CoTiMA User's Guide:
A package for R to perform
Continuous Time Meta-Analysis

Version 0.8.0

Christian Dormann, Olga Diener, & Markus Homberg

April 25, 2024

Chair of Business Education & Management, Johannes Gutenberg University Mainz, cdormann@uni-mainz.de

Contents

1 General Description

Continuous time meta-analysis (CoTiMA) performs meta-analyses of correlation matrices and/or raw data of repeatedly measured variables. Since variables are measured at discrete time points (e.g., today at 4pm, next week on Monday etc.) this imposes a problem for meta-analysis of longitudinal studies because the time intervals between measurements could vary across studies. However, so-called continuous time math can be used to extrapolate or interpolate the results from all studies to any desired time interval. By this, effects obtained in studies that used different time intervals can be meta-analyzed[1].

A critical assumption is the validity of the underlying causal model that describes the investigated process. CoTiMA is based on a rather general model, which can be restricted on demand. For instance, for a causal system that describes how a single variable that is measured repeatedly (e.g., x_0, x_1, x_2, etc.) develops over time, the default CoTiMA model assumes that x_0 affects x_1, x_1 affects x_2 and so forth. This is called a first order auto-regressive structure. In a two-variable model of x and y, the underlying CoTiMA model is a cross-lagged model with auto-regressive effects for x and y and, in addition, a cross-lagged effect of x_t to y_{t+1} and of y_t to x_{t+1}. Random intercepts cross-lagged panel models (RI-CLPM; e.g., Hamaker et al., 2015) can performed with the CoTiMA R package, too, but certain assumptions have to be met. More complex models (e.g., including x_t to y_{t+1} and x_t to y_{t+2}) can also be meta-analyzed, but they require user-specific adaptations. Restricted versions of the default CoTiMA model (e.g., x_t to y_{t+1} but not y_t to x_{t+1}) are easier to implement and several specific models (e.g., x_t to y_{t+1} exactly of the same size as y_t to x_{t+1}) could be optionally requested. Correlations of primary studies and/or raw data serve as input for CoTiMA and synthesized (i.e., meta-analytically aggregated) effect sizes represent the output of CoTiMA.

```
install.packages("CoTiMA")
library(CoTiMA)
```

Figure 1. Installing CoTiMA from CRAN

CoTiMA is a package for R (R Core Team, 2020). It can be downloaded from CRAN (https://cran.r-project.org) using the code shown in Figure 1. All codes and examples

[1] In a nutshell, CoTiMA fits models to empirical data using the structural equation model (SEM) package ctsem. The effects specified in a SEM are related (constrained) to parameters that are not directly included in the model (i.e., *continuous time parameters*; together, they represent the *continuous time structural equation model*, CTSEM) which is done in a fashion similar to other SEM programs (e.g., like a = b × c to test for mediation in MPLUS) using matrix algebra functions (e.g., matrix exponentiation, which is not available in MPLUS), and statistical model comparisons and significance tests are performed on the continuous time parameter estimates. For details see Dormann et al. (2020).

shown in this User's Guide were performed and tested with R version 4.3.4 and run using RStudio (Posit team, 2024). To get the most current beta version of the CoTiMA package consult Appendix A.

The next pages show how to conduct a CoTiMA. This involves several steps starting with entering primary study information (correlations etc.), initial fitting of a CTSEM to each primary study, fitting the CoTiMA, and plotting the results. We also highlight some common problems frequently encountered during the CoTiMA process.

2 A CoTiMA Example

To prepare a CoTiMA, users have to supply information about *i* primary studies to be meta-analyzed. Primary study information is stored into *objects* (everything in R is an object). Some objects have pre-defined names and are either always mandatory (`delta_ti`), mostly mandatory (`sampleSizei`, `empcovi`), or optional (`pairwiseNi`, `studyNumberi`, `moderatori`, etc., with *i* indicating the study number). User-defined object names could be added (e.g., `criticalRemarki`). Let's generate these objects in R with the code shown in Figure 2 and look closer at them in the next step[2].

The code in Figure 2 is sufficient for a small but nevertheless full CoTiMA based on two variables (Variable 1 = V1, Variable 2 = V2) measured in three primary studies. These studies are illustrated in Figure 3. The cross-lagged effects of earlier V1 on later V2 (*V1toV2*) and vice versa (*V2toV1*) are meant to be meta-analyzed. The first two studies are numbered 1 and 4 in our database. Note that the numeration itself could be arbitrarily chosen, but it should be assigned consistently within every study. Both studies 1 and 4 comprise two variables measured at two measurement occasions, which are represented in a correlation matrix with four rows (`nrow = 4`) and four columns (`ncol = 4`; i.e., a 4×4 correlation matrix; see Figure 2). The correlations reported in primary studies are stored in the objects `empcov1` and `empcov4`, respectively. The third Study 313 has three waves of measurement, and the empirical correlation matrix of Study 313 has, therefore, 6×6 entries. The order of the variables in the correlation matrices has to be V1 at Time 0, V2 at Time 0, V1 at Time 1, V2 at Time 1, etc. Note that in the continuous time literature it is common

[2] When it is desired, all R objects created in the following examples (e.g., `empocv1`, `delta_t1`, etc. in Figure 2 or CoTiMAstudyList_3 in Figure 4) can be created in the user's R environment in two ways. First, the code could be copied directly from this User Guide and then run. Second, the objects are *invisible* but actually available in the package:CoTiMA environment. For example, `empcov1 <- empcov1` copies `empcov1` from the package:CoTiMA environment into the global environment. Afterwards, `rm(empcov1)` removes `empcov1` from the global environment, but it still available in the package:CoTiMA environment. Objects that are available in the package:CoTiMA environment only, but not in the global environment, are not used when the user performs any CoTiMA analyses.

to number time points starting with 0. In the output files generated later, these two variables are labeled *V1* and *V2*. The matrices have to be symmetric. Lack of symmetry is automatically detected by CoTiMA, a warning is issued, and processing is interrupted.

```
empcov1 <- matrix(c(1.00, 0.45, 0.57, 0.18,
                    0.45, 1.00, 0.31, 0.66,
                    0.57, 0.31, 1.00, 0.40,
                    0.18, 0.66, 0.40, 1.00), nrow = 4, ncol = 4)
delta_t1 <- 3
sampleSize1 <- 148
empcov4 <- matrix(c(1.00, 0.43, 0.71, 0.37,
                    0.43, 1.00, 0.34, 0.69,
                    0.71, 0.34, 1.00, 0.50,
                    0.37, 0.69, 0.50, 1.00), nrow = 4, ncol = 4)
delta_t4 <- 3
sampleSize4 <- 88
empcov313 <- matrix(c(1.00, 0.38, 0.54, 0.34, 0.60, 0.28,
                      0.38, 1.00, 0.34, 0.68, 0.28, 0.68,
                      0.54, 0.34, 1.00, 0.47, 0.66, 0.39,
                      0.34, 0.68, 0.47, 1.00, 0.38, 0.72,
                      0.60, 0.28, 0.66, 0.38, 1.00, 0.38,
                      0.28, 0.68, 0.39, 0.72, 0.38, 1.00), nrow = 6, ncol = 6)
delta_t313 <- c(0.5, 0.5)
sampleSize313 <- 335
```

Figure 2. Entering information of three primary studies

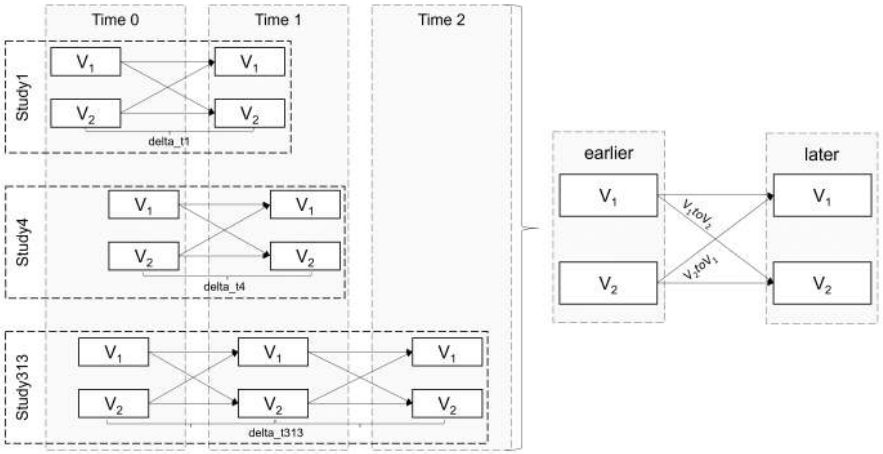

Figure 3. Visualization of a full CoTiMA

In addition to correlation matrices, a CoTiMA requires further information. Researchers need to provide time intervals (`delta_ti`) and sample sizes (`sampleSizei`). Primary Study 1 has a time lag of 3 quarters, which is stored in the object `delta_t1` (see Figure 2). One could also use 0.25 to indicate a quarter of a

one-year lag. Any time scale is possible, but it has to be used consistently across primary studies. It is recommended using a time scale that allows assigning a value of 6 or less to the longest of all time intervals[3], which usually results in better model convergence as we show later. Since Study 313 had three waves of observations, the corresponding two time intervals have to be provided as vector (`delta_t313 <- c(0.5, 0.5)`).

Primary Study 1 further had a sample size of 148, which is stored in the object `sampleSize1` (not `sampleSize01`). In cases, in which correlation matrices include correlations based on pairwise deletion of missing values, sample sizes vary between correlations, too. This could be specified as explained later.

```
activeDirectory <- "../../" # SET A VALID PATH
CoTiMAstudyList_3 <- ctmaPrep(selectedStudies = c(1, 4, 313),
                              activeDirectory = activeDirectory)
saveRDS(CoTiMAstudyList_3, paste0(activeDirectory, "CoTiMAstudyList_3.rds"))
```

Figure 4. Compiling a list of primary studies (ctmaPrep)

After all primary study information was entered, the next step is to compile them into a list[4] and store this list as an R object. This is done with the `ctmaPrep` function included in the CoTiMA R package. Before using `ctmaPrep`, define the `activeDirectory` (where to save results); this can then be used in all subsequent function calls. The created list object (e.g., `CoTiMAstudyList_3` in Figure 4) could be inspected as we demonstrate later. For the moment, it is sufficient to just have it available. Note that all functions provided by the CoTiMA R package start with *ctma* such as `ctmaPrep`. In general, we label the objects where results delivered by ctma-functions are stored starting with *CoTiMA*, such as `CoTiMAstudyList_3`.

After a list of primary study information has been complied with `ctmaPrep`, the next step is to fit a CTSEM to each primary study in a series of separate models using `ctmaInit`. This step is mandatory for subsequent CoTiMA for several reasons. One of the most important reasons is that at this stage one could check the results and identify possible problems with the data entered as, for example, the choice of a time scale that makes model convergence difficult.

The use of `ctmaInit` is shown in Figure 5. Before using `ctmaInit`, define the `activeDirectory` (where to save results) if not done yet and the number of computer cores to be used with `coresToUse`. This can then be used in all subsequent function calls. `ctmaInit` generates a fit-object `CoTiMAInitFit_3` from

[3] For example, if the longest time interval was 10 years, one could use 5-year intervals as the time scale, and to assign the value 2 to `delta_ti` if Study *i* had a 10-year interval.

[4] A list is a particular R object that is useful to collect a variety of information such as values, vectors, matrices, names etc.

the list of compiled studies `CoTiMAstudyList_3`. The fit-object `CoTiMAstudyList_3` will be used later for aggregating (i.e., meta-analyzing) drift effects, performing moderator analyses, estimating publication bias, calculation of expected power and required samples sizes for different time intervals, plotting, and much more. In virtually all cases, the CoTiMA functions to perform these tasks require `CoTiMAInitFit_3` as an argument. `ctmaInit` requires the number of latent variables (`n.latent`) per measurement occasion to be provided by the user as well as an `activeDirectory`, which is where `ctmaInit` saves the fitted CTSEM models for each primary study. These separate CTSEM model fit files become interesting later. More interesting at this stage it the complied list of all fitted models, which is stored in a fit-object named `CoTiMAInitFit_3`, and which can be saved to disk with `saveRDS`. Using `summary(CoTiMAInitFit_3)` displays the results, of which we selected the most interesting ones in Figure 6.

```
activeDirectory <- "../../" # SET A VALID PATH
CoTiMAInitFit_3 <- ctmaInit(primaryStudies = CoTiMAstudyList_3,
                            n.latent = 2,
                            activeDirectory = activeDirectory,
                            coresToUse = 2)
summary(CoTiMAInitFit_3)
saveRDS(CoTiMAInitFit_3, paste0(activeDirectory, "CoTiMAInitFit_3.rds"))
```

Figure 5. Fitting a ctsem model to each primary study (`ctmaInit`)

```
[[1]]
                                    V1toV1    SE       V2toV1   SE
Study No 1   "Reference not provided" "-0.2048" "0.0465" "0.0343" "0.0398"
Study No 4   "Reference not provided" "-0.132"  "0.0444" "0.0228" "0.0426"
Study No 313 "Reference not provided" "-1.249"  "0.1266" "0.4289" "0.1215"

                                    V1toV2    SE       V2toV2    SE
                                    "-0.0784" "0.0358" "-0.1079" "0.0353"
                                    "0.0438"  "0.0422" "-0.1486" "0.0444"
                                    "0.2777"  "0.1056" "-0.8499" "0.0954"

[[2]]
             discrete time          V1toV1 discrete time V2toV1 discrete time
Study No 1   "Reference not provided" "0.8137"           "0.0293"
Study No 4   "Reference not provided" "0.8768"           "0.0198"
Study No 313 "Reference not provided" "0.3066"           "0.1542"

                                    V1toV2 discrete time V2toV2 discrete time
                                    "-0.067"            "0.8965"
                                    "0.0381"            "0.8623"
                                    "0.0998"            "0.4501"
```

Figure 6. CTSEM results (`summary(CoTiMAInitFit_3)`)

The output shown in the first panel `[[1]]` of Figure 6 displays the so-called drift effects. The two auto effects (*V1toV1* & *V2toV2*) are negative as one would expect in continuous time modeling - we explain this later in Section 6. The two cross effects are mostly positive. Note that an effect is regarded as significant if its magnitude is more than 1.96 times its standard error (*SE*). Furthermore, when auto effects (*V1toV1* & *V2toV2*) were not significant, this represents a warning signal that proper

model fit might not be achieved. However, so-called credible intervals are available, too, and should be preferred (not shown in Figure 6). All of this – and a bit more – is displayed after entering `summary(CoTiMAInitFit_3)`. We do not show the full output here due to space reasons. The study numbers are repeated as row names. `"Reference not provided"` just indicates that, yes, we did not provide a reference for each study, which would improve readability of the table. We explain later how to provide references for labeling the output.

The second panel `[[2]]` of Figure 6 displays the discrete time counterparts of the two auto and the two cross effects across one quarter, which was the time scale used when entering primary study information in Figure 2. We explain the relation between continuous time and discrete time effects further below. The reason why we already show the discrete time effects here is that they can be interpreted as ordinary standardized lagged regression coefficients across one quarter. Inspecting the *V1toV1* and *V2toV2* auto-regressive effects shows reasonable effects for all three studies. Experienced readers would usually expect here moderate to high auto-regressive effects due to longitudinal analyses (e. g., .90 for personality variables or .80 for health variables). Small effects (e.g., .03), on the other hand, are very rare and unlikely.

Potential estimation problems are also reported in this summary section. A common cause for potential estimation problems is the user's choice of time scale for the values of `delta_ti`. While bigger time lags, for example, 12 months instead of 1 year, are sometimes a bit too large to ensure proper convergence, CoTiMA works extremely well for time lags in the range of 0.1 to 6, regardless of the measurement unit. The user is therefore usually well-advised to rescale the time lags when the numerical values used when performing `ctmaPrep` (see Figure 2) are larger than 6, which could be done in Figure 5 by adding the argument `scaleTime` (e.g., `scaleTime = 1/12` for months) – we explain this later in more detail.

A full CoTiMA, with *full* indicating that all drift parameters are simultaneously aggregated, is conducted by the code in Figure 7. The summary function displays a couple of results that we present here in reduced form and in two subsequent steps. Results not shown here are explained later.

```
activeDirectory <- "../../" # SET A VALID PATH
CoTiMAFullFit_3 <- ctmaFit(ctmaInitFit = CoTiMAInitFit_3,
                           coresToUse = 2)
saveRDS(CoTiMAFullFit_3, paste0(activeDirectory, "CoTiMAFullFit_3.rds"))
summary(CoTiMAFullFit_3)
```

Figure 7. Conducting a full CoTiMA (`ctmaFit`)

We reduced the `$estimates` section in Figure 8 compared to the actual output displayed on screen. Reason is that among the whole lot of estimates presented, only the four drift effects are of major interest. These are the meta-analytically aggregated effects as indicated by the additional label *invariant*. Invariant means that an effect

does not vary among primary studies and only a single overall effect is estimated. This is similar to traditional fixed effect analysis, where it is also assumed that a single overall (true) effect exists. This is what one usually wants from CoTiMA. We are done. All drift effects are significant by means of the T-values as well as by virtue of their credible intervals.

```
                          row col    Mean     sd    2.5%     50%   97.5% Tvalues
DRIFT V1toV1 (invariant)   1   1 -1.0797 0.1436 -1.3652 -1.0772 -0.8114 -7.5170
DRIFT V2toV1 (invariant)   1   2  0.5824 0.1048  0.3704  0.5826  0.7828  5.5567
DRIFT V1toV2 (invariant)   2   1  0.2816 0.0994  0.0744  0.2832  0.4741  2.8322
DRIFT V2toV2 (invariant)   2   2 -0.4370 0.0964 -0.6490 -0.4302 -0.2629 -4.5353
```

Figure 8. First part of summary(CoTiMAFullFit_3)

The second part of the output generated by `summary(CoTiMAFullFit_3)` is shown in Figure 9. It displays the *minus 2 loglikelihood* (*-2ll*) *value*, the number of estimated parameters (both are important if researchers want to compare nested models), and the optimal lag sensu Dormann and Griffin (2015), across which the effects become largest.

```
$minus2ll
[1] 7311.08

$n.parameters
[1] 22

$opt.lag.orig.time
     [,1] [,2]
[1,]   NA    2
[2,]    2   NA

$max.effects
       [,1]   [,2]
[1,]     NA 0.3036
[2,] 0.1468     NA
```

Figure 9. Second part of `summary(CoTiMAFullFit_3)`

The previous output in Figure 8 informed us that the effect of *V1toV2* is located in Row 2 and Column 1 and, conversely, the effect of *V2toV1* is located in Row 1 and Column 2. In this case, the optimal lag is two quarters for both effects, where the effects (see `$max.effects`) become .1468 for *V2toV1* and .3036 for *V1toV2*. The former seems to be much smaller than the latter, and we explain later how to test if the difference between the two effects is statistically significant.

Effects in continuous time are difficult to interpret. Therefore, they are usually translated into discrete time effects. More specifically, they are usually translated into the cross-lagged regression coefficients that can be expected across a range of different time intervals. This is achieved when plotting a CoTiMA fit-object (or several of the CoTiMA fit-objects in the same plot). Figure 10 shows how to plot both the effects of the three separately fitted primary studies and the aggregated effect into single figures. Actually, since there are four effects (auto effect *V1toV1*, auto

effect *V2toV2*, cross effect *V1toV2*, and cross effect *V2toV1*), four figures will be created.

```
plot(ctmaFitList(CoTiMAInitFit_3, CoTiMAFullFit_3),
     timeUnit = "Quarters",
     timeRange = c(1, 48, 1))
```

Figure 10. Plotting a full CoTiMA (`plot`)

To inform the plot function that we want to plot multiple CoTiMA fit-objects simultaneously, they have to be combined using the CoTiMA function `ctmaFitList`. For labeling of the x-axis, the time unit is defined by `timeUnit = "Quarters"` ranging from 1 to 48 in 1-quarter steps (the smaller the steps, the smoother the plot). For the effect *V1toV2*, the resulting plot is shown in Figure 11.

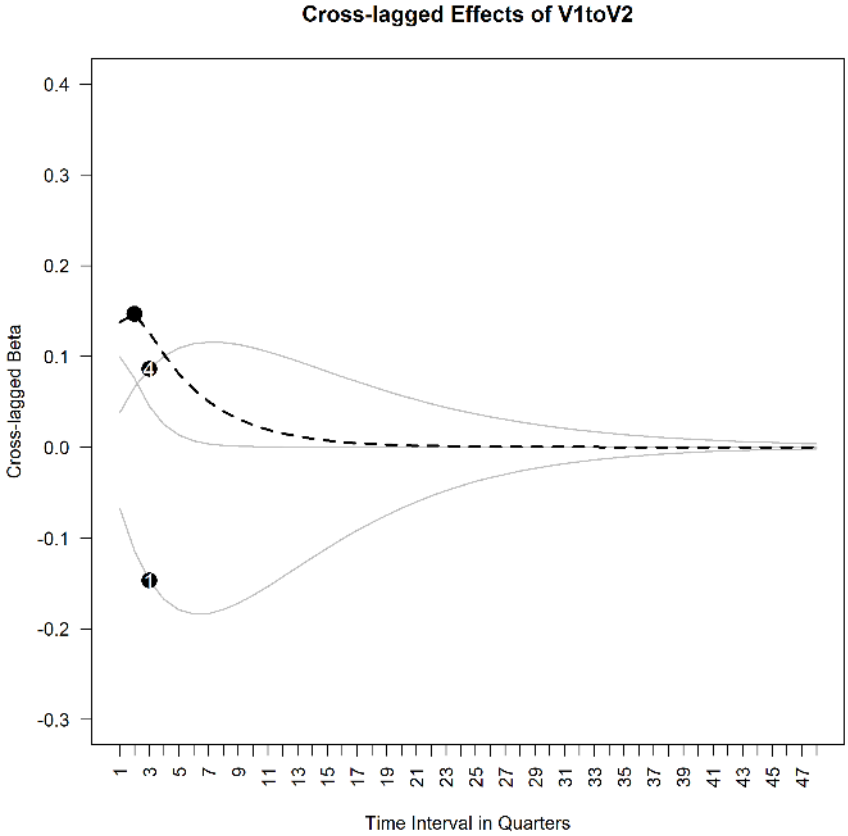

Figure 11. The cross-lagged effect *V1toV2* across 1 to 48 quarters

As can be seen, the dashed black line that represents the aggregated effect reaches its maximum across time intervals of two quarters, where it can be expected to be .1468 (see Figure 11 and compare to the results shown in Figure 9), and then becomes smaller eventually approaching zero. It is noteworthy, albeit not occurring very often and probably limited to CoTiMAs with very few primary studies, that the aggregated effect does not always have to be somewhere in between the smallest and largest effects observed among the primary studies. CoTiMA does not aggregate by taking a (weighted) average of single effects. Rather, it optimizes estimates of all effects simultaneously by minimizing the loglikelihood value of the fit-function, and the single set of the two auto effects and the two cross effects best explains the observed correlations across the three primary studies.

CoTiMA could be used to answer much more research questions than demonstrated up to this point. Capabilities include traditional fixed and random effects analyses, analyses of publication biases, assessing heterogeneity, comparing effect sizes within models, moderator analysis, and analysis of statistical power. However, for CoTiMA like for any kind of meta-analysis, the most time-consuming work is data collection and data management. Therefore, the two next sections deal with this topic. We make several recommendations of how to proceed and we introduce further functions and capabilities of CoTiMA, which could make the life of a meta-analyst more convenient. Subsequent sections then address additional types of analyses that could be conducted after a full CoTiMA.

3 EPIC-BiG-Power: A Recommended CoTiMA Workflow

Our recommended CoTiMA workflow can be summarized with the acronym EPIC-BiG-Power, which stands for **E**xtract, **P**repare, **I**nitFit, **C**oTiMAs, **Bi**as & **G**eneralizability, and statistical **Power**.

1. **E**PIC: **E**xtract correlations from the literature and save them to disk. There is no particular `ctma`-function available supporting this step. It is hard work! We make some suggestions in Section 4.
2. E**P**IC: In a **P**reparatory step, combine variables, correct correlations, add further study information, and add raw data if available. Finally, combine all information by compiling a *list* of primary studies to be used for subsequent analysis using `ctmaPrep` (and `ctmaEmpCov` if useful). This is elaborated in Section 5.
3. EP**I**C: Perform a series of **I**nitial fits, in which each primary study out of the list of primary studies is used to fit a CTSEM using `ctmaInit`. This is demonstrated in Section 6.
4. EPI**C**: The fit-object created in Step 3 is typically used to perform a **C**oTiMA using `ctmaFit`. This is the core of CoTiMA!

(a) We show how to perform a full CoTiMA in Subsection 7.1, in which an entire drift matrix is aggregated.
(b) In Subsection 7.2 a partial CoTiMA is demonstrated, in which subsets of drift coefficients are aggregated.
(c) To address the question whether two (or more) drift effects (e.g., the 2 cross effects) estimated in Step 4(b) are identical, or if one effect is significantly larger than the other one, use the CoTiMA fit-object delivered in Step 4(b) and `ctmaEqual` to test this. See Subsection 7.3 for details.
(d) To address the question whether one (or more) drift effects are moderated by certain characteristics of the primary studies (e.g., the year when they were published), use the CoTiMA fit-object delivered in Step 3 and `ctmaFit` to test this. See Subsection 7.4.

5. **BiG**: Analysis of publication **Bi**as including possible corrections can also be performed. Further, various measure of heterogeneity, which allow answering the question if effects could be **G**eneralized, are reported. This also involves z-curve analysis. Classical fixed and random effects of each single drift effect (not as a set) are estimated, too. Use the CoTiMA fit-object delivered in Step 3 and `ctmaBiG` to test this. This is demonstrated in Section 8.
6. **Power**: Calculation of the statistical (post hoc) **P**ower of the cross effects in each primary study (using the CoTiMA results as true effect estimates) as well as required sample sizes for future studies using a range of different time intervals could be performed, using by the CoTiMA fit-object delivered in Step 3 and `ctmaPower`. This is demonstrated in Section 9.
7. Results of the different analyses could be plotted with `plot(CoTiMaFitObjects)`. Funnel and forest plots will be created if `CoTiMaFitObjects` is a CoTiMA fit-object delivered by `ctmaBiG`. Plots of required sample sizes are delivered if `CoTiMaFitObjects` is a CoTiMA fit-object delivered by `ctmaPower`. Discrete time cross-lagged and auto-regressive effect size plots will be created if `CoTiMaFitObjects` is a CoTiMA fit-object delivered by `ctmaInit` or `ctmaFit`. This is demonstrated throughout 6 – Section 9.

4 Extraction of Correlations from the Literature

In the previous example, we used only the mandatory objects (`delta_t`*i*) and objects that are probably required in most instances (`sampleSize`*i*, `empcov`*i*). We show later how data management can be improved by using further objects. This section starts, however, with some recommendations and helpful functions that can make data entry easier and that offer new possibilities.

One of the most laborious steps is entering the correlation matrices of primary studies. Although it would be less laborious to enter only lower triangular correlation matrices, the requirement to have full correlation matrices serves to double check if correlations are entered correctly. Small typographical errors could have large consequences such as time-consuming and poor convergence in fitting the model to the data.

