

Supplementary Material: The Many Reliabilities of Psychological Dynamics: An Overview of Statistical Approaches to Estimate the Internal Consistency Reliability of Intensive Longitudinal Data

Sebastian Castro-Alvarez

Laura F. Bringmann

Jason Back

Siwei Liu

This document presents the supplementary material of the article titled “The Many Reliabilities of Psychological Dynamics: An Overview of Statistical Approaches to Estimate the Internal Consistency Reliability of Intensive Longitudinal Data”, which presents different approaches to estimate the reliability of the scales used in intensive longitudinal studies in psychology. Here, we show, by means of using empirical data, how to estimate the reliability of a set of items of positive affect based on each of the different approaches. We include step-by-step code to estimate the reliability based on each approach either in R (R Core Team, 2022) or Mplus (Muthén & Muthén, 1998-2023).

Next, we load all the packages needed for the following analyses. To install the package `esmpack`, one has to install the development version of the package available in [this GitHub repository](#). To install the package `dynr`, make sure to follow the [User Installation Instructions Manual](#) of the package.

```
library(lme4)
library(psych)
library(misty)
library(lavaan)
library(MplusAutomation)
library(esmpack)
library(dynr)
```

The Data

The data used in these analyses included daily diary data from 193 participants who completed self-report daily diary questionnaires up to 100 days (O’Laughlin et al., 2021; Williams et al., 2020). On average, participants completed the questionnaire for 82.7 days with a median of 93 days. The minimum number of days completed by a participant was 14 days, and the maximum was 100 days. In particular, we analyzed the responses to the 10 items of positive affect of the Positive and Negative Affect Schedule (PANAS). This scale included the following positive emotions: interested, excited, strong, enthusiastic, proud, inspired, determined, attentive, active, and alert.

First, we read the data into R:

```
ifit_com_pa <- read.table("../Data/ifit_pa_data.dat")
```

The first six rows look like this:

```
kable(head(ifit_com_pa))
```

Table 1: First six rows of the data

ID	day	inter	excit	stron	enthu	proud	inspi	deter	atten	activ	alert
1	1	7.2	5.0	5.0	7.7	5.2	7.5	9.3	8.0	5.0	7.8
1	2	7.5	7.5	7.6	8.0	9.6	8.4	8.8	8.4	8.6	9.8
1	3	NA	NA	NA	NA	NA	NA	NA	NA	NA	NA
1	4	NA	NA	NA	NA	NA	NA	NA	NA	NA	NA
1	5	5.1	5.9	6.6	6.2	7.1	5.5	5.0	5.3	5.9	5.5
1	6	8.2	8.2	7.5	7.8	8.6	8.7	8.9	8.0	8.6	8.4

This dataset includes 19300 records on 12 variables. There are 100 rows for each participant. The first two variables, *ID* and *time*, indicate the participants number and the day of the observation. The other ten variables include the response to each of the items of positive affect. Originally, these items were rated on a scale from 0 to 100, but we divided the responses by 10 when pre-processing the data to facilitate the computation of the models and the interpretation of the results.

Next, to facilitate the management and analysis of the data, we simply created a vector with the names of variables related to the items of positive affect.

```
pa <- c("inter", "excit", "stron", "enthu", "proud",  
        "inspi", "deter", "atten", "activ", "alert")
```

Furthermore, we restructured the data into different formats as required in some of the following analyses. Firstly, we removed all the records in which the participants missed the responses to all the items of positive affect, and we computed the compliance per person to filter based on the number of measurements completed.

```
## Remove rows with complete missingness for all pa items
# Identify number of missing values per row
n_miss <- apply(ifit_com_pa[, c(pa)], 1, function(x) sum(is.na(x)))
# Remove rows with complete missingness (i.e., number of missing values equal to 10)
ifit_pa <- ifit_com_pa[n_miss != 10, ]

# Compute compliance per person
compliance <- calc.nomiss(ID, ID, ifit_pa)

rm(n_miss)
```

Next, we created two new versions of the data: One that only included participants who completed the questionnaire on more than 50 days and one that restructured the data in a way that each row represents the response of a person on a day of one item. Excluding participants with less than 50 observations was necessary in methods that model the lag effects, such as the dynamic factor model (Fuller-Tyszkiewicz et al., 2017; Hu et al., 2016), the mixed-effect trait-state-occasion model (Castro-Alvarez et al., 2022), and the two-level random dynamic model (Xiao et al., 2023). On the other hand, restructuring the data in long format (one row per observation) was necessary in methods based on multilevel modeling (Cranford et al., 2006; Nezlek, 2017).

```
# Identify participants with less than 50 observations
id_exclude_50 <- as.numeric(names(which(compliance < 50)))

# Create subsets of data
# Subset: participants that completed 50 or more days including NAs
ifit_com_pa_50 <- ifit_com_pa[!(ifit_com_pa$ID %in% id_exclude_50), ]
# Subset: participants that completed 50 or more days excluding NAs
ifit_pa_50 <- ifit_pa[!(ifit_pa$ID %in% id_exclude_50), ]
# Subset: Restructure data in long format (one row per observation)
long_pa <- reshape(ifit_pa,
                    direction = "long",
                    idvar      = c("ID", "day"),
                    timevar    = "item",
                    varying    = list(pa),
                    v.names     = "resp")

rm(id_exclude_50)
```

Reliability in ESM

In this section, we estimated the reliability of the PANAS' items of positive affect based on six approaches especially proposed for intensive longitudinal data: Generalizability theory (Cranford et al., 2006), multilevel modeling (Nezlek, 2017), multilevel confirmatory factor analysis (Geldhof et al., 2014; Lai, 2021), P-technique and dynamic factor analysis (Fuller-Tyszkiewicz et al., 2017; Hu et al., 2016), latent state-trait theory (Castro-Alvarez et al., 2022), and two-level random dynamic model (Xiao et al., 2023).

As a comparison, we also estimated the reliability using a naive approach that ignores the nested structure and the time dependency of the data with Cronbach's alpha (Cronbach, 1951). For this, we used the function `alpha` available in the package `psych`.

```
alpha_pa <- psych::alpha(cor(ifit_pa[, pa], use = "complete.obs"),  
                          n.obs = nrow(ifit_pa))
```

In this case, the estimated reliability based on Cronbach's alpha was 0.95, which indicates that the scale is highly reliable. However, this estimated reliability coefficient is misleading because it assumes that all the observations are independent and, as mentioned before, it ignores the nested structure and the time dependency of the data. For these reasons, simply estimating Cronbach's alpha is not an appropriate approach to estimate the reliability of intensive longitudinal data.

In what is next, we showcase how to estimate the reliability based on each of the six approaches proposed specifically for intensive longitudinal data.

Generalizability Theory

The first method, which is relatively popular in the ESM literature, is based on the generalizability theory (Cranford et al., 2006). This model, decomposes the total variance into 7 components (three main effects, three two-way interactions, and error), which are used to compute 5 different coefficients of reliability.

Typically, the model and all the variance components can be estimated with an ANOVA. However, due to the missing observations, which result in an unbalanced data set, it is best to estimate the model by means of a mixed model (multilevel model). In this case, the data need to be structured in long format, meaning that each row represents the response that one person gave to one item on a given day. We stored the data structured this way in the data.frame `long_pa`, for which the first six rows look like this:

```
kable(head(long_pa))
```

Table 2: First six rows of the data structured in long format.

	ID	day	item	resp
1.1.1	1	1	1	7.2
1.2.1	1	2	1	7.5
1.5.1	1	5	1	5.1
1.6.1	1	6	1	8.2
1.7.1	1	7	1	8.3
1.9.1	1	9	1	7.9

To estimate the generalizability theory model using a multilevel model, we used the function `lmer`. To specify the model, we defined each of the main and two-way interaction effects as random intercepts. In this case, `resp` represents the observed score; and `ID`, `day`, and `item` are the corresponding indicator variables for the person, the occasion, and the item.

```
gt_pa <- lmer(resp ~ 1 + (1 | ID) + (1 | day) + (1 | item) +
              (1 | ID:day) + (1 | ID:item) + (1 | day:item),
              data = long_pa, REML = TRUE)
```

Now, to compute the reliability coefficients, we simply extracted the estimates of the variance components with the function `VarCorr` and used these estimates to compute the different reliability coefficients as suggested by Cranford et al. (2006). We stored these estimates in the vector `gt_rel_pa`.

```
var_gt_pa <- as.data.frame(VarCorr(gt_pa))

varP <- var_gt_pa[4, "vcov"] # ID effect variance
varI <- var_gt_pa[6, "vcov"] # item effect variance
var0 <- var_gt_pa[5, "vcov"] # day effect variance
varPxI <- var_gt_pa[2, "vcov"] # ID:item effect variance
varPx0 <- var_gt_pa[1, "vcov"] # ID:day effect variance
varIx0 <- var_gt_pa[3, "vcov"] # day:item effect variance
varE <- var_gt_pa[7, "vcov"] # residual variance

r1f <- (varP + varPxI / 10) /
      (varP + varPxI / 10 + varE / (10))
rkf <- (varP + varPxI / 10) /
      (varP + varPxI / 10 + varE / (100 * 10))
r1r <- (varP + varPxI / 10) /
      (varP + varPxI / 10 + var0 + varPx0 + varE / 10)
rkr <- (varP + varPxI / 10) /
```

```

    (varP + varPxI / 10 + varO / 100 + varPxO / (100) + varE / (100 * 10))
rc <- (varPxO) / (varPxO + varE / 10)

gt_rel_pa <- c(r1f, rkf, r1r,rkr, rc)

rm(r1f, rkf, r1r,rkr, rc, varP, varI, varO,
    varPxI, varPxO, varIxO, varE)

```

Alternatively, one can also use the function `mlr` from the `psych` package. This function estimates the reliability of intensive longitudinal data based on both the generalizability theory (Cranford et al., 2006) and the multilevel model (Nezlek, 2017). In the `mlr` function, data can be structured as in the data.frame `ifit_pa` (one row per person and time and items on different columns) or as in the data.frame `long_pa` (one row per response with indicator variables for the person, the day, and the item). Note that this function automatically computes the generalizability theory reliability coefficients. However, it does not report the R_{1f} . If a researcher is still interested in this coefficient, they have to compute it on their own based on the estimated variances.

```

mlr_pa <- mlr(ifit_pa, grp = "ID", Time = "day", items = pa,
    lmer = TRUE, lme = FALSE, alpha = FALSE, aov = FALSE)

```

Thus, the estimated reliability coefficients for PA as estimated with the functions `lmer` and `mlr` are presented below. The estimated coefficients are identical between functions.

```

gt_table <- data.frame("lmer" = gt_rel_pa,
    "mlr" = c(NA, mlr_pa$RkF, mlr_pa$R1R,
        mlr_pa$RkR, mlr_pa$Rc))
row.names(gt_table) <- c("R1F", "RkF", "R1R", "RkR", "Rc")
kable(gt_table, digits = 2)

```

Table 3: Reliability estimates based on the generalizability theory given the functions `lmer` and `mlr`

	lmer	mlr
R1F	0.95	NA
RkF	1.00	1.00
R1R	0.65	0.65
RkR	0.99	0.99
Rc	0.89	0.89

Multilevel Modeling

The second approach used to estimate the reliability in ESM data is using multilevel modeling, as suggested by Nezlek (2017). The idea in this approach is that the item responses are nested within measurement occasions, which are nested within persons, resulting in a three-level structure. Hence, fitting a three-level multilevel model, the variance coefficients at the different levels can be used to compute two reliability coefficients.

