convergence: 85.7
##
## Scaled residuals:
##      Min       1Q   Median       3Q      Max
## -1.41089 -0.54793 -0.05323  0.49528  1.82764
##
## Random effects:
##  Groups   Name        Variance Std.Dev.
##  id       (Intercept) 0.13287  0.3645
##  Residual             0.03485  0.1867
## Number of obs: 130, groups:  id, 65
##
## Fixed effects:
##              Estimate Std. Error        df t value Pr(>|t|)
## (Intercept)   3.45098    0.45794 125.94330   7.536  8.3e-12 ***
## ipq           0.05561    0.09692 125.94330   0.574    0.567
## time          0.01510    0.21816  63.00000   0.069    0.945
## ipq:time      0.01762    0.04617  63.00000   0.382    0.704
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Correlation of Fixed Effects:
##          (Intr) ipq    time
## ipq      -0.989
## time     -0.715  0.706
## ipq:time  0.706 -0.715 -0.989
##
## [[5]]
## Linear mixed model fit by REML. t-tests use Satterthwaite's method [
## lmerModLmerTest]
## Formula: env_pc ~ ipq * time + (1 | id)
##    Data: dat
##
## REML criterion at convergence: 348.5
##
## Scaled residuals:
##      Min       1Q   Median       3Q      Max
## -1.70780 -0.41684 -0.02183  0.43990  1.79021
##
## Random effects:
##  Groups   Name        Variance Std.Dev.
##  id       (Intercept) 1.210    1.100
```

```
##  Residual             0.254    0.504
## Number of obs: 130, groups:  id, 65
##
## Fixed effects:
##              Estimate Std. Error        df t value Pr(>|t|)
## (Intercept)  -2.05376    1.30141 125.23350  -1.578    0.117
## ipq           0.35511    0.27543 125.23350   1.289    0.200
## time         -0.01667    0.58900  62.99999  -0.028    0.978
## ipq:time      0.06329    0.12466  62.99999   0.508    0.613
##
## Correlation of Fixed Effects:
##          (Intr) ipq    time
## ipq      -0.989
## time     -0.679  0.671
## ipq:time  0.671 -0.679 -0.989
```

```
lapply(sod.models, FUN = lmerTest:::summary.lmerModLmerTest)
```

```
## Coercing object to class 'lmerModLmerTest'
## Coercing object to class 'lmerModLmerTest'
## Coercing object to class 'lmerModLmerTest'
## Coercing object to class 'lmerModLmerTest'
## Coercing object to class 'lmerModLmerTest'
```

```
## [[1]]
## Linear mixed model fit by REML. t-tests use Satterthwaite's method [
## lmerModLmerTest]
## Formula: iat ~ sod * time + (1 | id)
##    Data: dat
##
## REML criterion at convergence: 176.3
##
## Scaled residuals:
##      Min       1Q   Median       3Q      Max
## -2.80798 -0.56967  0.02637  0.60711  1.76928
##
## Random effects:
##  Groups   Name        Variance Std.Dev.
##  id       (Intercept) 0.1244   0.3528
##  Residual             0.1045   0.3233
## Number of obs: 138, groups:  id, 69
##
## Fixed effects:
##               Estimate Std. Error         df t value Pr(>|t|)
## (Intercept)   0.109157   0.481524 111.121118   0.227    0.821
## sod          -0.006195   0.094734 111.121118  -0.065    0.948
## time         -0.254809   0.273689  67.000002  -0.931    0.355
## sod:time      0.064295   0.053845  67.000002   1.194    0.237
##
## Correlation of Fixed Effects:
##          (Intr) sod    time
## sod      -0.980
## time     -0.853  0.835
## sod:time  0.835 -0.853 -0.980
##
```

```
## [[2]]
## Linear mixed model fit by REML. t-tests use Satterthwaite's method [
## lmerModLmerTest]
## Formula: ccs ~ sod * time + (1 | id)
##    Data: dat
##
## REML criterion at convergence: 135.9
##
## Scaled residuals:
##      Min       1Q   Median       3Q      Max
## -1.73181 -0.38435 -0.09369  0.32377  2.19186
##
## Random effects:
##  Groups   Name        Variance Std.Dev.
##  id       (Intercept) 0.22349  0.4727
##  Residual             0.04139  0.2034
## Number of obs: 138, groups:  id, 69
##
## Fixed effects:
##              Estimate Std. Error        df t value Pr(>|t|)
## (Intercept)   1.65984    0.39278 131.61473   4.226 4.42e-05 ***
## sod          -0.04011    0.07727 131.61473  -0.519    0.605
## time          0.07598    0.17224  67.00001   0.441    0.661
## sod:time     -0.01672    0.03389  67.00001  -0.493    0.623
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Correlation of Fixed Effects:
##          (Intr) sod    time
## sod      -0.980
## time     -0.658  0.644
## sod:time  0.644 -0.658 -0.980
##
## [[3]]
## Linear mixed model fit by REML. t-tests use Satterthwaite's method [
## lmerModLmerTest]
## Formula: nr ~ sod * time + (1 | id)
##    Data: dat
##
## REML criterion at convergence: 109.1
##
## Scaled residuals:
##      Min       1Q   Median       3Q      Max
## -1.91062 -0.53432 -0.01933  0.52449  1.83663
##
## Random effects:
##  Groups   Name        Variance Std.Dev.
##  id       (Intercept) 0.14449  0.3801
##  Residual             0.04107  0.2027
## Number of obs: 138, groups:  id, 69
##
## Fixed effects:
##              Estimate Std. Error        df t value Pr(>|t|)
## (Intercept)   3.98028    0.35411 133.57257  11.240   <2e-16 ***
```