Although it is preferred to analyze correlation matrices in meta-analyses rather than covariances, the option to analyze covariances is available; CoTiMA automatically switches to the analysis of covariances if vectors of variances (`empVari`) are provided. This is, however, not recommended because different variances imply that effect sizes between studies are on different scales, making aggregated effects impossible to interpret. Similarly, empirical mean values for all variables (`empMeansi`) could be provided, but we do not address these possibilities here.

```
empcov128 <- matrix(c(
          1.00,  0.48,  0.50,  0.50,  0.43,  0.40,  0.39, -0.51, -0.45,
          0.48,  1.00,  0.17,  0.23,  0.22,  0.00,  0.01, -0.10, -0.08,
          0.50,  0.17,  1.00,  0.63,  0.42,  0.45,  0.44, -0.52, -0.41,
          0.50,  0.23,  0.63,  1.00,  0.65,  0.59,  0.50, -0.50, -0.37,
          0.43,  0.22,  0.42,  0.65,  1.00,  0.49,  0.64, -0.41, -0.41,
          0.40,  0.00,  0.45,  0.59,  0.49,  1.00,  0.75, -0.54, -0.46,
          0.39,  0.01,  0.44,  0.50,  0.64,  0.75,  1.00, -0.48, -0.57,
         -0.51, -0.10, -0.52, -0.50, -0.41, -0.54, -0.48,  1.00,  0.70,
         -0.45, -0.08, -0.41, -0.37, -0.41, -0.46, -0.57,  0.70,  1.00), nrow = 9, ncol = 9)
pairwiseN128 <- matrix(c(
          100,  99,  88, 77,  66,  55,  44,  33,  22,
           99,  99,  99, 88,  77,  66,  55,  44,  33,
           88,  99,  88, 99,  88,  77,  66,  55,  44,
           77,  88,  99, 77,  99,  88,  77,  66,  55,
           66,  77,  88, 99,  66,  99,  88,  77,  66,
           55,  66,  77, 88,  99,  55,  99,  88,  77,
           44,  55,  66, 77,  88,  99,  44,  99,  88,
           33,  44,  55, 66,  77,  88,  99,  33,  99,
           22,  33,  44, 55,  66,  77,  88,  99,  22), nrow = 9, ncol = 9)
variableNames128 <- c("SPP_1", "SOP_1",
                      "role stress_1",
                      "exhaustion_1", "exhaustion_2",
                      "cynicism_1", "cynicism_2", "efficacy_1", "efficacy_2")
dimnames(empcov128) <- list(variableNames128, variableNames128)
activeDirectory <- "../../" # SET A VALID PATH
saveRDS(empcov128, paste0(activeDirectory, "empcov128.rds"))
saveRDS(pairwiseN128, paste0(activeDirectory, "pairwiseN128.rds"))
```

Figure 12. Entering correlation matrices

Figure 12 shows an example of how to enter and save correlation matrices. We recommend entering them as they are published and not change any signs or skip variables. This could be easily done later. Although it is no formal requirement, we also recommend labeling the variables (i.e., the row names and column names of the matrices) as they are labelled by the authors of the primary studies. The correlation matrices including the labels are then saved. For demonstration purposes, we change the original matrix reported by Childs and Stoeber (2012) by deleting one variable from the matrix shown in Figure 12. In the original study (Childs & Stoeber, Study 1), the variable role stress_2 was available, but sometimes researchers do not

measure all variable at all time points. Regardless the missing data (correlations) one has to deal with, such primary studies provide useful information (e.g., for estimating auto-regressive effects), and could therefore be included when conducting a CoTiMA.

A further possible challenge for CoTiMA are correlation matrices reported in primary studies that are based on pairwise deletion of missing values. One possible problem is that such matrices might not be suited at all for analysis if they are not positive definite. This cannot happen with listwise deletion. A not positive definite matrix is given, for example, if the correlation between A and B is $r = .90$, between A and C it is $r = .80$, and between B and C it is $r = .10$. Given the two large correlations, such a small correlation is impossible if all correlations are based on identical samples. If a matrix is not positive definite, we recommend contacting the authors of the primary study and ask for a correlation matrix based on listwise deletion, or for raw data. Another option is to drop one or more variables from the correlation matrix. One could check if the matrix is positive definite after dropping variables; the code `eigen(empcov128)$values` should deliver only positive eigenvalues then.

A second challenge resulting from pairwise deletion of missing values in primary studies is the sample size to be used for CoTiMA. Sometimes, authors report the range of pairwise *N* (e.g., pairwise $N = 22$ to 100) in a table note. We recommend using the smallest value then (e.g., `sampleSize128 = 22`), which prevents *SE*s from being estimated much lower than they actually are. Sometimes, however, authors report pairwise *N* for each correlation. Thus, we also have a matrix of pairwise *N*, which we illustrate in Figure 12. Recall that we also have to deal with the entirely missing variable role stress_2. Using a matrix of pairwise *N* rather than just the smallest of all *N* increases the statistical power of a CoTiMA. We recommend saving the matrix to disk (see Figure 12).

5 Preparatory Step (`ctmaEmpCov`, `ctmaCorRel`, `ctmaPrep`)

CoTiMA uses correlation matrices to generate *pseudo raw data* (also known as *synthetic data*; cf. Grund et al., 2022) using the MASS R package (Veneables & Ripley, 2002). Pseudo raw data exactly (!) reproduce the correlation matrices and offer a couple of interesting options. In the present section we show how data can be processed in terms of recoding variables, combining two or more variables into composite (mean) scores, and dealing with missing correlations.

We turn now to processing the correlations shown in Figure 12. Our aim is to analyze the reciprocal effects between job demands and burnout. In particular, we (1) want to correct the correlations for unreliability (aka correction for attenuation or disattenuation). Further, we (2) want to drop the variables SPP_1 and SOP_1

because these variables do not exist in other primary studies and because they are not of particular interest. We also (3) want to recode efficacy_1 and efficacy_2 so that they represent lack of efficacy rather than efficacy. Lack of efficacy, cynicism, and exhaustion are the three burnout symptoms, and we (4) want to combine them into a single variable[5]. Whereas a measure of demands is available for the first measurement occasion (role stress_1), such a measure is missing at the second measurement occasion. Thus, we (5) also have to deal with missing correlations.

To achieve our aims, we start with preparing the relevant data using the code shown in Figure 13. Note that the only computation done here is correction for unreliability using `ctmaCorRel`. No further computations are done until the CoTiMA function `ctmaEmpCov` in Figure 14 is applied. Here in Figure 13 we only prepare the required objects in R.

```
activeDirectory <- "../../" # SET A VALID PATH
empcov128 <- readRDS(paste0(activeDirectory, "empcov128.rds"))
pairwiseN128 <- readRDS(paste0(activeDirectory, "pairwiseN128.rds"))
delta_t128 <- 1.5
alphas128 <- c(.87, .88, .80, .94, .91, .88, .95, .81, .88)
empcov128 <- ctmaCorRel(empcov128, alphas128)
targetVariables128 <- c("role stress_1",
                        "exhaustion 1", "cynicism 1", "efficacy 1",
                        "exhaustion_2", "cynicism_2", "efficacy_2")
recodeVariables128 <- c("efficacy_1", "efficacy_2")
sampleSize128 <- mean(pairwiseN128)
combineVariables128 <- list("role stress_1",
                            c("exhaustion_1", "cynicism_1", "efficacy_1"),
                            c("exhaustion_2", "cynicism_2", "efficacy_2"))
combineVariablesNames128 <- c("Demands1", "Burnout1", "Burnout2")
missingVariables128 <- c(3)
```

Figure 13. Processing correlation matrices (`ctmaCorRel`)

We begin with reading the previously saved correlation matrix and the matrix of pairwise N from disk (see Figure 13) and assign them to R objects `empcov128` and `pairwiseN128`. With `colnames(empcov128)` (not shown in Figure 13) we could recall the variable names, which are `"SPP_1"`, `"SOP_1"`, `"role stress_1"`, `"exhaustion_1"`, `"exhaustion_2"`, `"cynicism_1"`, `"cynicism_2"`, `"efficacy_1"`, and `"efficacy_2"`.

First, we do the corrections for unreliability. This has to be done first because, for example, reliabilities would be no longer available after two or more variables are combined. To correct for unreliability, a vector of reliabilities (`alpha128`) has to be provided from primary studies, and then the `ctmaCorRel` is used to replace

[5] CoTiMA could also be used with measurement models, for example, with lack of efficacy, cynicism, and exhaustion as manifest indicators of a latent factor. However, in meta-analysis the most common case is that burnout would be measured using different (numbers of) variables. Therefore, combining the available variables for each primary study and then using a single manifest indicator in subsequent CoTiMA is frequently the only viable way.

`empcov128` by its disattenuated counterpart. Note that we usually do *not* recommend disattenuating correlations[6]!

Second, we reduce the number of variables. All variables except the two we want to drop (SPP_1 and SOP_1) are assigned to `targetVariables128`. Note that a formal requirement of CoTiMA is that the variables are ordered in Time (Time 0 variables, Time 1 variables, etc.). This is also achieved by ordering the variables accordingly when creating `targetVariables128`.

Third, the two variables we want to recode are assigned to the object `recodeVariables128`. If an `empcov`*i* does not include variable names (no dimnames), one could use the variables' positions (i.e., `recodeVariables128 <- c(4, 7)`). Note that if numbers are used, they should correspond to the positions in the `targetVariables`*i* object rather than the rows/columns in the `empcov`*i* object (i.e., recoding is done after `targetVariables`*i* were selected from `empcov`*i*). Although it is not necessary to assign any value to `sampleSize128`, we assigned the mean of the pairwise *N* (`mean(pairwiseN128)`), as a rough indicator of the overall contribution of the primary study to the result of CoTiMA. This is a reasonable value that will be used for descriptive statistics in the output of subsequent CoTiMAs. Other options could be `min(pairwiseN128)` or `max(pairwiseN128)`.

Fourth, we use a list (!) of variable names or vectors of variable names to define the variables that should or should not be combined. This list is stored in the object `combineVariables128`. We keep the variable role stress_1 as it is, whereas for the first and second measurement occasion the three burnout variables are combined into a single scale, respectively. The three final variables are then labeled as defined in `combineVariablesNames128`.

Fifth, since there is no variable for demands at the second time point, we declare it as missing. This is done by stating which variable is missing in the imagined set of Demands1, Burnout1, Demands2, Burnout2, which is the 3rd element. Thus, `missingVariables128 <- 3`.

[6] Correlations are disattenuated using the well-known formula developed by Spearman (1904). This formula is based on several assumptions. One of these assumptions is that underlying Cronbach's alpha (or any other estimate of reliability), which is usually used to measure reliability, are correct. While violations of the assumptions do usually not cause visible consequences when dealing with a single cross-sectional correlation coefficient, in the case of correlation matrices of longitudinal studies it might cause problems. One problem is that disattenuated test-retest correlations could become larger than 1.0, which is automatically corrected by `ctmaCorRel` (i.e., they are set to 1.0). Another problem is that the disattenuated matrices might not positive definite and could not be analyzed then.

```
results128 <- ctmaEmpCov(targetVariables = targetVariables128,
                         recodeVariables = recodeVariables128,
                         combineVariables = combineVariables128,
                         combineVariablesNames = combineVariablesNames128,
                         missingVariables = missingVariables128,
                         n.latent = 2,
                         pairwiseN = pairwiseN128,
                         Tpoints = 2,
                         empcov = empcov128)
empcov128 <- results128$rNew
pairwiseN128 <- results128$pairwiseNNew
```

Figure 14. Convert correlation matrices (`ctmaEmpCov`)

The CoTiMA package comes with the function `ctmaEmpCov`, which performs the desired operations (recoding, combining etc.) and yields the final correlation matrix that we want to use for our subsequent CoTiMA (see Figure 14). Bevor using this function no computations were applied to the data. Since we have a matrix of pairwise *N,* this will be processed by `ctmaEmpCov`, too. Note that a common problem resulting from copying/pasting the code in Figure 14 is failure to adjust the `Tpoints`. The function `ctmaEmpCov` returns a new correlation matrix, which is then used to replace the `empcov128` from which we started. Further, `ctmaEmpCov` returns a new matrix of pairwise *N*, which is then used to replace the `pairwiseN128`. Figure 15 shows the new correlation matrix and matrix of pairwise *N*.

```
          [,1]      [,2] [,3]      [,4]
[1,] 1.0000000 0.7361878   NA 0.5809288
[2,] 0.7361878 1.0000000   NA 0.8118634
[3,]        NA        NA   NA        NA
[4,] 0.5809288 0.8118634   NA 1.0000000

     [,1] [,2] [,3] [,4]
[1,]   88   77    0   44
[2,]   77   55    0   44
[3,]    0    0    0    0
[4,]   44   44    0   22
```

Figure 15. Results of applying `ctmaEmpCov` to the specifications of Study 128

Instead of correlation matrices, raw data can be used as well, and the arguments required to read raw data from disc have to be stored in a `rawDatai` object (see Figure 16). In R, a `list` is a list (sic!) that has elements, which have their own labels (like in a shopping list, in which you summarize the planned purchases in subitems like vegetables, cheese etc.). Unlike a vector, the elements of a list could be of different types, for example, characters, numbers, symbols, matrices etc. The list-object created in Figure 16 has seven elements: `fileName`, `studyNumbers`, `missingValues`, `standardize`, `header`, `dec`, and `sep`. Note for this example, data preparation has already been done (e. g., combining, eliminating variables). Consult the Appendix B for the `ctmaShapeRawData` function, which can be helpful to get raw data organized in the way required by CoTiMA (or ctsem).

```
activeDirectory <- "../../" # SET A VALID PATH
rawData228 <- list(fileName = paste0(activeDirectory, "rawdata228.txt"),
                   studyNumbers = 228, missingValues = -99,
                   standardize = TRUE, header = TRUE, dec = ".", sep = " ")
delta_t228 <- c(NA)
```

Figure 16. Specification for using raw data

The raw data have to be included in an ordinary text file, and the name of the file should be stored in the list element `fileName` (Note for this example, the raw data were not provided in this user's guide). Possibly missing values should be defined, and only a single value is possible (-99 is assumed by default) and stored in the list element `missingValues`. Whether or not the raw data should be standardized, which implies the analysis of correlations, or not, which implies the analysis of covariance, could be specified by setting the list element `standardize` to either `TRUE` (default and recommended) or to `FALSE`. Whether or not the raw data files include a header with variable names (as for the example data below) could be specified by setting the element `header` to either `TRUE` (default) or to `FALSE`. Finally, a decimal delimiter (default = ".") and the characters separating the values (default = " ") could be defined using the list elements `dec` and `sep`, respectively.

Note that in meta-analysis, moderators are usually study characteristics (e.g., the average age of a sample) rather than characteristics of individual study participants. Therefore, study-level moderator values are not included in raw data files, but they are defined directly for a primary study that does provide raw data by assigning values to the moderator-object; this is explained later[7]. Figure 17 shows the raw data file structure corresponding to the code used in Figure 16.

```
V1_T0   V2_T0   V1_T1   V2_T1   dT1
  0.835   2.328  -0.778   2.969 11
  1.555   2.634   1.977   1.807 12
  3.209   1.849   2.291   2.795 12
  0.416   2.351   0.127   1.705 13
-99.000 -99.000   0.476 -99.000 13
-99.000 -99.000   0.854 -99.000 11
-99.000 -99.000 -99.000   2.987 12
-99.000 -99.000 -99.000   2.087 12
-99.000 -99.000 -99.000   0.927 13
```

Figure 17. Raw data file structure

Raw data of a primary study has to be provided as a text (ascii) file. Data has to be in wide format (i.e., one row per individual). Assuming there are *t* measurement occasions, the order of the variables should be V1_T0, V2_T0, …, V1_Tt, V2_Tt, dT1, dT2, …dT(*t*-1), where dT*t* are the variables representing the time intervals (deltas) between measurements (see Figure 17). Note that if *t* measurement occasions

[7] Individual-level moderator variables could be modelled if raw data are available (see Appendix B for further details).

exist, there are *t*-1 time intervals. Compared to correlation matrices as input, raw data allow the time intervals to vary between the individuals within a study (average time intervals are automatically reported in CoTiMA fit-objects). However, for studies that supply raw data, it is mandatory to define the `delta_ti` object! It has to have as many NA as the largest number of possible time intervals in the respective study is, for example, in the case of three intervals, `delta_ti <- c(NA, NA)`. In the example in Figure 17 there are only two time points and, thus, one interval `dT1`. Thus, `delta_ti` is indeed the only mandatory object because `rawDatai` could substitute `empcovi` and `pairwiseNi` or `sampleSizei`.

So far, we introduced the objects `delta_ti`, `sampleSizei`, `empcovi`, `targetVariablesi`, `alphasi`, `pairwiseNi`, and `rawDatai`. Further predefined object names are:

- `moderatori`. A vector of numerical values either representing categorical or continuous variables, e.g., `moderator6 <- c(1, 2, 2, 0.76, 2.56, 2001)`
- `empMeansi`. Mean values of variables (default = 0). It is not recommended to change the default, but it is possible, e.g., `empMeans7 <- c(1, -2.5, 1.1, -2.4)`
- `empVarsi`. Variances of variables; (default = 1). It is not recommended to change the default, but it is possible, e.g., `empVars6 <- c(1, 2, 1.1, 1.9)`
- `studyNumberi`. A special number used for labeling in the outputs of subsequently fitted CoTiMA models, e.g., `studyNumber6 <- 66`
- `sourcei`. Useful to label the table displaying the estimated parameters for each primary study, rather than using the numbers used for the primary study-objects (e.g., 128 from `empcov128`), e.g., `source6 <- c("De Jonge", "Dormann", "Janssen", "Dollard", "Landeweerd", "& Nijhuis", "2001")`
- `ageMi`. A value indicating the mean age of participants in a primary study, e.g., `ageM6 <- 31.78`
- `malePercenti`. A value indicating the percentage of male participants in a primary study, e.g., `malePercent6 <- 0.11`
- `occupationi`. A vector of character strings representing the occupations of participants in a primary study. Of course, this has not to be taken literally. For example, it could be also used to represent the program in which student participants are enrolled and similar classifications, e.g., `occupation6 <- c("Health care workers")`
- `countryi`. A single character string representing the country in which a primary study was conducted, e.g., `country6 <- c("Netherlands")`

- `startValuesi`. A vector of start values, which was used in previous CoTiMA versions. Currently the use of start values is disabled, but this might change in the future.

In addition to these pre-defined object names, user-defined object names could be added (e.g., `demandsi` and `burnouti`, to add information about the type of measurement scale used in primary studies). The difference between pre-defined and user-defined objects is twofold. First, pre-defined objects are included in the Excel workbook that summarizes primary study information (see Figure 22). Second, user-defined objects have to be declared in `ctmaPrep` using the argument `addElements` (see Figure 20).

To proceed further with the example, in a first step documented in Figure 18 we add information to those four primary studies data already entered before (Study 1, 4, 128 and 313). In a second step, we add two further primary study information as shown in Figure 19.

```
ageM1 <- 39.3
ageSD1 <- 8.7
malePercent1 <- .60
occupation1 <- c("Bank employees")
country1 <- c("Netherlands")
demands1 <- c("Workload")
burnout1 <- c("Emotional Exhaustion")
targetVariables1 <- c("Demands1", "Burnout1", "Demands1", "Burnout2")
source1 <- c("Houkes, I,", "Janssen, P, P, M,", "de Jonge, J", "& Bakker, A, B",
             "Study1", "2003")
moderator1 <- c(1, 0.72)

ageM4 <- 47.4
ageSD4 <- 5.8
malePercent4 <- .70
occupation4 <- c("Teachers for adults")
country4 <- c("Netherlands")
demands4 <- c("Workload")
burnout4 <- c("Emotional exhaustion")
targetVariables4 <- c("Demands1", "Burnout1", "Demands1", "Burnout2")
source4 <- c("Houkes, I,", "Janssen, P, P, M,", "de Jonge, J", "& Bakker, A, B",
             "Study2", "2003")
moderator4 <- c(1, 0.72)

ageM313 <- 30
ageSD313 <- 6
malePercent313 <- 0.30
occupation313 <- c("Employment agency employees")
country313 <- c("Netherlands")
demands313 <- c("Work pressure")
burnout313 <- c("Exhaustion")
targetVariables313 <- c("Demands1", "Burnout1", "Demands1", "Burnout2",
                        "Demands3", "Burnout3")
source313 <- c("Demerouti", "Bakker", "& Bulters", "2004")
moderator313 <- c(2, 0.72)

ageM128 <- 41
ageSD128 <- 11.4
malePercent128 <- 0.203
occupation128 <- c("Managerial employees in NHS trusts")
country128 <- c("UK")
demands128 <- c("Role Stress")
burnout128 <- c("Exhaustion", "Cynicism")
source128 <- c("Childs, J. H.", "& Stoeber, J.", "Study1", "2012")
moderator128 <- c(2, 0.66)
```

Figure 18. Additional information for studies entered earlier (primary studies 1, 4, 313, & 128)

```
empcov18 <- matrix(c(1.00, 0.44, 0.62, 0.34,
                     0.44, 1.00, 0.41, 0.62,
                     0.62, 0.41, 1.00, 0.55,
                     0.34, 0.62, 0.55, 1.00), nrow = 4, ncol = 4)
variableNames18 <- c("Demands 1",  "Burnout 1", "Demands 2",  "Burnout 2")
dimnames(empcov18) <- list(variableNames18, variableNames18)
delta_t18 <- 3
sampleSize18 <- 174
ageM18 <- 41.33
ageSD18 <- 9.70
malePercent18 <- 0.03
occupation18 <- c("Service employees")
country18 <- c("Germany")
demands18 <- c("Workload")
burnout18 <- c("Emotional exhaustion", "Depersonalization")
source18 <- c("Diestel", "& Schmidt", "Study 1", "2012")
moderator18 <- c(1, 0.7)

empcov32 <- matrix(c(1.00, 0.45, 0.70, 0.40,
                     0.45, 1.00, 0.36, 0.66,
                     0.70, 0.36, 1.00, 0.43,
                     0.40, 0.66, 0.43, 1.00), nrow = 4, ncol = 4)
variableNames32 <- c("Demands_1",  "Burnout_1", "Demands_2", "Burnout_2")
dimnames(empcov32) <- list(variableNames32, variableNames32)
delta_t32 <- 2
sampleSize32 <- 433
ageM32 <- 41.5
ageSD32 <- 10.2
malePercent32 <- 0.199
occupation32 <- c("Teachers")
country32 <- c("Canada")
demands32 <- c("classroom overload")
burnout32 <- c("Emotional exhaustion", "Depersonalization")
source32 <- c("Fernet", "Guay", "Senecal", "& Austin", "2012")
moderator32 <- c(1, NA)
```

Figure 19. Information for two further primary studies (18 & 32)

The six studies are then compiled into a list as shown in Figure 20. Here we add the two user-defined object names `demandsi` and `burnouti`. We also provide a vector with the labels of the two moderators, and we provide a list of vectors to label the moderator values.

```
activeDirectory <- "../../" # SET A VALID PATH
CoTiMAstudyList_6 <- ctmaPrep(selectedStudies = c(1, 4, 313, 128, 18, 32),
                              activeDirectory = activeDirectory,
                              addElements = c("demands", "burnout"),
                              moderatorLabels = c( "Burnout Measure",
                                                   "Control at Work"),
                              moderatorValues = list(c(  "1 = Emotional Exhaustion",
                                                         "2 = Exhaustion"),
                                                     "continuous"))
saveRDS(CoTiMAstudyList_6, paste0(activeDirectory, "CoTiMAstudyList_6.rds"))
```

Figure 20. Compiling a list of primary studies with extended information (`ctmaPrep`)

To get a convenient overview of the information stored in this list, one could use the openxlsx R package (see Figure 21). An example of what is displayed when opening the excel workbook with its several sheets with openXL is shown in Figure 22. The

workbook could also be saved to disk using the `saveWorkbook` function of openxlsx.

```
openXL(CoTiMAstudyList_6$excelSheets)
activeDirectory <- "../../" # SET A VALID PATH
saveWorkbook(CoTiMAstudyList 6$excelSheets, overwrite = TRUE,
             file = paste0(activeDirectory, "myExcelSheet.xlsx") )
```

Figure 21. Open an Excel sheet with summary information included in a compiled list of primary studies (requires package openxlsx)

	A	B	C	D	E	F	G	H	I
1	Source Info 1	Source Info 2	Source Info 3	Source Info 4	Source Info 5	Source Info 6	Orig. Study No.	Moderator # 1	Moderator # 2
2	Houkes, I,	Janssen, P, P, M,	de Jonge, J	& Bakker, A, B	Study1	2003	1	1	0.72
3	Houkes, I,	Janssen, P, P, M,	de Jonge, J	& Bakker, A, B	Study2	2003	4	1	0.72
4	Demerouti	Bakker	& Bulters	2004			313	2	0.72
5	Childs, J. H.	& Stoeber, J.	Study1	2012			128	2	0.66
6	Diestel	& Schmidt	Study 1	2012			18	1	0.7
7	Fernet	Guay	Senecal	& Austin	2012		32	1	
8								Burnout Measure	Control at Work
9								1 = Emotional Exhaustion	continuous
10								2 = Exhaustion	

Figure 22. Excel sheet with summary information included in a compiled list of primary studies

6 Initial Fitting (`ctmaInit`)

Now the first two steps (**E**xtract & **P**repare) in the recommended EPIC-BiG-Power workflow are done and we can move forward to the ***Init*** step, for which the previously compiled `CoTiMAstudyList_6` is required. Initial fitting is done with the code in Figure 23 (analogous to Figure 5), and the result is then displayed on the console (see Figure 24).

```
activeDirectory <- "../../" # SET A VALID PATH
CoTiMAInitFit_6 <- ctmaInit(primaryStudies = CoTiMAstudyList_6,
                            n.latent = 2,
                            activeDirectory = activeDirectory,
                            coresToUse = 2)
summary(CoTiMAInitFit_6)
saveRDS(CoTiMAInitFit_6, paste0(activeDirectory, "CoTiMAInitFit_6.rds"))
```

Figure 23. Fitting a ctsem model for each primary study (`ctmaInit`)

For Study 128, which we used to demonstrate how to deal with missing variables, some unusual estimates (e.g., large *SEs* and non-significant auto effects) emerged, which was not unexpected in this case. In Study 128, which comprised two waves of measurement, the variable V1_T1 was missing (demands T1, i.e., role stress_2). Obviously, this makes it impossible to validly estimate parameter involving V1_T1. These parameters are called *non-identified.* Thus, all estimates involving V1_T1 are not trustworthy. And even if only a single parameter was not identified, consequently the entire model is not identified. Thus, even the seemingly reasonable drift effect *V2toV2* in Figure 24 is not trustworthy. We show later why Study 128 could

nevertheless be used for CoTiMA. Anyway, we will use the current case to review some of the general principles of continuous time structural equation modeling (CTSEM).

```
                                                    V1toV1    SE        V2toV1    SE
Study No 1   "Houkes et al., Study1, 2003"           "-0.208"  "0.0454"  "0.0363"  "0.0389"
Study No 4   "Houkes et al., Study2, 2003"           "-0.1301" "0.0459"  "0.0217"  "0.0422"
Study No 313 "Demerouti et al., 2004"               "-1.2538" "0.1364"  "0.4298"  "0.1195"
Study No 128 "Childs, & Stoeber, Study1, 2012"      "-7.5969" "10.1265" "-0.2537" "1.938"
Study No 18  "Diestel, & Schmidt, Study 1, 2012"    "-0.2166" "0.0421"  "0.1037"  "0.0411"
Study No 32  "Fernet, Guay, Senecal, & Austin, 2012" "-0.2031" "0.0306"  "0.0435"  "0.03"


Study No 1   "Houkes et al., Study1, 2003"           V1toV2    SE       V2toV2    SE
Study No 4   "Houkes et al., Study2, 2003"          "-0.0777" "0.037"  "-0.1081" "0.0359"
Study No 313 "Demerouti et al., 2004"              "0.0438"  "0.0425" "-0.1516" "0.0467"
Study No 128 "Childs, & Stoeber, Study1, 2012"     "0.2761"  "0.1039" "-0.8486" "0.0957"
Study No 18  "Diestel, & Schmidt, Study 1, 2012"   "-0.0991" "0.208"  "-0.1756" "0.1433"
Study No 32  "Fernet, Guay, Senecal, & Austin, 2012" "0.0501"  "0.0396" "-0.1914" "0.0412"
```

Figure 24. Some results for the primary studies (`ctmaInit`)

Table 1. Overview of the parameters/terms used in discrete and continuous time modelling

discrete time	**continuous time**
variable at *Time t* affect variable at *Time t+1* - *auto-regressive* effect: e.g., from $V1_t$ to $V1_{t+1}$ - *cross-lagged* effect: e.g., from $V1_t$ to $V2_{t+1}$	*earlier* variable affect *later* variable - *auto* effect: e.g., from *earlier* V1 to *later* V1 - *cross* effect: e.g., from *earlier* V1 to *later* V2
[structural equations, regression slopes/paths] (matrices gamma $\boldsymbol{\Gamma}$ or beta $\boldsymbol{B}$)	[drift effects] (drift matrix $\boldsymbol{A}$)
(co-)variance of *residuals* [unexplained/ residual/ error variance, structural disturbance] (matrix psi $\boldsymbol{\Psi}$)	(co-)variance of *innovations* [system noise, random change, prediction error] (diffusion matrix $\boldsymbol{G}$)
intercept [constant] (matrix alpha $\boldsymbol{\alpha}$)	*continuous time* intercept (matrices **b**, CINT)
measurement error (matrices theta $\boldsymbol{\Theta}$, $\boldsymbol{\Theta}_\delta$, or $\boldsymbol{\Theta}\varepsilon$)	observational noise [measurement error] (matrix theta $\boldsymbol{\Theta}$)
(co-)variance of *exogeneous variables* (matrix phi $\boldsymbol{\Phi}$)	(co-)variance of *variables at Time 0* [initial (co-)variance] (matrices T0var or T0covar)

Note. Parameters/commonly used terms and phrases, [synonyms], (matrices).