Similarly to the generalizability theory, data need to be in long format (one row per response). To estimate the model, we also used the function `lmer`. To specify three-level models in `lmer`, we used the syntax `resp ~ (1 | ID/day)`, which implies that the responses to the items `resp` are modeled with a random intercept model where `ID` (persons) is the third level and `day` (occasions) is the second level. Thus, the three-level multilevel model is estimated as follows:

```
ml3_pa <- lmer(resp ~ (1 | ID/day), data = long_pa)
```

Then, the estimates of the variances are extracted and used to estimate the reliability coefficients as proposed by Nezlek (2017):

```
var_ml3_pa <- as.data.frame(VarCorr(ml3_pa))

varP    <- var_ml3_pa[2, "vcov"]
varPx0  <- var_ml3_pa[1, "vcov"]
varE    <- var_ml3_pa[3, "vcov"]

rkrn <- varP / (varP + varPx0 / 100 + varE / (10 * 100))
rcn  <- varPx0 / (varPx0 + varE / 10)

ml3_rel_pa <- c(rkrn, rcn)

rm(varP, varPx0, varE, rkrn, rcn)
```

These reliability coefficients are also automatically estimated in the function `mlr`. We can retrieve the estimates from the same object (i.e., `mlr_pa`) that we used to estimate the generalizability theory approach. The following table presents the estimated reliability coefficients for PA according to the multilevel modeling approach (Nezlek, 2017) based on the functions `lmer` and `mlr`. The estimated reliability coefficients are identical between functions.

```
ml3_table <- data.frame("lmer" = ml3_rel_pa,
                        "mlr"  = c(mlr_pa$RkRn, mlr_pa$Rcn))
row.names(ml3_table) <- c("RkRn", "Rcn")

kable(ml3_table, digits = 2)
```

Table 4: Reliability estimates based on the multilevel model given the functions lmer and mlr

	lmer	mlr
RkRn	0.99	0.99
Rcn	0.87	0.87

Multilevel CFA

Geldhof et al. (2014) proposed computing the reliability through a multilevel confirmatory factor analysis. In this approach, a reliability coefficient is estimated for both the within- and the between-level, which are denoted as ω^w and $\tilde{\omega}^b$, respectively. This approach was proposed for data with a nested structure, such as students nested in schools or repeated measures nested within persons, and it has been extensively used in ESM studies. Recently, Lai (2021) discussed that the reliability coefficient at the between-level, as proposed by Geldhof et al. (2014), indicates the reliability of the latent variable but not of the composite score (i.e., observed sum scores). To solve this, Lai (2021) proposed alternative reliability coefficients at the between-level denoted as ω^b and ω^{2l} , which are an improvement over Geldhof et al. (2014) between-level coefficient.

In short, the procedure involves fitting a multilevel unidimensional confirmatory factor model and using the factor loadings and the residual variances at the within- and the between-level to estimate the reliability coefficients based on McDonald’s omega (McDonald, 1999). McDonald’s omega is an alternative reliability coefficient to Cronbach’s alpha.

To estimate the model, we used `lavaan` and computed the reliability estimates as suggested by both, Geldhof et al. (2014) and Lai (2021). To account for different cluster sizes, the between-person reliability coefficient proposed by Lai (2021) uses the *harmonic mean*¹ of the cluster sizes in the formula. In this case, the number of observations per person defines the cluster size, and the harmonic mean of these is 70.11. The syntax for the model in `lavaan` is presented below. This syntax also includes the estimation of the reliability coefficients. To identify the model, we fixed the factor variances to 1 and estimated all the factor loadings. In order to free the estimation of the first factor loading, we multiplied the first indicator by “NA” and repeated the indicator in the list of indicators for a given factor (i.e., `wf =~ NA*inter + Lw1*inter`).

```
mlcfa_model_pa <- "
level: 1
  wf =~ NA*inter + L1*inter + L2*excit + L3*stron + L4*enthu + L5*proud +
        L6*inspi + L7*deter + L8*atten + L9*activ + L10*alert
  wf =~ 1*wf
```

¹The harmonic mean is an alternative mean, typically used in situations when the average rate is desired. Its formula is $n/(\sum_{i=1}^n (1/x_i))$.

```

inter ~~ ew1 * inter
excit ~~ ew2 * excit
stron ~~ ew3 * stron
enthu ~~ ew4 * enthu
proud ~~ ew5 * proud
inspi ~~ ew6 * inspi
deter ~~ ew7 * deter
atten ~~ ew8 * atten
activ ~~ ew9 * activ
alert ~~ ew10 * alert
level: 2
bf =~ NA*inter + L1*inter + L2*excit + L3*stron + L4*enthu + L5*proud +
      L6*inspi + L7*deter + L8*atten + L9*activ + L10*alert
bf ~~ bfvar * bf
inter ~~ eb1 * inter
excit ~~ eb2 * excit
stron ~~ eb3 * stron
enthu ~~ eb4 * enthu
proud ~~ eb5 * proud
inspi ~~ eb6 * inspi
deter ~~ eb7 * deter
atten ~~ eb8 * atten
activ ~~ eb9 * activ
alert ~~ eb10 * alert

# within-level omega (Geldhof, 2014; Lai 2021)
tilomgw := (L1 + L2 + L3 + L4 + L5 + L6 + L7 + L8 + L9 + L10)^2 * 1/
            ((L1 + L2 + L3 + L4 + L5 + L6 + L7 + L8 + L9 + L10)^2 +
            (ew1 + ew2 + ew3 + ew4 + ew5 + ew6 + ew7 + ew8 + ew9 + ew10))

# between-level 'tilde' omega (Geldhof, 2014)
tilomgb := (L1 + L2 + L3 + L4 + L5 + L6 + L7 + L8 + L9 + L10)^2 * bfvar/
            ((L1 + L2 + L3 + L4 + L5 + L6 + L7 + L8 + L9 + L10)^2 * bfvar +
            (eb1 + eb2 + eb3 + eb4 + eb5 + eb6 + eb7 + eb8 + eb9 + eb10))

# between-level omega (Lai, 2021)
omgb := ((L1 + L2 + L3 + L4 + L5 + L6 + L7 + L8 + L9 + L10)^2 * bfvar)/
          ((L1 + L2 + L3 + L4 + L5 + L6 + L7 + L8 + L9 + L10)^2 * (bfvar + 1 / 70.11) +
          (eb1 + eb2 + eb3 + eb4 + eb5 + eb6 + eb7 + eb8 + eb9 + eb10) +
          (ew1 + ew2 + ew3 + ew4 + ew5 + ew6 + ew7 + ew8 + ew9 + ew10) / 70.11)
omg2l := ((L1 + L2 + L3 + L4 + L5 + L6 + L7 + L8 + L9 + L10)^2 * (bfvar + 1))/
          ((L1 + L2 + L3 + L4 + L5 + L6 + L7 + L8 + L9 + L10)^2 * (bfvar + 1) +

```

```
(eb1 + eb2 + eb3 + eb4 + eb5 + eb6 + eb7 + eb8 + eb9 + eb10) +
(ew1 + ew2 + ew3 + ew4 + ew5 + ew6 + ew7 + ew8 + ew9 + ew10))

"
```

To fit the model in `lavaan`, we used the following code:

```
mlcfa_pa_full <- cfa(mlcfa_model_pa, data = ifit_pa, cluster = "ID")
```

With this, we can easily extract the reliability estimates of interest:

```
cfa_pa_est <- parameterEstimates(mlcfa_pa_full)

ml_omega_pa <- cfa_pa_est$est[cfa_pa_est$label %in%
                             c("tilomgw", "tilomgb", "omgb", "omg2l")]
```

Alternatively, we also estimated the within- and between-level reliability by means of the function `multilevel.omega` from the package `misty`. This function automatically fits the multilevel confirmatory model to the data and estimates the reliability coefficients as presented by Lai (2021). This function does not estimate $\tilde{\omega}^b$, which is the between-level reliability as initially proposed by Geldhof et al. (2014).

```
mlcfa_pa <- multilevel.omega(ifit_pa[, pa],
                           cluster = ifit_pa$ID,
                           missing = "listwise")
```

As a result, the multilevel reliability coefficients for PA as estimated in `lavaan` and `misty` are presented in the following table. The results are identical.

```
ml_omega_table <- data.frame("lavaan" = ml_omega_pa,
                             "misty" = mlcfa_pa$result$omega$omega[c(1, NA, 2, 3)])
row.names(ml_omega_table) <- c("$\\omega^w$",
                              "$\\tilde{\\omega}^b$",
                              "$\\omega^b$",
                              "$\\omega^{2l}$")

kable(ml_omega_table, digits = 2, escape = FALSE)
```

Table 5: Reliability estimates based on the multilevel CFA model

	lavaan	misty
ω^w	0.90	0.90
$\tilde{\omega}^b$	0.98	NA
ω^b	0.98	0.98
ω^{2l}	0.95	0.95

P-Technique Factor Analysis and Dynamic Factor Analysis

Another approach to estimate the reliability in ESM studies is to fit P-technique factor analysis or dynamic factor analysis and estimate a reliability coefficient per person based on McDonald’s omega (Fuller-Tyszkiewicz et al., 2017; Hu et al., 2016).

Here, we implemented both P-technique factor analysis and dynamic factor analysis to estimate the reliability per person, for the sake of comparison. The main difference between P-technique factor analysis and dynamic factor analysis is that the former does not account for the lagged effects that are likely to be present in intensive longitudinal data. On the other hand, dynamic factor analysis takes into account auto-regressive and cross-regressive effects. Furthermore, in P-technique factor analysis, the computation of the reliability coefficients based on McDonald’s omega is straightforward. However, when using dynamic factor analysis, the computation of McDonald’s omega must be adjusted to account for the auto-regressive and cross-regressive effects in the model. For the purpose of this study, we adjusted the computation of McDonald’s omega as indicated in Equation 17 of the main article. Also, for these analyses, we only included participants who completed at least 50 measurements (172 participants). We did this because it is generally recommended to have at least 50 time points per participant (Molenaar & Nesselroade, 2009). To proceed, we first separated the data per person into a list:

```
ids <- unique(ifit_com_pa_50$ID)

ifit_list_pa <- list()

for (i in 1:length(ids)) {
  ifit_list_pa[[i]] <- ifit_com_pa_50[ifit_com_pa_50$ID == ids[i], ]
}
rm(i)
```

P-technique factor analysis in lavaan

To fit the p-technique factor analysis to each individual, we used the package `lavaan`. In this case, we identified the model per person by fixing the factor variance at 1. The syntax of the

model in lavaan is as follows:

```
pfa_model_pa <- "  
  wf =~ NA*inter + Lw1*inter + Lw2*excit + Lw3*stron + Lw4*enthu + Lw5*proud +  
        Lw6*inspi + Lw7*deter + Lw8*atten + Lw9*activ + Lw10*alert  
  wf ~~ 1*wf  
  inter ~~ ew1 * inter  
  excit ~~ ew2 * excit  
  stron ~~ ew3 * stron  
  enthu ~~ ew4 * enthu  
  proud ~~ ew5 * proud  
  inspi ~~ ew6 * inspi  
  deter ~~ ew7 * deter  
  atten ~~ ew8 * atten  
  activ ~~ ew9 * activ  
  alert ~~ ew10 * alert  
  
# individual omega  
omega := (Lw1 + Lw2 + Lw3 + Lw4 + Lw5 + Lw6 + Lw7 + Lw8 + Lw9 + Lw10)^2 * 1/  
          ((Lw1 + Lw2 + Lw3 + Lw4 + Lw5 + Lw6 + Lw7 + Lw8 + Lw9 + Lw10)^2 * 1 +  
           (ew1 + ew2 + ew3 + ew4 + ew5 + ew6 + ew7 + ew8 + ew9 + ew10))  
"
```