```
## sod          -0.03485    0.06967 133.57257  -0.500    0.618
## time         -0.03995    0.17159  67.00000  -0.233    0.817
## sod:time      0.02341    0.03376  67.00000   0.693    0.490
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Correlation of Fixed Effects:
##          (Intr) sod    time
## sod      -0.980
## time     -0.727  0.712
## sod:time  0.712 -0.727 -0.980
##
## [[4]]
## Linear mixed model fit by REML. t-tests use Satterthwaite's method [
## lmerModLmerTest]
## Formula: nep ~ sod * time + (1 | id)
##    Data: dat
##
## REML criterion at convergence: 91.4
##
## Scaled residuals:
##      Min       1Q   Median       3Q      Max
## -1.46069 -0.58390 -0.04105  0.50130  1.89785
##
## Random effects:
##  Groups   Name        Variance Std.Dev.
##  id       (Intercept) 0.13382  0.3658
##  Residual             0.03445  0.1856
## Number of obs: 138, groups:  id, 69
##
## Fixed effects:
##              Estimate Std. Error        df t value Pr(>|t|)
## (Intercept)   3.33827    0.33121 133.97744  10.079   <2e-16 ***
## sod           0.07682    0.06516 133.97744   1.179    0.241
## time          0.18737    0.15714  67.00000   1.192    0.237
## sod:time     -0.01900    0.03092  67.00000  -0.615    0.541
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Correlation of Fixed Effects:
##          (Intr) sod    time
## sod      -0.980
## time     -0.712  0.697
## sod:time  0.697 -0.712 -0.980
##
## [[5]]
## Linear mixed model fit by REML. t-tests use Satterthwaite's method [
## lmerModLmerTest]
## Formula: env_pc ~ sod * time + (1 | id)
##    Data: dat
##
## REML criterion at convergence: 371.8
##
## Scaled residuals:
```

```
##      Min       1Q   Median       3Q      Max
## -1.76166 -0.43974  0.00888  0.44844  1.81302
##
## Random effects:
##  Groups   Name        Variance Std.Dev.
##  id       (Intercept) 1.2802   1.1315
##  Residual             0.2437   0.4937
## Number of obs: 138, groups:  id, 69
##
## Fixed effects:
##              Estimate Std. Error         df t value Pr(>|t|)
## (Intercept)  -0.956276   0.946384 132.028220  -1.010    0.314
## sod           0.115718   0.186190 132.028220   0.622    0.535
## time         -0.001958   0.417982  66.999999  -0.005    0.996
## sod:time      0.052956   0.082233  67.000000   0.644    0.522
##
## Correlation of Fixed Effects:
##          (Intr) sod    time
## sod      -0.980
## time     -0.662  0.649
## sod:time  0.649 -0.662 -0.980
```

Export

Correlation table

```
lower <- round(cor(data[dvs[-5]]),2)
lower[upper.tri(lower)] <- ""
lower <- as.data.frame(lower)
rownames(lower) <- toupper(rownames(lower))
colnames(lower) <- toupper(colnames(lower))
print(xtable(lower, label = "tab:cortable",
       caption = "Correlation table of the four environmental attitudes measures. Computed on the untran
```

Tables for models

First, we define a couple of helping functions

```
#lmermods = all.models
#  x <- lmermods[[1]]

make.x.table <- function(lmermods, save = TRUE, add = ""){
  #lapply(lmermods, function(x) formula(x)[2])
  dvs <- as.character(lapply(lmermods, function(x) as.character(formula(x)[2])))
  iv <- sub("\\ .*", "", lmermods[[1]]@call$formula[3])


  helpf <- function(x){
    # get fixed effects table
    fix.temp <- round(as.data.frame(lmerTest:::summary.lmerModLmerTest(x)$coefficients), 3)

    # calculate and round confidence intervals
    conf.temp <- as.data.frame(confint(x, method = "profile"))
    conf.temp <- round(conf.temp, 3)
```

```
    conf.temp$CI <- paste("[", conf.temp[,1], "; ", conf.temp[,2],"]", sep = "")

    # put together
    fix.temp$CI <- conf.temp$CI[!startsWith(rownames(conf.temp), ".")]

   fix.temp["DV"] <- ""
   #fix.temp[paste(dv, " ~", sep = "")][1,] <-paste(formula(x))[3]
   fix.temp["DV"][1,] <- "deleteme"

    fix.temp <- rownames_to_column(fix.temp, var = "Coef")
    fix.temp <- fix.temp[, c(8, 1, 2, 7, 5, 4, 6)]
    colnames(fix.temp)[7] <- "p value"

    # return value
    return(fix.temp)
  }



  # confidence intervals:
  fixed.effects.list <- lapply(lmermods, FUN = helpf)



  fixed.effects.df <- do.call("rbind", fixed.effects.list)
  fixed.effects.df$DV[fixed.effects.df$DV=="deleteme"] <- dvs

  #l.temp <- length(rownames(fixed.effects.df))
  #cond <- toupper(strsplit(rownames(fixed.effects.df)[l.temp], split = " ")[[1]][5])

  rownames(fixed.effects.df) <- NULL
  tabl <- xtable(fixed.effects.df,
                 caption = paste(toupper(iv), "Models",  add),
                 label = paste("tab:",add , iv, "-models", sep = ""))

  if(save){

  print.xtable(tabl, file = paste("analysisOutputs/tables/", add, iv, "-modeltable.tex", sep = ""),
               include.rownames=FALSE,
               hline.after = c(-1, c(which(fixed.effects.df[1]!="")-1), nrow(tabl))#,
               #add.to.row = list(list(-1), c(paste("\\hspace{-3mm}", toupper(dv), "$\\sim$%")))
               )
  }else{
  print.xtable(tabl, #file = paste("analysisOutputs/tables/", add, dv, "-by", cond, "-modeltable.tex", s
               include.rownames=FALSE,
               hline.after = c(-1, c(which(fixed.effects.df[1]!="")-1), nrow(tabl))#,
               #add.to.row = list(list(-1), c(paste("\\hspace{-3mm}", toupper(dv), "$\\sim$%")))
               )}

  # which(fixed.effects.df[2]!="")-1
  # this saves directly into my .tex folder
}
```

Then we create and save the tables:

```r
make.x.table(all.models[-5], save = FALSE)
```

```
## Coercing object to class 'lmerModLmerTest'

## Computing profile confidence intervals ...

## Coercing object to class 'lmerModLmerTest'

## Computing profile confidence intervals ...

## Coercing object to class 'lmerModLmerTest'

## Computing profile confidence intervals ...

## Coercing object to class 'lmerModLmerTest'

## Computing profile confidence intervals ...

## Warning in nextpar(mat, cc, i, delta, lowcut, upcut): unexpected decrease in
## profile: using minstep

## Warning in FUN(X[[i]], ...): non-monotonic profile for .sig02

## Warning in confint.thpr(pp, level = level, zeta = zeta): bad spline fit
## for .sig02: falling back to linear interpolation