First, in CTSEM any pair of subsequent measurement occasions is regarded as equivalent except the length of the time interval, which may vary. Therefore, continuous time coefficients do not describe, for example, the relations between demands at Time 0 and burnout at Time 1. Rather, *earlier* demands affect *later* burnout. Thus, in CoTiMA, the effect *V1toV1* means the auto effect of earlier V1 to later V1. Similarly, the effect *V1toV2* means the cross effect of earlier V1 to later V2. Note that in continuous time, the terms *auto* effect and *cross* effect are used, whereas in discrete time, the terms *auto-regressive* effect and *cross-lagged* effect are used. In a similar vein, the terms *innovation* and their associated (co-)variances (*diffusion* matrix) in continuous time substitute the term *error* (and *unexplained* variance) in discrete time, and the term *continuous time intercept* substitutes the term *intercept* (cf. Table 1 and for more details see Driver et al., 2017; Voelkle et al., 2012).

Between continuous time and discrete time coefficients, well-defined mathematical relations exist. The only reason why continuous time coefficients are used is that the math is known to describe how coefficients change across time. To translate auto and cross effects into auto-regressive and cross-lagged effects, put the former into a matrix, multiply the matrix by length of time interval, and then apply the matrix (!) exponential function. The resulting matrix contains the auto-regressive effects in the diagonal and the cross-lagged effects off the diagonal (cf. Dormann et al., 2020; Voelkle et al., 2012).

Figure 25 shows how the continuous time drift effects obtained for Study 313 (see Figure 24) relate to 1-quarter auto-regressive and cross-lagged effects in discrete time. Demands have slightly smaller carry-over effects (*V1toV1*) than burnout (*V2toV2*). The negative auto effects in continuous time thus translate into positive auto-regressive effects in discrete time. Thus, in continuous time, the more negative an auto effect is, the smaller are the effects that a variable carries over time. Further, the effect of earlier demands on later burnout is smaller (*V1toV2*) than the effect of earlier burnout on later stressors (*V2toV1*). Note that multiplying the matrix with, for example, 2 (i.e., `expm(A313 * 2)`) yields the effects across a 2-quarter lag (i.e., half a year). This is the way how discrete time effect sizes are computed and plotted (see Figure 11).

```
library(expm)
A313 <- matrix(c(-1.2538, 0.4298, 0.2761, -0.8486), nrow = 2, ncol = 2, byrow = TRUE)
A313
       [,1]     [,2]
[1,] -1.2538   0.4298
[2,]  0.2761  -0.8486

expm(A313 * 1)
      [,1]       [,2]
[1,] 0.30509068 0.1542537
[2,] 0.09909133 0.4505155
```

Figure 25. Relation between continuous time drift coefficients of Study 313 and its discrete time effects across one quarter

The result of applying the same transformation to the suspicious drift effects of Study 128 is shown in Figure 26. The non-identified auto effect *V1toV1* corresponds to an auto-regressive effect of 0.0009 across one quarter: A person's level of demands at work does virtually not predict at all the person's level of demands one quarter later, which one would usually regard as implausible. In fact, this out-of-range estimate is a consequence that in Study 128 later demands was a missing variable. Thus, we cannot expect meaningful results from fitting a ctsem model to Study 128.

```
library(expm)
A128 <- matrix(c(-7.5969, -0.2537, -0.0991, -0.1756), nrow = 2, ncol = 2, byrow = TRUE)
A128
      [,1]    [,2]
[1,] -7.5969 -0.2537
[2,] -0.0991 -0.1756

expm(A128 * 1)
      [,1]           [,2]
[1,]  0.0008838301  -0.02873391
[2,] -0.0112240047   0.84141568
```

Figure 26. Relation between continuous time drift coefficients of Study 128 and its discrete time effects

Again, model results could also be opened as excel workbook with `openXL(CoTiMAInitFit_6$excelSheets)`. For example, effects, their standard errors (*SEs*) and lower limit (*LL*) and upper limit (*UL*) credible intervals are shown in Figure 27. Excel sheet with summary information included in a compiled list of primary studies. From the workbook, coefficients could be easily copied into a word processing app to build proper results tables.

	A	B	C	D	E	F	G	H	I
1	Study	V1toV1LL	V1toV1UL	V2toV1LL	V2toV1UL	V1toV2LL	V1toV2UL	V2toV2LL	V2toV2UL
2	1	-0,3062	-0,1284	-0,0408	0,1148	-0,1483	-0,0108	-0,1915	-0,0549
3	4	-0,2361	-0,0645	-0,0552	0,1039	-0,0421	0,1319	-0,2648	-0,0832
4	313	-1,4976	-1,0342	0,1903	0,6718	0,0571	0,4789	-1,0364	-0,6883
5	128	-122,8276	0	-27,3917	22,0153	-0,198	0,1404	-2,1477	-0,0077
6	18	-0,3069	-0,1446	0,0245	0,183	-0,0267	0,134	-0,282	-0,1237
7	32	-0,2685	-0,1507	-0,0123	0,1075	0,0413	0,167	-0,336	-0,2012

Figure 27. Excel sheet with summary information included in a compiled list of primary studies

Doing the initial fitting of ctsem models to all primary studies allows specifying several arguments, for example, constraining some drift effects to be 0.0, or using different estimators such as Bayesian instead of maximum likelihood estimation (default). The arguments to select estimators are introduced next, and the entire list of possible arguments of the different CoTiMA functions are listed in the Appendix B. Note that the `optimize` argument should be used and not be confused with `optimise`, which is used by ctsem.

One particular option is to use Bayesian estimation. Thus, Bayesian estimates will be drawn from posterior probability distributions. The Stan Math library (Carpenter et al., 2015), which is used by ctsem and CoTiMA for estimation, offers therefore a No U-Turn Sampler (NUTS). However, this sampler is much (!) slower than the default estimator maximum likelihood estimation or the maximum a posteriori estimation. In fact, most desktop computers in 2024 probably would need a several days for a full CoTiMA with Bayesian estimation if 20 or more primary studies are analyzed. Table 2 gives an overview of how the different estimators can be requested by setting the `optimize` and the `priors` argument. This applies to all CoTiMA fitting functions (`ctmaInit`, `ctmaFit`, `ctmaEqual`, & `ctmaPower`).

Table 2. Estimators available for CoTiMA

Estimator	Argument Settings	
	`optimize`	`priors`
Bayesian estimation via Stan's NUTS (No U-Turn) sampler	`FALSE`	`TRUE`
Maximum a posteriori estimation	`TRUE`	`TRUE`
Maximum likelihood estimation (default)	`TRUE`	`FALSE`

Weakly informative priors for Bayesian estimation with the NUTS sampler and for maximum a posteriori estimation are provided by ctsem. They work well under most circumstances, however, sometimes they might not work well because the priors provided by ctsem have been optimized for time measured in years. For example, one could use the argument `scaleTime = 1/365.25` if time was measured in days and previous fitting attempts did not yield meaningful results.

Figure 28 shows how Bayesian estimates using the NUTS sampler could be obtained. Since estimation requires long time (expect several hours), it is recommended to save the model fits for each primary study using the `saveSingleStudyModelFit` argument. If further studies are added later, re-estimating these models could be avoided by the corresponding `readSingleStudyModelFit` argument. In the example in Figure 28, we used `chains = 2` and `coresToUse = 2`. Three chains and three cores are recommended before publishing results. Since Bayesian estimation takes a long time, we want to take care that we get precise results in our first fitting attempt; we set `finishsamples = 10000` for this purpose; parameter estimates and credible intervals will be sampled 10000 times from the estimated parameter distribution, rather than only 1000 sample, which is the default for `finishsamples`.

```
activeDirectory <- "../../" # SET A VALID PATH
CoTiMAInitFit_6_NUTS <- ctmaInit(primaryStudies = CoTiMAstudyList_6,
                                 n.latent = 2,
                                 activeDirectory = activeDirectory,
                                 saveSingleStudyModelFit =
                                   c("InitFit_6_NUTS", 1, 4, 313, 128, 18, 32),
                                 optimize = FALSE,
                                 priors = TRUE,
                                 chains = 2,
                                 coresToUse = 2,
                                 finishsamples = 10000)
summary(CoTiMAInitFit_6_NUTS)
saveRDS(CoTiMAInitFit_6_NUTS, paste0(activeDirectory, "CoTiMAInitFit_6_NUTS.rds"))
```

Figure 28. Using Bayesian estimation via Stan's NUTS sampler (`ctmaInit`)

Part of the results obtained from the code in Figure 28 printed to the console with `summary(CoTiMAInitFit_6_NUTS)` is shown in Figure 29[8]. A comparison with the maximum likelihood effects and their standard errors in Figure 24 reveals no substantial differences except for Study 128, for which results are not trustworthy anyway. We should note, further, that Bayesian estimation is sensitive to priors, and default priors are only appropriate if the time scale is appropriately chosen, too. This could require using an appropriately chosen `scaleTime` argument.

```
                                                    V1toV1    SE       V2toV1    SE
Study No 1   "Houkes et al., Study1, 2003"                "-0.2123" "0.0455" "0.0388"  "0.0397"
Study No 4   "Houkes et al., Study2, 2003"                "-0.135"  "0.0439" "0.0245"  "0.0435"
Study No 313 "Demerouti, Bakker, & Bulters, 2004"        "-1.2559" "0.1285" "0.4319"  "0.1197"
Study No 128 "Childs, J. H., & Stoeber, J., Study1, 2012" "-3.3878" "2.3539" "-0.1908" "0.8782"
Study No 18  "Diestel, & Schmidt, Study 1, 2012"         "-0.2171" "0.0424" "0.1043"  "0.0427"
Study No 32  "Fernet, Guay, Senecal, & Austin, 2012"     "-0.2038" "0.0297" "0.0461"  "0.0304"


Study No 1   "Houkes et al., Study1, 2003"                V1toV2    SE       V2toV2    SE
Study No 4   "Houkes et al., Study2, 2003"                "-0.0786" "0.0376" "-0.1073" "0.0344"
Study No 313 "Demerouti, Bakker, & Bulters, 2004"        "0.0483"  "0.0484" "-0.1542" "0.0467"
Study No 128 "Childs, J. H., & Stoeber, J., Study1, 2012" "0.277"   "0.1062" "-0.8489" "0.0989"
Study No 18  "Diestel, & Schmidt, Study 1, 2012"         "0.0832"  "0.4813" "-0.2162" "0.1715"
Study No 32  "Fernet, Guay, Senecal, & Austin, 2012"     "0.0551"  "0.0431" "-0.1996" "0.0489"
```

Figure 29. Estimates for the primary studies using Bayesian estimation (`ctmaInit`)

7 CoTiMA (ctmaFit)

Now the first three steps (**E**xtract, **P**repare, & **I**nitFit) in the recommended EPIC-BiG-Power workflow are done, and we can move forward to do **C**oTiMAs, for which the now available `CoTiMAInitFit_6` (or `CoTiMAInitFit_6_NUTS`) object is required. In the first subsection, we demonstrate how a *full CoTiMA* with all drift effects could be fitted. A distinction is therefore made between two special model types depending on the structure of the data: the *all-invariant-model* (see 7.1.1) and the *regular model* (see 7.1.2). In the second subsection, we show how a

[8] In addition, several warning messages are issued. They are all related to Study 128, for which we introduced missing data. This does not happen if doing the analysis again without Study 128.

partial CoTiMA could be fitted, and we use this subsection to introduce the possibilities to analyze specific invariance constraints. In the third subsection, we show how to statistically *test the equality* of drift effects, that is, a CoTiMA with equality constraints. Finally, in the fourth subsection, we show how a *moderated CoTiMA* can be performed.

7.1 Full CoTiMA (`ctmaFit`)

We shall note that the first full CoTiMA we present here is a very special case that is probably rarely applied, and we will move on to the regular case a bit further below. The reason why the first full CoTiMA is a very special case is, again, Study 128, which was a 2-wave study with one missing variable. Such studies prevent applying the usually recommended CoTiMA.

7.1.1 Full CoTiMA as All-Invariant Model (`ctmaFit`)

Usually, CoTiMA aggregates the drift coefficients by constraining them to be invariant across primary studies, whereas the correlations at Time 0 and the diffusion terms (i.e., innovation (co-)variances) are freely estimated within each primary study. This is impossible with the current set of primary studies because for Study 128, demands (role stress) was measured at Time 0 only, so diffusions for demands cannot be estimated for Study 128. As we shall later, missing variables do not impose problems if each variable is measured at least twice, which is possible in studies comprising more than two waves, but Study 128 had only two waves. In such instances, one could either decide to exclude critical studies from CoTiMA, or one could estimate a very restrictive CoTiMA that restricts all parameters (Time 0 correlations, drift effects, diffusions) to be invariant across all studies. This is called an *all-invariant-model*, and estimating such a model can be achieved by using the argument `allInvModel = TRUE`. Usually, we do *not* recommend using this argument, but in this case there is no other option except excluding Study 128, which we do further below in Section 7.1.2.

```
activeDirectory <- "../../" # SET A VALID PATH
CoTiMAFullFit_6 <- ctmaFit(ctmaInitFit = CoTiMAInitFit_6_NUTS,
                           allInvModel = TRUE,
                           coresToUse = 2)
summary(CoTiMAFullFit_6)
saveRDS(CoTiMAFullFit_6, paste0(activeDirectory, "CoTiMAFullFit_6.rds"))
```

Figure 30. Full CoTiMA with six studies (`ctmaFit`)

Fitting this type of a very restrictive CoTiMA is done with the code in Figure 30, and with `summary(CoTiMAFullFit_6)` the results are displayed[9]. The term

[9] Fitting will issue a warning that an *approximate* Hessian was used and standard errors are not trustworthy. This is caused by the missing variables in Study 128.

full CoTiMA is used to refer to a model in which all possible auto effects and all possible cross effects are simultaneously aggregated. Later, we show how some effects could be excluded from the model (i.e., fixed to 0.0), and how some effects could be exempted from being invariant across primary studies. It is noteworthy that the estimator used for initial fitting, which was NUTS, does not affect which estimator is used in a CoTiMA; it is maximum likelihood in the present example, which is the default estimator. Other estimators could be specified as shown in Table 2.

The results in Figure 31 show the names of all parameters of the full (and *all-invariant*) CoTiMA model, their respective row and column numbers in the matrices in which they are used, their estimated mean population values, their standard *errors* (labelled *sd*), their 2.5% lower credible interval, mean, and 97.5% upper credible interval, and the T-values.

The four rows starting with `DRIFT` show the estimates for the continuous time drift coefficients, and their discrete time counterparts, that is, the auto-regressive and cross-lagged effects, across one quarter, are again shown closer to the bottom (`dtDRIFT`). As explained earlier, only the four rows containing the drift coefficients are usually important for reporting CoTiMA results. Nevertheless, we briefly explain what the other parameters stand for. For a more detailed description see Driver et al. (2017) and exact mathematical definitions can be found in Driver and Voelkle (2018).

`T0MEANS` at the top of Figure 31 represent the initial (T0) means of the latent variables. Closer to the bottom in Figure 31, `T0cov` shows correlation of the latent factors at T0, which is identical to their covariance because we deal with standardized variables here[10].

`LAMBDA` is a matrix with the factor loadings of the manifest variables on the latent factors. In the present example, this is a diagonal matrix in which the diagonal was fixed to 1.0. By this, each manifest variable loads on a single latent factor. Conversely, each latent factor is identified by a single manifest variable.

`MANIFESTMEANS` is a matrix (with a single column only) containing the means of the intercepts of the manifest variables. Again, all values were fixed to 0.0 because we deal with standardized variables here.

`CINT` are the continuous time intercepts, which in case of standardized variables are usually zero. `asymCint` are the asymptotic continuous time intercepts. They reflect the intercept values to which the process converges after infinite time. These values should also be 0.0 in the case of CoTiMA, where we use standardized variables (correlations).

[10] Variances in CoTiMA are typically slightly smaller than 1.0. They are computed with $n - 1$ in the denominator. For example, in the case of two primary studies with both $N = 5$ and variances = 1.0 (computed with 4 as denominator), when merged the variance will be computed with 9 instead of 8 as denominator, making the resulting estimate smaller than 1.0.

	row	col	Mean	sd	2.5%	50%	97.5%	Tvalues
T0MEANS 1 1 (invariant)	1	1	0.0000	0.0000	0.0000	0.0000	0.0000	NaN
T0MEANS_2_1 (invariant)	2	1	0.0000	0.0000	0.0000	0.0000	0.0000	NaN
LAMBDA_1_1	1	1	1.0000	0.0000	1.0000	1.0000	1.0000	Inf
LAMBDA_1_2	1	2	0.0000	0.0000	0.0000	0.0000	0.0000	NaN
LAMBDA_2_1	2	1	0.0000	0.0000	0.0000	0.0000	0.0000	NaN
LAMBDA_2_2	2	2	1.0000	0.0000	1.0000	1.0000	1.0000	Inf
DRIFT V1toV1 (invariant)	1	1	-0.7901	0.0695	-0.9409	-0.7904	-0.6691	-11.3683
DRIFT V2toV1 (invariant)	1	2	0.3328	0.0575	0.2266	0.3323	0.4456	5.7878
DRIFT V1toV2 (invariant)	2	1	0.2253	0.0475	0.1311	0.2257	0.3208	4.7432
DRIFT V2toV2 (invariant)	2	2	-0.5528	0.0439	-0.6432	-0.5519	-0.4737	-12.5923
MANIFESTMEANS_1_1 (invariant)	1	1	0.0000	0.0000	0.0000	0.0000	0.0000	NaN
MANIFESTMEANS_2_1 (invariant)	2	1	0.0000	0.0000	0.0000	0.0000	0.0000	NaN
CINT_1_1	1	1	0.0000	0.0000	0.0000	0.0000	0.0000	NaN
CINT_2_1	2	1	0.0000	0.0000	0.0000	0.0000	0.0000	NaN
asymCINT_1_1	1	1	0.0000	0.0000	0.0000	0.0000	0.0000	NaN
asymCINT_2_1	2	1	0.0000	0.0000	0.0000	0.0000	0.0000	NaN
asymDIFFUSIONcov_1_1	1	1	1.0900	0.0517	0.9916	1.0896	1.1932	21.0832
asymDIFFUSIONcov_1_2	1	2	0.4900	0.0437	0.4138	0.4892	0.5779	11.2128
asymDIFFUSIONcov_2_1	2	1	0.4900	0.0437	0.4138	0.4892	0.5779	11.2128
asymDIFFUSIONcov_2_2	2	2	1.0780	0.0544	0.9780	1.0762	1.1872	19.8162
DIFFUSIONcov_1_1 (invariant)	1	1	1.3915	0.0887	1.2304	1.3873	1.5754	15.6877
DIFFUSIONcov_1_2 (invariant)	1	2	0.0534	0.0472	-0.0444	0.0534	0.1478	1.1314
DIFFUSIONcov 2 1 (invariant)	2	1	0.0534	0.0472	-0.0444	0.0534	0.1478	1.1314
DIFFUSIONcov_2_2 (invariant)	2	2	0.9679	0.0506	0.8742	0.9650	1.0735	19.1285
MANIFESTcov_1_1	1	1	0.0000	0.0000	0.0000	0.0000	0.0000	NaN
MANIFESTcov 1 2	1	2	0.0000	0.0000	0.0000	0.0000	0.0000	NaN
MANIFESTcov_2_1	2	1	0.0000	0.0000	0.0000	0.0000	0.0000	NaN
MANIFESTcov_2_2	2	2	0.0000	0.0000	0.0000	0.0000	0.0000	NaN
T0cov 1 1 (invariant)	1	1	0.9971	0.0401	0.9230	0.9978	1.0760	24.8653
T0cov_1_2 (invariant)	1	2	0.4390	0.0310	0.3804	0.4387	0.5018	14.1613
T0cov_2_1 (invariant)	2	1	0.4390	0.0310	0.3804	0.4387	0.5018	14.1613
T0cov_2_2 (invariant)	2	2	0.9941	0.0402	0.9198	0.9922	1.0801	24.7289
dtDRIFT_1_1	1	1	0.4735	0.0291	0.4164	0.4721	0.5285	16.2715
dtDRIFT_1_2	1	2	0.1720	0.0255	0.1227	0.1723	0.2194	6.7451
dtDRIFT_2_1	2	1	0.1164	0.0224	0.0712	0.1167	0.1586	5.1964
dtDRIFT_2_2	2	2	0.5960	0.0222	0.5519	0.5960	0.6373	26.8468

Figure 31. Results (Part 1) of a full all-invariant CoTiMA with six studies (`ctmaFit`)

Similarly, `DIFFUSIONcov` are the continuous time error variances (usually referred to as diffusion term in the literature), and `asymDIFFUSIONcov` reflect asymptotic diffusion (error) variances and covariances. One might speculate that the asymptotic diffusion (error) variances should be 1.0 since one cannot explain any variance across infinite time. However, these estimates are based on internal transformations, which are internally useful to reduce the time to fit the model but have no inherent meaning.

`MANIFESTcov` is a matrix of variances and covariances among the manifest variables at each measurement occasion. All values were fixed to 0.0 because we had only a single manifest indicator per latent factor.

```
$minus2ll
[1] 14072.99

$n.parameters
[1] 10

$opt.lag.orig.time
     [,1] [,2]
[1,]   NA    2
[2,]    2   NA

$max.effects
       [,1]   [,2]
[1,]     NA 0.1843
[2,] 0.1248     NA
```

Figure 32. Results (Part 2) of a full all-invariant CoTiMA with six studies (`ctmaFit`)

Part 2 of the results generated by the code in Figure 30 is shown in Figure 32. The *-2ll values* and number of estimated parameters are reported first. Then the optimal time interval according to Dormann and Griffin (2015) and the sizes of effects across the optimal interval are reported[11].

7.1.2 Full CoTiMA as Regular Model (`ctmaFit`)

As noted in the last subsection, 2-wave studies with missing variables could be used, but they require constraining all parameters to be invariant across primary studies. Such strict assumptions are not necessary if variables (correlations) are not missing, or if each variable in a primary studies is measured at least twice. When a variable is available at two measurement occasions and a primary study comprises more than two waves, it does not impose problems for CoTiMA if this variable is missing at further waves. Only two measurements are required, whenever they were carried out during multi-wave studies. This is demonstrated in the current section, where we add such a study (Study 201), which is then used in subsequent examples as replacement for Study 128.

The workflow for replacing Study 128 by Study 201 and conducting a full CoTiMA is shown in Figure 33. Study 201 comprised three waves of measurement, and burnout was not measured at the third measurement occasion so that the correlations were not available (`NA`). A new list of primary studies is compiled (`CoTiMAstudyList_6_new`), and the initial fitting of each primary study is re-done with the fit stored in the object `CoTiMAInitFit_6_new`. `CoTiMAInitFit_6_new` is then used as the `ctmaInitFit` argument to fit a regular full CoTiMA using `ctmaFit`.

[11] When performing a CoTiMA, the user will notice several empty slots displayed after the `summary` function is applied (e.g., no random effects, no cluster effects). These represent additional functionalities of CoTiMA that we introduce later.

```
# Enter primary Study 201 with missing variables but each variable measured at least
twice
empcov201 <- matrix(c(1.00, 0.43, 0.64, 0.32, 0.57, NA,
                      0.43, 1.00, 0.30, 0.61, 0.26, NA,
                      0.64, 0.30, 1.00, 0.48, 0.69, NA,
                      0.32, 0.61, 0.48, 1.00, 0.37, NA,
                      0.57, 0.26, 0.69, 0.37, 1.00, NA,
                        NA,   NA,   NA,   NA,   NA, NA), nrow = 6, ncol = 6)
variableNames201 <- c("Demands_1",  "Burnout_1", "Demands_2",  "Burnout_2",
                      "Demands_3", "Burnout_3")
dimnames(empcov201) <- list(variableNames201, variableNames201)
delta_t201 <- c(3, 3)
sampleSize201 <- 999
ageM201 <- 39.4
ageSD201 <- 10.55
malePercent201 <- .689
occupation201 <-  c("different occupations")
country201 <- c("Switzerland")
demands201 <- c("Time Pressure")
burnout201 <- c("Exhaustion")
source201 <-  c("Brauchli", "Schaufeli", "Jenny", "Fuellemann", "& Bauer", "2013")
moderator201 <- c(2, NA)

# Compiling a revised list of primary studies with Study 201 replacing Study 128
activeDirectory <- "../../" # SET A VALID PATH
CoTiMAstudyList_6_new <- ctmaPrep(selectedStudies = c(1, 4, 313, 18, 32, 201),
                                  activeDirectory = activeDirectory,
                                  addElements = c("demands", "burnout"),
                                  moderatorLabels = c("Burnout Measure",
                                                      "Control at Work"),
                                  moderatorValues = list(c("1 = Emotional Exhaustion",
                                                           "2 = Exhaustion"),
                                                           "continuous"))

# Initial fitting of revised list of primary studies
CoTiMAInitFit_6_new <- ctmaInit(primaryStudies = CoTiMAstudyList_6_new,
                                n.latent = 2,
                                activeDirectory = activeDirectory)
summary(CoTiMAInitFit_6_new)

# The full CoTiMA
CoTiMAFullFit_6_new <- ctmaFit(ctmaInitFit = CoTiMAInitFit_6_new, coresToUse = 2)
summary(CoTiMAFullFit_6_new)
```

Figure 33. Workflow for replacing Study 128 by Study 201 and conducting a regular full CoTiMA

The results of the full CoTiMA are shown in Figure 34 and Figure 35. The interpretation of results is analogous to the interpretation of the all-invariant CoTiMA discussed in 7.1.1[12].