Then, we fitted the model for each individual and extracted the reliability estimate of the items of PA using the following loop. Within this loop, we (a) retrieved the data from each person and removed the rows with missing values, (b) verified whether the data matrix was positive definite, (c) estimated the model per person when the data matrix was positive definite, (d) checked whether the model had converged, and (e) extracted and saved fit measures and the estimate of McDonald's omega for each individual.

```
# Empty list to store: the lavaan object, the reliability estimate,  
# and the indices of model fit for each person.  
pfa_pa <- pfa_pa_omega <- pfa_pa_fit <- list()  
  
for (i in 1:length(ifit_list_pa)) {  
  # Get dat for participant i and remove row with NAs  
  tmp <- na.omit(ifit_list_pa[[i]][, pa])  
  # Fit the model if all eigen values are larger than 0  
  if (!any(eigen(cov(tmp, use = "complete.obs"))$values <= 0)) {  
    # Fit the P-technique model to the person i  
    pfa_pa[[i]] <- cfa(pfa_model_pa, tmp)  
    # Save model output if the model converged
```

```

    if (lavInspect(pfa_pa[[i]], what = "converged")) {
      tmp_est      <- parameterEstimates(pfa_pa[[i]])
      pfa_pa_omega[[i]] <- tmp_est$est[tmp_est$label == "omega"]
      pfa_pa_fit[[i]] <- all(c(fitMeasures(pfa_pa[[i]])[c("cfi", "tli")] > 0.9,
                             fitMeasures(pfa_pa[[i]])["rmsea"] < 0.08))
    } else {
      pfa_pa_omega[[i]] <- NA
      pfa_pa_fit[[i]] <- NA
    }
  } else {
    pfa_pa[[i]] <- NULL
    pfa_pa_omega[[i]] <- NA
    pfa_pa_fit[[i]] <- NA
  }
}
rm(tmp, tmp_est, i)

```

The model did not converge for 2 participants, reducing the number of estimated models to 170. The mean and median of the estimated reliability of the items of PA per person were 0.87 and 0.89, respectively, with a minimum of 0.35 and a maximum of 0.98. The distribution of the estimated reliability per person based on the p-technique is shown in the following figure:

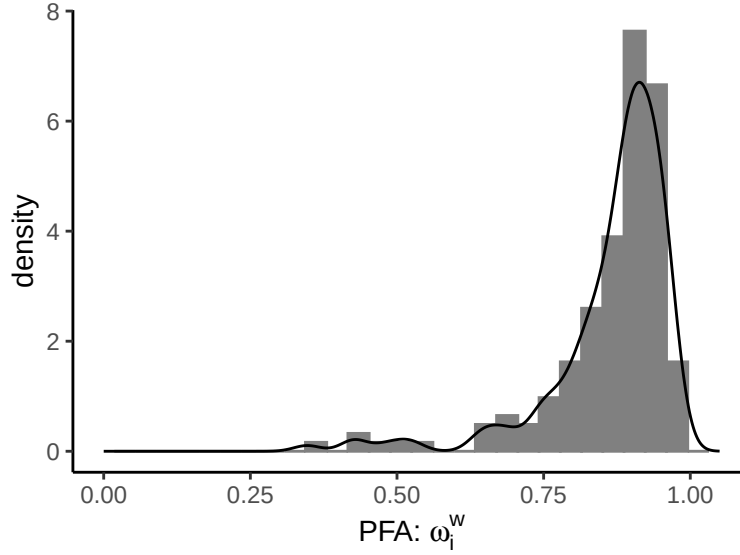
```

pfa_pa_omega <- as.data.frame(unlist(pfa_pa_omega))
names(pfa_pa_omega) <- "pfa_omg"
pfa_pa_fit   <- as.data.frame(unlist(pfa_pa_fit))
names(pfa_pa_fit) <- "pfa_fit"

ggplot(na.omit(pfa_pa_omega), aes(x = pfa_omg)) +
  geom_histogram(aes(y = after_stat(density)),
                colour = gray(1/2),
                fill = gray(1/2),
                bins = 30) +
  geom_density() +
  xlab(expression(paste("PFA: ", omega[i]^w))) +
  xlim(0, 1.05)

```

Figure 1: Distribution of the reliability coefficients per person of the items of PA based on P-technique factor analysis.



Dynamic factor analysis in dynr

To fit the dynamic factor model of order 1 to each participant, we used the package **dynr**. This is a general package that can fit all kinds of dynamic models, including discrete time, continuous time, and regime switching models. This package uses Kalman filtering techniques for the estimation of the model. The workflow in **dynr** consists of (a) specifying each component of the model through their “recipe” functions, (b) defining the **dynr** model, and (c) estimating the model with the function **dynr.cook**. To specify the dynamic factor model in **dynr**, we needed four components: dynamics, measurement, noise, and initial states. These components are specified through their corresponding “recipe” functions: **prep.matrixDynamics**, **prep.measurement**, **prep.noise**, and **prep.initial**.

Next, we defined each of the components for the dynamic factor model. In the dynamic component, **prep.matrixDynamics**, one indicates the initial values for the auto-regressive effects (**values.dyn**), the labels of the auto-regressive effects (**params.dyn**), and whether the dynamic process is continuous. When the label is “fixed”, the parameter is fixed to their initial value. In the measurement component, **prep.measurement**, one specifies the initial values and the labels of the factor loadings and the item intercepts, as well as the names of the latent and the observed variables. In the noise component, **prep.noise**, one specifies the initial values and the labels of the error terms. In this case, the error terms of the observed variables are the measurement error and the error terms of the latent variables are the latent innovation variance of the dynamic process. Lastly, the initial state component, **prep.initial**, is used to specify different states in regime-switching models. As here, we only have one state, this

component simply indicates the mean and the variance of the latent variable which are used to initiate the algorithm.

In preliminary analyses, we noticed that the dynamic factor model as implemented in **dynr** did not converge or resulted in untrustworthy estimates for a handful of participants. To resolve this issue and ease the estimation of the model, we decided to provide reasonable starting values based on the fitted p-technique factor model per person. This allowed us to obtain proper results for almost all individuals. To extract the starting values, we identified the p-technique model by fixing the factor loading of the first item at 1. The new p-technique model in **lavaan** is defined as

```
pfa_model_pa_v2 <- "
  wf =~ inter + Lw2*excit + Lw3*stron + Lw4*enthu + Lw5*proud +
        Lw6*inspi + Lw7*deter + Lw8*atten + Lw9*activ + Lw10>alert
  wf ~~ fvar*wf
  inter ~~ ew1 * inter
  excit ~~ ew2 * excit
  stron ~~ ew3 * stron
  enthu ~~ ew4 * enthu
  proud ~~ ew5 * proud
  inspi ~~ ew6 * inspi
  deter ~~ ew7 * deter
  atten ~~ ew8 * atten
  activ ~~ ew9 * activ
  alert ~~ ew10 * alert

# individual omega
omega := (1 + Lw2 + Lw3 + Lw4 + Lw5 + Lw6 + Lw7 + Lw8 + Lw9 + Lw10)^2 * fvar /
        ((1 + Lw2 + Lw3 + Lw4 + Lw5 + Lw6 + Lw7 + Lw8 + Lw9 + Lw10)^2 * fvar +
        (ew1 + ew2 + ew3 + ew4 + ew5 + ew6 + ew7 + ew8 + ew9 + ew10))
"
```

Next, in the following loop, we fitted the P-technique model for each person to extract the estimated parameters and defined the dynamic factor model in **dynr** using the “recipe” functions.

```
# Create list objects to store the dynr components per person
dynamics <- meas <- ecov <- initial <- list()

for (i in 1:length(ifit_list_pa)) {
  # Get the data for the i-th person
  tmp <- na.omit(ifit_list_pa[[i]][, pa])
  # Fit the model if all eigen values are larger than 0
```

```

if (!any(eigen(cov(tmp, use = "complete.obs"))$values <= 0)) {
  # Fit the P-technique model to the person i
  tmp_pfa <- cfa(pfa_model_pa_v2, tmp, meanstructure = TRUE)
  # Extract estimated parameters
  if (lavInspect(tmp_pfa, what = "converged")) {
    tmp_est <- parameterEstimates(tmp_pfa)[, 5]
    rm(tmp_pfa)
  } else {
    # Define starting values when the model does not converge
    tmp_est <- rep(c(1, 0.5, 0), times = c(11, 10, 10))
  }
} else {
  # Define starting values when it is impossible to fit the model
  tmp_est <- rep(c(1, 0.5, 0), times = c(11, 10, 10))
}

# Dynamic component. Defining Autoregressive effect
dynamics[[i]] <- prep.matrixDynamics(
  values.dyn = matrix(c(0), nrow = 1), # AR effect initial value set to 0
  params.dyn = matrix(c("phi"), nrow = 1), # Autoregressive effect label
  isContinuousTime = FALSE)
# Measurement component. Defining factor model.
meas[[i]] <- prep.measurement(
  values.load = matrix(tmp_est[1:10], 10, 1), # Loadings initial values
  params.load = matrix(c('fixed', paste0("lambda_", 2:10)), 10, 1), # labels
  values.int = matrix(tmp_est[22:31], 10, 1), # Intercepts
  params.int = matrix(paste0("int", 1:10), 10, 1), #labels
  state.names = c("Theta"), # Latent variable name
  obs.names = pa) # Items
# Noise component. Defining measurement error and innovation error terms.
ecov[[i]] <- prep.noise(
  values.latent = diag(tmp_est[11], 1), # Innovation Variance initial value
  params.latent = diag(c("innovar"), 1), # Innovation variance label
  values.observed = diag(tmp_est[12:21]), # Measurement error variance initial value
  params.observed = diag(paste0("merror_", 1:10))) # Measurement error labels
# Initial states component. Defining number of states and initial conditions.
initial[[i]] <- prep.initial(
  values.inistate = c(0), # Initial mean of the latent variable
  params.inistate = c('fixed'), # Fixed mean of latent variable
  values.inicov = diag(1, 1), # Initial variance
  params.inicov = diag('fixed', 1)) # Fixed variance

```

```

    # Clear temporal objects to avoid overwriting
    rm(tmp, tmp_est)
  }

  rm(i)

```

Once all the components are defined for each person, we can specify and fit the models to the data. We carried out this process in the following loop. Within this loop, we (a) retrieved the data from each person, (b) structured the data according to `dynr` requirements using the function `dynr.data`, (c) specified the dynamic factor model in `dynr`, (d) estimated the model for each person, and (e) computed and saved McDonald's omega for each person based on the estimated parameters.

```

dynr_dfa_pa    <- dynr_dfa_pa_omega <- list()
dynr_dfa_phi  <- list()

for (i in 1:length(ifit_list_pa)) {
  cat("\nThis is the analysis of person ", i, "\n")
  # Get data for participant i
  tmp <- ifit_list_pa[[i]]

  # Structure data with dynr.data
  data_dynr <- dynr.data(tmp,
                        id    = "ID",
                        time  = "day",
                        observed = pa)

  # Specify dfa model in dynr
  model_dynr_dfa <- dynr.model(dynamics    = dynamics[[i]],
                              measurement = meas[[i]],
                              noise       = ecov[[i]],
                              initial     = initial[[i]],
                              data        = data_dynr)

  # Estimate the dynamic factor model and compute omega
  cook_dfa <- dynr.cook(model_dynr_dfa, verbose = FALSE)
  # Save results if model "converged" (exitflag 1, 2, or 3) and if
  # the estimated autoregressive effect is lower than 1.
  if (cook_dfa$exitflag %in% c(1:3) && coef(cook_dfa)["phi"] < 1) {
    # Save dynr object
    dynr_dfa_pa[[i]] <- cook_dfa
    # Save estimated omega

```

```

dynr_dfa_pa_omega[[i]] <-
  (sum(c(1, coef(cook_dfa)[paste0("lambda_", 2:10)]))^2 *
    (coef(cook_dfa)["innovar"]/(1 - coef(cook_dfa)["phi"]^2))) /
  (sum(c(1, coef(cook_dfa)[paste0("lambda_", 2:10)]))^2 *
    (coef(cook_dfa)["innovar"]/(1 - coef(cook_dfa)["phi"]^2)) +
    sum(coef(cook_dfa)[paste0("merror_", 1:10)]))
# Save autoregressive effect
dynr_dfa_phi[[i]] <- coef(cook_dfa)["phi"]
} else {
  dynr_dfa_pa[[i]] <- NULL
  dynr_dfa_pa_omega[i] <- NA
  dynr_dfa_phi[[i]] <- NA
}

# Clear temporal objects to avoid overwriting
rm(tmp, data_dynr, model_dynr_dfa,
    cook_dfa)
}
rm(i)

```

The dynamic factor model did not converge or returned an estimated autoregressive effect larger than 1 for 3 participants, reducing the number of participants to 169. In these analyses, the mean and median of the estimated reliability of the items of PA per person were 0.86 and 0.89. The minimum reliability was 0.25 and the maximum was 1.