## % latex table generated in R 4.1.2 by xtable 1.8-4 package
## % Mon Feb 28 14:15:24 2022
## \begin{table}[ht]
## \centering
## \begin{tabular}{llrlrrr}
##   \hline
## DV & Coef & Estimate & CI & t value & df & p value \\
##   \hline
## iat & (Intercept) & 0.23 & [0.106; 0.355] & 3.63 & 225.16 & 0.00 \\
##    & time & -0.01 & [-0.085; 0.066] & -0.23 & 20.40 & 0.82 \\
##    & type1 & -0.15 & [-0.276; -0.028] & -2.39 & 225.16 & 0.02 \\
##    & time:type1 & 0.07 & [-0.001; 0.15] & 1.83 & 20.40 & 0.08 \\
##    \hline
## ccs\_bc & (Intercept) & 0.23 & [0.199; 0.27] & 12.84 & 278.36 & 0.00 \\
##    & time & -0.01 & [-0.029; 0.006] & -1.32 & 140.00 & 0.19 \\
##    & type1 & -0.00 & [-0.038; 0.034] & -0.12 & 278.36 & 0.91 \\
##    & time:type1 & -0.00 & [-0.02; 0.015] & -0.32 & 140.00 & 0.75 \\
##    \hline
## nr & (Intercept) & 3.82 & [3.71; 3.925] & 69.64 & 268.10 & 0.00 \\
##    & time & 0.04 & [-0.009; 0.085] & 1.51 & 9.57 & 0.16 \\
##    & type1 & -0.01 & [-0.118; 0.096] & -0.20 & 268.10 & 0.84 \\
##    & time:type1 & 0.04 & [-0.008; 0.086] & 1.56 & 9.57 & 0.15 \\
##    \hline
## nep & (Intercept) & 3.76 & [3.663; 3.849] & 79.18 & 276.21 & 0.00 \\
##    & time & 0.07 & [0.03; 0.117] & 3.23 & 14.89 & 0.01 \\
##    & type1 & -0.04 & [-0.128; 0.058] & -0.74 & 276.21 & 0.46 \\
##    & time:type1 & 0.02 & [-0.025; 0.063] & 0.83 & 14.89 & 0.42 \\
##    \hline
## \end{tabular}
## \caption{TIME Models }
## \label{tab:time-models}
## \end{table}
```

```r
make.x.table(vr.models[-5], save = FALSE)
```

```
## Coercing object to class 'lmerModLmerTest'
## Computing profile confidence intervals ...

## Coercing object to class 'lmerModLmerTest'

## Computing profile confidence intervals ...

## Coercing object to class 'lmerModLmerTest'

## Computing profile confidence intervals ...

## Coercing object to class 'lmerModLmerTest'

## Computing profile confidence intervals ...

## % latex table generated in R 4.1.2 by xtable 1.8-4 package
## % Mon Feb 28 14:15:26 2022
## \begin{table}[ht]
## \centering
## \begin{tabular}{llrlrrr}
##   \hline
## DV & Coef & Estimate & CI & t value & df & p value \\
##   \hline
## iat & (Intercept) & 0.08 & [-0.109; 0.266] & 0.81 & 106.86 & 0.42 \\
##    & condition1 & 0.10 & [-0.166; 0.364] & 0.72 & 106.86 & 0.47 \\
##    & condition2 & -0.23 & [-0.491; 0.039] & -1.65 & 106.86 & 0.10 \\
##    & time & 0.07 & [-0.043; 0.173] & 1.17 & 66.00 & 0.24 \\
##    & condition1:time & -0.08 & [-0.23; 0.076] & -0.98 & 66.00 & 0.33 \\
##    & condition2:time & 0.04 & [-0.114; 0.192] & 0.50 & 66.00 & 0.62 \\
##    \hline
## ccs\_bc & (Intercept) & 0.23 & [0.181; 0.283] & 8.84 & 131.56 & 0.00 \\
##    & condition1 & -0.06 & [-0.129; 0.014] & -1.54 & 131.56 & 0.12 \\
##    & condition2 & 0.01 & [-0.057; 0.086] & 0.38 & 131.56 & 0.70 \\
##    & time & -0.01 & [-0.038; 0.009] & -1.21 & 66.00 & 0.23 \\
##    & condition1:time & 0.01 & [-0.018; 0.048] & 0.88 & 66.00 & 0.38 \\
##    & condition2:time & 0.01 & [-0.026; 0.04] & 0.41 & 66.00 & 0.68 \\
##    \hline
## nr & (Intercept) & 3.81 & [3.674; 3.939] & 55.49 & 129.34 & 0.00 \\
##    & condition1 & 0.10 & [-0.091; 0.283] & 0.99 & 129.34 & 0.33 \\
##    & condition2 & -0.29 & [-0.476; -0.102] & -2.98 & 129.34 & 0.00 \\
##    & time & 0.08 & [0.009; 0.144] & 2.21 & 66.00 & 0.03 \\
##    & condition1:time & 0.01 & [-0.081; 0.11] & 0.30 & 66.00 & 0.77 \\
##    & condition2:time & 0.04 & [-0.058; 0.132] & 0.76 & 66.00 & 0.45 \\
##    \hline
## nep & (Intercept) & 3.72 & [3.591; 3.85] & 55.49 & 132.00 & 0.00 \\
##    & condition1 & -0.04 & [-0.228; 0.138] & -0.48 & 132.00 & 0.63 \\
##    & condition2 & -0.06 & [-0.243; 0.123] & -0.63 & 132.00 & 0.53 \\
##    & time & 0.09 & [0.031; 0.154] & 2.94 & 66.00 & 0.01 \\
##    & condition1:time & 0.02 & [-0.064; 0.11] & 0.52 & 66.00 & 0.60 \\
##    & condition2:time & 0.03 & [-0.055; 0.119] & 0.71 & 66.00 & 0.48 \\
##    \hline
## \end{tabular}
## \caption{CONDITION Models }
## \label{tab:condition-models}
## \end{table}
```

```
make.x.table(all.models[-5], save = TRUE)
```

```
## Coercing object to class 'lmerModLmerTest'

## Computing profile confidence intervals ...

## Coercing object to class 'lmerModLmerTest'

## Computing profile confidence intervals ...

## Coercing object to class 'lmerModLmerTest'

## Computing profile confidence intervals ...

## Coercing object to class 'lmerModLmerTest'

## Computing profile confidence intervals ...

## Warning in nextpar(mat, cc, i, delta, lowcut, upcut): unexpected decrease in
## profile: using minstep

## Warning in FUN(X[[i]], ...): non-monotonic profile for .sig02

## Warning in confint.thpr(pp, level = level, zeta = zeta): bad spline fit
## for .sig02: falling back to linear interpolation
```

```
make.x.table(vr.models[-5], save = TRUE)
```

```
## Coercing object to class 'lmerModLmerTest'
## Computing profile confidence intervals ...