[12] There is one notable difference. Whereas in the all-invariant CoTiMA estimated T0 correlations and diffusions apply to the full sample of primary studies, in Figure 33 they apply to the last of the primary studies (i.e., Study 32). This is due to technical reasons inherent in the `ctsem` R-package used by CoTiMA. In `ctsem`, *k* - 1 dummy variables for the overall *k* primary studies are used as so-called time independent predictors (TI), which modify (add or subtract values) the T0-correlations and the diffusion parameters estimated for the *k*th primary study. However, T0 correlations and diffusion parameters are usually of very little interest to researcher applying CoTiMA, so these technical details are only important in the probably rare case estimated T0 correlations and diffusions should be reported in publications.

```
                          row col  Mean    sd     2.5%    50%     97.5%  Tvalues
T0MEANS_1_1                 1   1  0.0000 0.0000  0.0000  0.0000  0.0000      NaN
T0MEANS_2_1                 2   1  0.0000 0.0000  0.0000  0.0000  0.0000      NaN
LAMBDA_1_1                  1   1  1.0000 0.0000  1.0000  1.0000  1.0000      Inf
LAMBDA_1_2                  1   2  0.0000 0.0000  0.0000  0.0000  0.0000      NaN
LAMBDA_2_1                  2   1  0.0000 0.0000  0.0000  0.0000  0.0000      NaN
LAMBDA_2_2                  2   2  1.0000 0.0000  1.0000  1.0000  1.0000      Inf
DRIFT V1toV1 (invariant)    1   1 -0.1671 0.0090 -0.1857 -0.1669 -0.1507 -18.4741
DRIFT V2toV1 (invariant)    1   2  0.0322 0.0090  0.0139  0.0325  0.0499   3.5902
DRIFT V1toV2 (invariant)    2   1  0.0489 0.0116  0.0276  0.0490  0.0708   4.1993
DRIFT V2toV2 (invariant)    2   2 -0.2010 0.0124 -0.2254 -0.2005 -0.1784 -16.2699
MANIFESTMEANS_1_1           1   1  0.0000 0.0000  0.0000  0.0000  0.0000      NaN
MANIFESTMEANS_2_1           2   1  0.0000 0.0000  0.0000  0.0000  0.0000      NaN
CINT_1_1                    1   1  0.0000 0.0000  0.0000  0.0000  0.0000      NaN
CINT_2_1                    2   1  0.0000 0.0000  0.0000  0.0000  0.0000      NaN
asymCINT_1_1                1   1  0.0000 0.0000  0.0000  0.0000  0.0000      NaN
asymCINT_2_1                2   1  0.0000 0.0000  0.0000  0.0000  0.0000      NaN
asymDIFFUSIONcov_1_1        1   1  1.3081 0.0538  1.2006  1.3065  1.4152  24.3253
asymDIFFUSIONcov_1_2        1   2  0.5871 0.0460  0.5022  0.5880  0.6753  12.7614
asymDIFFUSIONcov_2_1        2   1  0.5871 0.0460  0.5022  0.5880  0.6753  12.7614
asymDIFFUSIONcov_2_2        2   2  1.2270 0.0591  1.1109  1.2263  1.3386  20.7661
DIFFUSIONcov_1_1            1   1  0.3984 0.0113  0.3770  0.3985  0.4212  35.3336
DIFFUSIONcov_1_2            1   2  0.1122 0.0099  0.0930  0.1123  0.1317  11.3817
DIFFUSIONcov_2_1            2   1  0.1122 0.0099  0.0930  0.1123  0.1317  11.3817
DIFFUSIONcov_2_2            2   2  0.4344 0.0154  0.4059  0.4340  0.4645  28.1915
MANIFESTcov_1_1             1   1  0.0000 0.0000  0.0000  0.0000  0.0000      NaN
MANIFESTcov_1_2             1   2  0.0000 0.0000  0.0000  0.0000  0.0000      NaN
MANIFESTcov_2_1             2   1  0.0000 0.0000  0.0000  0.0000  0.0000      NaN
MANIFESTcov_2_2             2   2  0.0000 0.0000  0.0000  0.0000  0.0000      NaN
T0cov_1_1                   1   1  0.9993 0.0312  0.9401  0.9995  1.0598  32.0630
T0cov_1_2                   1   2  0.4287 0.0230  0.3852  0.4277  0.4746  18.6006
T0cov_2_1                   2   1  0.4287 0.0230  0.3852  0.4277  0.4746  18.6006
T0cov_2_2                   2   2  0.9972 0.0296  0.9400  0.9948  1.0567  33.6564
dtDRIFT_1_1                 1   1  0.8468 0.0076  0.8311  0.8471  0.8606 111.6018
dtDRIFT_1_2                 1   2  0.0268 0.0075  0.0116  0.0271  0.0414   3.5973
dtDRIFT_2_1                 2   1  0.0406 0.0096  0.0230  0.0408  0.0588   4.2355
dtDRIFT_2_2                 2   2  0.8186 0.0101  0.7987  0.8190  0.8375  81.2755
```

Figure 34. Results (Part 1) of a regular full CoTiMA with six studies (`ctmaFit`)

```
$minus2ll
[1] 25440.1

$n.parameters
[1] 40

$opt.lag.orig.time
     [,1] [,2]
[1,]   NA    6
[2,]    6   NA

$max.effects
      [,1]   [,2]
[1,]    NA 0.0638
[2,] 0.098     NA
```

Figure 35. Results (Part 2) of a regular full CoTiMA with six studies (`ctmaFit`)

7.2 Partial CoTiMA (`ctmaFit`)

Figure 36 demonstrates some further possibilities for conducting a CoTiMA; additional capabilities are explained in Appendix B. The CoTiMA model specified in

Figure 36 fixes the effect of *V2toV1* to 0.0 (which we do not generally recommend - let the evidence decide rather theoretical expectations), by labeling the according drift `0` or `"0"`. Further, only the effect *V1toV2* is invariant across primary studies as defined in the `invariantDrift` argument (which could be reasonable – and which could be decided based upon a statistical test see Subsection 7.3). The estimated drift coefficients of this partial CoTiMA are shown in Figure 37.

```
activeDirectory <- "../../" # SET A VALID PATH
CoTiMAPart134Inv3Fit_6 <- ctmaFit(ctmaInitFit = CoTiMAInitFit_6_new,
                                  drift = matrix(c("V1toV1", 0,
                                                   "V1toV2", "V2toV2"),
                                                   nrow = 2, ncol = 2, byrow = TRUE),
                                  invariantDrift = c("V1toV2"),
                                  coresToUse = 2)
saveRDS(CoTiMAPart134Inv3Fit 6, paste0(activeDirectory,"CoTiMAPart134Inv3Fit 6.rds"))
summary(CoTiMAPart134Inv3Fit_6)
```

Figure 36. A partial CoTiMA with a subset of primary studies, with one cross effect fixed to 0.0 invariant across primary studies and only one effect invariant across primary studies (`ctmaFit`)

```
                         row col    Mean      sd    2.5%     50%   97.5%  Tvalues
DRIFT V1toV1               1   1 -0.2066 0.0102 -0.2274 -0.2063 -0.1874 -20.2936
DRIFT V2toV1               1   2  0.0000 0.0000  0.0000  0.0000  0.0000      NaN
DRIFT V1toV2 (invariant)   2   1  0.0497 0.0115  0.0268  0.0498  0.0722   4.3117
DRIFT V2toV2               2   2 -0.2501 0.0132 -0.2772 -0.2498 -0.2251 -19.0090

$minus2ll
[1] 25265.27

$n.parameters
[1] 49
```

Figure 37. Results of the partial CoTiMA specified in Figure 36 (`ctmaFit`)

7.3 CoTiMA with Equality Constraints (`ctmaFit`, `ctmaEqual`, `ctmaCompFit`)

A *-2ll difference test* can be applied whenever researchers want to compare two model fits. Note, however, that the result is only valid if the two models are nested, that is, the second model is derived from the first model by constraining parameters. Such constraints are present, for example, if parameters are eliminated from a model by constraining them to be 0.0 (like demonstrated in Figure 36), or by constraining other parameters to be equal. To statistically test if two or more effects are equal is a bit complex and requires three steps: (1) ensuring correct coding (polarity), (2) fitting a partially invariant CoTiMA using `ctmaFit`, and (3) testing equality using `ctmaEqual`.

First, (1) one has to take care that the effects to be compared have equal signs. For example, consider a model with three latent variables such as demands, resources, and burnout. Work-related resources, such as supervisor support, can be supposed to reduce burnout whereas demands increase burnout. To compare the

effect sizes, one would need to go back to square one and re-start the EPIC part of the workflow. When preparing the correlations with `ctmaEmpCov`, one would need to use the `recode` argument to recode supervisor support so that it becomes lack of supervisor support. Then, one has to use `ctmaInit` again for initial fitting.

In the second step (2), one could start testing the equality of the effect sizes of supervisor support and of demands on burnout. This requires two CoTiMAs to be performed. The first CoTiMA has to specify those two (or more) effects as invariant across studies that should be tested for equality in the subsequent step. This is done with `ctmaFit`. We call this the *invariance model*.

```
activeDirectory <- "../../" # SET A VALID PATH
CoTiMAFullInv23Fit_6 <- ctmaFit(ctmaInitFit = CoTiMAInitFit_6_new,
                                invariantDrift = c("V2toV1", "V1toV2"),
                                coresToUse = 2)
saveRDS(CoTiMAFullInv23Fit_6, paste0(activeDirectory,
                                     "CoTiMAFullInv23Fit_6.rds"))
summary(CoTiMAFullInv23Fit 6)

CoTiMAFullInvEq23Fit_6 <- ctmaEqual(ctmaInvariantFit = CoTiMAFullInv23Fit_6,
                                    coresToUse = 2)
saveRDS(CoTiMAFullInvEq23Fit_6, paste0(activeDirectory,
                                       "CoTiMAFullInvEq23Fit_6.rds"))

summary(CoTiMAFullInvEq23Fit_6)
```

Figure 38. Two-step procedure for testing the equality of two cross effects (`ctmaFit`, `ctmaEqual`)

Third (3), the CoTiMA fit-object returned then serves as an argument for `ctmaEqual`. The code for Step 2 and 3 is shown in Figure 38, in which *V1toV2* and *V2toV1* are first declared to be invariant and then tested for equality. We do not display all estimated drift parameters returned from `summary(CoTiMAFull-Inv23Fit_6)` in a Figure here because it is sufficient to note that *V1toV2* = .0444, *V2toV1* = .0307, *-2ll* = 25253.17, and the number of estimated parameters = 50. *V1toV2* and *V2toV1* were the only parameters that were aggregated, that is, invariant across primary studies. This is recognized by `ctmaEqual`, which, in addition to their invariance, constrains *V1toV2* and *V2toV1* to be equal. We call this the *equality model*. Again, we do not display all estimated drift parameters returned from `summary(CoTiMAFullEq23Fit_6)` in a Figure here because it is sufficient to note that *V1toV2* = *V2toV1* = .0364, *-2ll* = 25253.89, and the number of estimated parameters = 49.

```
[1] "  ###    NEXT MODEL COMPARISON    ###"
[2] "Diff_Minus2LL:  0.719788864123984"
[3] "Diff_df (= Diff_n.params):  1"
[4] "prob:  0.396213174274355"
[5] "Message1:  A prob value < .05 indicates a significant difference."
```

Figure 39. Result of the *-2ll difference test* comparing the fit of the invariance model with the fit of the equality to test if two cross effects are equal (`ctmaEqual`)

The *-2ll difference test* examines if the fit (*-2ll value*) of the equality model is not statistically worse than the fit of the invariance model. If this would be the case, then the hypothesis that both effects are equal has to be rejected and the alternative hypothesis that one effect (*V2toV1* in this example) is significantly larger than the other one (*V1toV2* in this example), will be retained. The *-2ll difference test* is automatically performed by `ctmaEqual`, too, it is displayed at the end of the `summary(CoTiMAFullInv23Fit_6)`, and it is shown in Figure 39. In our example, the *-2ll difference test* was not significant. Thus, we could not reject the hypothesis that *V1toV2 = V2toV1*.

Finally, we shall mention the `ctmaCompFit` function that comes with the CoTiMA package. The `ctmaCompFit` function is automatically used by `ctmaEqual`. It can also be applied whenever researchers want to compare two model fits with a *-2ll difference test* by using `ctmaCompFit(CoTiMAFit1, CoTiMAFit2)`. Note, however, that the result is only valid if the two models are nested, that is, the second model is derived from the first model by constraining parameters. Such constraints are present, for example, if parameters are eliminated from a model by constraining them to be 0.0, or by constraining other parameters to be equal. The former is achieved by setting the desired drift effect to `"0"`, and the latter is achieved by assigning identical labels to the desired drift effects. This could be done with the `ctmaInit` and `ctmaFit` functions. For example, the argument `drift = matrix(c("V1toV1", 0, 0, "V1toV1"), nrow = 2, ncol = 2, byrow = TRUE)` could be used to fit a model that has no cross effects and equal auto effects. This model is nested in a full CoTiMA model because it is more constrained.

7.4 Moderated CoTiMA (`ctmaFit`)

CoTiMA can handle multiple continuous moderators and multiple categorical moderators, however, it is not yet possible to mix categorical and continuous ones. In general, we recommend starting with a single moderator to foster understanding how they operate before analyzing multiple moderators combined.

Recalling from Figure 20 that we entered information about two moderators. The first was the type of burnout measure applied in a primary study, which was either exhaustion or emotional exhaustion, and which was a categorical moderator. If there were two or more categorical moderators, the moderator numbers and moderator names would have to be provided as vectors (e.g., `mod.number = c(1, 3)`, `mod.names = c("Burnout Measure", "Study Quality")`). However, in the present example in Figure 40, we use the first potential moderator variable only (`mod.number = 1`), which was categorial (`mod.type = "cat"`) representing two types of burnout measures (`mod.names = "Burnout Measure"`). By default, CoTiMA does (!) standardize moderators from version

0.5.3 onwards. In the present example, we did overwrite the default by including the argument `scaleMod = FALSE`. Thus, the *k* - 1 dummy variables created from the *k* categories of the moderator variable use values 0 and 1.

```
activeDirectory <- "../../" # SET A VALID PATH
CoTiMAMod1onFullFit_6 <- ctmaFit(ctmaInitFit = CoTiMAInitFit_6_new,
                                 mod.number = 1,
                                 mod.type = "cat",
                                 mod.names = "Burnout Measure",
                                 coresToUse = 2,
                                 scaleMod = FALSE)
summary(CoTiMAMod1onFullFit_6)
saveRDS(CoTiMAMod1onFullFit_6, paste0(activeDirectory,
                                      "CoTiMAMod1onFullFit_6.rds"))
```

Figure 40. A full moderated CoTiMA with a single categorical moderator (`ctmaFit`)

Part of the results are shown in Figure 41. The drift effects shown in the Section `$estimates` are those in the reference group, which is always the group with the smallest category number. In the present example, these are the primary studies for which the moderator value was 1 (and internally recoded to 0 by CoTiMA) meaning they used an emotional exhaustion scale to measure burnout.

The Section `$mod.effects` in Figure 41 shows the effects belonging group with the 2nd category number. In case there were more categories, one would find here four additional rows starting with `"3(category value)"` etc. It is important to note that this section does not show the drift effects. Rather, it shows how for primary studies of this category, which used an exhaustion compared to emotional exhaustion scale to measure burnout, the drift effects change compared to the reference group. Neither auto effects nor cross effects were significantly affected by the type of burnout measure. Leaving lack of significance aside, the effect of demands on burnout (*V1toV2*) was increased if an exhaustion scale was used in primary studies and the effect of burnout on demands (*V2toV1*) was reduced if an exhaustion rather than emotional exhaustion scale was used. We call this a *positive moderating effect* and a *negative moderating effect* of the exhaustion scale, respectively[13].

[13] Plotting the moderator effects is straightforward because for each time interval the change in the drift parameter introduced by the moderator can be depicted as shown in Figure 42. However, summarizing the effect of a moderator in continuous time is not as straightforward because of the non-linearities involved. To do so, the moderator effect is *linearized* at the mean of the drift effect, and this linearized effect is reported in the `$mod.effects` section.

```
$estimates
                          row col    Mean     sd    2.5%     50%   97.5%  Tvalues
DRIFT V1toV1 (invariant)   1   1 -0.1884 0.0186 -0.2276 -0.1874 -0.1544 -10.1084
DRIFT V2toV1 (invariant)   1   2  0.0488 0.0188  0.0124  0.0491  0.0851   2.6001
DRIFT V1toV2 (invariant)   2   1  0.0373 0.0190 -0.0008  0.0377  0.0757   1.9676
DRIFT V2toV2 (invariant)   2   2 -0.1882 0.0191 -0.2310 -0.1872 -0.1544  -9.8449

$minus2ll
[1] 25434.78

$n.parameters
[1] 44

$opt.lag.orig.time
     [,1] [,2]
[1,]   NA    5
[2,]    5   NA

$max.effects
       [,1]   [,2]
[1,]     NA 0.0959
[2,] 0.0733     NA

$mod.effects
                                                        mean     sd    2.5%     50%  97.5% Tvalues
2  (category value) of Burnout Measure_on_V1toV1  0.0328 0.0249 -0.0125  0.0305 0.0857  1.3166
2  (category value) of Burnout Measure_on_V2toV1 -0.0226 0.0213 -0.0632 -0.0234 0.0198 -1.0620
2  (category value) of Burnout Measure_on_V1toV2  0.0173 0.0243 -0.0304  0.0167 0.0652  0.7110
2  (category value) of Burnout Measure_on_V2toV2 -0.0184 0.0237 -0.0593 -0.0199 0.0334 -0.7767
```

Figure 41. Part of the results moderated full CoTiMA (`ctmaFit`)

As always, the sizes of continuous time effects are virtually impossible to interpret. For example, the effect *V1toV2* is .0373 for emotional exhaustion and .0373 + .0173 = .0546 (linearized; see Footnote 13) for exhaustion. However, how these effects unfold over time also depends on the other three effects *V1toV1*, *V2toV2*, and *V2toV1*. We used `plot(CoTiMAMod1onFullFit_6, timeUnit = "Quarters", timeRange = c(1, 36, 1))` to plot the moderated discrete time effects. For *V1toV2*, the course of the moderated effect over discrete time is shown in Figure 42.

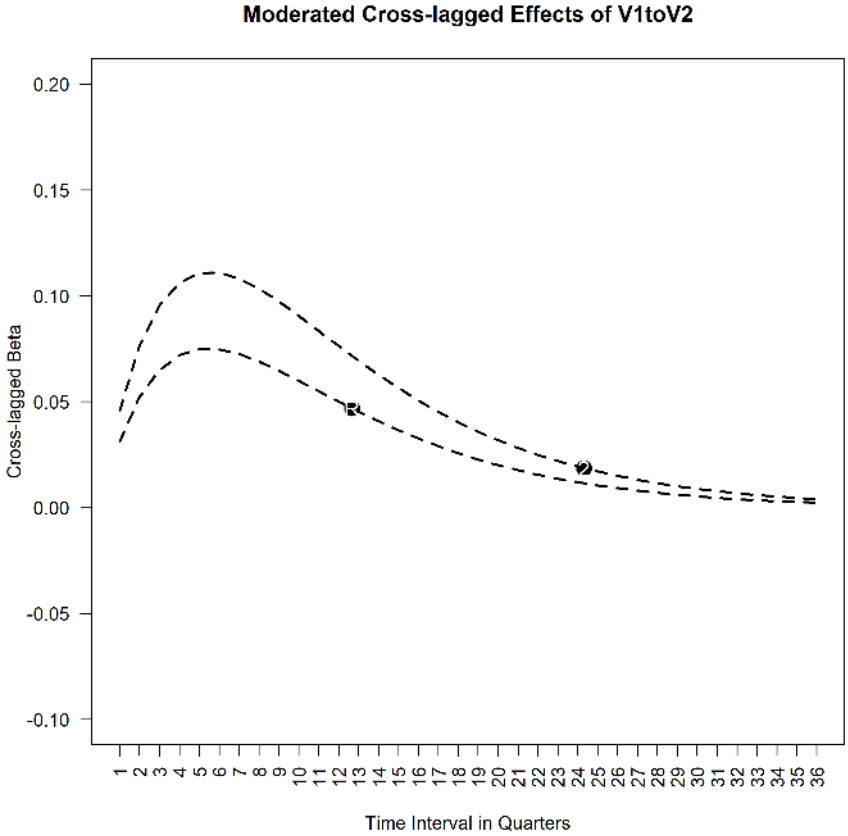

Figure 42. The cross-lagged effect *V1toV2* moderated by type of burnout measure (R = Reference category: emotional exhaustion, 2 = exhaustion) from 1 to 36 quarters (the horizontal location of the category indicators R (reference category) and 2 has no inherent meaning; `plot`)

Since only two categories exist, `catsToCompare = c(1, 2)` is the only viable option in the present example. In the fitted model the moderating effects of Category 1 and Category 2 are restricted to be invariant. If this assumption is valid (i.e., moderating effects are not different for the two categories), the *-2ll value* of the restricted model should not be significantly different from the *-2ll value* of the unrestricted model. This is tested with the `ctmaCompFit` function at the bottom of Figure 43, which shows (not displayed in a figure) that the difference in the *-2ll values* given 4 degrees of freedom is not significant ($\Delta(-2ll) = 5.3201$; $\Delta(df) = 4$; $p = 0.2560$). In

fact, with only two categories available, restricting their effects to be invariant is conceptually identical to assuming there is no moderating effect. Hence, comparing the (unrestricted) moderator model with the full CoTiMA model estimated earlier (which had not moderator effect included), should yield virtually identical results, and indeed `ctmaCompFit(CoTiMAFullFit_6_new, CoTiMAMod1on-FullFit_6)` yields (Δ(*-2ll*) = 5.3201; Δ(*df*) = 4; *p* = 0.2560). However, with three or more categories these two *-2ll difference tests* will yield diverging results[14].

```
activeDirectory <- "../../" # SET A VALID PATH
CoTiMAMod1onFullFit_6_cats12 <- ctmaFit(ctmaInitFit = CoTiMAInitFit_6_new,
                                        mod.number = 1,
                                        mod.type = "cat",
                                        mod.names = "Burnout Measure",
                                        catsToCompare = c(1,2),
                                        scaleMod = FALSE)
saveRDS(CoTiMAMod1onFullFit_6_cats12,
        paste0(activeDirectory, "CoTiMAMod1onFullFit_6_cats12.rds"))
ctmaCompFit(CoTiMAMod1onFullFit_6_cats12, CoTiMAMod1onFullFit_6)
```

Figure 43. Comparing the effect of two categories of a categorical moderator (`ctmaFit`, `ctmaCompFit`)

```
activeDirectory <- "../../" # SET A VALID PATH
tmpStudyList <- ctmaPrep(selectedStudies = c(1, 4, 313, 18),
                         activeDirectory = activeDirectory,
                         addElements = c("demands", "burnout"),
                         moderatorLabels = c("Burnout Measure", "Control at Work"),
                         moderatorValues = list(c("1 = Emotional Exhaustion",
                                                  "2 = Exhaustion"),
                                                  "continuous"))

CoTiMAMod2on23Fit_6 <- ctmaFit(ctmaInitFit = CoTiMAInitFit_6_new,
                               x = tmpStudyList,
                               mod.number = 2,
                               mod.type = "cont",
                               mod.names = "Control",
                               moderatedDrift = c("V1toV2", "V2toV1"),
                               scaleMod = TRUE,
                               coresToUse = 2)
summary(CoTiMAMod2on23Fit_6)
saveRDS(CoTiMAMod2on23Fit_6, paste0(activeDirectory,
                                    "CoTiMAMod2on23Fit_6.rds"))
plot(CoTiMAMod2on23Fit_6, timeUnit = "Quarters", timeRange = c(1, 36, 1))
```

Figure 44. Comparing the effect of two categories of a categorical moderator (`ctmaFit`, `ctmaCompFit`)

The code for a partially moderated CoTiMA with a single *continuous* moderator is shown in Figure 44. Again, the types of primary studies we use in our example impose a difficulty that is likely to occur in many practical circumstances: For some

[14] Instead of `c(1, 2)`, it would also be possible to use indices such as `c(i, j)` and then use a double loop in R to compare all possible combinations of categories. For example:

```
for(i in 1:(numberOfCats-1)) { for(j in (i+1):numberOfCats) {
 tmpFit <- ctmaFit(ctmaInitFit = CoTiMAInitFit_object, mod.number = 1,
                   mod.type = "cat", catsToCompare = c(i, j))
saveRDS(tmpFit, paste0(activeDirectory, "CoTiMAModFit_cat", i, "_", j, ".rds"))}}
```

studies the moderator variable is not available, and the moderator was therefore coded as NA. In our example, this was the case for Study 201 and Study 32. However, instead of going back to square one and compiling a reduced study list followed by applying `ctmaInit` again, we create a temporary study list using `ctmaPrep`, which does no longer include Study 201 and Study 32 (`tmpStudyList`). We use this temporary study list to specify an optional argument of the `ctmaFit` function (i.e., `primaryStudyList = tmpStudyList`).

To conduct a moderated CoTiMA, further arguments have to be specified. In the current example in Figure 44 only the cross effects are specified to be moderated. It is recommended to standardize continuous moderators, which is achieved by `scaleMod = TRUE`. When continuous moderators are standardized, the estimated drift parameters are those for a prototypical study with a mean moderator value (average effect). The summary (not shown) reveals that control does not significantly reduce *V2toV1* (i.e., the moderating effect) by -.0379 from the average effect, which is *V2toV1* = .0685 (i.e., the main effect).

The plot function shown in Figure 44 yields the plot shown in Figure 45. Across all time intervals, for people who have low levels of control at work, effects of demands on burnout are larger than for those with high levels of control. In most empirical articles that visualize moderator effects for moderator values at +2*SD* and -2*SD* are not shown. This could be achieved by using `mod.values = c(-1, 0, 1)` as additional argument for the plot function in Figure 44.