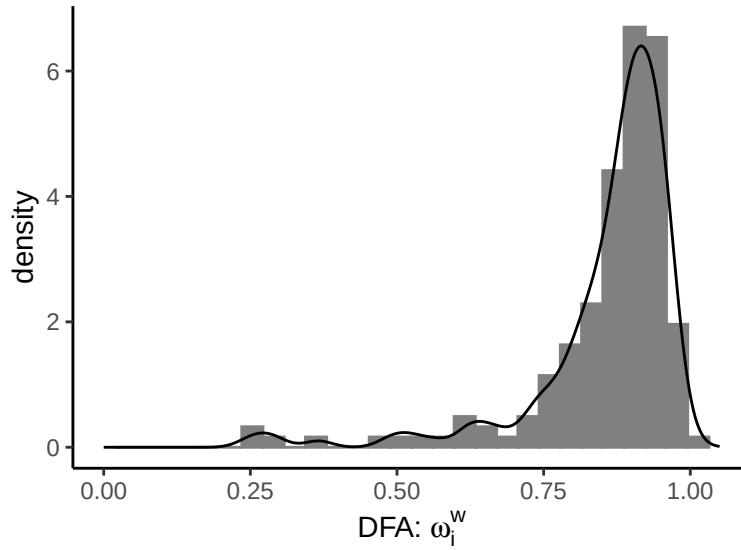
```

dynr_dfa_pa_omega <- as.data.frame(unlist(dynr_dfa_pa_omega))
names(dynr_dfa_pa_omega) <- "dfa_omg"

ggplot(na.omit(dynr_dfa_pa_omega), aes(x = dfa_omg)) +
  geom_histogram(aes(y = after_stat(density)),
    colour = gray(1/2),
    fill = gray(1/2),
    bins = 30) +
  geom_density() +
  xlab(expression(paste("DFA: ", omega[i]^w))) +
  xlim(0, 1.05)

```

Figure 2: Distribution of the reliability coefficients per person of the items of PA based on the dynamic factor model.

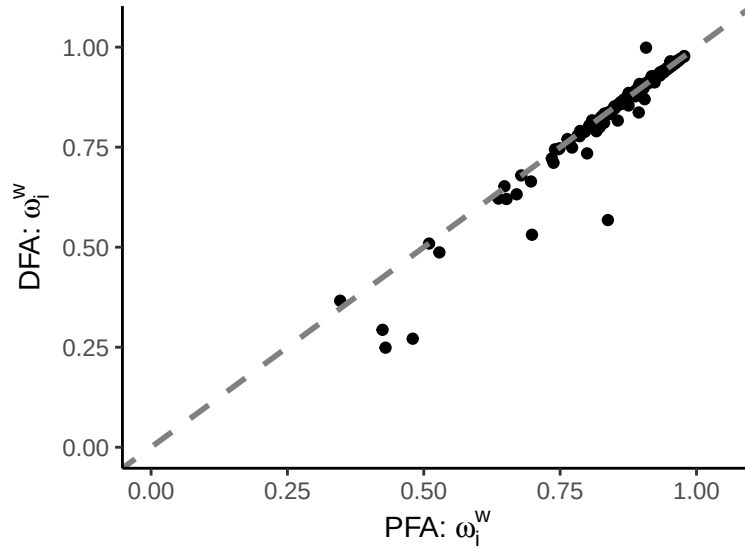


Comparison: P-technique and dynamic factor

To compare the reliability estimates between the P-technique factor analysis and the dynamic factor model, we plotted the estimated reliabilities against each other. Overall, the reliability coefficients estimated based on each of these methods seem to agree to a large extent. When the differences were large, the reliability coefficient based on dynamic factor analysis tended to be lower than the reliability coefficients based on P-technique factor analysis.

```
p0 <- ggplot(pfa_pa_omega, aes(x = pfa_omg, y = dynr_dfa_pa_omega$dfa_omg)) +
  geom_point() +
  xlab(expression(paste("PFA: ", omega[i]^w))) +
  ylab(expression(paste("DFA: ", omega[i]^w))) +
  xlim(0, 1.05) +
  ylim(0, 1.05) +
  geom_abline(intercept = 0,
              slope = 1,
              colour = gray(1/2),
              linetype = 2,
              linewidth = 1)
# Display plot
print(p0)
```

Figure 3: Scatterplot of the estimated reliability coefficients per person based on the P-technique and the dynamic factor model.



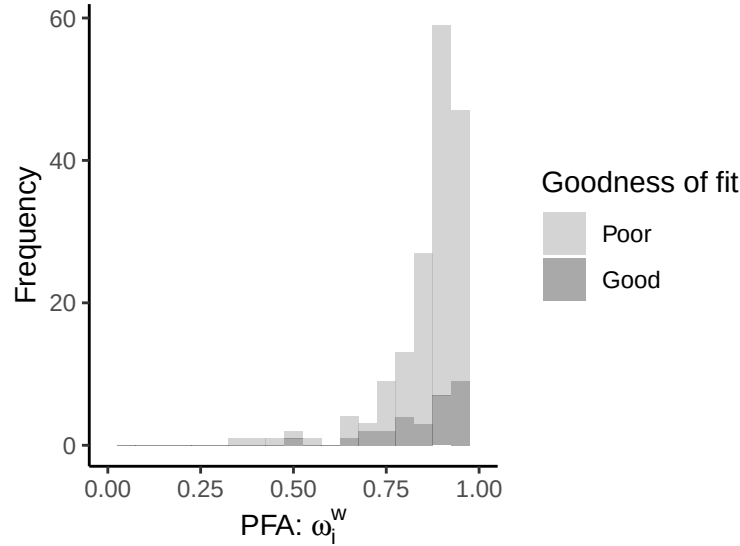
Furthermore, when estimating the P-technique factor analysis, it was possible to assess the goodness-of-fit of the model by means of standard fit indices such as CFI, TLI, or RMSEA. We wanted to see whether there was any difference between the estimated reliability when the model showed a good fit versus when it did not. In our data, it does not seem that there is an association. Therefore, the fact that the model fits the data well does not necessarily imply that the reliability is satisfactory, and vice versa.

```
tmp <- data.frame(pfa_pa_omega, pfa_pa_fit)
tmp <- na.omit(tmp)

ggplot(tmp, aes(x = pfa_omg, fill = pfa_fit)) +
  geom_histogram(alpha = 0.5, bins = 21) +
  scale_fill_manual(values = gray((2:1)/3),
                    labels = c("Poor", "Good")) +
  xlab(expression(paste("PFA: ", omega[i]^w))) +
  ylab("Frequency") +
  xlim(0, 1) +
  labs(fill = "Goodness of fit")

rm(tmp)
```

Figure 4: Distribution of the reliability coefficients per person of the items of PA based on the p-technique factor analysis given the goodness of fit of the model.



Latent State-Trait Theory

Next, we estimated the reliability based on the ME-TSO model (Castro-Alvarez et al., 2022). This model is based on the Latent State-Trait theory and is suitable for analyzing intensive longitudinal data. The ME-TSO aims to distinguish between the trait and state components of the measurement and indicates to what extent the measured construct is trait- or state-like.

For these analyses, we first loaded two custom functions, each of which is available at the following GitHub repositories: [LST_Analyses](#) and [ME-TSO](#). The first function aids in writing the Mplus syntax of the model from R and the second function is used to compute the reliability and all other variance coefficients of the ME-TSO. One can directly load the function from the GitHub repository as follows:

```
eval(parse(
text = RCurl::getURL(paste0("https://raw.githubusercontent.com/",
                             "secastroal/LST_Analyses/master/",
                             "R/write.mltso.to.Mplus.R"),
                             ssl.verifypeer = FALSE)))
eval(parse(
text = RCurl::getURL(paste0("https://raw.githubusercontent.com/",
                             "secastroal/ME-TSO/master/",
                             "R/metso.var.coef.R"),
                             ssl.verifypeer = FALSE)))
```

Here, we exported the data (participants with at least 50 measurements) to Mplus using the package `MplusAutomation`. Also, we used the custom function `write.mltso.to.Mplus` to generate part of the Mplus syntax for the model. Note that this function does not precisely generate the ME-TSO, but rather a simplified version of the model. Therefore, one still has to adjust the syntax before running the model.

```
prepareMplusData(ifit_pa_50, filename = "../Mplus/metso_pa.dat", inpfiler = TRUE)

cat(write.mltso.to.Mplus(ifit_pa_50[, pa]))
```

Once the syntax is complete, the Mplus ME-TSO model for the items of PA looks like this:

```
cat(readLines("../Mplus/metso_pa.inp"), sep = "\n")
```

```
TITLE: Your title goes here
DATA: FILE = "Mplus/metso_pa.dat";
VARIABLE:
NAMES = ID day inter excit stron enthuproud inspi deter atten activ alert;
MISSING=.;
USEVAR = inter excit stron enthuproud inspi deter atten activ alert;
CLUSTER = ID;
TINTERVAL = day(1);

ANALYSIS:
TYPE = TWOLEVEL RANDOM;
ESTIMATOR = BAYES;
BITERATIONS = 50000(20000);
PROCESSORS = 4;
CHAINS = 4;
THIN = 10;

MODEL:

%WITHIN%

! Latent occasion specific variable
eta BY inter@1
excit (lamb2)
stron (lamb3)
enthuproud (lamb4)
inspi (lamb6)
```

```

deter (lamb7)
atten (lamb8)
activ (lamb9)
alert (&1 lamb10);

! Random Autoregressive effect
ar | eta ON eta&1 ;

! Variances of the random measurement error and
! the latent state residual.
inter(o_err1) ;
excit(o_err2) ;
stron(o_err3) ;
enthu(o_err4) ;
proud(o_err5) ;
inspi(o_err6) ;
deter(o_err7) ;
atten(o_err8) ;
activ(o_err9) ;
alert(o_err10) ;
eta(s_var) ;

%BETWEEN%

! Latent trait-indicator variables
theta1 BY inter@1 ;
theta2 BY excit@1 ;
theta3 BY stron@1 ;
theta4 BY enthu@1 ;
theta5 BY proud@1 ;
theta6 BY inspi@1 ;
theta7 BY deter@1 ;
theta8 BY atten@1 ;
theta9 BY activ@1 ;
theta10 BY alert@1 ;

!Fix 0 intercepts and estimate latent trait-indicator means.
!Estimate random coefficients means
[inter@0];
[excit@0];
[stron@0];
[enthu@0];
[proud@0];

```

```

[inspi@0];
[deter@0];
[atten@0];
[activ@0];
[alert@0];
[theta1*];
[theta2*];
[theta3*];
[theta4*];
[theta5*];
[theta6*];
[theta7*];
[theta8*];
[theta9*];
[theta10*];
[ar];

!Fix 0 between residual variance. Estimate latent trait-indicator
!variances and random coefficients variances.
inter@0.001;
excit@0.001;
stron@0.001;
enthu@0.001;
proud@0.001;
inspi@0.001;
deter@0.001;
atten@0.001;
activ@0.001;
alert@0.001;
theta1(t_var1);
theta2(t_var2);
theta3(t_var3);
theta4(t_var4);
theta5(t_var5);
theta6(t_var6);
theta7(t_var7);
theta8(t_var8);
theta9(t_var9);
theta10(t_var10);
ar;

! Covariances
theta1-theta10 WITH theta1-theta10 ar;

```

```

SAVEDATA: BPARAMETERS = metso_pa_samples.dat;
          FILE is metso_pa_fscores.dat;
          SAVE = FSCORES (200 10);
          FACTORS = all;

```

```

DATA IMPUTATION: THIN = 10;

```

```

OUTPUT: TECH8;

```

In addition to specifying the syntax of the model, in Mplus, one should also indicate other arguments for the estimation and the output to be saved. To estimate this model, we used Bayesian estimation with the following setup: 4 chains, thinning 10, and a minimum of 20000 (maximum 50000) iterations per chain. We also asked Mplus to add missing values between observations to ensure that the observations are approximately equally spaced over time. This was done using the command `tinterval` in Mplus. For the output, we required to saved all the MCMC samples of the estimated parameters and the estimated values of the factor scores. We required an additional 200 iterations to compute the factor scores in order to save a posterior distribution of these values.