## Coercing object to class 'lmerModLmerTest'

## Computing profile confidence intervals ...

## Coercing object to class 'lmerModLmerTest'

## Computing profile confidence intervals ...

## Coercing object to class 'lmerModLmerTest'

## Computing profile confidence intervals ...
```

Plots

I would like to get a table / data structure that gives me the mean values for each combination of relevant time and condition ### CI plot helping functions

```
ci.plot <- function(colorby = "condition"){
#colorby = "condition"
#df <- vr.res.md.comp.mean
  pos <- ifelse(colorby=="condition", "dodge2", "identity")

  function(df){
    df$cond <- df[[colorby]]

  if(colorby != "condition"){
    df <- df %>% dplyr::filter(condition=="a"|condition=="text" )

  }

    ggplot(df, aes(x = time, y = ml.value, color = cond)) +
      facet_wrap(~dv, scales = "free") +
```

```
      geom_line(position = position_jitterdodge(dodge.width = 0.2, jitter.width = 0, jitter.height = 0)
      geom_point(position = position_jitterdodge(dodge.width = 0.2, jitter.width = 0, jitter.height = 0

      geom_errorbar(aes(ymin = df$`2.5 %`, ymax = df$`97.5 %`), position = "dodge2", width = 0.2) +
      ggthemes::theme_tufte() +
      ylab("scale value") +

      scale_x_discrete() +
      xlim(c("before", "after"))
    }
}

ci.plot.vr <- ci.plot("type")
ci.plot.condition <- ci.plot("condition")

#ci.plot.condition(vr.res.md.comp.mean)
#ci.plot.vr(vr.nonvr.mean)
```

```
cond.mean.ci <- function(data.temp){
  predict.fun <- function(my.lmm) {
   # my.lmm <- md
  #data.temp <- df.predicted.vr
    predict(my.lmm, newdata = data.temp, re.form = NA)   # This is predict.merMod
    #data.temp$x <- x
    #data.temp$y <- rep(x[data.temp$time==1]-x[data.temp$time==2], each = 2)
    #y <- x
  }

  function(md){
    data.temp$ml.value <- predict.fun(md)
    # Make predictions in 100 bootstraps of the LMM. Use these to get confidence
    # intervals.
    dv <- as.character(formula(md))[2]
    lmm.boots <- bootMer(md, predict.fun, nsim = 1000,
                         #parallel = "multicore",
                         ncpus = (detectCores(all.tests = FALSE, logical = TRUE)-1),
                         type = "semiparametric",
                         use.u = T)

    data.temp <- cbind("dv" = dv, data.temp, confint(lmm.boots))
    return(data.temp)
  }
}
```

```
df.predicted.all <- data.frame(
  id = rep(as.factor(1:6), each = 2),
  time = rep(1:2, times = 6),
  condition = rep(levels(as.factor(data$condition)), each = 2),
  vr = rep(c(TRUE, FALSE), each = 6),
  type = rep(c("vr", "control"), each = 6)
)

df.predicted.vr <- df.predicted.all %>% filter(vr)
```

```
## filter: removed 6 rows (50%), 6 rows remaining
```

Create function for each option:

```r
cond.mean.ci.all <- cond.mean.ci(data = df.predicted.all)
cond.mean.ci.vr <- cond.mean.ci(data = df.predicted.vr)
```

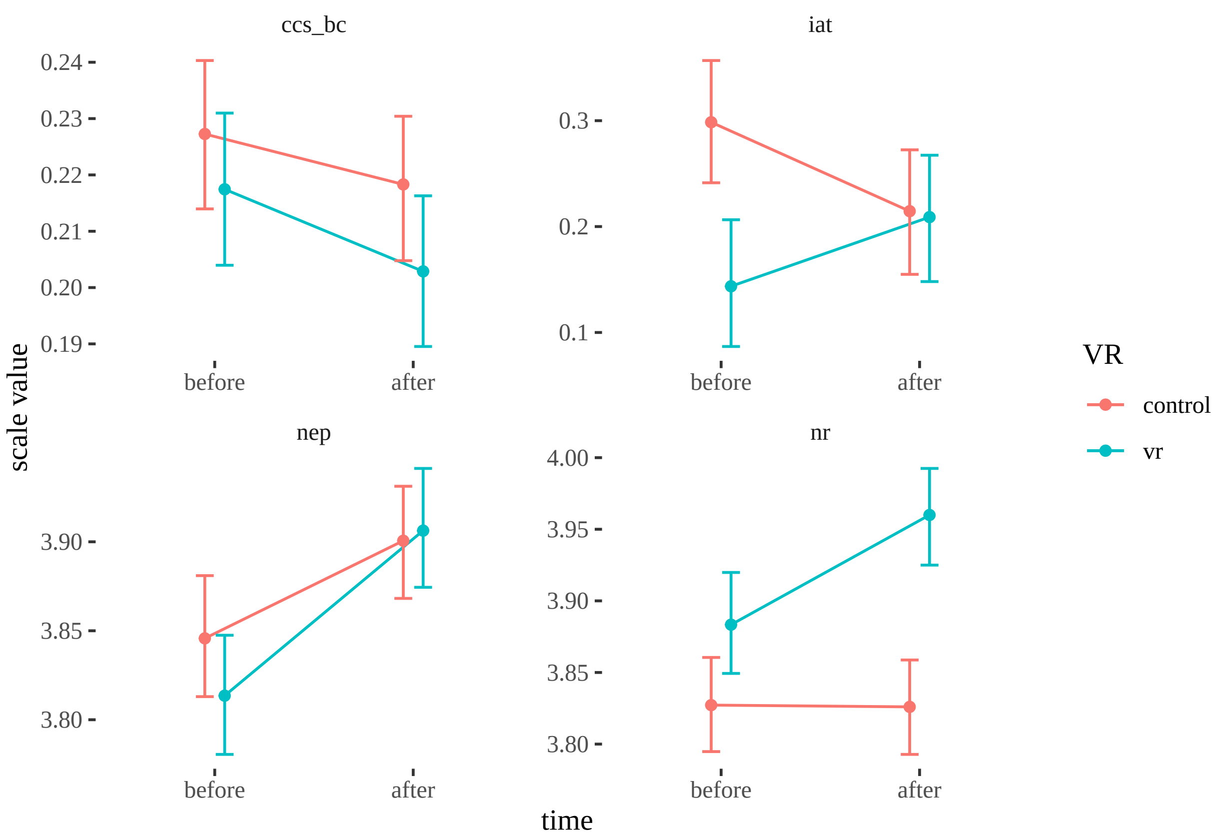
Creating the plots

First comparing VE to control conditions

```r
# Bootstrap CI's
vr.nonvr.mean.list <- lapply(all.models[1:4], cond.mean.ci.all)
vr.nonvr.mean <- do.call("rbind", vr.nonvr.mean.list)
```

```r
(vr.ci.plot <- ci.plot.vr(vr.nonvr.mean) +
  guides(color=guide_legend(title="VR")))
```

```
## Scale for 'x' is already present. Adding another scale for 'x', which will
## replace the existing scale.