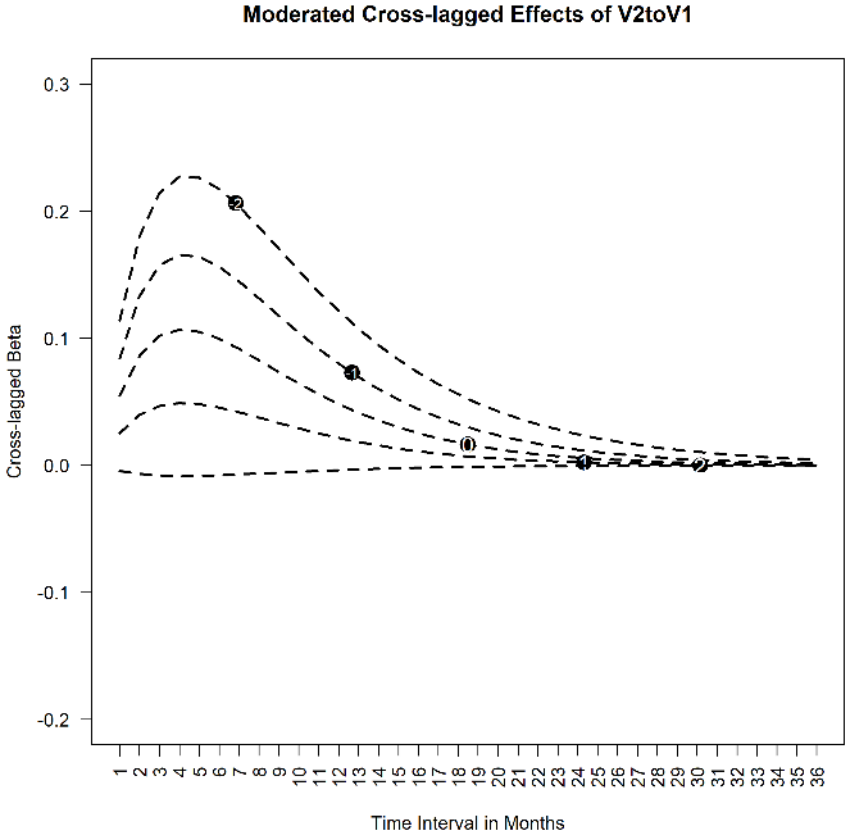

Figure 45. The cross-lagged effect *V2toV1* moderated (not significantly) by control at work from 1 to 36 quarters. The lines show the effect of *V2toV1* for control at -2*SD* below the mean of control (-2), -1*SD* below the mean of control (-1), at the mean of control (0), +1*SD* above the mean of control (1), and +2*SD* above the mean the mean of control (2). The horizontal location of the *SD* values has no inherent meaning.

8 Bias & Generalizability (`ctmaBiG`)

After finishing the EPIC part of the EPIC-BiG-Power workflow, we can now turn to the first part of the **BiG** workflow, which is done by using `ctmaBiG`. It performs Egger's tests for drift coefficients (e.g., Sterne & Egger, 2001) and provides PET-PEESE corrections of fixed effect estimates (Stanley & Doucouliagos, 2014). Random effect estimates are also computed. Various measures of heterogeneity (cf.

Borenstein et al., 2009) as well as measures of expected replications rates (ERR) and expected discovery rates (EDR; Bartoš & Schimmack, 2022; Brunner & Schimmack, 2020) are also provided by `ctmaBiG`. The return object of `ctmaBiG` can be used to plot funnel plots and forest plots.

To proceed with `ctmaBiG`, we use the `init` fit file and data of primary studies published in the online repository of Dormann et al. (2020), which belongs to their CoTiMA of job stressors and burnout. The file containing their `init` fit-object can be retrieved from the website of the Open Science Foundations with the code shown in Figure 46. Note that Guthier et al. (2020) used a preliminary CoTiMA version that was based on the OpenMx R-package (Boker et al., 2011), whereas the file we suggest downloading was created with the rstan R-package (Stan Development Team, 2020). The latter samples parameter estimates from generated parameter distribution and results thus slightly change from analysis to analysis (unless the argument `finishsamples` is set to a large value, e.g., 10000). So, one could expect minor differences compared to the results reported in Guthier et al. (2020). On the other hand, the `init` fit-object contains all information required to replicate all their results with minor deviations[15]. Note, however, computations could last a few hours except `ctmaBiG`. This is the major reason why we did not use their `init` fit-object before.

```
activeDirectory <- "../../" # SET A VALID PATH
dl_link <- "https://osf.io/download/qhpae/"
target_file <- paste0(activeDirectory, "CoTiMAInitFit_D_BO_stanct.rds")
download.file(dl_link, target_file) # Note on windows computers add mode="wb"
CoTiMAInitFit_D_BO <- readRDS(target_file)
saveRDS(CoTiMAInitFit_D_BO, paste0(activeDirectory, "CoTiMAInitFit_D_BO.rds"))
```

Figure 46. Downloading the `Init`-Fit file of Guthier et al. (2020)

The analysis of bias and generalizability, summarizing the results, and plotting forest plots and funnel plots is achieved with the code in Figure 47. First, results of fixed effects analyses of *single* drift coefficients are displayed. Recall that in CoTiMA all drift effects (full CoTiMA) or a subset (partial CoTiMA) is aggregated simultaneously, thereby taking the entire causal system into account. Thus, CoTiMA estimates a *set* of fixed effects by constraining a set of drift effects to be invariant across groups (i.e., primary studies). Estimation is based on minimization of the discrepancy between the model implied covariance matrices and their empirical counterparts.

[15] In addition to the fitted ctsem models of each primary study, it is possible to extract all information from an `init` fit-object that were originally complied with `ctmaPrep` by, e.g., `originalStudyList <- initFitObject$primaryStudyList`. Thus, replicability of CoTiMA results is easily enabled by making one's `init` fit-object available for download in a repository, for example, using the Open Science Framework http://osf.io/

```
activeDirectory <- "../../" # SET A VALID PATH
CoTiMABiG_D_BO <- ctmaBiG(CoTiMAInitFit_D_BO)
summary(CoTiMABiG_D_BO)
plot(CoTiMABiG_D_BO, activeDirectory = activeDirectory)
```

Figure 47. Analysis of bias and generalizability, summary of results, and plotting (`ctmaBiG`)

Contrary, in terms of a traditional fixed and random effects analysis, the drift effects of all primary studies, which resulted from the initial fitting of ctsem models one by one rather than as a set, are analyzed. Estimation is based on the standard errors of the drift effects rather than on minimizing discrepancies between implied and empirical covariance matrices. The fixed effect estimates of the two cross effects reported in the section `$`Fixed Effects of Drift Coefficients`` of Figure 48 were *V1toV*2 = .0024 ($p < .001$) and *V2toV1* = .0053 ($p < .001$).

The next section in Figure 48 is `$Heterogeneity`. Here τ^2 , H^2 , and I^2 are shown, of which I^2 is usually of most interest. Note that estimates of τ^2 were small so even four decimal places are not sufficient to show this. Consequently, between-study heterogeneity as indicated by I^2 was large with the exception of the (small) effect *V1toV2*.

The third section (`$`Random Effects of Drift Coefficients``) in Figure 48 displays the random effect estimates, their *SE*, confidence intervals (`Limit`), and the *z*-values with their associated probability levels. In addition, prediction intervals (`LimitPI`) also allow assessing the degree of heterogeneity. Prediction intervals describe a region in which about 95% of the true study effects are expected to be found (e.g., Guddat et al., 2012). The effects *V1toV2* = .0061 ($p < .001$) and *V2toV1* = .0114 ($p < .001$) were larger than their fixed effects counterparts reported earlier. Note that the corresponding CoTiMA (fixed) effects reported by Guthier et al. (2020) were *V1toV2* = .0039 ($p < .001$) and *V2toV1* = .0084 ($p < .001$), and they were right in the middle between the traditional fixed and random effects estimates.

```
$estimates$`Fixed Effects of Drift Coefficients`
                             V1toV1   V2toV1   V1toV2    V2toV2
MeanOfDriftValues           -0.0590   0.0219   0.0112   -0.0539
FixedEffect_Drift           -0.0219   0.0053   0.0024   -0.0133
FixedEffect_DriftVariance    0.0000   0.0000   0.0000    0.0000
FixedEffect_DriftSE          0.0004   0.0004   0.0003    0.0003
FixedEffect_DriftUpperLimit -0.0211   0.0061   0.0030   -0.0128
FixedEffect_DriftLowerLimit -0.0227   0.0046   0.0018   -0.0139
FixedEffect_DriftZ          -54.3759  14.8119   7.5051  -46.5553
FixedEffect_DriftProb        0.0000   0.0000   0.0000    0.0000
tau2Drift                    0.0001   0.0001   0.0000    0.0001
Q_Drift                    772.8941 534.5197 217.5015 1235.3390
H2_Drift                    16.4446  11.3728   4.6277   26.2838
H2DriftUpperLimit           18.0378  12.6111   5.2907   28.4719
H2DriftLowerLimit           14.9920  10.2560   4.0477   24.2639
I2_Drift                    93.9190  91.2071  78.3910   96.1954
I2DriftUpperLimit           94.9458  92.8491  83.4677   96.7577
I2DriftLowerLimit           92.6835  89.1880  71.7552   95.5355

$estimates$Heterogeneity
                   V1toV1   V2toV1   V1toV2    V2toV2
tau2Drift          0.0001   0.0001   0.0000    0.0001
Q_Drift          772.8941 534.5197 217.5015 1235.3390
H2_Drift          16.4446  11.3728   4.6277   26.2838
H2DriftUpperLimit 18.0378  12.6111   5.2907   28.4719
H2DriftLowerLimit 14.9920  10.2560   4.0477   24.2639
I2_Drift          93.9190  91.2071  78.3910   96.1954
I2DriftUpperLimit 94.9458  92.8491  83.4677   96.7577
I2DriftLowerLimit 92.6835  89.1880  71.7552   95.5355

$estimates$`Random Effects of Drift Coefficients`
                                  V1toV1  V2toV1  V1toV2   V2toV2
RandomEffecttot_Drift            -0.0402  0.0114  0.0061  -0.0380
RandomEffecttot_DriftVariance     0.0000  0.0000  0.0000   0.0000
RandomEffecttot_DriftSE           0.0021  0.0017  0.0011   0.0021
RandomEffecttot_DriftUpperLimit  -0.0360  0.0147  0.0082  -0.0339
RandomEffecttot_DriftLowerLimit  -0.0444  0.0080  0.0039  -0.0420
RandomEffecttot_DriftZ           -18.8218  6.6937  5.5527 -18.2167
RandomEffecttot_DriftProb         0.0000  0.0000  0.0000   0.0000
RandomEffecttot_DriftUpperLimitPI -0.0169 0.0289  0.0153  -0.0149
RandomEffecttot_DriftLowerLimitPI -0.0636 -0.0062 -0.0032 -0.0611
```

Figure 48. Part 1 of the results of `ctmaBiG`

Part 2 of the results returned from `ctmaBiG` is shown in Figure 49. These results address possible publication bias. Egger's tests (e.g., Sterne & Egger, 2001) is a statistical test of funnel plot asymmetry. Significant results indicate that small-*N* studies produced larger effect sizes (i.e., more positive, if the true effect is positive & more negative, if the true effect is negative), suggesting that the aggregated effects are biased. Thus, the results in the `$`Egger's tests`` part of Figure 49 suggest that the cross effects are biased upwards, and the two auto effects are biased downwards. The latter means that demands and burnout in small-*N* studies have smaller carry-over effects than in large-*N* studies. This could have many reasons. For instance, if job stress studies with small-*N* were based on single organizations or single occupations, variance might be restricted, implying lower test-retest correlations eventually resulting in smaller auto effects. However, this reasoning would also imply smaller cross effects, which was not the case. Selective reporting might be a more plausible reason here.

Precision-effect test and precision effect estimates with standard errors (PET-PEESE; Stanley & Doucouliagos, 2014) removes small sample bias (selective reporting) from the fixed effect estimates providing an "aggressive approach" (Stanley et al., 2018, p. 1333). PET-PEESE involves a decision rule when PET or PEESE is more important. The result of this decision is the `PET_PEESE_Drift` row in the section `$`PET-PEESE corrections`` of Figure 49. The `WLS_Drift` estimates of the auto effects *V1toV1* and *V2toV2*, which are identical to the fixed effect estimates in Figure 48 (but have more appropriate *SE*), are more negative compared to their corrected `PET_PEESE_Drift` counterparts, but the differences are not very large. This also applies to the *V2toV1* cross effects, representing the effect of earlier burnout on later burnout. However, PET-PEESE of *V1toV2* = .0010, which is less than 1/5 of the fixed effect. Hence, the true effect of earlier demands on later burnout is probably much smaller than suggested by the fixed effect estimate.

```
$estimates$`PET-PEESE corrections`
                  V1toV1 V2toV1 V1toV2  V2toV2
PET_Drift       -0.0149 0.0031 0.0010 -0.0079
PET_SE           0.0014 0.0015 0.0008  0.0010
PEESE_Drift     -0.0206 0.0048 0.0021 -0.0126
PEESE_SE         0.0013 0.0012 0.0007  0.0013
PET_PEESE_Drift -0.0206 0.0048 0.0010 -0.0126
PET_PEESE_SE     0.0013 0.0012 0.0008  0.0013
WLS_Drift       -0.0219 0.0053 0.0024 -0.0133
WLS_SE           0.0016 0.0012 0.0007  0.0015

$estimates$`Egger's tests`
            V1toV1 V2toV1 V1toV2  V2toV2
Egger's b0 -3.9450 1.4756 1.0961 -4.9811
SE(b0)      0.5038 0.5854 0.3512  0.5145
T          -7.8297 2.5207 3.1211 -9.6814
p           0.0000 0.0152 0.0031  0.0000
```

Figure 49. Part 2 of the results of `ctmaBiG`

Funnel plots and forest plots could be obtained with `plot(CoTiMABiG_D_BO, activeDirectory = activeDirectory)`. Before plotting, define the `activeDirectory` (where to save results), which could then be used in all subsequent function calls. Funnel plots represent the graphical counterpart of Egger's tests, and they plot standard error of effects (an indicator of small-*N* bias; y-axis; large at the bottom & low at the top) against the effect size (x-axis). Without small-*N* bias, funnel plots would be symmetric. Conversely, funnel plot asymmetry indicates small-*N* bias. The funnel plot of *V1toV2*, for which Eggers's test and PET-PEESE indicated large bias, is shown in Figure 50. Effect sizes are clearly asymmetrically distributed on the right-hand side, particularly at the bottom where effect sizes of small-*N* studies (with large *SE*) are located.

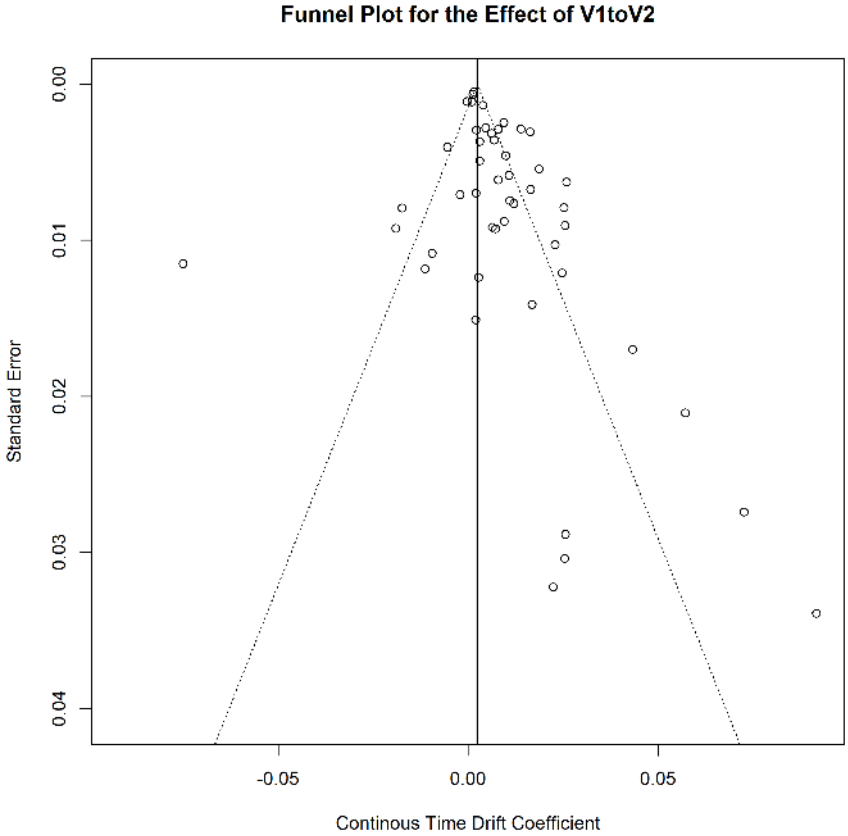

Figure 50. Funnel plot of the effect *V1toV2* of the fit-object returned by `ctmaBiG` (`plot`)

A better impression of the effects obtained in all primary studies is provided in forest plots. The effects for each of the primary studies is represented by a square and their confidence intervals are represented by horizontal lines through these squares. A forest plot of the *V1toV2* effect is shown in Figure 46. The squares vary in size depending on their sample sizes, and they are sometimes small because sample sizes varied considerably across primary studies. The diamond at the bottom shows the aggregated fixed effect. There is no visible horizontal line for its confidence interval because the overall *SE* was very small and, thus, the confidence interval is rather narrow.

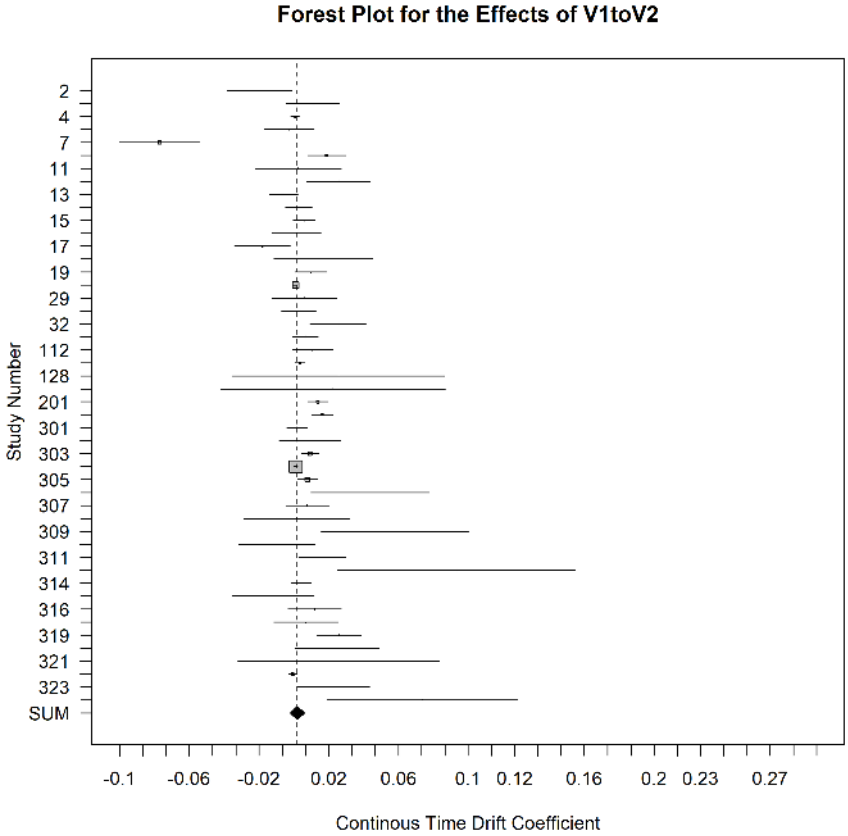

Figure 51. Forest plot of the effect *V1toV2* of the fit-object returned by `ctmaBiG` (`plot`)

9 Statistical Power (`ctmaPower`)

Finally, we can turn to the **Power** part of the EPIC-BiG-Power workflow, which can be performed with `ctmaPower`. It conducts two types of analyses. First, it estimates required sample sizes for a range of different time intervals to achieve a desired statistical power. This is important for designing future studies. Second, it calculates the expected power for all primary studies (sometimes also referred to as post hoc power or retrospective power). This is important to know if past studies might have failed to replicate effects with statistical significance because they were under-powered.

```
activeDirectory <- "../../" # SET A VALID PATH
CoTiMAPower_D_BO <- ctmaPower(ctmaInitFit = CoTiMAInitFit_D_BO,
                              statisticalPower = c(.50, .80, .95),
                              coresToUse = -1, # use all available cores except 1
                              finishsamples = 10000)
saveRDS(CoTiMAPower_D_BO, paste0(activeDirectory, "CoTiMAPower_D_BO.rds"))
summary(CoTiMAPower_D_BO)
```

Figure 52. Calculating expected (post hoc) power for three different probability levels and required sample sizes (`ctmaPower`)

To calculate statistical power, a highly restrictive CoTiMA model is estimated. In the regular full CoTiMA, all drift effects are constrained to be invariant across primary studies. To calculate statistical power, a more restrictive model is required that, in addition, constraints the variance and covariances at T0 as well as the diffusion coefficients to be invariant. This all-invariant model was previously used to include studies with two waves only and a missing variable (see Subsection 7.1.1). Stated differently, it has to be assumed all samples analyzed in the primary studies were drawn from the same population. There are several arguments that can be used with `ctmaPower`, and they are enumerated in the Appendix B. In most cases, requesting the desired levels of power in addition to the `init` fit-object is probably sufficient.

We used the code in Figure 52 for generating the subsequently discussed output and the figures. Then, `summary(CoTiMAPower_D_BO)` creates a large output on the console that we again discuss in parts. Figure 53 displays the parameter estimates of the all-invariant-model. These are the parameter estimates that are regarded as the *true* effects (mean of the distribution of true effects). In concert with the sample sizes and the time intervals of the primary studies (both are taken from `CoTiMAInitFit_D_BO` and do not need to be explicitly provided as arguments) the true effects determine the statistical power of the primary studies to achieve significance levels of $\alpha = .05$ and $\alpha = .01$. Further, across a range of time intervals (could be provided with the argument `timeRange`; otherwise, it is from 1 to 1.5 times the longest interval used in primary studies), the true effects determine the required sample sizes to and achieve the requested levels of statistical power.

```
                        V1toV1   (SE)    Tvalue V2toV1   (SE)   Tvalue
Fixed Effects Drift    -0.0525 0.0009 -58.3333 0.0164 0.0008 20.5000
Fixed Effects Diffusion 0.0976 0.0013  75.0769 0.0096 0.0008 12.0000
Fixed Effects T0Var     0.9983 0.0087 114.7471 0.3757 0.0066 56.9242

Fixed Effects Drift     V1toV2   (SE)  Tvalue  V2toV2   (SE)   Tvalue
Fixed Effects Diffusion 0.0119 0.0007 17.0000 -0.0428 0.0007 -61.1429
Fixed Effects T0Var     0.0096 0.0008 12.0000  0.0818 0.0010  81.8000
```

Figure 53. Estimates of drift parameters using an all-invariant-model with all variances and covariances at T0, all drift effects, and all diffusion coefficients invariant across primary studies (`ctmaPower`)

The next section in the generated output reports the expected power of primary studies. For the effect of *V1toV2*, this is displayed in Figure 54. Note that in Guthier et al. (2020) we reported numerical problems in estimating the expected statistical power across short time intervals – since then we solved this issue. We left out several studies (6 to 23 & 28 to 47) for space reasons here. Assuming the aggregated effects in Figure 53 are the true effects, the probability values in Figure 54 represent the statistical power each primary study had to detect the focal true *V1toV2* effect (i.e., .0119; see Figure 53) with $p < .05$ and $p < .01$. For those studies with more than two measurement occasions, the statistical power is reported for all adjacent time intervals. At the bottom, median and mean statistical power across all primary studies is shown. For instance, the median statistical power was .5042 to find a significant *V2toV1* effect with $p <$. 05. As in most meta-analyses, this demonstrates that many primary studies are heavily under-powered and finding a significant effect is less likely than like getting heads-up when flipping a coin.

	N	Time Lag	Power (α=.05)	Power (α=.01)	Time Lag	Power (α=.05)	Power (α=.01)
Study_No_1	148	12.0	0.3255	0.1411	NA	NA	NA
Study_No_2	188	12.0	0.3981	0.1893	NA	NA	NA
Study_No_3	556	96.0	0.0683	0.0176	NA	NA	NA
Study_No_4	261	12.0	0.5191	0.2832	NA	NA	NA
Study_No_5	1378	18.0	0.9976	0.9861	NA	NA	NA
...	...	...	...	...	...	...	...
Study_No_24	195	3.0	0.1926	0.0684	NA	NA	NA
Study_No_25	999	12.0	0.9761	0.9132	12.0	0.9761	0.9132
Study_No_26	668	12.0	0.8961	0.7390	12.0	0.8961	0.7390
Study_No_27	370	12.0	0.6676	0.4258	12.0	0.6676	0.4258
...	...	...	...	...	...	...	...
Study_No_48	171	3.0	0.1740	0.0595	NA	NA	NA
Mean	NA	NA	0.5397	0.3684	NA	NA	NA
Median	NA	NA	0.5042	0.2707	NA	NA	NA

Figure 54. Expected (post hoc) power across primary studies (`ctmaPower`)

The generated output further shows the required samples sizes for (future) studies to obtain significant effects across different time intervals (Figure 55). Note again, that in Guthier et al. (2020) we reported numerical problems in estimating the required samples across short time intervals; this issue is now solved. For most effects and most desired levels of statistical power, required sample sizes are lowest around 16-18-month intervals. We show how to plot required sample sizes against time interval later. Note that the output showing the required sample sizes would also display the expected (discrete time) effect sizes, which we omitted from Figure 55.