As running this model can be time consuming due to the MCMC algorithm and the number of iterations, we ran the model directly in Mplus once and simply read the output into R for this document. Usually, one can import Mplus output into R with the function `readModels` of the package `MplusAutomation`. However, when saving factor scores, the function does not work as expected as it fails to load the output into R. For this reason, we had to import each of the Mplus output variables on our own. Below, we read the output file and retrieved the total number of observations and participants. Notice that the total number of observations used in Mplus might be different from the total number of observations that we had in the exported data due to using the `tinterval` command.

```

# Import output file
metso_mplus_out <- readLines("../Mplus/metso_pa.out")
# Extract total number of observations
metso_nobs <- grep("Number of observations",
                  metso_mplus_out,
                  value = TRUE)
metso_nobs <- sub("Number of observations\\s+(\\d+)", "\\1", metso_nobs)
metso_nobs <- as.numeric(metso_nobs)
# Extract number of participants
metso_nid <- grep("Number of clusters",
                 metso_mplus_out,
                 value = TRUE)

```

```
metso_nid <- sub("Number of clusters\\s+(\\d+)", "\\1", metso_nid)
metso_nid <- as.numeric(metso_nid)
```

Next, we imported the factor scores, which were saved in the file “*metso_pa_fscores.dat*”. This file includes the factor scores corresponding to each of the 1.7134×10^4 observations for each of the latent variables and each of the extra iterations. Since the file is extremely large, we used the function `scan` instead of `read.table` to import the data into R. After importing the output with `scan`, we had to restructure the data into a matrix.

```
metso_fscores_raw <- scan("../Mplus/metso_pa_fscores.dat", what = "c",
                          na.strings = "*")
metso_fscores_raw <- as.numeric(metso_fscores_raw)
metso_fscores_raw <- matrix(metso_fscores_raw,
                            nrow = metso_nobs,
                            ncol = length(metso_fscores_raw)/metso_nobs,
                            byrow = TRUE)
```

This matrix with the factor scores has 4728 variables. From the data in the matrix `metso_fscores_raw`, we extracted the factor scores of the autoregressive effect, which is a random effect in the ME-TSO. As we used 200 iterations to compute the factor scores, we saved values of the autoregressive effect per person for each iteration, meaning that we are interested in 200 columns of the matrix `metso_fscores_raw`. To identify the variables, we obtained the names of these columns from the output file.

```
# Get lines where the names of the factor scores are
line_start <- grep("Order and format of variables", metso_mplus_out)
line_end   <- grep(
  "\\+ variables that have a value for each of the \\d+ imputations",
  metso_mplus_out)
# Extract number of iterations used to compute the factor scores
n_fs_samples <- sub(
  ".*\\+ variables that have a value for each of the (\\d+) imputations.*",
  "\\1", metso_mplus_out[line_end])
n_fs_samples <- as.numeric(n_fs_samples)
# Obtain the name of the factor scores
fscoresnames <- metso_mplus_out[(line_start + 2):(line_end - 2)]
# Identify names that need to be repeated for the number of iterations
which_samples <- grep("^\\s+\\+", fscoresnames)
# Clean the factor scores names by removing unnecessary spaces
fscoresnames <- gsub("^\\s+|^\\s+\\+|^\\s+_", "", fscoresnames)
fscoresnames <- gsub("\\s{2,}..+", "", fscoresnames)
```

Additionally, we had to complete the names of the factor scores by repeating the variable names of the variables for which we computed 200 factor scores.

```
fcoresnames <-
lapply(fcoresnames, function(x) {
  if (x %in% fcoresnames[which_samples]) {
    return(paste0(x, sprintf("_i%03d", 1:n_fs_samples)))
  } else {
    return(x)
  }
})

fcoresnames <- unlist(fcoresnames)
```

With this, we were able to rename all the variables in the matrix `metso_fcores_raw`, and we simplified the matrix by selecting the columns that included the factor scores of the autoregressive effect. At the same time, we only selected one row per participant as the factor score of the autoregressive effect was the same across observations for each participant.

```
# Rename variables in the factor scores
colnames(metso_fcores_raw) <- fcoresnames
# Get the first row of each participant in the data
firstrow_id <- match(unique(metso_fcores_raw[, "ID"]),
                     metso_fcores_raw[, "ID"])
# Simplify dataset of the factor scores
metso_fcores_pa <- metso_fcores_raw[firstrow_id, grep("AR_i", fcoresnames)]
row.names(metso_fcores_pa) <- unique(metso_fcores_raw[, "ID"])
```

To compute the reliability based on the ME-TSO, we also needed the MCMC samples of the other parameters of the model such as the between-level factor variances or the factor loadings. To do this, we imported the MCMC samples of these parameters, which are stored in the file *“metso_pa_samples.dat”*. Similarly, as with the factor scores, we had to find the names of these variables in the output file. Moreover, this file includes up to 80200 samples from all the parameters, but we only used the last 200, which correspond to the iterations used to compute the factor scores.

```
# Import MCMC samples
metso_samples <- read.table("../Mplus/metso_pa_samples.dat")
# Get lines to find variables names in the output file
line_start <- grep("Order of parameters saved", metso_mplus_out)
line_end    <- grep("Convergence iterations 1-\\d+",
                    metso_mplus_out)
```

```

# Get names of the variables in the MCMC samples
samplesnames <- metso_mplus_out[(line_start + 2):(line_end - 2)]
samplesnames <- gsub("^\\s+", "", samplesnames)
# Rename the MCMC samples
names(metso_samples) <- samplesnames
# Get total number of iterations from the output file
niter_final <- sub(".*Convergence iterations 1-(\\d+).*",
                  "\\1", metso_mplus_out[line_end])
niter_final <- as.numeric(niter_final)
# Select the samples from the last 200 iterations
metso_samples <- metso_samples[metso_samples$`Chain number` == 1 &
                               metso_samples$`Iteration number` > niter_final, ]

```

After extracting the factor scores of the autoregressive effects stored in the matrix `metso_fscores_pa` and the MCMC samples of the other parameters in the model stored in the matrix `metso_samples`, we had all the output needed to compute the variance coefficients of the ME-TSO. However, before computing the variance coefficients, we created the following function that allows structuring the estimated parameters as required for the function `metso.var.coeff`.

```

extract.metso.bpar <- function(bpar, I) {
  # Turn bayesian samples of the parameters for a given iteration
  # into a vector.
  bpar <- bpar[, grep("Parameter", names(bpar))]
  bpar <- t(bpar)

  # Set number of situation covariates to 0.
  D <- 0

  # Number of elements in the between-level var-cov matrix
  NC <- (I * (I + 1)) / 2

  # Retrieve within and between parameters from bpar
  within_par <- list(lambda.state = c(1, bpar[1:(I - 1)]), # Factor Loadings
                    epsilon.var = bpar[(I):(2 * I - 1)], # Measurement Error
                    zeta.var = bpar[2 * I]) # Latent State Variance
  between_par <- list(beta1 = matrix(0, I, 1), # Person-Situation Effect
                    beta0 = matrix(0, I, 1), # Person-Situation Effect
                    xi.means = bpar[(2 * I + 1):(3 * I)], # Latent Traits Mean
                    ar.mean = bpar[3 * I + 1], # Mean Autoregressive Effect
                    xi.var = bpar[cumsum(1:I) + 3 * I + 1], # Latent Traits Var
                    ar.var = bpar[4 * I + NC + 2], # AR Effect Variance

```

```

        omega.var = 0) # Person-Situation Residual

out <- list(within = within_par, between = between_par)

return(out)
}

```

Next, we computed the different variance coefficients of the ME-TSO discussed in Castro-Alvarez et al. (2022) for each of the 200 iterations. In the following loop, we used the custom function `extract.metso.bpar` to extract the estimated parameters into a list that distinguishes between the within-level and the between-level parameters, and we used the custom function `metso.var.coeff` to actually compute all the variance coefficients for a given iteration. Also, since we did not include any situational dummy variables in the model, in order for the function to work, we set the effects of the parameters associated with the situational dummy variables to 0.

```

metso_varcoeff_samples <- list()

I <- 10

for (i in 1:n_fs_samples) {
  # Get the ith samples of the parameter estimates
  bpar_tmp <- extract.metso.bpar(metso_samples[i, ], I)
  # Get the ith samples of the individual AR effects
  ar_tmp <- metso_fscores_pa[, i]
  # Compute ME-TSO variance coefficients for the ith samples
  coeff_tmp <- metso.var.coeff(
    within.parameters = bpar_tmp$within,
    between.parameters = bpar_tmp$between,
    ind.ar = ar_tmp,
    id = names(ar_tmp))

  metso_varcoeff_samples[[i]] <- coeff_tmp
}
rm(bpar_tmp, ar_tmp, coeff_tmp, i)

```

Using the previous loop, all of the variance coefficients of the ME-TSO were computed for each iteration and saved in the list `metso_varcoeff_samples`. Since we were only interested in the reliability estimates, we extracted the samples of the reliability parameter into an array.

```
# Get only the reliability samples
metso_rel_samples <- lapply(metso_varcoeff_samples, function(x) {
  x$random.situations$Xi_j0$reliability
})
# Restructure the reliability sample into an array
metso_rel_samples <- array(
  data      = unlist(metso_rel_samples),
  dim       = c(metso_nid, I, n_fs_samples),
  dimnames = list(id      = unique(metso_fscores_raw[, "ID"]),
                  items = pa,
                  iter  = paste0("it_", 1:n_fs_samples)))
```

This array contains the samples of the reliability parameters per person and item. In other words, we stored the posterior distribution of the reliability as computed in the ME-TSO. By having it in an array, it was straightforward to summarize the data and compute the estimated reliability per person and item (i.e., the median of the posterior distribution).

```
# Compute reliability estimates
metso_rel_sum <- apply(metso_rel_samples, 1, function(x) {
  apply(x, 1, median, na.rm = TRUE)
})
metso_rel_sum <- as.data.frame(t(metso_rel_sum))
```

The following table presents the median, minimum, and maximum of the reliability coefficients across individuals for each item of PA. Overall, there was not much variability in these coefficients across individuals.

```
metso_rel_summaries <- rbind(apply(metso_rel_sum, 2, median),
                             apply(metso_rel_sum, 2, min),
                             apply(metso_rel_sum, 2, max))
metso_rel_summaries <- as.data.frame(metso_rel_summaries)
row.names(metso_rel_summaries) <- c("Median", "Min", "Max")

kable(metso_rel_summaries, digits = 2)
```

Table 6: Summaries of the Reliability estimates based on the ME-TSO. Median, minimum, and maximum per item across persons.

	inter	excit	stron	enthu	proud	inspi	deter	atten	activ	alert
Median	0.79	0.78	0.79	0.83	0.78	0.80	0.73	0.77	0.66	0.71

Table 6: Summaries of the Reliability estimates based on the ME-TSO. Median, minimum, and maximum per item across persons.

	inter	excit	stron	enthu	proud	inspi	deter	atten	activ	alert
Min	0.78	0.77	0.78	0.82	0.78	0.79	0.72	0.76	0.65	0.70
Max	0.91	0.90	0.89	0.93	0.89	0.91	0.88	0.89	0.83	0.84