## Warning: Use of `df$`2.5 %`` is discouraged. Use `2.5 %` instead.

## Warning: Use of `df$`97.5 %`` is discouraged. Use `97.5 %` instead.
```

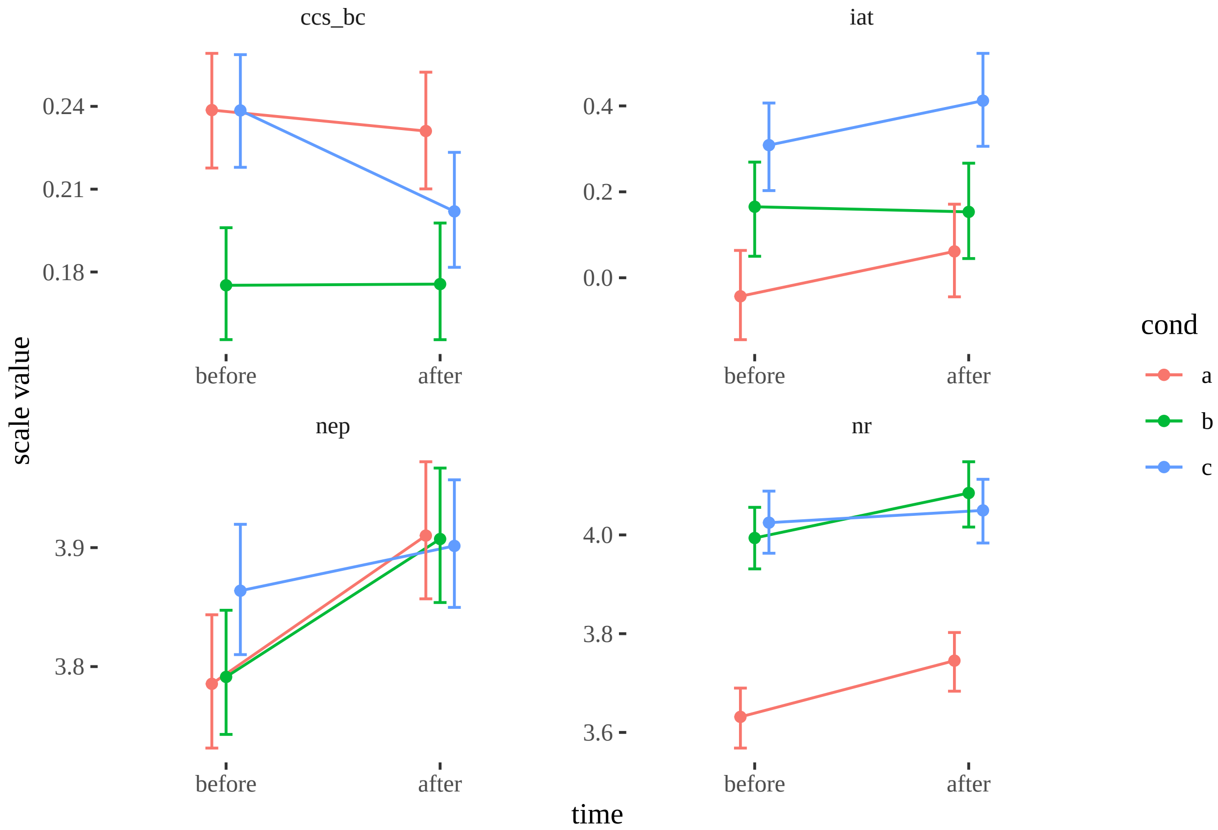


Then for only between the VE conditions:

```r
onlyvr.cond.mean.list <- lapply(vr.models[1:4], cond.mean.ci.vr)
onlyvr.cond.mean <- do.call("rbind", onlyvr.cond.mean.list)
```

```r
(cond.ci.plot <- ci.plot.condition(onlyvr.cond.mean))
```

```
## Scale for 'x' is already present. Adding another scale for 'x', which will
## replace the existing scale.

## Warning: Use of `df$`2.5 %`` is discouraged. Use `2.5 %` instead.

## Warning: Use of `df$`97.5 %`` is discouraged. Use `97.5 %` instead.
```



backtransform ccs

```
reverse <- function(lambda){

  function(x){
    exp(log(lambda * x + 1) / lambda)
  }
}


reverse_ccs <- reverse(lambda = lambda_ccs)
```

```
which.ccs <- which(onlyvr.cond.mean$dv=="ccs_bc")
which.ccs2 <- which(vr.nonvr.mean$dv=="ccs_bc")

vr.nonvr.mean2 <- vr.nonvr.mean
vr.nonvr.mean2[which.ccs2,c("ml.value", "2.5 %","97.5 %")] <- reverse_ccs(vr.nonvr.mean[which.ccs2,c("ml