The last interesting output deals with combinations of possible time intervals and samples sizes, and it informs about the range of time intervals across which one could expect significant effects. If neither a sample size (`failSafeN`) nor a *p*-level (`failSaveP`) is provided as function argument, the average sample size of the primary studies is used (otherwise the values assigned to `failSafeN`) and $p < .01$ (otherwise the values assigned to `failSaveP`) are used. As the `$estimates$`Range of significant effects`` section in Figure 56 reports, with *N* corresponding to the average $N = 549$ across primary studies, one should

select time intervals between 8-32 months to find a significant *V2toV1* effect. With the average *N* used in primary studies, one cannot expect finding a significant *V1toV2* effect across neither time interval.

```
         V2toV1 Power=0.5 V2toV1 Power=0.8 V2toV1 Power=0.95 V1toV2
1                    1704             3479              5759
1.5                  1163             2374              3929
2                     893             1823              3016
3                     625             1274              2108
4                     492             1003              1659
...                   ...              ...               ...
15                    236              480               793
16                    234              476               787
17                    234              475               785
18                    234              476               786
19                    235              478               790
20                    237              482               797
21                    240              488               806
...                   ...              ...               ...
142                180113           368006            609274
143                192184           392668            650104
144                205069           418994            693691
...                   ...              ...               ...
Min N                 234              475               785
```

Figure 55. Required sample sizes to achieve requested levels of statistical power across a range of time intervals from 1 to 144 months (`ctmaPower`)

```
[1] "The shortest interval across which the effect (V2toV1) is significant with
    p < 0.01 assuming N = 549 (= avg. N) is 8.  The longest interval across which the
    effect (V2toV1) is significant with p < 0.01 assuming N = 549 (= avg. N) is 32.
    Note that you have not provided an explicit time range for analysis of statisti-
    cal power. The time intervals used ranged from 1 to 1.5 times the longest inter-
    val used in the primary studies, using integer steps of 1.0. These intervals were
    then augmented by time intervals found in primary studies that were non-inte-
    gers."
[2] "There is no shortest interval across which the effect (V1toV2) is significant
    with p < 0.01 assuming N = 549 (= avg. N).  There is no longest interval across
    which the effect (V1toV2) is significant with p < 0.01 assuming N = 549.  Note
    that you have not provided an explicit time range for analysis of statistical
    power. The time intervals used ranged from 1 to 1.5 times the longest interval
    used in the primary studies, using integer steps of 1.0. These intervals were
    then augmented by time intervals found in primary studies that were non-inte-
    gers."
```

Figure 56. Expected range across which significant effects could be expected (`ctmaPower`)

Finally, required sample sizes can be plotted. We used the code in Figure 57 to generate the plot displayed in Figure 58. This figure is based on the values previously shown in parts in Figure 55.

```
activeDirectory <- "../../" # SET A VALID PATH
plot(CoTiMAPower_D_BO, timeUnit = "Months",
                       activeDirectory = activeDirectory,
                       timeRange = c(1, 84, 1))
```

Figure 57. Plotting required sample sizes (`plot`)

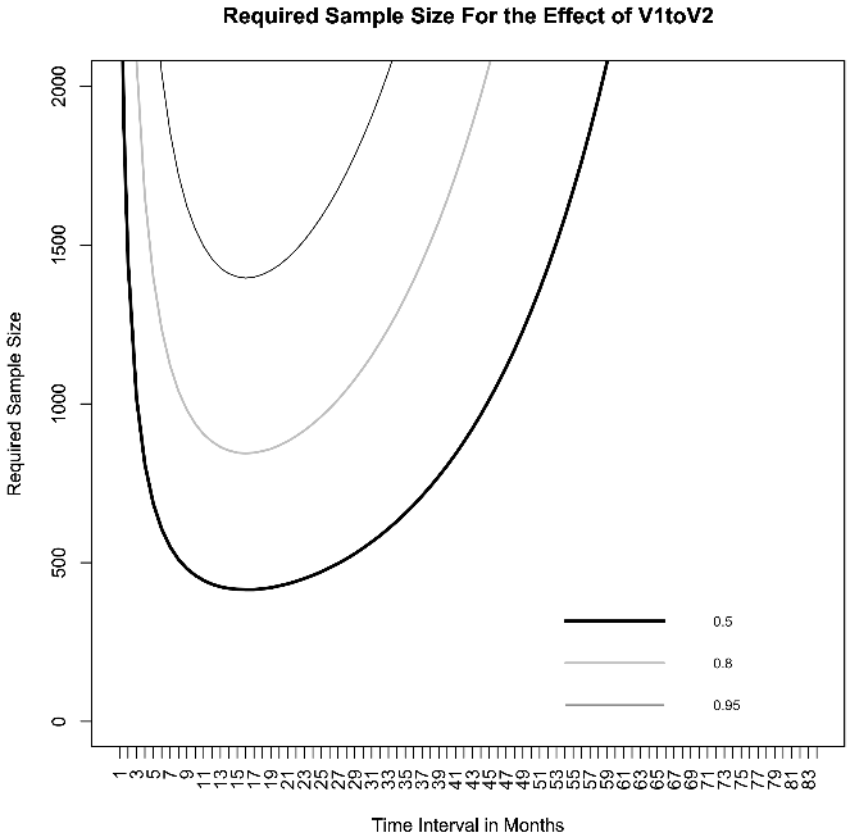

Figure 58. Required sample sizes across time to achieve a statistical power of .50, .80, and .95 for finding a significant effect of *V1toV2* (`plot`)

10 Special Topics

In Chapter 10, several special topics are discussed. We start in Section 10.1 with explaining how a CoTiMA fit-object could be used to go back to the P-Step of the EPIC-Big-Power Workflow and obtain the list of primary studies, which was once compiled using the `ctmaPrep` function. This could be useful for re-doing CoTiMAs with different setting across a team of authors because only the fit files need to be shared, and it is achieved by the `ctmaFitToPrep` function. One possible

reason why researchers want to do this is because they want to apply a CoTiMA to a subset of studies, which we demonstrate in Section 10.2, where we also introduce random intercept models (RI-CoTiMA) and four different specifications how random intercept models can be fitted. Another good reason for re-doing CoTiMAs is to check if results are replicable – sometimes initially obtained results differ slightly depending on the operating systems, time scaling used etc. Therefore, in Section 10.3, we continue with demonstrating how the user could test if a better fit than previously obtained could be achieved. This is useful to ensure that the previously obtained did not suffer from the fitting algorithm being stuck in a local minimum, and it is done with the function `ctmaOptimizeFit`. Optimizing model fit with `ctmaOptimizeFit` could take a lot of time. However, once an optimal fit is obtained, reproducibility of the optimal fit could be facilitated by extracting fitting parameters including starting values from the optimized fit, and by explicitly including them in subsequent (and fast) fitting attempts. This is demonstrated in Section 10.4. In Section 10.5, we deal with the relation between CoTiMAs with random intercepts (RI-CoTiMA) and latent change score (LCS) models and we show how the results of a RI-CoTiMA can be transformed into a continuous time dual latent change score meta-analysis using the `ctmaLCS` function.

10.1 From Fit back to Prep (`ctmaFitToPrep`)

A fit-object created with `ctmaInit` or with `ctmaFit` could serve as argument of the `ctmaFitToPrep` function to recover the original list of primary studies, and to modify it if it is required. The resulting object has one notable difference compared to the original study list created during the P-Step of the EPIC process: the summary function is not operational. In Figure 59, the previously (see Figure 46) downloaded `init` fit file created by Guthier et al. (2020) is used to create a new study list.

```
activeDirectory <- "../../" # SET A VALID PATH
CoTiMAInitFit_D_BO <- readRDS(paste0(activeDirectory, "CoTiMAInitFit_D_BO.rds"))
studyList_D_BO <- ctmaFitToPrep(ctmaFitObject = CoTiMAInitFit_D_BO,
                                reUseEmprawData = TRUE)
```

Figure 59. Using `ctmaFitToPrep` to create a new study list to be used as input for subsequent `ctmaInit`

The created study list (i.e., `studyList_D_BO`) could be directly used as argument for `ctmaInit` or for `ctmaFit`, or it could be modified before. This requires some R code and some experience with list objects in R. For example, the R code shown in Figure 61 could be used to delete all primary studies with two waves only. Further, Study 313 is also excluded even though it comprised three time points because it could not be fitted properly, which we explain further below. The code in Figure 60 generates the output shown in Figure 61.

```
studies3plus <- lapply(studyList_D_BO$deltas, function(x) (length(x) > 1))
head(studies3plus)
targetPos <- which(unlist(studies3plus == TRUE)); targetPos
studyNumbers <- unlist(studyList_D_BO$studyNumbers); studyNumbers
targetStudies <- c(studyNumbers[targetPos], 313); targetStudies
excludedStudies <- studyNumbers[!(studyNumbers %in% targetStudies)]
excludedStudies
study3plusList <- ctmaPrep(ctmaPrepObject = studyList_D_BO,
                           excludedStudies = excludedStudies)
unlist(study3plusList$studyNumbers)
```

Figure 60. R code for deleting all studies with 2 time points from a `ctmaPrep`-object using `ctmaFitToPrep` to create a new study list to be used as input for subsequent `ctmaInit`

```
# Step 1: Create a list containing a FALSE for every primary study with only one
# time interval
[[1]]
[1] FALSE
[[2]]
[1] FALSE
[[3]]
[1] FALSE
[[4]]
[1] FALSE
[[5]]
[1] FALSE
[[6]]
[1] FALSE

...

# Step 2: identify the target list positions that contain a TRUE instead of a FALSE
[1] 22 25 26 27 38 39

# Step 3: Create a vector with all available studyNumbers
 [1]   2   3   4   6   7  10  11  12  13  14  15  16  17  18  19  25  29  31  32
[20]  34 112 127 128 129 201 203 301 302 303 304 305 306 307 308 309 310 311 313
[39] 314 315 316 317 319 320 321 322 323 324

# Step 4: Use the vector of all studyNumbers to identify the targetStudy numbers
[1] 127 201 203 301 314

# Step 5: Exclude in subsequent model fitting studies identified in Step 4 and the
# study with the study number  =  313
[1]   2   3   4   6   7  10  11  12  13  14  15  16  17  18  19  25  29  31  32
[20]  34 112 128 129 302 303 304 305 306 307 308 309 310 311 315 316 317 319 320
[39] 321 322 323 324 313

# Step 6: Compile a new list of primary studies and check that it includes the pri-
mary studies with three or more waves only.
[1] 127 201 203 301 314
```

Figure 61. Corresponding R output for the code used in Figure 60

As shown in Figure 61, first, a list is created that contains a `FALSE` for every primary study that has only one time interval (i.e., 2 time points); the results for the first six studies only is displayed on the console (all `FALSE`). Then we identify the target list positions that contain a `TRUE` instead of a `FALSE`, which are six studies (22, 25, etc.; note that these are the positions in the list of all 48 study numbers and not yet

the study number themselves). Further, we create a vector with all available `studyNumbers`, which is then used to identify the numbers of the target studies. In the next step, we identify the study numbers that we want to exclude (`excludedStudies`) in subsequent model fitting, which are the `studyNumbers` that are not (indicated by `!`) included (`%in%`) the `targetStudies` plus the study with the study number = 313. Finally, we use `ctmaPrep` again and the `excludedStudies` object to compile a new list of primary studies (`study3plusList`) and check that it indeed includes the primary studies with three or more waves only.

10.2 Random Intercept CoTiMA (`ctmaInit, ctmaFit`)

In so-called static models (e.g., multi-level models (MLM); without autoregressive effects), researchers are frequently interested in separating *between-person* differences in *mean* levels of variables from *changes* over time *within-persons*. In dynamic models with autoregressive structures, between-person differences could be modelled by allowing the intercepts in the model equations to vary between persons.

So far, we have not yet dealt much with *continuous time intercepts*. Since CoTiMA is based on standardized variables (i.e., correlations instead of covariances), the mean levels of variables are 0.0 at all time points. Therefore, the mean intercepts are also 0.0 at all time points. This is because the intercepts are added to the values predicted by earlier states of the variables, and positive intercepts would therefore cause the mean level of the variables to become increasingly larger over time instead of staying at 0.0.

The fact that mean intercepts are 0.0 does nevertheless allow for individual differences in intercepts, which is also known as *random intercepts* (RI). Individuals with a positive intercept in one of the variables show an increasing trend in this variable over time, and individuals with negative intercepts exhibit a decreasing trend. Hence, whereas static MLM statistically control for between-person mean *levels* in variables, dynamic models including CoTiMA control for differences in possible linear *trends* or *growth factors* over time (Hamaker et al., 2018; Voelkle et al., 2012).

These trends may covary across latent variables and may covary with the initial latent states at Time 0. Frequently, people with large/low scores at Time 0 tend to maintain their large/low scores over time, but there is typically also a regression to the mean effect, so that, conversely, the continuous time intercepts are low/large. When there is no actual trend, but rather fluctuating scores around a stable level, the continuous time intercepts could also be interpreted as the *trait* level.

The CoTiMA package offers two different alternatives to include random intercepts, that is, to estimate the variance of the random intercepts, their covariances, and their covariances with the latent variables at Time 0 (e.g., in the case of two latent variables a 4 × 4 covariance matrix). In the case of initial fitting with `ctmaInit`, these two alternatives should yield identical fit and identical parameter

estimates. In practice, this is frequently not the case, and one should generally trust the model with the smallest *-2ll value* (see also Section 10.3 on optimizing model fit).

In case of a *full* or a *moderated* CoTiMA with random intercepts performed with `ctmaFit`, however, differences between the two alternatives exists. Out of the two alternatives, only one is the common random intercepts model referred to in the literature (e.g., Hamaker et al., 2018), whereas the others one is a restricted version. Recall that the CoTiMA package is based on the ctsem package (Driver & Voelkle., 2018), which imposes the restrictions on that random intercepts cannot be modelled as nested in primary studies. In other words, only a single covariance matrix of random intercepts is estimated that applies to all primary studies. Further, the correlation of the latent variables at Time 0, which is part of the random intercept covariance matrix, is therefore no longer nested in primary studies. This way to model random intercepts is used with the argument `indVarying`. This limitation could be overcome, and a true random intercepts model could be estimated by specific adaptations implemented in the CoTiMA package using the argument `randomIntercepts`, but this is at the cost of some complex internal model structure which could challenge the fitting algorithm. In case the model with `randomIntercepts` has the smallest *-2ll value* and no estimation problems are reported, these results should be preferred over the model using `indVarying`. When this is not the case, see Section 10.3 on optimizing model fit.

Note that using the argument `randomIntercepts` requires that all primary studies have at least three time points each. This is because the covariance matrix of random intercepts, which is estimated for each study, cannot be estimated when only two time points are available. With the argument `indVarying`, only "some" primary studies have to have three or more time points because a single covariance matrix of random intercepts is estimated across all studies. There have been no Monte Carlo simulation studies yet to suggest how many "some" primary studies should be; however, a rough estimate probably is 15%. Typically, the fit achieved with the argument `randomIntercepts` is better, but fitting an additional model with the argument `indVarying`, could be useful to check for consistency of results.

Both alternatives could be used in two different ways each. Both ways are algebraically identically, but sometimes either of them is numerically easier to fit and leads to better model convergence. In case latent variables have a single indicator only, it is algebraically identical to either let the continuous time intercepts or the *means* of the manifest indicators vary across persons. In the former case, an individual intercept is added to the latent variable at each time point, and in the latter case an individual *mean* is added to the manifest indicator. The former could be achieved by `indVarying = "CINT"` or by `randomIntercepts = "CINT"`, and the latter could be achieved by `indVarying = "MANIFEST"` or by `random-`

`Intercepts = "MANIFEST"`. This is demonstrated in Figure 62, where we use the list of primary studies (`study3plusList`) created in Section 10.1.

As mentioned earlier, random intercept models are very useful because they allow investigating within-person processes, and at least three time points have to be available in a primary study for mathematical reasons. Otherwise, a random intercept model is *not identified*. However, sometimes even three time points are insufficient for numerical reasons, and the four different ways to model random intercepts may not yield consistent results. When inconsistency is high, that is, when the resulting estimates vary much, one should treat them with care.

There are multiple ways to check the trustworthiness of results delivered by `ctmaInit`. We recommend applying all four alternatives and comparing their *-2ll values* and the resulting estimates. In case estimates of a model with poor *-2ll values* are very different from estimates of models with better and similar *-2ll values*, the former could be ignored, and the latter could eventually be trusted.

```
activeDirectory <- "../../" # SET A VALID PATH
CoTiMAInitFit_D_BO_3plus_IV <-
  ctmaInit(activeDirectory = activeDirectory, primaryStudies = study3plusList,
           n.latent = 2, scaleTime = 1/12, coresToUse = 2,
           indVarying = "CINT")

CoTiMAInitFit_D_BO_3plus_IV_mm <-
  ctmaInit(activeDirectory = activeDirectory, primaryStudies = study3plusList,
           n.latent = 2, scaleTime = 1/12, coresToUse = 2,
           indVarying = "MANIFEST")

CoTiMAInitFit_D_BO_3plus_RI <-
  ctmaInit(activeDirectory = activeDirectory, primaryStudies = study3plusList,
           n.latent = 2, scaleTime = 1/12, coresToUse = 2,
           randomIntercepts = "CINT")

CoTiMAInitFit_D_BO_3plus_RI_mm <-
  ctmaInit(activeDirectory = activeDirectory, primaryStudies = study3plusList,
           n.latent = 2, scaleTime = 1/12, coresToUse = 2,
           randomIntercepts = "MANIFEST")
```

Figure 62. Four different ways to model random intercepts with `ctmaInit` (`saveRDS` and `summary` functions not shown for reasons of space)

The R code shown in Figure 62 demonstrates one way to compare *-2ll values* and check for consistency of estimates. Figure 63 shows the corresponding output. First, the *-2ll values* of the four models fitted in Figure 62 are displayed. This shows that the models using random manifest means instead of random continuous time intercepts yielded better fit (44407.06 & 44407.06 compared to 44434.48 & 44416.57). As mentioned earlier, probable reason is that it is numerically easier for the fitting algorithm to find a proper solution. In particular, what is sometimes considered as the default way to fit a model with random intercepts (i.e., by `indVarying = "CINT"`) yielded to worst fit, that is, the largest *-2ll value*.

```
#Step 1: Display the -2ll values of the four models fitted in Figure 62
c(CoTiMAInitFit_D_BO_3plus_IV$summary$minus2ll, CoTiMAInitFit_D_BO_3plus_IV_mm$summary$minus2ll,
  CoTiMAInitFit_D_BO_3plus_RI$summary$minus2ll, CoTiMAInitFit_D_BO_3plus_RI_mm$summary$minus2ll)

# Step 2: Compaing and checking for consistency of estimates across the four models fitted
# in Figure 62
fits <- list(CoTiMAInitFit_D_BO_3plus_IV, CoTiMAInitFit_D_BO_3plus_IV_mm,
             CoTiMAInitFit_D_BO_3plus_RI, CoTiMAInitFit_D_BO_3plus_RI_mm)
allDrift <- as.data.frame((do.call(rbind, lapply(fits, function(x)
                      x$summary$drift_estimates_original_time_scale[1:4,]))))
allDrift[order(allDrift[,1],decreasing = FALSE),][,2:9]
```

Figure 63. R code to compare *-2ll values* of the four models fitted in Figure 62 and check for consistency of estimates

```
# Corresponding output for Step 1
[1] 44434.48 44407.06 44416.57 44407.06

# Corresponding output for Step 2
                V1toV1     SE  V2toV1   SE.1  V1toV2   SE.2  V2toV2   SE.3
Study.No.127   -0.2332 0.0459  0.0147 0.1445  0.0426 0.0913 -0.2461 0.0899
Study.No.127.1 -0.0294 0.0048 -0.0114 0.0116  0.0094 0.0083 -0.0542 0.0157
Study.No.127.2 -0.0383 0.0058  0.0152 0.0092  0.1197 0.0844 -0.2609 0.1376
Study.No.127.3 -0.0294 0.0049 -0.0117 0.0115  0.0095 0.0083 -0.0546 0.0155
Study.No.201   -0.1068 0.0198 -0.0118 0.0161  0.0072 0.0174 -0.1276 0.0223
Study.No.201.1 -0.1067 0.0199 -0.0117 0.0159  0.0076 0.0177 -0.1282 0.0223
Study.No.201.2 -0.1062 0.0197 -0.0120 0.0161  0.0072 0.0174 -0.1277 0.0222
Study.No.201.3 -0.1065 0.0200 -0.0120 0.0160  0.0073 0.0176 -0.1282 0.0225
Study.No.301   -0.1013 0.0203  0.0579 0.0134  0.0507 0.0148 -0.0720 0.0165
Study.No.301.1 -0.1014 0.0204  0.0582 0.0136  0.0510 0.0150 -0.0711 0.0165
Study.No.301.2 -0.1013 0.0199  0.0576 0.0131  0.0504 0.0146 -0.0715 0.0165
Study.No.301.3 -0.1009 0.0203  0.0580 0.0131  0.0510 0.0151 -0.0712 0.0158
Study.No.203   -0.1847 0.0283 -0.0272 0.0241  0.0019 0.0238 -0.1789 0.0265
Study.No.203.1 -0.1843 0.0281 -0.0273 0.0243  0.0023 0.0239 -0.1791 0.0270
Study.No.203.2 -0.1841 0.0280 -0.0273 0.0241  0.0021 0.0237 -0.1791 0.0270
Study.No.203.3 -0.1845 0.0281 -0.0269 0.0241  0.0022 0.0234 -0.1786 0.0268
Study.No.314   -0.5725 0.7090 -0.1815 1.0214 -0.2335 1.3111 -0.8973 1.1202
Study.No.314.1 -0.1819 0.0601 -0.1027 0.0300 -0.1323 0.0152 -0.2518 0.1011
Study.No.314.2 -0.2502 0.0848 -0.1784 0.0854 -0.2320 0.0774 -0.3686 0.1667
Study.No.314.3 -0.1919 0.0995 -0.1035 0.0584 -0.1335 0.0723 -0.2718 0.1698
```

Figure 64. Corresponding R output for the code used in Figure 63 (*suspicious* estimates are shown in bold face)

Despite the differences in *-2ll values*, the estimates shown in Figure 64 were mostly consistent across the four models; in particular, this applies to the two models using randomly varying manifest means, which also had the best *-2ll values*. Some *supicious* estimates are marked in bold face. Our decision to mark effects as supicious was based on the size of effects, in particular we regarded large effects and standard errors in models with large *-2ll values* as supicious. Overall, for a subsequent full CoTiMA, we should chose the `"MANIFEST"` options, which seems to work well for all primary studies, and we should use it together with the arugment `randomIntercepts = "MANIFEST"` rather than with `indVarying`, which would be more restrictive.

Fitting a full CoTiMA with random intercepts modelled as varying manifest means is demonstrated in Figure 65. Not shown here, we obtained *-2ll* = 44568.95 with 89 estimated parameters. More important, a warning was issued: `"Warning: ***Generalized inverse required for Hessian inversion -`

- standard errors not trustworthy". This indicates problems in proper model convergence. Section 10.3 shows how this and similar problems could be dealt with.

```
activeDirectory <- "../../" # SET A VALID PATH
CoTiMAFullFit_D_BO_3plus_RI <-
  ctmaFit(activeDirectory = activeDirectory,
          ctmaInitFit = CoTiMAInitFit_D_BO_3plus_RI,
          scaleTime = 1/12, coresToUse = 2,
          randomIntercepts = "MANIFEST")
```

Figure 65. Fitting a full CoTiMA with random intercepts modelled as randomly varying manifest means (`ctmaFit`; `saveRDS` and `summary` functions not shown for reasons of space)

10.3 Optimizing Fit (`ctmaOptimizeFit`)

The CoTiMA package offers a couple of possibilities for automatically varying certain fitting parameters during a series of fit attempts. For example, the time scale chosen may not be well-suited for the build-in starting values, which may prevent proper convergence.

Automatically varying certain fitting parameters during a series of fit attempts possibilities are implemented in the function, `ctmaOptimizeFit`, which generates a series of model fits (either using `ctmaInit` or `ctmaFit`), in which different time scales could be used. For example, the argument `randomScaleTime = c(1/24, 1/2)`, would vary the original time scale in subsequent model fits by multiplying it with a randomly selected factor between `1/24` and `1/2`.

Whereas the argument `randomScaleTime` could be used both when optimizing using `ctmaInit` or `ctmaFit`, there are arguments that apply either to optimizing with `ctmaFit` or `ctmaInit`. A first, but not much obvious way to facilitate model fit, is to vary the order of primary studies that were previously used with `ctmaInit`. Recall that in case of k primary studies, $k - 1$ dummy variables are internally used to control for the nested data structure (persons in studies). The last study in the list of k studies is used as the reference study for which no dummy variable is created. With the argument `shuffleStudyList = TRUE` the original order of the primary studies could be randomly changed in subsequent fit attempts with `ctmaFit`, which may also facilitate model convergence. Note that this argument has no impact when a single model fit should be optimized with `ctmaInit`.

A second way to facilitate fit is to standardize the dummy variables (time independent predictors, TIpreds, in ctsem terminology), so that they are no longer 0 or 1, but rather 0 on average with a standard deviation = 1.0. While this will not change the interpretation of the estimated drift coefficients, it may facilitate model convergence, too, and it is achieved by `randomScaleTI = TRUE`, which randomly switches between standardized and unstandardized dummy variables in subsequent fit attempts.

The `ctmaOptimizeFit` function could be used also to optimize the fit for a single primary study using `ctmaInit`. For this purpose, one could use the argument, for example, `problemStudy = 3`, which re-fits the third study in the study list subsequently using varying fitting parameters. We do not present this in a Figure here. Rather, we focus on optimizing a full CoTiMA.

```
activeDirectory <- "../../" # SET A VALID PATH
CoTiMAFullFit_D_BO_3plus_RI_opt <-
  ctmaOptimizeFit(activeDirectory = activeDirectory,
                  ctmaFitFit = CoTiMAFullFit_D_BO_3plus_RI,
                  ctmaInitFit = CoTiMAInitFit_D_BO_3plus_RI,
                  finishsamples = 10000, coresToUse = 2,
                  randomScaleTime = c(1/24, 1/2),
                  shuffleStudyList = TRUE,
                  saveModelFits = "/OptFitFiles/3plusRIopt",
                  reFits = 50)
```

Figure 66. Optimizing a full CoTiMA with random intercepts modelled as randomly varying manifest means (`ctmaOptimizeFit`; `saveRDS` and `summary` functions not shown for reasons of space)

Figure 66 demonstrates how to optimize the full RI-CoTiMA fitted as shown in Figure 65. Note that the model setup is not changed, and the arguments (e.g., `randomIntercepts = "MANIFEST"`) is taken from the `ctmaFitFit` object provided. In Figure 66, we requested 10,000 finishsamples to get precise estimates, we let the time scale factor randomly vary between 1/24 and 1/2 (of the original time scale, which was 1 month), we shuffle the list of primary study in subsequent fit attempts, we re-fit the model 50 times, and each fit is saved for inspecting the respective estimates later. Without providing the `saveModelFits` argument, the `ctmaOptimizeFit` function would return only the estimates of the best fitting model. The `saveModelFits` argument also prevents that already computed fits are lost in case very poor fitting parameters were randomly chosen, which may cause the abortion of `ctmaOptimizeFit`.