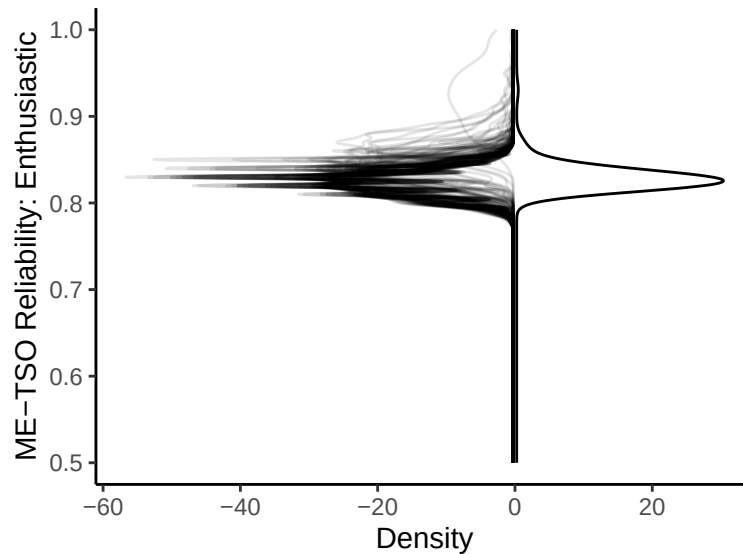
To conclude this section, we created the following plots to illustrate the distribution of the estimated reliability coefficients across individuals for two items: Enthusiastic and active. On the left side of the plots, each density plot represents the posterior distribution of the reliability coefficient for a specific person. The median of this distribution served as the estimated reliability coefficient of the person. On the right side, the density plot represents the distribution of the estimated reliability coefficients across individuals, which would be similar to the histograms presented for the reliability coefficients of the P-technique factor analysis and the dynamic factor analysis (see Figure 1 and Figure 2).

```
tmp <- as.data.frame(metso_rel_samples[, "enthu", ])
tmp_long <- reshape(tmp,
                    direction = "long",
                    varying   = colnames(tmp),
                    ids       = row.names(tmp),
                    v.names   = "it")

tmp_long$median <- ave(tmp_long$it,
                      tmp_long$id,
                      FUN = function(x) median(x, na.rm = TRUE))

plot01 <- ggplot(na.omit(tmp_long), aes(x = x)) +
  geom_density(adjust = 12,
              aes(x = median, y = after_stat(density) + 0.25)) +
  xlim(c(0.5, 1)) +
  geom_density(aes(x = it, group = id, y = -after_stat(density) - 0.25),
              color = rgb(0, 0, 0, 0.1)) + # Overlaid density plots
  labs(x = "ME-TSO Reliability: Enthusiastic",
       y = "Density") +
  coord_flip()
# Display the plot
print(plot01)
```

Figure 5: Posterior distribution per person and distribution of the reliability estimates for the item: Enthusiastic

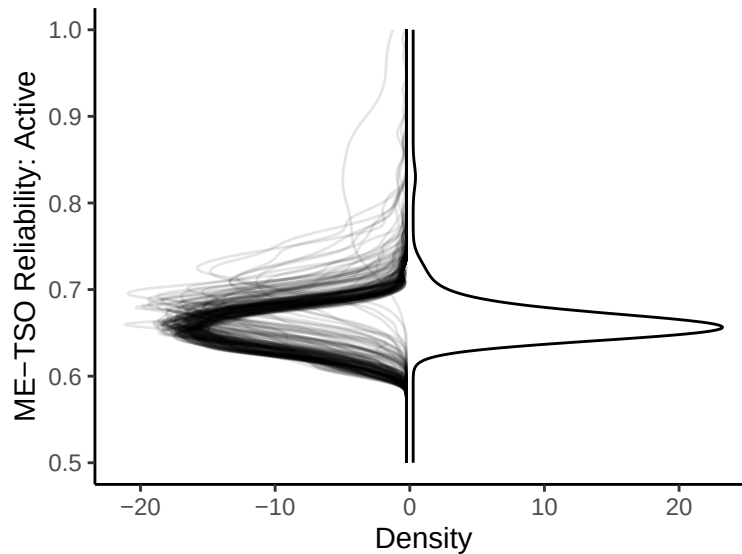


```
tmp <- as.data.frame(metso_rel_samples[, "activ", ])
tmp_long <- reshape(tmp,
                    direction = "long",
                    varying    = colnames(tmp),
                    ids        = row.names(tmp),
                    v.names    = "it")

tmp_long$median <- ave(tmp_long$it,
                      tmp_long$id,
                      FUN = function(x) median(x, na.rm = TRUE))

plot02 <- ggplot(na.omit(tmp_long), aes(x = x)) +
  geom_density(adjust = 16,
              aes(x = median, y = after_stat(density) + 0.25)) +
  xlim(c(0.5, 1)) +
  geom_density(aes(x = it, group = id, y = -after_stat(density) - 0.25),
              color = rgb(0, 0, 0, 0.1)) + # Overlaid density plots
  labs(x = "ME-TSO Reliability: Active",
       y = "Density") +
  coord_flip()
# Display the plot
print(plot02)
```

Figure 6: Posterior distribution per person and distribution of the reliability estimates for the item: Active



2RDM

The last approach is the two-level random dynamic model (2RDM, Xiao et al., 2023). This model was proposed within the DSEM framework and it is basically a multilevel dynamic factor model. In this model, Xiao et al. (2023) proposed estimating one reliability coefficient at the between-level and reliability coefficients per person at the within-level. To fit this model, we used Mplus. As it can be seen in the following steps, the procedure to import and preprocess the output needed to compute the reliability based on the 2RDM is very similar to the one we used for the ME-TSO. In this case, there are no custom functions available to aid in writing the syntax of the model or to compute the between-level and within-level reliability coefficients. To start, we exported the data to Mplus:

```
prepareMplusData(ifit_pa_50, filename = "../Mplus/2rdm_pa.dat", inpfiler = TRUE)
```

Then, we wrote the syntax of the model as shown in Xiao et al. (2023). The final syntax was:

```
cat(readLines("../Mplus/2rdm_pa.inp"), sep = "\n")
```

```
TITLE: Your title goes here
DATA: FILE = "Mplus/2rdm_pa.dat";
VARIABLE:
NAMES = ID day inter excit stron enthu proud inspi deter atten activ alert;
```

```

MISSING=.;
USEVAR = inter excit stron enthuproud inspi deter atten activ alert;
CLUSTER = ID;
TINTERVAL = day(1);

ANALYSIS:
TYPE = TWOLEVEL RANDOM;
ESTIMATOR = BAYES;
BITERATIONS = 200000(120000);
PROCESSORS = 4;
CHAINS = 4;
THIN = 10;

MODEL:

%WITHIN%

! Within person factor with random loadings
wf BY inter@1 (&1);
wl2-wl10 | wf BY excit stron enthuproud inspi deter atten activ alert (&1);

! Random Autoregressive effect
ar | wf ON wf&1 ;

! Random variances of the random measurement error and
! the within person factor.
werr1-werr10 | inter excit stron enthuproud inspi deter atten activ alert;
wvar | wf;

%BETWEEN%

! Between person factor
bf BY inter@1;
bf BY excit stron enthuproud inspi deter atten activ alert (bl2-bl10);

! Between person measurement errors and factor variance.
! Intercepts and factor mean.

[inter excit stron enthuproud inspi deter atten activ alert];
inter excit stron enthuproud inspi deter atten activ alert (berr1-berr10);

[bf@0];
bf (bvar);

```

```

! Means of random effects
[werr1-werr10] (mwe1-mwe10);
[w12-w110] (mw12-mw110);
[ar] (phi);
[wvar] (mwvar);

! Variances of random effects
werr1-werr10 (vwe1-vwe10);
w12-w110 (vw12-vw110);
ar (vphi);
wvar (vwvar);

! No covariances among the random effects and the between person factor

MODEL PRIORS:
! Use similar Priors as in Xiao et al. (2023)
vwe1-vwe10 ~ U(0.00001,3);
vw12-vw110 ~ U(0.00001,0.5);
vwvar ~ U(0.00001,3);
vphi ~ U(0.00001,1);

MODEL CONSTRAINT:
! Use model constraint to estimate between person reliability.
NEW(unexp, explain, brel);
explain = (1 + b12 + b13 + b14 + b15 + b16 + b17 + b18 + b19 + b110)^2 * bvar;
unexp = berr1 + berr2 + berr3 + berr4 + berr5 + berr6 + berr7 + berr8 + berr9 + berr10;
brel = explain/(explain + unexp);

SAVEDATA: BPARAMETERS = 2rdm_pa_samples.dat;
          FILE is 2rdm_pa_fscores.dat;
          SAVE = FSCORES (200, 10);
          FACTORS = all;

DATA IMPUTATION: THIN = 10;

OUTPUT: TECH8;

```

Similar to the ME-TSO, we used the `tinterval` command to make the data approximately equally spaced over time. We also used 4 chains and a thinning of 10 for the MCMC algorithm. However, this time, we increased the minimum number of iterations to 120000. Then, to compute the factor scores, we ran an additional 200 iterations. Furthermore, in this case, the

syntax also computes the between-person reliability, which is defined in the section **MODEL CONSTRAINT**.

Next, we read the output file into R and obtained the total number of observations and participants from this file.

```
# Import output file
rdm_mplus_out <- readLines("../Mplus/2rdm_pa.out")
# Get total number of observations
rdm_nobs <- grep("Number of observations",
                 rdm_mplus_out,
                 value = TRUE)
rdm_nobs <- sub("Number of observations\\s+(\\d+)", "\\1", rdm_nobs)
rdm_nobs <- as.numeric(rdm_nobs)
# Get total number of participants
rdm_nid <- grep("Number of clusters",
               rdm_mplus_out,
               value = TRUE)
rdm_nid <- sub("Number of clusters\\s+(\\d+)", "\\1", rdm_nid)
rdm_nid <- as.numeric(rdm_nid)
```

We also read the factor scores.

```
# Read factor scores
rdm_fscores_raw <- scan("../Mplus/2rdm_pa_fscores.dat", what = "c",
                       na.strings = "*")
rdm_fscores_raw <- as.numeric(rdm_fscores_raw)
rdm_fscores_raw <- matrix(rdm_fscores_raw,
                          nrow = rdm_nobs,
                          ncol = length(rdm_fscores_raw)/rdm_nobs,
                          byrow = TRUE)
```

The matrix `rdm_fscores_raw` had a total of 7188 columns. First, the names of these columns had to be located in the output file.

```
# Get lines where the names of the factor scores are
line_start <- grep("Order and format of variables", rdm_mplus_out)
line_end   <- grep(
  "\\+ variables that have a value for each of the \\d+ imputations",
  rdm_mplus_out)
# Extract number of iterations used to compute the factor scores
n_fs_samples <- sub(
```

```

    ".*\\+ variables that have a value for each of the (\\d+) imputations.*",
    "\\1", rdm_mplus_out[line_end])
n_fs_samples <- as.numeric(n_fs_samples)
# Obtain the names of the factor scores
fscoresnames <- rdm_mplus_out[(line_start + 2):(line_end - 2)]
# Identify names that need to be repeated for the number of iterations
which_samples <- grep("^\\s+\\+", fscoresnames)
# Clean the factor scores names by removing unnecessary spaces
fscoresnames <- gsub("^\\s+|^\\s+\\+|^\\s+_", "", fscoresnames)
fscoresnames <- gsub("\\s{2,}..+", "", fscoresnames)

```

The variable names of the factor scores are incomplete as Mplus only adds a sign to the variables that had as many columns as there were iterations to compute the factor scores. For this reason, the variable names found in the output file have to be extended to account for samples of the factor scores.

```

fscoresnames <-
lapply(fscoresnames, function(x) {
  if (x %in% fscoresnames[which_samples]) {
    return(paste0(x, sprintf("_i%03d", 1:n_fs_samples)))
  } else {
    return(x)
  }
})

fscoresnames <- unlist(fscoresnames)