onlyvr.cond.mean2 <- onlyvr.cond.mean
onlyvr.cond.mean2[which.ccs,c("ml.value", "2.5 %","97.5 %")] <- reverse_ccs(onlyvr.cond.mean[which.ccs,c
```

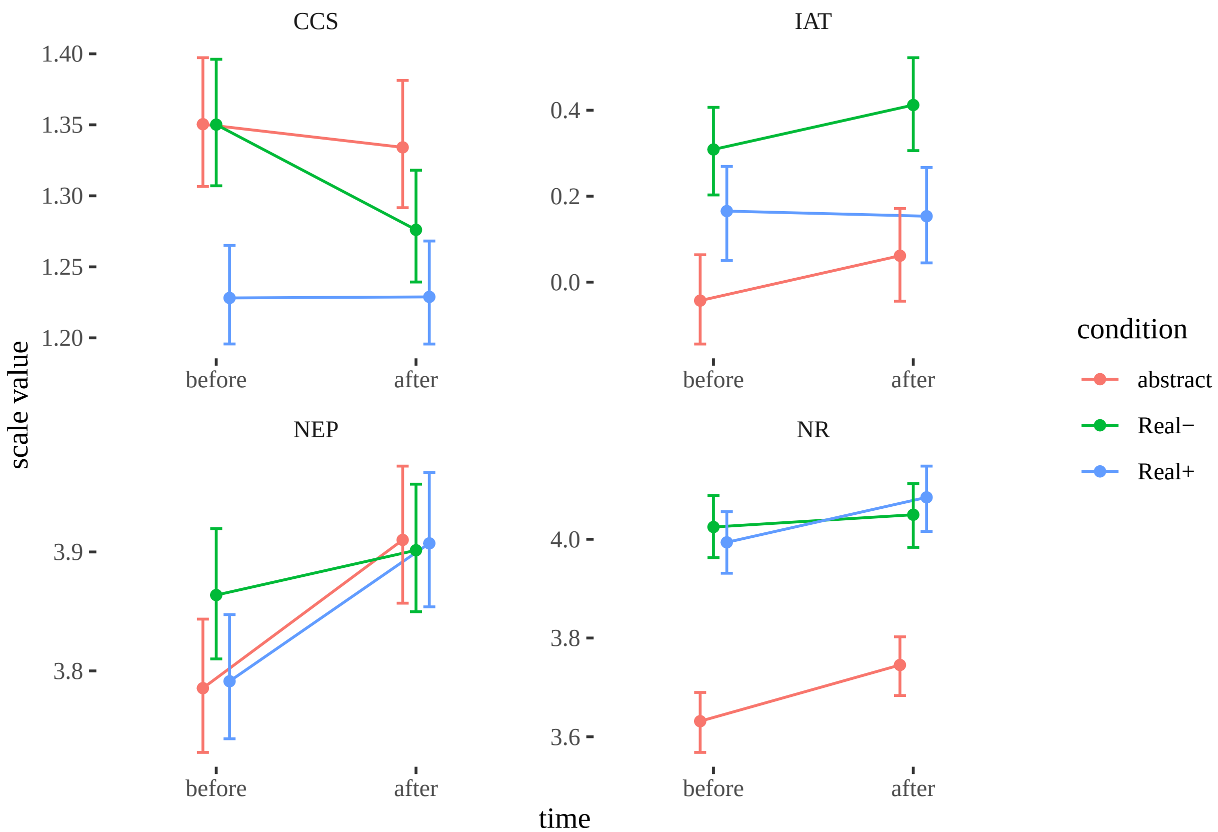
```r
# change dv names:
onlyvr.cond.mean2$dv[onlyvr.cond.mean2$dv=="ccs_bc"] <- "ccs"
vr.nonvr.mean2$dv[vr.nonvr.mean2$dv=="ccs_bc"] <- "ccs"


onlyvr.cond.mean2$dv <- toupper(onlyvr.cond.mean2$dv)
vr.nonvr.mean2$dv   <- toupper(vr.nonvr.mean2$dv)

# change condition names:
onlyvr.cond.mean2 <-tibble( onlyvr.cond.mean2 ) %>%
  dplyr::mutate(condition = ifelse(condition =="b", "Real+",
                            ifelse(condition == "c", "Real-",
                                ifelse(condition == "a", "abstract", condition))))
```

```r
(cond.ci.plot2 <- ci.plot.condition(onlyvr.cond.mean2)+
  guides(color=guide_legend(title="condition")))
```

```
## Scale for 'x' is already present. Adding another scale for 'x', which will
## replace the existing scale.
```

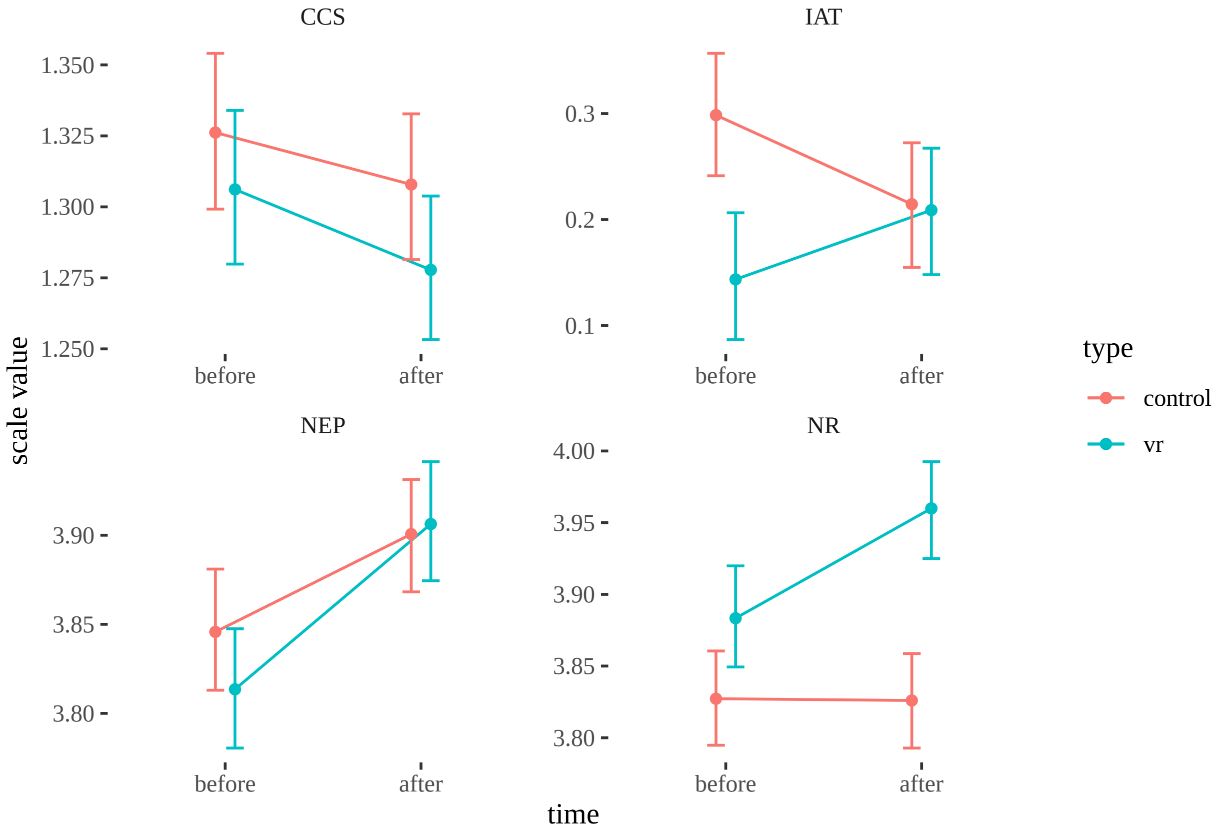
```
## Warning: Use of `df$`2.5 %`` is discouraged. Use `2.5 %` instead.
```

```
## Warning: Use of `df$`97.5 %`` is discouraged. Use `97.5 %` instead.
```




```r
(vr.ci.plot2 <- ci.plot.vr(vr.nonvr.mean2) +
  guides(color=guide_legend(title="type")))
```

```
## Scale for 'x' is already present. Adding another scale for 'x', which will
## replace the existing scale.