The R code shown in Figure 67 can be used to compare *-2ll values* and check for consistency of estimates for the optimized models re-fitted in Figure 70. Figure 68 shows the corresponding output. At the top of Figure 68, the rounded *-2ll values* of all 50 fit-attempts are displayed. Seven attempts yielded *-2ll values* < 44474. And the index (no.) of these fit-attempts is determined next. Then, the fit files of these indexed fit-attempts are read, the estimated drift effects are extracted, sorted, and displayed. There is very little variation in any of the four drift effects and their associated standard errors across the seven fit files. Hence, one would regard these results as trustworthy. Interesting to note in passing is the fact the neither of the two cross effects is significant, which would lead to the conclusion that once random intercepts are statistically modelled, there is no evidence for an effect of workload on exhaustion nor vice versa from exhaustion to workload.

```r
# Step 1: Display -2ll values of all 50 fit attempts
table(round(CoTiMAFullFit_D_BO_3plus_RI_opt$all_minus2ll,0))

# Step 2: Determine the index (no.) of the best fit attempts (here: -2ll < 44474)
index <- which(CoTiMAFullFit_D_BO_3plus_RI_opt$all_minus2ll < 44475)
index

# Step 3: The fit files of the indexed fit attempts are read, the estimated drift
# effects are extracted, sorted, and displayed.
activeDirectory <- "../../" # SET A VALID PATH

fits <- list()
for (i in 1:length(index)) { fits[[i]] <-
  readRDS(paste0(activeDirectory, "/OptFitFiles/3plusRIopt ", index[i], " .rds"))}
allDrift <- as.data.frame((do.call(
   rbind, lapply(fits, function(x) x$summary$estimates_original_time_scale[1:4,]))))
allDrift <-round(allDrift, 4)
allDrift[order(allDrift[,1], allDrift[,2], decreasing = FALSE),]
```

Figure 67. R code to compare *-2ll values* for the best-fitting models optimized and re-fitted in Figure 66 and their resulting drift estimates

```
# Corresponding output for Step 1
44474 44486 44521 44528 44566 44569 44574 44592 44614 44622 44630 44631 44668 44680 44686
    7     1     9     1     1     2     2    11     3     1     3     4     1     2     2
# Corresponding output for Step 2
[1] 13 14 33 35 37 41 50

# Corresponding output for Step 3
                          row col    Mean     sd    2.5%     50%   97.5%  Tvalues
DRIFT V1toV1 (invariant)    1   1 -0.1382 0.0126 -0.1645 -0.1377 -0.1149 -10.9936
DRIFT V1toV1 (invariant)1   1   1 -0.1378 0.0127 -0.1640 -0.1374 -0.1146 -10.8915
DRIFT V1toV1 (invariant)2   1   1 -0.1377 0.0126 -0.1634 -0.1373 -0.1139 -10.8977
DRIFT V1toV1 (invariant)3   1   1 -0.1380 0.0127 -0.1636 -0.1376 -0.1142 -10.9018
DRIFT V1toV1 (invariant)4   1   1 -0.1378 0.0127 -0.1639 -0.1374 -0.1147 -10.8690
DRIFT V1toV1 (invariant)5   1   1 -0.1380 0.0127 -0.1639 -0.1375 -0.1141 -10.9006
DRIFT V1toV1 (invariant)6   1   1 -0.1378 0.0125 -0.1635 -0.1372 -0.1145 -11.0153
DRIFT V2toV1 (invariant)    1   2 -0.0012 0.0100 -0.0207 -0.0012  0.0184  -0.1235
DRIFT V2toV1 (invariant)1   1   2 -0.0007 0.0101 -0.0201 -0.0006  0.0195  -0.0693
DRIFT V2toV1 (invariant)2   1   2 -0.0008 0.0100 -0.0204 -0.0008  0.0190  -0.0769
DRIFT V2toV1 (invariant)3   1   2 -0.0006 0.0099 -0.0202 -0.0006  0.0189  -0.0608
DRIFT V2toV1 (invariant)4   1   2 -0.0007 0.0100 -0.0209 -0.0006  0.0190  -0.0715
DRIFT V2toV1 (invariant)5   1   2 -0.0015 0.0100 -0.0210 -0.0014  0.0182  -0.1469
DRIFT V2toV1 (invariant)6   1   2 -0.0006 0.0101 -0.0207 -0.0005  0.0192  -0.0620
DRIFT V1toV2 (invariant)    2   1  0.0106 0.0108 -0.0106  0.0106  0.0320   0.9817
DRIFT V1toV2 (invariant)1   2   1  0.0108 0.0108 -0.0100  0.0110  0.0317   1.0053
DRIFT V1toV2 (invariant)2   2   1  0.0107 0.0106 -0.0100  0.0108  0.0317   1.0101
DRIFT V1toV2 (invariant)3   2   1  0.0108 0.0106 -0.0102  0.0108  0.0316   1.0156
DRIFT V1toV2 (invariant)4   2   1  0.0107 0.0108 -0.0103  0.0106  0.0318   0.9931
DRIFT V1toV2 (invariant)5   2   1  0.0105 0.0107 -0.0106  0.0105  0.0317   0.9802
DRIFT V1toV2 (invariant)6   2   1  0.0106 0.0107 -0.0102  0.0105  0.0317   0.9915
DRIFT V2toV2 (invariant)    2   2 -0.1341 0.0129 -0.1609 -0.1335 -0.1104 -10.4019
DRIFT V2toV2 (invariant)1   2   2 -0.1328 0.0126 -0.1585 -0.1325 -0.1096 -10.5301
DRIFT V2toV2 (invariant)2   2   2 -0.1328 0.0126 -0.1585 -0.1325 -0.1087 -10.5499
DRIFT V2toV2 (invariant)3   2   2 -0.1330 0.0125 -0.1588 -0.1326 -0.1098 -10.6058
DRIFT V2toV2 (invariant)4   2   2 -0.1326 0.0127 -0.1590 -0.1322 -0.1088 -10.3982
DRIFT V2toV2 (invariant)5   2   2 -0.1340 0.0126 -0.1600 -0.1336 -0.1101 -10.6008
DRIFT V2toV2 (invariant)6   2   2 -0.1328 0.0127 -0.1592 -0.1322 -0.1095 -10.4617
```

Figure 68. Corresponding R output for the code used in Figure 67

Thus far, it was quite a long way from initial fitting with `ctmaInit` to get trustworthy results for a full RI-CoTiMA. With the open science-movement, however, not only replicability of findings but also reproducibility of results became important. Facilitating reproducibility is demonstrated next.

10.4 Facilitating Reproducibility of Results by Providing Start Values

Reproducibility of findings could be facilitated much by using the best fitting model so far, extract starting values and other fitting parameters, and provide them as argument to the `ctmaFit` function. How this can be done is shown in Figure 69.

```
# Step 1: Extracting modelling parameter scaleTime
scaleTime <- CoTiMAFullFit_D_BO_3plus_RI_opt$usedTimeScale; scaleTime

# Step 2: Extracting modelling parameter scaleTI
scaleTI <- CoTiMAFullFit_D_BO_3plus_RI_opt$usedScaleTI; scaleTI

# Step 3: Extracting starting values (inits)
inits <- CoTiMAFullFit_D_BO_3plus_RI_opt$bestFit$studyFitList$stanfit$rawest
round(inits, 4)

# Step 4: Create the object primaryStudyList, which is used in Figure 71 below
primaryStudyList <- CoTiMAFullFit_D_BO_3plus_RI_opt$usedStudyList
```

Figure 69. R code for extracting modelling parameters and starting values (`inits`) from the best fitting model obtained by using the code in Figure 66

```
# Corresponding output for Step 1
[1] 0.29

# Corresponding output for Step 2
[1] TRUE

# Corresponding output for Step 3
[1]   0.0000  0.0000  0.0000  0.0000  0.6587 -0.0047  0.0365  0.6759 -1.4039  0.3049 -1.4064
[12] -0.9191  0.3693 -0.9200 -0.1402 -0.0790 -0.8780 -0.0935 -0.1274  0.7232 -0.8852  0.0000
[23]  0.0000  0.0000  0.0000  0.0000 -0.0001  0.0000  0.0000  0.0000  0.0000  0.0000  0.0000
[34]  0.0000  0.0001  0.0000  0.0000 -0.0399  0.0132 -0.0568 -0.0040 -0.0371 -0.0820 -0.0618
[45] -0.0674  0.0096  0.0636  0.0102  0.0553  0.0047 -0.0001 -0.0270 -0.0080 -0.1384 -0.0594
[56] -0.0427  0.0653  0.0245  0.0120 -0.0131  0.0304 -0.0613  0.0075 -0.0233  0.0229  0.0957
[67]  0.0398  0.0480 -0.0312  0.0241 -0.0068  0.0245  0.0052  0.1288  0.0063  0.0243 -0.0706
[78] -0.0418  0.0496  0.0159  0.0186 -0.2078 -0.0375 -0.0749  0.0271 -0.0009 -0.0381 -0.0033
[89] -0.0261
```

Figure 70. Corresponding R output for the code used in Figure 69

The values for `scaleTime (0.29)` and `scaleTI (TRUE)` as well as the entire set of `inits` displayed in Figure 70 are now used in a final call of the `ctmaFit` function in Figure 71. By this, the optimal fit should be recovered, and this is usually achieved much faster than without proper starting values.

```
activeDirectory <- "../../" # SET A VALID PATH
CoTiMAFullFit D BO 3plus RI fin <-
  ctmaFit(activeDirectory = activeDirectory,
          inits = c( 0.0000, 0.0000, 0.0000, 0.0000, 0.6587,-0.0047, 0.0365, 0.6759,
                    -1.4039, 0.3049,-1.4064,-0.9191, 0.3693,-0.9200,-0.1402,-0.0790,
                    -0.8780,-0.0935,-0.1274, 0.7232,-0.8852, 0.0000, 0.0000, 0.0000,
                     0.0000, 0.0000,-0.0001, 0.0000, 0.0000, 0.0000, 0.0000, 0.0000,
                     0.0000, 0.0000, 0.0001, 0.0000, 0.0000,-0.0399, 0.0132,-0.0568,
                    -0.0040,-0.0371,-0.0820,-0.0618,-0.0674, 0.0096, 0.0636, 0.0102,
                     0.0553, 0.0047,-0.0001,-0.0270,-0.0080,-0.1384,-0.0594,-0.0427,
                     0.0653, 0.0245, 0.0120,-0.0131, 0.0304,-0.0613, 0.0075,-0.0233,
                     0.0229, 0.0957, 0.0398, 0.0480,-0.0312, 0.0241,-0.0068, 0.0245,
                     0.0052, 0.1288, 0.0063, 0.0243,-0.0706,-0.0418, 0.0496, 0.0159,
                     0.0186,-0.2078,-0.0375,-0.0749, 0.0271,-0.0009,-0.0381,-0.0033,
                    -0.0261),
          randomIntercepts = "MANIFEST",
          ctmaInitFit = CoTiMAInitFit_D_BO_3plus_RI,
          primaryStudyList = primaryStudyList,
          scaleTime = .29,
          finishsamples = 100000,
          scaleTI = TRUE,
          coresToUse = 2)
```

Figure 71. Model specification for a full CoTiMA with random intercepts modelled as randomly varying manifest means with optimized `scaleTime`, `scaleTI`, and `inits` arguments (`ctmaOptimizeFit`)

The results are shown in parts in Figure 72 and Figure 73. Figure 72 is limited to the first out of the five studies. The section `$randomIntercepts$popcov_mean` shows the covariance matrix of the random intercepts for the first study, and the section `$randomIntercepts$popcov_sd` shows their standard errors. The covariances could be easily transformed into correlations by using the cov2cor function available in R (e.g., `cov2cor(…$popcov_mean[[1]])`), and T-values to assess the significance of the covariances could be obtained by dividing the two matrices (e.g., `…$popcov_mean[[1]]/…$ popcov_sd[[1]]`)

```
$randomIntercepts$popcov_mean
$randomIntercepts$popcov_mean[[1]]
            [,1]        [,2]        [,3]        [,4]
[1,]  0.75544101  0.20699384 -0.06486421 -0.05119978
[2,]  0.20699384  0.69241780 -0.06442819 -0.07796963
[3,] -0.06486421 -0.06442819  0.77227645  0.39082200
[4,] -0.05119978 -0.07796963  0.39082200  0.84741154
...

$randomIntercepts$popcov_sd
$randomIntercepts$popcov_sd[[1]]
            [,1]       [,2]       [,3]       [,4]
[1,] 0.01487585 0.01418275 0.01443790 0.01489576
[2,] 0.01418275 0.01407399 0.01435217 0.01521140
[3,] 0.01443790 0.01435217 0.01525741 0.01646765
[4,] 0.01489576 0.01521140 0.01646765 0.01824070
...
```

Figure 72. Part 1 of selected results of the partial CoTiMA specified in *Figure 36* (`ctmaFit`)

The section `$estimates_original_time_scale` at the bottom of Figure 73 reveals estimates that are very close to those previously obtained when optimizing the fit. This also applies to the fit, which was *-2ll* = 44473.94.

```
$estimates
                           row col    Mean     sd    2.5%     50%   97.5%  Tvalues
…
DRIFT V1toV1 (invariant)   1   1 -0.4756 0.0438 -0.5658 -0.4740 -0.3950 -10.8566
DRIFT V2toV1 (invariant)   1   2 -0.0021 0.0345 -0.0696 -0.0022  0.0657  -0.0612
DRIFT V1toV2 (invariant)   2   1  0.0372 0.0371 -0.0356  0.0373  0.1096   1.0046
DRIFT V2toV2 (invariant)   2   2 -0.4585 0.0438 -0.5494 -0.4567 -0.3777 -10.4782
…
dtDRIFT_1_1                1   1  0.6222 0.0270  0.5682  0.6226  0.6737  23.0095
dtDRIFT_1_2                1   2 -0.0013 0.0216 -0.0435 -0.0014  0.0413  -0.0608
dtDRIFT_2_1                2   1  0.0233 0.0232 -0.0224  0.0233  0.0686   1.0036
dtDRIFT_2_2                2   2  0.6330 0.0277  0.5772  0.6336  0.6857  22.8832

$minus2ll
[1] 44473.94

$n.parameters
[1] 89

$opt.lag.orig.time
     [,1] [,2]
[1,]   NA    7
[2,]    7   NA

$opt.lag.scaled.time
     [,1] [,2]
[1,]   NA 2.03
[2,] 2.03   NA

$max.effects
       [,1]   [,2]
[1,]     NA 0.0017
[2,] 0.0293     NA


$estimates_original_time_scale
                           row col    Mean     sd    2.5%     50%   97.5%  Tvalues
DRIFT V1toV1 (invariant)   1   1 -0.1379 0.0127 -0.1641 -0.1375 -0.1146 -10.8566
DRIFT V2toV1 (invariant)   1   2 -0.0006 0.0100 -0.0202 -0.0006  0.0191  -0.0612
DRIFT V1toV2 (invariant)   2   1  0.0108 0.0108 -0.0103  0.0108  0.0318   1.0046
DRIFT V2toV2 (invariant)   2   2 -0.1330 0.0127 -0.1593 -0.1324 -0.1095 -10.4782
```

Figure 73. Part 2 of selected Results of the partial CoTiMA specified in *Figure 36* (`ctmaFit`)

10.5 The Relation between RI-CoTiMA and Latent Change-Score Models (LCS)

Both the autoregressive continuous time cross-lagged panel model used for CoTiMA, and latent change score models (LCS) are dynamic models. There is a close relation between them (Voelkle & Oud, 2015), in particular between CoTiMA and multivariate dual LCS models. Continuous time models, however, are less restrictive than LCS model. LCS model use difference equations as an approximation of the

differential equations that is applied by CoTiMA and that are typically used to describe changes across time in a variety of disciplines such as physics or econometry. Further, LCS typically require equally spaced time points whereas CoTiMA does not.

Like in CoTiMA, when applying LCS models, researchers are typically interested in the *rate* of change across a particular time interval. In dual change LCS models, there are individual differences in rate of change, which are conceptually related to random intercepts (see Section 10.2), but the former is frequently referred to as a *latent slope* whereas the latter as a *trait* of *intercept* (cf. Voelkle & Oud, 2015). Still, as we noted earlier, the trait could also be interpreted as a growth factor (see Section 10.2).

When all time intervals were invariant across people, time points, and primary studies, the "true" rate of change, the individual differences in growth, and the unexplained variance in latent factors can be expected to be identical for continuous time and dual LCS models across this particular time interval (Voelkle & Oud, 2015). Nevertheless, the metrics in which the estimated parameters are presented are different, as well is the terminology. Instead of cross effects, LCS results frequently contain estimated *coupling parameters*, and instead of auto effects, LCS results frequently contain estimated *proportional effects*. Further, when the assumption of invariant time intervals is violated, continuous time models could yield unbiased effects only. These unbiased effects, fortunately, could be translated into what could have been expected under ideal conditions with LCS modeling.

Since dual LCS model have latent slopes, in order to be translated into dual LCS effects, an appropriate CoTiMA model has to include traits, that is, random intercepts, too. At this stage, it is the `indVarying` argument that is helpful rather than the `randomIntercepts` argument because a single overall covariance matrix of intercepts is required rather than covariance matrices of intercepts for each primary study. In other words, the LCS model is more restrictive than the RI-CoTiMA model used with the argument `randomIntercepts`.

To obtain an optimally fitting CoTiMA model with the argument `indVarying = "CINT"`, we need to go through the same series of steps used in Figure 65, Figure 66, Figure 67, Figure 69 and Figure 71. First, we create a fit-object with `ctmaFit`, then, second, an optimized fit-object with `ctmaOptimizeFit`, and finally, third, check if the results are reproducible in a final fit-object named `CoTiMAFullFit_D_BO_3plus_IV_fin` (see Figure 74).

Dual LCS results are eventually obtained by applying the `ctmaLCS` function to the final fit-object, that is, `ctmaLCS(CoTiMAFullFit_D_BO_3plus_IV_fin)`. This is shown in Figure 75 . The results are displayed in Figure 76.

```
activeDirectory <- "../../" # SET A VALID PATH

# Step 1: Create a fit-object
CoTiMAFullFit_D_BO_3plus_IV <-
  ctmaFit(activeDirectory = activeDirectory,
          ctmaInitFit = CoTiMAInitFit_D_BO_3plus_IV,
          scaleTime = 1/12, coresToUse = 2, indVarying = "CINT")

# Step 2: Create an optimized fit-object
CoTiMAFullFit_D_BO_3plus_IV_opt <-
  ctmaOptimizeFit(activeDirectory = activeDirectory,
                  ctmaFitFit = CoTiMAFullFit_D_BO_3plus_IV,
                  ctmaInitFit = CoTiMAInitFit_D_BO_3plus_IV,
                  finishsamples = 10000, coresToUse = 2,
                  randomScaleTime = c(1/24, 1/2), shuffleStudyList = TRUE,
                  saveModelFits = "/OptFitFiles/3plusIVopt",
                  reFits = 50)

# Step 3: Compare -2ll values for the best-fitting models optimized and re-fitted
table(round(CoTiMAFullFit D BO 3plus IV opt$all minus2ll,0))
index <- which(CoTiMAFullFit_D_BO_3plus_IV_opt$all_minus2ll < 44664)
index
fits <- list()
for (i in 1:length(index)) { fits[[i]] <-
  readRDS(paste0(activeDirectory, "/OptFitFiles/3plusIVopt ", index[i], " .rds"))}
allDrift <- as.data.frame((do.call(rbind, lapply(fits, function(x)
                           x$summary$estimates_original_time_scale[1:4,]))))
allDrift <- round(allDrift, 4)
allDrift[order(allDrift[,1], allDrift[,2], decreasing = FALSE),]

# Step 4: Extracting modelling parameters and starting values (inits)
scaleTime <- CoTiMAFullFit_D_BO_3plus_IV_opt$usedTimeScale; scaleTime
scaleTI <- CoTiMAFullFit_D_BO_3plus_IV_opt$usedScaleTI; scaleTI
inits <- CoTiMAFullFit_D_BO_3plus_IV_opt$bestFit$studyFitList$stanfit$rawest
round(inits, 4)
primaryStudyList <- CoTiMAFullFit_D_BO_3plus_IV_opt$usedStudyList
length(primaryStudyList)

# Step 5: Check if the results are reproducible in a final fit-object named Co-
TiMAFullFit_D_BO_3plus_IV_fin
CoTiMAFullFit_D_BO_3plus_IV_fin <-
  ctmaFit(activeDirectory = activeDirectory,
          inits = c( 0.0000,  0.0000, -0.3146, -0.1116,  0.1541, -0.3786,
                    -1.0016,  0.3125, -0.9842,  0.0000,  0.0000, -0.6265,
                    -0.6265, -0.3338, -0.3334,  0.4358,  0.8459,  0.2513,
                     0.3288,  0.8568,  0.6563,  0.0401,  0.0365, -0.0312,
                     0.0522,  0.1388,  0.1669,  0.0304,  0.0748,  0.0253,
                    -0.0260, -0.0084,  0.0352),
          indVarying = "CINT"
          ctmaInitFit = CoTiMAInitFit_D_BO_3plus_IV,
          primaryStudyList = primaryStudyList,
          scaleTime = .06,
          finishsamples = 100000,
          scaleTI = TRUE,
          coresToUse = 2)
```

Figure 74. R Code to obtain an optimally fitting CoTiMA model with the argument `indVarying = "CINT"`

```
activeDirectory <- "../../" # SET A VALID PATH
resultsLCS <- ctmaLCS(CoTiMAFullFit D BO 3plus IV fin)
readRDS(resultsLCS, paste0(activeDirectory, "LCS_D_BO_3plus_IV_fin.rds"))
resultsLCS
```

Figure 75. R code for the dual LCS results obtained in Figure 76

		est	SD	LL	UL
1	InitialMean_eta1	0.0000	0.0184	-0.0361	0.0362
2	InitialMean eta2	0.0000	0.0185	-0.0361	0.0361
3	ManifestMean_eta1	0.0000	0.0000	0.0000	0.0000
4	ManifestMean_eta2	0.0000	0.0000	0.0000	0.0000
5	SlopeMean Cint eta1	0.0003	0.0370	-0.0727	0.0729
6	SlopeMean_Cint_eta2	-0.0002	0.0374	-0.0736	0.0731
7	CTauto_V1toV1	-0.1263	0.0114	-0.1495	-0.1048
8	CTauto V2toV2	-0.1365	0.0124	-0.1618	-0.1130
9	CTcross_V1toV2	0.0090	0.0103	-0.0111	0.0292
10	CTcross_V2toV1	-0.0070	0.0096	-0.0256	0.0118
11	Proportion_eta1	-0.1186	0.0100	-0.1389	-0.0995
12	Proportion_eta2	-0.1275	0.0108	-0.1494	-0.1068
13	Autoregressions_eta1 (Delta=1)	0.8814	0.0100	0.8611	0.9005
14	Autoregressions_eta2 (Delta=1)	0.8725	0.0108	0.8506	0.8932
15	Coupling_V1toV2	0.0079	0.0090	-0.0097	0.0255
16	Coupling_V2toV1	-0.0061	0.0084	-0.0225	0.0103
17	CrossLagged_V1toV2 (Delta=1)	0.0079	0.0090	-0.0097	0.0255
18	CrossLagged_V2toV1 (Delta=1)	-0.0061	0.0084	-0.0225	0.0103
19	Diffusion_eta1	0.0964	0.0058	0.0856	0.1083
20	Diffusion_eta2	0.1028	0.0066	0.0905	0.1162
21	Diffusion_eta2_eta1	0.0292	0.0046	0.0203	0.0384
22	DynErrVar_eta1 (Delta=1)	0.0850	0.0043	0.0768	0.0937
23	DynErrVar_eta2 (Delta=1)	0.0901	0.0048	0.0810	0.0998
24	DynErrCov_eta2_eta1 (Delta=1)	0.0257	0.0034	0.0191	0.0325
25	measurementErrorVar_eta1	0.0000	0.0000	0.0000	0.0000
26	measurementErrorVar_eta2	0.0000	0.0000	0.0000	0.0000
27	measurementError V2 with V1	0.0000	0.0000	0.0000	0.0000
28	InitialVar_eta1	0.9987	0.0515	0.9008	1.1028
29	InitialVar_eta2	0.9988	0.0517	0.9004	1.1027
30	InitialCov eta2 eta1	0.4333	0.0400	0.3570	0.5132
31	SlopeVariance_TraitVariance_eta1	3.0095	0.6087	1.9895	4.3619
32	SlopeVariance_TraitVariance_eta2	3.0127	0.6314	1.9524	4.4286
33	SlopeCov_TraitCov_eta2_eta1	1.7176	0.4849	0.8426	2.7439
34	SlopeCor_TraitCor_eta2_eta1	0.5686	0.1021	0.3435	0.7419
35	Minus2LL	44663.8705	NA	NA	NA
36	NumEstParams	33.0000	NA	NA	NA

Figure 76. LCS results and CoTiMA results of a CoTiMA model with the argument `indVarying = "CINT"` created in Figure 74 and Figure 75(eta1 = demands; eta2 = burnout; `ctmaLCS`)

Figure 76 shows that the initial means of the two latent variables, that is the mean latent scores at Time 0, were 0.0 for both variables. The same applies to the mean of their manifest indicators because we deal with standardized variables here.

As noted earlier, what is termed a (linear) slope in dual LCS models corresponds to the continuous time intercepts in CoTiMA. The slopes/intercepts are also 0.0 for both variables because variables are standardized.

The next four lines #7 to #10 show the continuous time auto and cross effects. Their sizes were very similar to those presented at the bottom of Figure 73, even though the ones presented here were based on a more restrictive CoTiMA.

Lines #11 to #14 each show the carry-over effects of latent variables over time and their mutual influences, respectively, over a time interval of 1 (= one month). The proportional effects represent the carry-over effects in LCS terminology, and the autoregressions the counterpart in discrete time cross-lagged panel models. Whereas continuous time auto effects and LCS proportional effects are typically negative, discrete time autoregressive effects usually are positive.

Lines #15 to #18 each show the mutual influences of latent variables over time. Like the continuous time cross effects in lines #9 and #10, the coupling effects and cross-lagged effects were not significant either.

Prediction errors are shown next. Lines #22 to #24 display the continuous time covariance of the innovation terms as summarized in the diffusion matrix, and lines #22 to #24 the discrete time dynamic error covariances.

Although it is possible to separate prediction error from measurement error in CoTiMA, this typically requires more than three time points in practice. Therefore, the measurement error covariances matrix was fixed to 0.0 (default in CoTiMA) and line #25 to #27 contain only zeros.

Lines #28 to #30 displays the Time 0 covariance matrix of the two latent variables. Since we deal with standardized variables, this is actually a correlation matrix.

Finally, lines #31 and #32 show the variances of the slopes in LCS parlance or the variance of the continuous time intercepts in CoTiMA terminology, respectively. Further, lines #33 and #34 display the covariance and correlation of the slopes/intercepts of the two latent variables. Thus, this is the between-person covariance/correlation of demands and exhaustion, which is positive and significant, and which has been demonstrated in literally thousands of studies.

11 References

Bartoš, F., & Schimmack, U. (2022). Z-curve 2.0: Estimating Replication and Discovery Rates. Meta-Psychology, 2022, vol 6, MP.2021.2720. https://doi.org/10.15626/MP.2021.2720

Boker, S. M., Neale, M., Maes, H., Wilde, M., Spiegel, M., Brick, T., . . ., & Fox, J. (2011). OpenMx: An open source extended structural equation modeling framework. *Psychometrika, 76,* 306–317. https://doi.org/10.1007/s11336-010-9200-6

Borenstein, M., Hedges, L. V., Higgins, J. P., & Rothstein, H. R. (2009). *Introduction to meta-analysis.* John Wiley & Sons Inc.