```

Once the names of the factor scores were completed, we reduced the data by only selecting the variables of interest. In this case, the variables of interest were the samples of the factor loadings, the autoregressive effect, the measurement error variances, and the innovation variance.

```

# Rename variables in the factor scores
colnames(rdm_fscores_raw) <- fscoresnames
# Get the first row for each participant in the data
firstrow_id <- match(unique(rdm_fscores_raw[, "ID"]),
                     rdm_fscores_raw[, "ID"])
# Simplify dataset of the factor scores
rdm_fscores_pa <- rdm_fscores_raw[firstrow_id,
                                   grep("WL\\d+_i|WERR\\d+_i|^AR_i|WVAR_i",
                                       fscoresnames)]
row.names(rdm_fscores_pa) <- unique(rdm_fscores_raw[, "ID"])

```

By extracting the factor scores of these parameters, we have everything that is needed for the computation of the within-person reliabilities of the 2RDM. To facilitate working with the data, we turn the samples of the factor scores into an array:

```
rdm_fscores_array_pa <- array(
  rdm_fscores_pa,
  dim = c(rdm_nid, n_fs_samples, 21),
  dimnames = list(ids = unique(rdm_fscores_raw[, "ID"]),
    iter = paste0("it", 1:n_fs_samples),
    varname = c(paste0("WL", 2:I), "AR",
      paste0("WERR", 1:I), "WVAR")))
```

When there are random variances in Mplus, these are estimated on the log scale. Therefore, before continuing with the computations, we transformed the variances back to the normal scale.

```
# Transform variances in log scale back to normal
rdm_fscores_array_pa[, , c(paste0("WERR", 1:I), "WVAR")] <-
  exp(rdm_fscores_array_pa[, , c(paste0("WERR", 1:I), "WVAR")))
```

In the following loop, we computed the samples of the within-person reliability coefficients based on the factor score samples stored in `rdm_fscores_array_pa`.

```
rdm_wrel_pa <- matrix(NA, rdm_nid, n_fs_samples)

for (i in 1:n_fs_samples) {
  tmp <- rdm_fscores_array_pa[, i, ]
  # Observed Variance
  ovar <- tmp[, "WVAR"]/(1 - tmp[, "AR"]^2)
  # Squared sum of loadings
  sqload <- rowSums(cbind(1, tmp[, paste0("WL", 2:10)]))^2
  # Sum of measurement error
  ersum <- rowSums(tmp[, paste0("WERR", 1:10)])

  # Compute within person reliability
  rdm_wrel_pa[, i] <- (sqload * ovar)/(sqload * ovar + ersum)
}
rm(i, tmp)
```

The result of this was a matrix with 172 rows (one row per person) and 200 columns (one column per iteration). Using this matrix, we easily computed the estimated within-person reliability per person (i.e., the median of the posterior distribution).

```
rdm_wrel_pa <- as.data.frame(rdm_wrel_pa)
names(rdm_wrel_pa) <- paste0("it_", 1:n_fs_samples)
row.names(rdm_wrel_pa) <- unique(rdm_fscores_raw[, "ID"])
# Remove within-person reliability estimates lower than 0
rdm_wrel_pa[rdm_wrel_pa < 0] <- NA
# Compute the median across iterations to obtain the reliability estimates
rdm_wrel_pa_sum <- apply(rdm_wrel_pa, 1, median, na.rm = TRUE)
```

When looking at the within-person reliability coefficients based on the 2RDM, the mean reliability across persons was 0.86 and the median was 0.9. Moreover, the minimum and the maximum reliability coefficients were 0.16 and 0.98, respectively. In contrast, the between-person reliability (found in the output file) was estimated at 0.991 with a credibility interval of (0.989, 0.993).

Lastly, we also plotted the posterior distributions of the within-person reliability coefficients alongside the density of the estimated within-person reliability coefficients. Again, the densities of the posterior distributions of the within-level reliability coefficients are shown on the left side for each individual and the distribution of the estimated within-level reliability coefficients is shown on the right side.

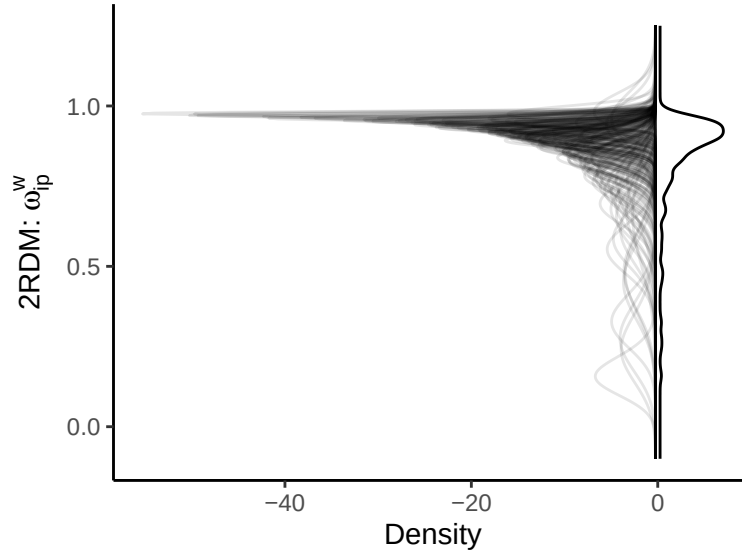
```
tmp <- as.data.frame(rdm_wrel_pa)
tmp_long <- reshape(tmp,
                    direction = "long",
                    varying   = colnames(tmp),
                    ids       = row.names(tmp),
                    v.names   = "it")

tmp_long$median <- ave(tmp_long$it,
                      tmp_long$id,
                      FUN = function(x) median(x, na.rm = TRUE))

plot <- ggplot(na.omit(tmp_long), aes(x = x)) +
  geom_density(adjust = 2,
               aes(x = median, y = after_stat(density) + 0.25)) +
  xlim(c(-0.1, 1.25)) +
  geom_density(adjust = 4,
               aes(x = it, group = id, y = -after_stat(density) - 0.25),
               color = rgb(0, 0, 0, 0.1)) + # Overlaid density plots
  labs(x = expression(paste("2RDM: ", omega[ip]^w)),
       y = "Density") +
  coord_flip()
```

```
# Display the plot
print(plot)
```

Figure 7: Posterior distribution per person and distribution of the within-person reliability estimates based on the 2RDM



Comparing the Person-Specific Reliability Coefficients

Three of the approaches discussed in these supplementary materials estimate reliability coefficients per person. These approaches were the P-technique factor analysis and the dynamic factor analysis (Fuller-Tyszkiewicz et al., 2017; Hu et al., 2016), the ME-TSO (Castro-Alvarez et al., 2022), and the 2RDM (Xiao et al., 2023). The comparison between the reliability coefficients based on the P-technique/dynamic factor analysis and the 2RDM is straightforward because these approaches estimate one coefficient per participant. In contrast, the comparison to the ME-TSO is more difficult because in the ME-TSO there is a reliability coefficient per person and item. Furthermore, while in the P-technique/dynamic factor analysis and the 2RDM, the (within-person) reliability coefficients account only for the within-person variance, in the ME-TSO, the reliability coefficients are more general as they account for stable (trait) and non-stable (state) sources of variance. Nonetheless, to compare the estimates based on these approaches, we plotted the estimated reliability coefficient against each other. For the ME-TSO, we selected two items for this comparison, the item with the mean highest reliability, *enthusiastic*, and the item with the mean lowest reliability, *active*.

For the ME-TSO and the 2RDM, we first had to reorder the reliability estimates to match the order of the *id* variable given that Mplus modifies the order of the participants and their

estimates during the estimation of the model.

```
metso_rel_sum <- metso_rel_sum[order(as.numeric(row.names(metso_rel_sum))), ]
rdm_wrel_pa_sum <- rdm_wrel_pa_sum[order(as.numeric(names(rdm_wrel_pa_sum)))]

rdm_wrel_pa_sum <- as.data.frame(rdm_wrel_pa_sum)
names(rdm_wrel_pa_sum) <- "rdm_wrel"
```

Finally, in the following Figure, we present eight scatter plots that compare the reliability estimates across approaches. The first row presents the reliability estimates of the two items based on the ME-TSO against the reliability estimates of the scale based on the P-technique factor analysis. The second and third rows do the same but against the reliability estimates based on the dynamic factor analysis and the 2RDM, respectively. The last row compares the reliability estimates based on the 2RDM against the reliability estimates based on the P-technique factor analysis on the left and against the reliability estimates based on dynamic factor analysis on the right. The comparison between the reliability estimates based on the P-technique factor analysis and the dynamic factor analysis was shown in Figure 3.

```
tmp <- data.frame(pfa_pa_omega,
                  dynr_dfa_pa_omega,
                  metso_rel_sum,
                  rdm_wrel_pa_sum)

p1 <- ggplot(tmp, aes(x = pfa_omg, y = enthu)) +
  geom_point() +
  geom_smooth(method='lm', colour = gray(1/2),
             linetype = 2, alpha = 0.2, se = FALSE) +
  xlab(expression(paste("PFA: ", omega[i]^w))) +
  ylab("Enthusiastic") +
  xlim(0, 1) +
  ylim(0.75, 1)

p2 <- ggplot(tmp, aes(x = pfa_omg, y = activ)) +
  geom_point() +
  geom_smooth(method='lm', colour = gray(1/2),
             linetype = 2, alpha = 0.2, se = FALSE) +
  xlab(expression(paste("PFA: ", omega[i]^w))) +
  ylab("Active") +
  xlim(0, 1) +
  ylim(0.6, 0.85)

p3 <- ggplot(tmp, aes(x = dfa_omg, y = enthu)) +
  geom_point() +
```

```

geom_smooth(method='lm', colour = gray(1/2),
             linetype = 2, alpha = 0.2, se = FALSE) +
xlab(expression(paste("DFA: ", omega[i]^w))) +
ylab("Enthusiastic") +
xlim(0, 1) +
ylim(0.75, 1)
p4 <- ggplot(tmp, aes(x = dfa_omg, y = activ)) +
geom_point() +
geom_smooth(method='lm', colour = gray(1/2),
             linetype = 2, alpha = 0.2, se = FALSE) +
xlab(expression(paste("DFA: ", omega[i]^w))) +
ylab("Active") +
xlim(0, 1) +
ylim(0.6, 0.85)

p5 <- ggplot(tmp, aes(x = rdm_wrel, y = enthu)) +
geom_point() +
geom_smooth(method='lm', colour = gray(1/2),
             linetype = 2, alpha = 0.2, se = FALSE) +
xlab(expression(paste("2RDM: ", omega[ip]^w))) +
ylab("Enthusiastic") +
xlim(0, 1) +
ylim(0.75, 1)
p6 <- ggplot(tmp, aes(x = rdm_wrel, y = activ)) +
geom_point() +
geom_smooth(method='lm', colour = gray(1/2),
             linetype = 2, alpha = 0.2, se = FALSE) +
xlab(expression(paste("2RDM: ", omega[ip]^w))) +
ylab("Active") +
xlim(0, 1) +
ylim(0.6, 0.85)

p7 <- ggplot(tmp, aes(x = pfa_omg, y = rdm_wrel)) +
geom_point() +
geom_smooth(method='lm', colour = gray(1/2),
             linetype = 2, alpha = 0.2, se = FALSE) +
xlab(expression(paste("PFA: ", omega[i]^w))) +
ylab(expression(paste("2RDM: ", omega[ip]^w))) +
xlim(0, 1) +
ylim(0, 1)
p8 <- ggplot(tmp, aes(x = dfa_omg, y = rdm_wrel)) +
geom_point() +