## Warning: Use of `df$`2.5 %`` is discouraged. Use `2.5 %` instead.

## Warning: Use of `df$`97.5 %`` is discouraged. Use `97.5 %` instead.
```



save plots

```
pdf(file = "analysisOutputs/plots/vr_ci_plot2.pdf", width = 7, height = 5)
vr.ci.plot2
```

```
## Warning: Use of `df$`2.5 %`` is discouraged. Use `2.5 %` instead.

## Warning: Use of `df$`97.5 %`` is discouraged. Use `97.5 %` instead.
```

```
dev.off()
```

```
## pdf
##   2
```

```
pdf(file = "analysisOutputs/plots/cond_ci_plot2.pdf", width = 7, height = 5)
cond.ci.plot2
```

```
## Warning: Use of `df$`2.5 %`` is discouraged. Use `2.5 %` instead.

## Warning: Use of `df$`97.5 %`` is discouraged. Use `97.5 %` instead.
```

```
dev.off()
```

```
## pdf
##   2
```

```r
# save the plots
tikz(file = "analysisOutputs/plots/check.tex", width = 7, height = 5)
grid::grid.draw(shift_legend(vr_eval_plot))
```

```
## Warning: Removed 304 rows containing non-finite values (stat_ydensity).
```

```
## Warning: Removed 304 rows containing missing values (geom_point).
```

```r
dev.off()
```

```
## pdf
##   2
```

```r
contrasts(data$vr) <- NULL
contrasts(data$condition) <- NULL
class(data$vr)
```

```
## [1] "factor"
```

```r
data.desc <- data %>% filter(vr == TRUE, time == 1) %>%
  ungroup()
```

```
## filter: removed 215 rows (76%), 69 rows remaining
```

```
## ungroup: no grouping variables
```

```r
scls <- c("excitement", "graphically pleasing", "pleasant", "realistic", "enjoyment")



forms <- paste("vr_eval",1:5, " ~ condition", sep = "")
vr_evals <- list()
for(i in 1:5){
  paste("vr_eval", i, " ~ condition", sep = "")
  formul <- as.formula(paste("vr_eval", i, " ~ condition", sep = ""))
  vr_evals[["lm"]][[paste("vr_eval", i, ": ", scls[i])]] <- lm.temp <- lm(data.desc, formula  = formul)
  vr_evals[["comparisons"]][[scls[i]]] <- anova(lm.temp)

}
vr_evals
```

```
## $lm
## $lm$`vr_eval 1 :  excitement`
##
## Call:
## lm(formula = formul, data = data.desc)
##
## Coefficients:
## (Intercept)   conditiona   conditionc
##     6.34783     -0.17391      0.08696
##
##
## $lm$`vr_eval 2 :  graphically pleasing`
##
## Call:
## lm(formula = formul, data = data.desc)
##
## Coefficients:
```

```
## (Intercept)   conditiona   conditionc
##      5.9565      -1.0435      -0.6087
##
##
## $lm$`vr_eval 3 :  pleasant`
##
## Call:
## lm(formula = formul, data = data.desc)
##
## Coefficients:
## (Intercept)   conditiona   conditionc
##      6.0000      -0.1739       0.1739
##
##
## $lm$`vr_eval 4 :  realistic`
##
## Call:
## lm(formula = formul, data = data.desc)
##
## Coefficients:
## (Intercept)   conditiona   conditionc
##      5.4348      -1.0000      -0.3478
##
##
## $lm$`vr_eval 5 :  enjoyment`
##
## Call:
## lm(formula = formul, data = data.desc)
##
## Coefficients:
## (Intercept)   conditiona   conditionc
##     6.30435     -0.08696      0.17391
##
##
##
## $comparisons
## $comparisons$excitement
## Analysis of Variance Table
##
## Response: vr_eval1
##           Df Sum Sq Mean Sq F value Pr(>F)
## condition  2  0.812  0.4058  0.6667 0.5168
## Residuals 66 40.174  0.6087
##
## $comparisons$`graphically pleasing`
## Analysis of Variance Table
##
## Response: vr_eval2
##           Df  Sum Sq Mean Sq F value  Pr(>F)
## condition  2  12.638  6.3188   3.208 0.04682 *
## Residuals 66 130.000  1.9697
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
```

```
## $comparisons$pleasant
## Analysis of Variance Table
##
## Response: vr_eval3
##           Df  Sum Sq Mean Sq F value Pr(>F)
## condition  2   1.391 0.69565  0.4307 0.6519
## Residuals 66 106.609 1.61528
##
## $comparisons$realistic
## Analysis of Variance Table
##
## Response: vr_eval4
##           Df Sum Sq Mean Sq F value  Pr(>F)
## condition  2 11.855  5.9275    4.49 0.01485 *
## Residuals 66 87.130  1.3202
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## $comparisons$enjoyment
## Analysis of Variance Table
##
## Response: vr_eval5
##           Df Sum Sq Mean Sq F value Pr(>F)
## condition  2  0.812 0.40580   0.552 0.5784
## Residuals 66 48.522 0.73518
```

```
summary(vr_evals$lm$`vr_eval 2 :  graphically pleasing`)
```

```
##
## Call:
## lm(formula = formul, data = data.desc)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -3.3478 -0.9565  0.0870  1.0435  2.0870
##
## Coefficients:
##             Estimate Std. Error t value Pr(>|t|)
## (Intercept)   5.9565     0.2926  20.354   <2e-16 ***
## conditiona   -1.0435     0.4139  -2.521   0.0141 *
## conditionc   -0.6087     0.4139  -1.471   0.1461
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 1.403 on 66 degrees of freedom
## Multiple R-squared:  0.0886, Adjusted R-squared:  0.06098
## F-statistic: 3.208 on 2 and 66 DF,  p-value: 0.04682
```