```

```

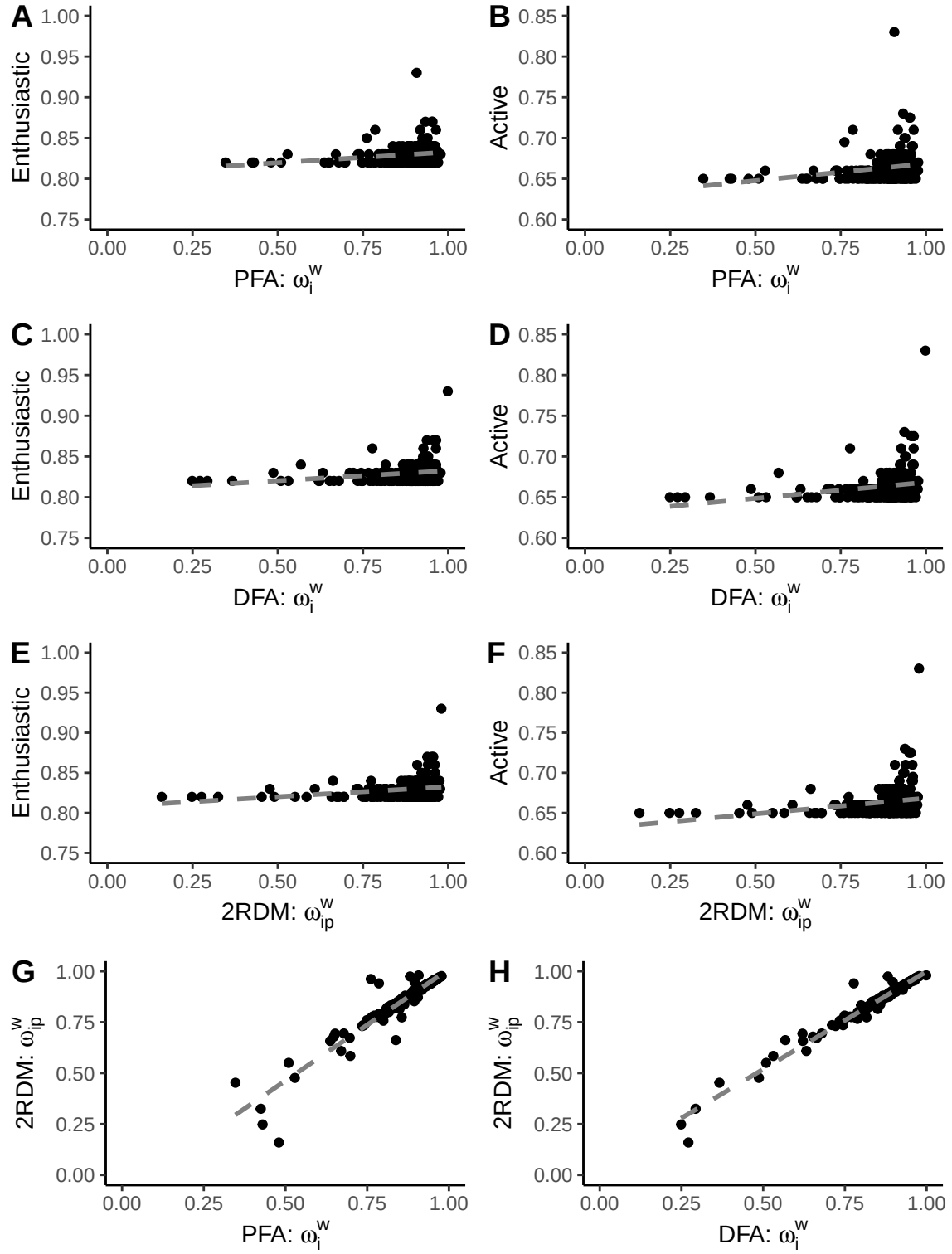
geom_smooth(method='lm', colour = gray(1/2),
             linetype = 2, alpha = 0.2, se = FALSE) +
xlab(expression(paste("DFA: ", omega[i]^w))) +
ylab(expression(paste("2RDM: ", omega[ip]^w))) +
xlim(0, 1) +
ylim(0, 1)

ggpubr::ggarrange(p1, p2, p3, p4, p5, p6, p7, p8,
                  labels = LETTERS[1:8],
                  ncol = 2, nrow = 4)

rm(tmp)

```

Figure 8: Comparison of the person-specific reliability coefficients based on the different approaches.



R Session

R version 4.4.3 (2025-02-28)
Platform: x86_64-pc-linux-gnu
Running under: Ubuntu 24.04.3 LTS

Matrix products: default

BLAS: /usr/lib/x86_64-linux-gnu/blas/libblas.so.3.12.0

LAPACK: /usr/lib/x86_64-linux-gnu/lapack/liblapack.so.3.12.0

locale:

[1] LC_CTYPE=en_US.UTF-8	LC_NUMERIC=C
[3] LC_TIME=en_GB.UTF-8	LC_COLLATE=en_US.UTF-8
[5] LC_MONETARY=en_GB.UTF-8	LC_MESSAGES=en_US.UTF-8
[7] LC_PAPER=en_GB.UTF-8	LC_NAME=C
[9] LC_ADDRESS=C	LC_TELEPHONE=C
[11] LC_MEASUREMENT=en_GB.UTF-8	LC_IDENTIFICATION=C

time zone: Europe/Amsterdam

tzcode source: system (glibc)

attached base packages:

[1] stats	graphics	grDevices	utils	datasets	methods	base
-----------	----------	-----------	-------	----------	---------	------

other attached packages:

[1] dynr_0.1.16-114	esmpack_0.1-20	MplusAutomation_1.1.1
[4] lavaan_0.6-17	misty_0.6.5	psych_2.4.6.26
[7] lme4_1.1-35.5	Matrix_1.7-0	ggplot2_3.5.1
[10] knitr_1.45		

loaded via a namespace (and not attached):

[1] Rdpack_2.6	mnormt_2.1.1	bitops_1.0-7
[4] rlang_1.1.3	magrittr_2.0.3	compiler_4.4.3
[7] mgcv_1.9-1	vctr_0.6.5	reshape2_1.4.4
[10] quadprog_1.5-8	stringr_1.5.1	pkgconfig_2.0.3
[13] shape_1.4.6.1	fastmap_1.1.1	backports_1.4.1
[16] labeling_0.4.3	pbivnorm_0.6.0	pander_0.6.5
[19] utf8_1.2.4	deSolve_1.40	rmarkdown_2.25
[22] pracma_2.4.4	nloptr_2.0.3	purrr_1.0.2
[25] xfun_0.41	glmnet_4.1-8	jomo_2.7-6
[28] jsonlite_1.8.8	pan_1.9	broom_1.0.5
[31] parallel_4.4.3	cluster_2.1.8.1	R6_2.5.1

[34]	stringi_1.8.3	car_3.1-2	boot_1.3-31
[37]	rainbow_3.8	rpart_4.1.24	numDeriv_2016.8-1.1
[40]	Rcpp_1.0.12	iterators_1.0.14	hdrclde_3.4
[43]	splines_4.4.3	nnet_7.3-20	tidyselect_1.2.0
[46]	rstudioapi_0.15.0	abind_1.4-5	yaml_2.3.8
[49]	codetools_0.2-19	lattice_0.22-5	tibble_3.2.1
[52]	plyr_1.8.9	ks_1.14.2	withr_2.5.0
[55]	coda_0.19-4.1	evaluate_0.23	fastDummies_1.7.3
[58]	survival_3.8-3	mclust_6.1.1	texreg_1.39.4
[61]	ggpubr_0.6.0	pillar_1.9.0	fda_6.3.0
[64]	carData_3.0-5	mice_3.16.0	KernSmooth_2.23-26
[67]	checkmate_2.3.2	foreach_1.5.2	stats4_4.4.3
[70]	pcaPP_2.0-4	generics_0.1.3	RCurl_1.98-1.14
[73]	munsell_0.5.0	scales_1.3.0	minqa_1.2.6
[76]	xtable_1.8-4	glue_1.7.0	tools_4.4.3
[79]	fds_1.8	data.table_1.15.4	ggsignif_0.6.4
[82]	gsubfn_0.7	mvtnorm_1.3-1	cowplot_1.1.3
[85]	grid_4.4.3	tidyr_1.3.1	rbibutils_2.2.16
[88]	colorspace_2.1-0	nlme_3.1-168	proto_1.0.0
[91]	cli_3.6.2	latex2exp_0.9.6	fansi_1.0.5
[94]	dplyr_1.1.4	gtable_0.3.4	rstatix_0.7.2
[97]	digest_0.6.34	farver_2.1.1	htmltools_0.5.7
[100]	lifecycle_1.0.4	httr_1.4.7	mitml_0.4-5
[103]	MASS_7.3-65		

References

- Castro-Alvarez, S., Tendeiro, J., Jonge, P. de, Meijer, R. R., & Bringmann, L. (2022). Mixed-effects trait-state-occasion model: Studying the psychometric properties and the person–situation interactions of psychological dynamics. *Structural Equation Modeling: A Multidisciplinary Journal*, 29(3), 438–451. <https://doi.org/10.31234/osf.io/4ext3>
- Cranford, J. A., Shrout, P. E., Iida, M., Rafaeli, E., Yip, T., & Bolger, N. (2006). A Procedure for Evaluating Sensitivity to Within-Person Change: Can Mood Measures in Diary Studies Detect Change Reliably? *Personality and Social Psychology Bulletin*, 32(7), 917–929. <https://doi.org/10.1177/0146167206287721>
- Cronbach, L. J. (1951). Coefficient alpha and the internal structure of tests. *Psychometrika*, 16(3), 297–334. <https://doi.org/10.1007/BF02310555>
- Fuller-Tyszkiewicz, M., Hartley-Clark, L., Cummins, R. A., Tomy, A. J., Weinberg, M. K., & Richardson, B. (2017). Using dynamic factor analysis to provide insights into data reliability in experience sampling studies. *Psychological Assessment*, 29(9), 1120–1128. <https://doi.org/10.1037/pas0000411>
- Geldhof, G. J., Preacher, K. J., & Zyphur, M. J. (2014). Reliability estimation in a multilevel confirmatory factor analysis framework. *Psychological Methods*, 19(1), 72–91. <https://doi.org/10.1037/a0032138>
- Hu, Y., Nesselroade, J. R., Erbacher, M. K., Boker, S. M., Burt, S. A., Keel, P. K., Neale, M. C., Sisk, C. L., & Klump, K. (2016). Test reliability at the individual level. *Structural Equation Modeling: A Multidisciplinary Journal*, 23(4), 532–543. <https://doi.org/10.1080/10705511.2016.1148605>
- Lai, M. H. C. (2021). Composite reliability of multilevel data: It’s about observed scores and construct meanings. *Psychological Methods*, 26(1), 90–102. <https://doi.org/10.1037/met0000287>
- McDonald, R. P. (1999). *Test Theory: A Unified Treatment*. Psychology Press. <https://doi.org/10.4324/9781410601087>
- Molenaar, P. C. M., & Nesselroade, J. R. (2009). The Recoverability of P-technique Factor Analysis. *Multivariate Behavioral Research*, 44(1), 130–141. <https://doi.org/10.1080/00273170802620204>
- Muthén, L. K., & Muthén, B. O. (1998-2023). *Mplus User’s Guide. Eighth Edition*. Muthén & Muthén.
- Nezlek, J. B. (2017). A practical guide to understanding reliability in studies of within-person variability. *Journal of Research in Personality*, 69, 149–155. <https://doi.org/10.1016/j.jrp.2016.06.020>
- O’Laughlin, K. D., Liu, S., & Ferrer, E. (2021). Use of composites in analysis of individual time series: Implications for person-specific dynamic parameters. *Multivariate Behavioral Research*, 56(3), 408–425. <https://doi.org/10.1080/00273171.2020.1716673>
- R Core Team. (2022). *R: A Language and Environment for Statistical Computing*. R Foundation for Statistical Computing.
- Williams, D. R., Martin, S. R., Liu, S., & Rast, P. (2020). Bayesian Multivariate Mixed-Effects

Location Scale Modeling of Longitudinal Relations Among Affective Traits, States, and Physical Activity. *European Journal of Psychological Assessment: Official Organ of the European Association of Psychological Assessment*, 36(6), 981–997. <https://doi.org/10.1027/1015-5759/a000624>

Xiao, Y., Wang, P., & Liu, H. (2023). Assessing intra- and inter-individual reliabilities in intensive longitudinal studies: A two-level random dynamic model-based approach. *Psychological Methods*. <https://doi.org/10.1037/met0000608>