```
summary(vr_evals$lm$`vr_eval 4 :  realistic`)
```

```
##
## Call:
## lm(formula = formul, data = data.desc)
##
## Residuals:
```

```
##     Min      1Q  Median      3Q     Max
## -3.0870 -0.4348  0.5652  0.5652  1.9130
##
## Coefficients:
##             Estimate Std. Error t value Pr(>|t|)
## (Intercept)   5.4348     0.2396  22.685  < 2e-16 ***
## conditiona   -1.0000     0.3388  -2.951  0.00438 **
## conditionc   -0.3478     0.3388  -1.027  0.30836
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 1.149 on 66 degrees of freedom
## Multiple R-squared:  0.1198, Adjusted R-squared:  0.09309
## F-statistic:  4.49 on 2 and 66 DF,  p-value: 0.01485
```

Here we see that only the realism was rated clearly differently between the conditions, although excitement and graphical pleasantness are borderline.

Now sod and presence

```
vars <- c("sod", "ipq")

forms <- paste(vars, " ~ condition", sep = "")
sod_ipq_list <- list()
for(i in 1:2){
  formul <- as.formula(forms[i])
  sod_ipq_list[["lm"]][[vars[i]]] <- lm.temp <- lm(data.desc, formula  = formul)
  sod_ipq_list[["comparisons"]][[scls[i]]] <- anova(lm.temp)

}
sod_ipq_list
```

```
## $lm
## $lm$sod
##
## Call:
## lm(formula = formul, data = data.desc)
##
## Coefficients:
## (Intercept)   conditiona   conditionc
##    4.888889     0.280193    -0.009662
##
##
## $lm$ipq
##
## Call:
## lm(formula = formul, data = data.desc)
##
## Coefficients:
## (Intercept)   conditiona   conditionc
##      4.7112      -0.2554       0.1324
##
##
##
```

```
## $comparisons
## $comparisons$excitement
## Analysis of Variance Table
##
## Response: sod
##           Df Sum Sq Mean Sq F value Pr(>F)
## condition  2  1.247 0.62337  0.5808 0.5623
## Residuals 66 70.834 1.07324
##
## $comparisons$`graphically pleasing`
## Analysis of Variance Table
##
## Response: ipq
##           Df Sum Sq Mean Sq F value Pr(>F)
## condition  2  1.635 0.81748   1.632 0.2038
## Residuals 62 31.057 0.50092
```

Now comparing vr to non-vr

```
vars <- c("sod", "ipq")

forms <- paste(vars, " ~ vr", sep = "")
sod_ipq_vr_list <- list()
for(i in 1:2){
  formul <- as.formula(forms[i])
  sod_ipq_vr_list[["lm"]][[vars[i]]] <- lm.temp <- lm(data, formula  = formul, subset = time == 1)
  sod_ipq_vr_list[["comparisons"]][[scls[i]]] <- anova(lm.temp)

}
sod_ipq_vr_list
```

```
## $lm
## $lm$sod
##
## Call:
## lm(formula = formul, data = data, subset = time == 1)
##
## Coefficients:
## (Intercept)       vrTRUE
##      4.6250       0.3541
##
##
## $lm$ipq
##
## Call:
## lm(formula = formul, data = data, subset = time == 1)
##
## Coefficients:
## (Intercept)       vrTRUE
##       3.586        1.085
##
##
##
## $comparisons
## $comparisons$excitement
```

```
## Analysis of Variance Table
##
## Response: sod
##           Df  Sum Sq Mean Sq F value Pr(>F)
## vr         1   2.232  2.2323   1.886  0.173
## Residuals 91 107.706  1.1836
##
## $comparisons$`graphically pleasing`
## Analysis of Variance Table
##
## Response: ipq
##           Df Sum Sq Mean Sq F value    Pr(>F)
## vr         1 20.639  20.639  37.323 2.739e-08 ***
## Residuals 87 48.110   0.553
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

```
data.desc.all <- subset(data, subset = time==1) %>%
  ungroup()
```

```
## ungroup: no grouping variables
```

```
scls <- c("excitement", "graphically pleasing", "pleasant", "realistic", "enjoyment")



forms <- paste("vr_eval",1:5, " ~ condition", sep = "")
vr_evals_all <- list("lm" = list(), "comparisons" = list())
for(i in 1:5){
#i <- 1
    paste("vr_eval", i, " ~ condition", sep = "")
  formul <- as.formula(paste("vr_eval", i, " ~ vr + (1|condition)", sep = ""))
  vr_evals_all[["lm"]][[scls[i]]] <- mem.temp <- lmer(data.desc.all, formula  = formul)
  vr_evals_all[["comparisons"]][[scls[i]]] <- anova(mem.temp)

}
```

```
## boundary (singular) fit: see ?isSingular
```

```
summary(lm(data.desc.all, formula = ipq ~ vr))
```

```
##
## Call:
## lm(formula = ipq ~ vr, data = data.desc.all)
##
## Residuals:
##      Min       1Q   Median       3Q      Max
## -1.65774 -0.52857 -0.02857  0.54286  1.62798
##
## Coefficients:
##             Estimate Std. Error t value Pr(>|t|)
## (Intercept)   3.5863     0.1518  23.626  < 2e-16 ***
## vrTRUE        1.0851     0.1776   6.109 2.74e-08 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
```

```
## Residual standard error: 0.7436 on 87 degrees of freedom
##   (53 observations deleted due to missingness)
## Multiple R-squared:  0.3002, Adjusted R-squared:  0.2922
## F-statistic: 37.32 on 1 and 87 DF,  p-value: 2.739e-08
```

```
summary(lm(data.desc.all, formula = sod ~ vr))
```

```
##
## Call:
## lm(formula = sod ~ vr, data = data.desc.all)
##
## Residuals:
##      Min       1Q   Median       3Q      Max
## -2.73611 -0.53462 -0.09018  0.70833  2.37500
##
## Coefficients:
##             Estimate Std. Error t value Pr(>|t|)
## (Intercept)   4.6250     0.2221  20.827   <2e-16 ***
## vrTRUE        0.3541     0.2578   1.373    0.173
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 1.088 on 91 degrees of freedom
##   (49 observations deleted due to missingness)
## Multiple R-squared:  0.0203, Adjusted R-squared:  0.009539
## F-statistic: 1.886 on 1 and 91 DF,  p-value: 0.173
```

Looks like VR has an impact on enjoyment and on excitement. It seems to be a positive impact! VR also leads to larger presence, but not to larger suspension of disbelief than video.