Brunner, J., & Schimmack, U. (2020). Estimating population mean power under conditions of heterogeneity and selection for significance. *Meta-Psychology, 4,* MP.2018.874, https://doi.org/10.15626/MP.2018.874

Carpenter, B., Hoffman, M. D., Brubaker, M., Lee, D. D., Li, P., & Betancourt, M. (2015). *The Stan Math Library: Reverse-Mode Automatic Differentiation in C++.* CoRR abs/1509.07164

Childs, J. H., & Stoeber, J. (2012). Do you want me to be perfect? Two longitudinal studies on socially prescribed perfectionism, stress and burnout in the workplace. *Work & Stress, 26,* 347–364. http://dx.doi.org/10.1080/02678373.2012.737547

Dormann C., & Homberg, M. (2022). *CoTiMA.* GitHub repository, https://github.com/CoTiMA/CoTiMA

Dormann, C., & Griffin, M. A. (2015). Optimal time lags in panel studies. *Psychological Methods, 20,* 489 –505. http://dx.doi.org/10.1037/met0000041

Dormann, C., Guthier, C., & Voelkle, M. (2020). *CoTiMA Burnout.* Retrieved from osf.io/e92jd

Driver, C. C., Oud, J. H. L., & Voelkle, M. C. (2017). Continuous Time Structural Equation Modeling with R Package ctsem. *Journal of Statistical Software, 77*(5), 1–35. https://doi.org/10.18637/jss.v077.i05

Driver, C. C., & Voelkle, M. C. (2018). Hierarchical Bayesian continuous time dynamic modeling. *Psychological Methods, 23*(4), 774–799. http://dx.doi.org/10.1037/met0000168

Guddat, C., Grouven, U., Bender, R., & Skipka, G., (2012). A note on the graphical presentation of prediction intervals in random-effects meta-analyses. *Systematic Reviews, 1,* 1–34. https://doi.org/10.1186/2046-405

Grund, S., Lüdtke, O., & Robitzsch, A. (2022). Using Synthetic Data to Improve the Reproducibility of Statistical Results in Psychological Research. *Psychological Methods.* Advance online publication. http://dx.doi.org/10.1037/met0000526

Guthier, C., Dormann, C., & Voelkle, M. C. (2020). Reciprocal effects between job stressors and burnout: A continuous time meta-analysis of longitudinal studies. *Psychological Bulletin, 146*(12), 1146–1173. https://doi.org/10.1037/bul0000304

Hamaker, E. L., Kuiper, R. M., & Grasman, R. P. P. P. (2015). A critique of the cross-lagged panel model. *Psychological Methods, 20,* 102-116. http://dx.doi.org/10.1037/a0038889

Hamaker, E. L., Asparouhov, T., Brose, A., Schmiedek, F., & Muthén, B. (2018). At the Frontiers of Modeling Intensive Longitudinal Data: Dynamic Structural Equation

Models for the Affective Measurements from the COGITO Study. *Multivariate Behavioral Research, 53*, 820–841. https://doi.org/10.1080/00273171.2018.1446819
R Core Team (2020). *R: A language and environment for statistical computing.* R Foundation for Statistical Computing. http://www.R-project.org/
Posit team (2024). *RStudio: Integrated Development for R.* Posit Software PBC. http://www.posit.co/
Spearman, C. (1904). The proof and measurement of association between two things. *The American Journal of Psychology, 15*(1), 72–101. JSTOR 1412159
Stan Development Team (2020). *RStan: the R interface to Stan. R package version 2.21.2.* http://mc-stan.org/
Stanley, T. D., & Doucouliagos, H. (2014). Meta-regression approximations to reduce publication selection bias. *Research Synthesis Methods, 5*(1), 60–78. http://dx.doi.org/10.1002/jrsm.1095
Stanley, T. D., Carter, E. C., & Doucouliagos, H. (2018). What meta-analyses reveal about the replicability of psychological research. *Psychological Bulletin, 144*(12), 1325–1346. http://dx.doi.org/10.1037/bul0000169
Sterne, J. A., & Egger, M. (2001). Funnel plots for detecting bias in meta-analysis: Guidelines on choice of axis. *Journal of Clinical Epidemiology, 54*(10), 1046–1055. http://dx.doi.org/10.1016/S0895-4356(01)00377-8
Venables W. N., & Ripley, B. D. (2002). *Modern Applied Statistics with S, Fourth edition.* Springer. http://www.stats.ox.ac.uk/pub/MASS4/
Voelkle, M. C., & Oud, J. H. L. (2015). Relating latent change score and continuous time models. *Structural Equation Modeling: A Multidisciplinary Journal, 22*, 366–381. DOI: 10.1080/10705511.2014.935918
Voelkle, M. C., Oud, J. H., Davidov, E., & Schmidt, P. (2012). An SEM approach to continuous time modeling of panel data: Relating authoritarianism and anomia. *Psychological Methods, 17,* 176–192. http://dx.doi.org/10.1037/a0027543

12 Appendix A: Release Notes

Functions described in this guide bevor are available via the R package CoTiMA downloaded from CRAN. Additional functions and new arguments are continuously developed and can be installed in the beta version from our GitHub repository (Dormann & Homberg, 2022). Therefore, after the devtools R package is installed (`install.packages("devtools")`), the latest version of the R package CoTiMA can also be installed using the code shown in *Figure 77*.

```
library(devtools)
install_github("CoTiMA/CoTiMA")
library(CoTiMA)
```

Figure 77. Installing CoTiMA from GitHub

13 Appendix B: Overview of CoTiMA Functions and their Arguments

Argument	Default	Possible Values	Explanation
`ctmaBiG (EPIC-BiG-Power)`			Performs fixed effect analysis, random effect analysis, and analysis of publication bias (Egger's tests, PET-PEESE corrections, z-curve analysis).
`ctmaInitFit`	`NULL`	CoTiMA fit-object	CoTiMA fit-object created with `ctmaInit`.
`activeDirectory`	`NULL`	character string	Specifies the directory where required files are found and saved. Should end with `"/"`. For example, `"/Users/GDC/CoTiMA/"`.
`activateRPB`	`FALSE`	`FALSE/TRUE`	Messages (warning, finished fitting) could be sent to mobile phone if set to `TRUE`.
`digits`	`4`	value > 0	Rounding used in output.
`PETPEESEalpha`	`.10`	values between 0 and 1	Probability level (condition) below which to change from PET to PEESE.
`undoTimeScaling`	`TRUE`	`FALSE/TRUE`	Uses the time scale used in the Extract-Step (i.e., as defined in the `delta_ti` objects when `TRUE`).
`zcurve`	`FALSE`	`FALSE/TRUE`	Performs z-curve analysis. Could fail if too few studies (e.g., around 10) are supplied.
`dT`	`NULL`	value > 0	Performs all analyses for one discrete time interval, too, using the interval assigned to `dT`.

Argument	Default	Possible Values	Explanation
`ctmaCompFit (EPIC-BiG-Power)`			Compares the fit of two nested CoTiMA models (**l**iberal model1 on the **l**eft, **r**estricted model2 on the **r**ight) via *-2ll difference test*. Note that the nested structure of the two models is assumed to be given and is not checked.
`model1`	`NULL`	CoTiMA fit-object	CoTiMA fit-object created with `ctmaInit`, `ctmaFit`, or `ctmaEqual`.
`model2`	`NULL`	CoTiMA fit-object	CoTiMA fit-object created with `ctmaInit`, `ctmaFit`, or `ctmaEqual`.

Argument	Default	Possible Values	Explanation
`ctmaCorRel (EPIC-BiG-Power)`			Corrects (disattenuates) correlation matrix for unreliability.
`alphas`	`NULL`	vector of the same length as dimensionality of `empcov`	
`empcov`	`NULL`	symmetric correlation matrix	

Argument	Default	Possible Values	Explanation
`ctmaEmpCov` (E**P**IC-BiG-Power)			Changes a full correlation matrix by selecting target variables, recode them, combine them (add), and add rows/columns with NA if focal variables are missing.
`combineVariables`	`c()`	list of (vectors of) variable positions	Creates composite variables (i.e., means of one or more variables). Variables that should be combined have to be listed in a vector. Variables that should not be combined have to be listed, too. For example, `list(1, c(2, 3), 4, c(5, 6))` combines the 2nd and 3rd as well as the 5th and 6th variables of `empcovi`. and leaves the 1st and 4th variable untouched. Instead of positions, variable names could be used if they were also used in the argument `targetVariables`.
`CombineVariablesNames`	`c()`	vector variable names for combined variables	not yet operational
`empcov`	`NULL`	symmetric correlations matrix	Correlation matrix reported in a primary study.
`n.latent`	`NULL`	value > 0	The number of (latent) variables. Actually, it is the number of all variables at T0. A distinction between latent and manifest variables is not made here.
`missingVariables`	`c()`	vector of variable positions	Augments `empcovi` and `pairwiseNi` by rows and columns containing NA in order to create matrices of the desired dimension. For example, if the desired matrix should contain correlations of the four variables x_0, y_0, x_1 *and* y_1, but a primary study did not measure y_1, then the 4th variable is missing and the correlation matrix returned by `ctmaEmpCov` will be a 4 × 4 `empcovi` and a 4 × 4 `pairwiseNi` with NA in the 4th row and in the 4th column, respectively.
`pairwiseN`	`NULL`	symmetric matrix of same dimensions as `empcov` containing possible `pairwiseN`.	A matrix with sample sizes for each correlation of `empcovi`.
`recodeVariables`	`c()`	vector of variable positions or variable names	Recodes desired variables in `empcovi` (i.e., changes the signs of the correlations). For example, `c(1, 4)` changes the signs of the correlations in the 1st and 4th row of `empcovi`. Instead of positions, variable names could be used if `dimnames` were assigned to `empcovi`. Note that if numbers are used, they should correspond to the positions in the `targetVariablesi` object rather than the rows/columns in the `empcovi` object (i.e., recoding is done after `targetVariablesi` were selected from `empcovi`).
`sampleSize`	`NULL`	value > 0	The sample size. It does not need to be specified if `pairwiseNi` is provided instead.
`targetVariables`	`NULL`	vector of variable positions or variable names	Selects desired variables in `empcovi` (i.e., deletes those variables that should not be analyzed). For example, `c(1, 2, 4, 5)` deletes the 3rd and 6th row and column in a 6 × 6 `empcovi`. Instead of positions, variable names could be used if `dimnames` were assigned to `empcovi`.
`Tpoints`	`NULL`	value > 1	The number of time points.

Argument	Default	Possible Values	Explanation
`ctmaEqual (EPI`**`C`**`-BiG-Power)`			Statistically tests if the two or more invariant drift parameters in the CoTiMA fit-object supplied are equal.
`activateRPB`	`FALSE`	`FALSE/TRUE`	Messages (warning, finished fitting) could be sent to mobile phone if set to `TRUE`.
`activeDirectory`	`NULL`	path to directory	Specifies the directory where required files are found and saved. Should end with `"/"`. For example, `"/Users/GDC/Co-TiMA/"`.
`coresToUse`	`2`	value > 0 or < 0	The number of cores (threads) to be used for fitting. If a negative value is provided, the value is subtracted from available cores, else the value sets the number of cores to be used.
`ctmaInvariantFit`	`NULL`	CoTiMA fit-object	CoTiMA fit-object that was returned by `ctmaFit`. In most cases this is probably the fit of a model in which two effects were specified with the `invariantDrift` argument (e.g., two cross effects).
`digits`	`4`	value > 0	Rounding used in output.

Argument	Default	Possible Values	Explanation
`ctmaFit (EPIC-BiG-Power)`			Fits a CoTiMA model.
`activateRPB`	`FALSE`	`FALSE/TRUE`	Messages (warning, finished fitting) could be sent to mobile phone if set to `TRUE`.
`activeDirectory`	`NULL`	path to directory	Specifies the directory where required files are found and saved. Should end with `"/"`. For example, `"/Users/GDC/CoTiMA/"`.
`allInvModel`	`FALSE`	`FALSE/TRUE`	Whether or not a model should be tested in which all (!) parameters are assumed to be invariant across primary studies. If set to `TRUE`, other specifications (e.g., specified with the `equalDrift` argument) will be ignored. An all-invariant model is also used by `ctmaPower`.
`binaries`	`NULL`	vector consisting of 0 and 1	Experimental argument. The vector of 0 (for continuous) and 1 (for dichotomous) defines the scale of manifest variables. May require many (e.g., > 30) time points for valid results.
`catsToCompare`	`NULL`	vector of categories' values that should be compared	This argument is the 2nd out of 3 used to specify contrast among categorical moderators. Compared to an unconstrained moderator model, the effects of the categories of the vector specified are set equal. This will reduce the fit (i.e., increase the *-2ll value*), which can be used for comparing it with the unconstrained model. A significant difference indicates that the categories' effects are not equal. For example, `catsToCompare = c(2, 3)` sets the drift effects (see below) for categories 2 and 3 equal. Note that the smallest category selected will become the new comparison category (instead of the lowest of all category numbers) and the output will be labelled accordingly.
`chains`	`NULL` (= 2)	value > 0 and < available cores	The `chains` argument is passed to `ctStanFit` and defines the number of chains to be used for Bayesian estimation.
`CINT`	0	string vector of names of means of continuous time intercepts	Usually, CoTiMA assumes that standardized variables (correlations) are analyzed, which should result in estimates of `manifestMeans` (and `T0mean`) to be 0.0. To facilitate convergence, these parameters are set to 0.0 by default. They can be set free by providing names. Note that this is automatically done if `indVarying = "CINT"` is specified.
`cluster`	`NULL`	vector of same length as number of primary studies	Vector with cluster variables (e.g., countries), e.g., `c(1, 1, 1, 3, 3, 6, 7, 8)`. Has to be set up carefully. Will be included in `ctmaPrep` in later developmental stages of the CoTiMA package.
`coresToUse`	2	value > 0 or < 0	The number of cores (threads) to be used for fitting. If a negative value is provided, the value is subtracted from available cores, else the value sets the number of cores to be used.
`CoTiMAStanctArgs`	`NULL`	list of further fitting parameters	All fitting parameters that are allowed in `ctStanFit` can be specified here, too.
`ctmaInitFit`	`NULL`	CoTiMA fit-object	Object to which all single ctsem fits of primary studies has been assigned to (i.e., what has been returned by `ctmaInit`).
`customPar`	`FALSE`	`FALSE/TRUE`	If set to `TRUE` some starting values usually used by `ctStanFit` will be used by CoTiMA specific settings. Not recommended to be used in combination with Bayesian estimation. It was introduced to improve handling of large values used in `delta_ti`. Setting it to `FALSE` and use `scaleTime` instead could be a better alternative if estimation problems will nevertheless occur.
`digits`	4	value > 0	Rounding used in output.
`drift`	`NULL` (= all)	vector (!) of rowwise drift matrix elements	Labels for drift effects that should or should not be included. Have to be either of the type *V1toV2* or 0 for effects to be excluded, which is usually not recommended, e.g., `c("V1toV1", "V2toV1", 0, "V2toV2")`.

Table continues ...

Argument	Default	Possible Values	Explanation
`ctmaFit (EPIC-BiG-Power)`			*... continued*
`driftsToCompare`	`NULL`	string vector of drift names	This argument is the 3rd out of 3 used to specify contrasts among categorical moderators. The strings in the vector define which drift effects is analyzed for possible differences among categories of the moderator(s). For example, `driftsToCompare = c("V1toV2")` together with `catsToCompare = c(2, 3)` and `modsToCompare = 2` fits a model in which the effect of category 2 and 3 on the drift parameter *V1toV2* of the second moderator is set equal.
`equalDrift`	`NULL`	vector of 0 or 2 or more drift effects	Labels for drift effects that should be set equal. Have to be of the type *V1toV2*, e.g., `c("V2toV1", "V1toV2")`. Constrains all listed effects to be equal (e.g., `equalDrift = c("V1toV2", "V2toV1")`). Note that this is not required for testing the assumption that two effects are equal in the population. Use the `invariantDrift` argument and then `ctmaEqual` to fit second model.
`finishsamples`	`NULL` (= 1000)	values > 0	The `finishsamples` argument is passed to `ctStanFit`. It specifies the number of samples to draw for final results computation. Larger (e.g., 10.000) values make results more exactly replicable. Larger values are recommended before manuscripts are submitted. Very large values (e.g., 100.000) might be helpful if very small effects (e.g., 0.0002) result from estimation.
`fit`	`TRUE`	`FALSE/TRUE`	If set to `FALSE` only compiled data frame and ctsem model is returned. Useful for customizing ctsem model.

`ind.mod.names`	`NULL`	string vector of names	Vector of names for individual level (!) moderators used in output. Can only be used with primary studies providing raw data. Individual level moderators are usually provided as last columns in raw data files, and this is the specified in `rawDatai` objects by adding the number of individual level moderators, e.g.,
			`rawData2 <- list(fileName = paste0(activeDirectory, "raw-Data2.txt"), studyNumbers = 6, missingValues = c(-99), standardize = TRUE, header = TRUE, dec = ".", sep = " ", n.ind.mod = 2)`
			However, individual level moderators can also be added later to existing fit-objects created with `ctmaInit`. In the following example, raw data are extracted from an existing fit-object created with `ctmaInit`, then the mean of all time intervals per individual is computed, which is then added to a copy of the fit-object created with `ctmaInit` and saved. Time interval length as study-level of individual-level moderator could indicate that qualitatively different processes are captured by different sets of discrete time points.
			`CoTiMAInitFit <- readRDS(paste0(activeDirectory, "CoTiMAInitFit.rds"))`
			`ind.mod.List <- list()`
			`for (i in 1:length(CoTiMAInitFit$studyFitList)) {`
			`wide <- CoTiMAInitFit$studyFitList[[i]]$empraw`
			`dtCols <- grep("dT", colnames(wide))`
			`naPos <- which(wide[ ,dtCols] <= .001, arr.ind = TRUE)`
			`wide[,dtCols][naPos] <- NA # replace missing (coded as <.001) by NA`
			`ind.mod.List[[i]] <- data.frame(matrix(NA, ncol = 1, nrow = nrow(wide)))`
			`colnames(ind.mod.List[[i]]) <- "TI1"`
			`ind.mod.List[[i]]$TI1 <- c(apply(as.matrix(wide[, dtCols], nrow = nrow(wide)), 1, mean, na.rm = TRUE))`
			`}`
			`CoTiMAInitFit_iml <- CoTiMAInitFit`
			`CoTiMAInitFit_iml$ind.mod.List <- ind.mod.List`
			`saveRDS(CoTiMAInitFit_iml, paste0(activeDirectory, "CoTiMAInitFit_iml.rds"))`

`ind.mod.number`	`NULL`	vector of positions of moderators	Which moderator (in the vector of individual level (!) moderators) shall be used (e.g., 2 for a single moderator or 1:3 for 3 moderators simultaneously). Can only be used with primary studies providing raw data.

Table continues ...

Argument	Default	Possible Values	Explanation
ctmaFit (EPI**C**-BiG-Power)			*... continued*
ind.mod.type	"CONT"	"cont" or "cat"	"cont" or "cat" of the individual level (!) moderators (mixing them in a single model not yet possible). Can only be used with primary studies providing raw data.
indVarying	FALSE	FALSE/TRUE/ "CINT"	Specifies a restricted random intercept (RI) model, however, without allowing the variance of the intercepts to vary between studies (see argument randomIntercepts). RI models could be specified using individually varying ct intercepts (indVarying = "CINT") or individually varying manifests (indVarying == TRUE). In case n-manifest = n.latent, both models are algebraically identical. However, numerically, models with individually varying manifests are easier to fit. Note that *random intercept* models are referred to as *fixed effect* models in the econometric literature. Is set to FALSE if the argument randomIntercepts is provided.
indVaryingT0	NULL	FALSE/TRUE	Deprecated
inits	NULL	vector of raw inits	Typical use is to provide raw parameter estimates of a previous fit (of an identical model). Could ensure replicability of results and reduces computation time. Raw estimates could be extracted from previous fit stored in, e.g., previousFit, by `inits <- previousFit$studyFitList$stanfit$rawest`
invariantDrift	NULL (= all)	vector of drift effects or "none"	Labels for drift effects that should be aggregated. Have to be of the type *V1toV2*, e.g., c("V2toV1").
iter	NULL (= 1000)	values > 0	The iter argument is passed to ctStanFit. It specifies the number of iterations used for Bayesian estimation, half of which will be devoted to warmup.
lambda	NULL (= identity matrix)	n.latent × n.manifest matrix	Matrix with pattern of fixed (= 1) or free (any string) loadings.
manifestMeans	0	string vector of names of means of manifest variables	Usually, CoTiMA assumes that standardized variables (correlations) are analyzed, which should result in estimates of manifestMeans (and T0mean) to be 0.0. To facilitate convergence, these parameters are set to 0.0 by default. They can be set free by providing names. Note that this is automatically done if indVarying = TRUE is specified. When many waves of data exist, it could be possible to separate latent means from manifest means by setting either of them free.
manifestVars	0	0 or n.manifest × n.manifest matrix with values or strings	Lower triangular matrix with error(co-)variances of manifest indicators. Usually, CoTiMA assumes that a single indicator is used per latent. This typically requires assuming error variances to be 0.0 if only few waves are available. Alternatively, they can be assigned a particular value, e.g., 1- Cronbach's alpha. In cases where many waves of observation are available, the error variance of single manifest indicators can be estimated, too. This is achieved by assigning labels.

Table continues ...

Argument	Default	Possible Values	Explanation
`ctmaFit (EPIC-BiG-Power)`			*... continued*
`mod.names`	`NULL`	(vector of) character object(s)	Names used to label moderators in the output.
`mod.number`	`NULL`	vector of positions of moderators	The position(s) of the moderator(s) in the vector of moderator values compiled with `ctmaPrep`, which should be used in a moderated CoTiMA, e.g., `c(1, 3)`.
`mod.type`	`"cont"`	`"cont"` or `"cat"`	Type of moderator(s). Categorical and continuous moderators could not be used in combination, but more than one continuous or more than one categorical moderator is possible.
`moderatedDrift`	`NULL` (= all)	vector of drift effects	Labels for drift effects that should be moderated. Have to be of the type *V1toV2*, e.g., `c("V2toV1")`. Is only used if moderators are specified.
`modsToCompare`	`NULL`	vector of positions of selected (!) moderators	This argument is the 1st out of 3 used to specify contrast among categorical moderators. The values define the position(s) of the moderator(s) in the vector of moderator values selected with the `mod.number` argument (see above). For example, if `mod.number = c(1, 4, 6)` was specified before, `modsToCompare = 2` specifies that subsequent contrasts will be performed for the moderator at the 4th position of the moderator(s) in the vector of moderator values compiled with `ctmaPrep`.
`nopriors`	`TRUE`	`FALSE/TRUE`	Deprecated
`optimize`	`TRUE`	`FALSE/TRUE`	The `optimize` argument is passed to `ctStanFit`. If `FALSE`, Bayesian estimations is used. The chosen sampler is conditional on the `priors` argument. Note that this works differently than the `optimise` argument of `ctStanFit`.
`primaryStudyList`	`NULL`	list of primary studies	A list of primary studies compiled with `ctmaPrep` that contains a subset of studies included in `ctmaInit`. Useful to exclude studies without the need to use `ctmaInit` again.
`priors`	`FALSE`	`FALSE/TRUE`	Replaces previously used and now deprecated `nopriors` argument. Consequences of `TRUE` or `FALSE` are conditional on the `optimize` argument. `optimize = TRUE` & `priors = FALSE` implies maximum likelihood estimation, `optimize = TRUE` & `priors = TRUE` implies maximum a posteriori estimation, `optimize = FALSE` & `priors = TRUE` implies Bayesian estimation using NUTS (No U-Turn Sampler).
`randomIntercepts`	`FALSE`	`FALSE/TRUE/ "MANIFEST"`	Implements an unrestricted random intercepts model where either the intercepts (`TRUE`) or the manifest indicator" (`"MANIFEST"`) vary within and between primary studies. Sometimes difficult to fit. It requires that all (!) primary studies have 3 or more waves!
`sameInitialTimes`	`FALSE`	`FALSE/TRUE`	Useful argument when raw data are used. When set to `FALSE` (default), all initial times points with fully missing data are eliminated from raw data frames, which is prevented when set to `TRUE`. Could be useful when it is desired to interpret continuous time intercepts as growth factors with the same initial starting point (e.g., school enrolment).

Table continues ...

Argument	Default	Possible Values	Explanation
`ctmaFit (EPIC-BiG-Power)`			*... continued*
`scaleClus`	`TRUE`	`FALSE/TRUE`	The argument `scaleClus = TRUE` leads to centering of dummy variables representing different clusters. If the argument `scaleClus = FALSE` is used, the resulting drift effects are those for the reference cluster. The reference cluster either is a cluster that consists of all cluster values that are unique (see main text for explanation) or the cluster with the largest number. If the argument `scaleClus = TRUE` is used, the internally created dummy variables are centered (but not standardized). The resulting drift effects are no longer those for the reference cluster. Rather, they represent the average effect across all studies and all clusters. The effect of the centered cluster dummies now represents the mean difference between studies belonging to a particular cluster and all other studies. Since one is usually interested in both the size of a drift effect for a particular cluster, we recommend not centering the dummy variables representing clusters (i.e., the argument `scaleClus = FALSE` should be used).
`scaleMod`	`TRUE`	`FALSE/TRUE`	Whether or not continuous moderators should be standardized or categorical moderators should be centered. Recommended for continuous moderators. With categorical moderators, the argument `scaleMod = TRUE` leads to centering rather than standardization, which requires some explanation. Categorical moderators are internally transformed into dummy variables. The smallest category value represents the reference category. For instance, if 1 & 2 are unused, 3 is < 19 yrs old, 4 = 19-60 yrs, 5 = all ages mixed, two dummy variables are created. Dummy 1 is 1 if a primary study included people aged 19-60 only, and Dummy 1 is 0 in all other cases. Dummy 2 is 1 if a primary study included people with ages mixed, and dummy 2 is 0 in all other cases. If the argument `scaleMod = FALSE` is used in combination with categorical moderators, the resulting drift effects are those for the reference category, that is, for studies including people < 19 yrs only. The effect of Dummy 1 the represents the changes that result if a study includes people aged 19-60 rather than < 19 yrs. If this effect is significant, the interpretation is that the effect is significantly different compared to the effect found in studies using people < 19 yrs only. Correspondingly, the effect of Dummy 2 the represents the changes that result if a study includes people with all ages mixed rather than < 19 yrs. If the argument `scaleMod = TRUE` is used in combination with categorical moderators, the internally created dummy variables are centered (but not standardized; see the argument `transfMod` to circumvent this default behavior and standardize them anyway). The resulting drift effects are no longer those for the reference category. Rather, they represent the average effect across all studies and all moderator categories. The effect of the centered Dummy 1 now represents the mean difference between studies including people aged 19-60 and all other studies. If the effect of Dummy 1 is significant, its interpretation is that this category has a significantly different effect compared to all other categories. It does, however, no longer inform about the sizes of the drift effects resulting for this category. Since one is usually interested in both the size of a drift effect for a particular category and if it is significantly different, we recommend not centering the dummy variables representing categorical moderators (i.e., the argument `scaleMod = FALSE` should be used). See the argument `catsToCompare` to analyzed difference between categories anyway.

Table continues ...

Argument	Default	Possible Values	Explanation
ctmaFit (EPI**C**-BiG-Power)			*... continued*
scaleTI	TRUE	FALSE/TRUE	With the move from the omx (OpenMX) version of ctsem to the stanct (stan) version, CoTiMA moved from fitting a multi group model to a model in which the groups are represented by dummy variables. These are internally handled as time independent (TI) predictors, and scaleTI specifies whether or not these dummy variables should be standardized. Recommended since version 0.5.3.1 and set to TRUE. Note that standardizing dummies for primary studies does not affect estimation of aggregated effects. However, without standardization the study-specific effects (in which meta-analysist are usually not interested in) cannot be interpreted as within-study effects.
scaleTime	NULL (= FALSE)	value > 0	This argument can facilitate model convergence. It internally changes the time scale assigned to delta_ti. For example, scaleTime = 1/12 could change the time scale from months to years. It is usually recommended to avoid delta_ti larger than 6.
transfMod	NULL	character vector applying R functions to x	Can be used as a replacement of scaleMod if more than one moderator is analyzed and standardization is not desired for all moderators. Then, it could take the form transfMod = c("scale(x)", "x", "scale(x)"). Alternatively, users can define their own functions, e.g., transfMod=("(x - mean(x))"); this function centralizes a single moderator without standardization. Another example is transfMod=("scale(x) - min(x)"), which standardizes x and then shift values to a scale beginning with 0.0. This yields the (unmoderated) drift effects for the reference group with them smallest moderator value.
T0means	0	string vector of names of means of T0 latent variable	Usually, CoTiMA assumes that standardized variables (correlations) are analyzed, which should result in estimates of T0mean (and manifestMeans) to be 0.0. To facilitate convergence, these parameters are set to 0.0 by default. They can be set free by providing names. Note that this is automatically done if indVarying = TRUE is specified.
T0var	"auto"	NULL/"auto"/ lower triangular matrix	Could be used to specify the Time 0 covariance matrix by setting certain elements free or fixing them to particular values. Usually not recommended.
useSampleFraction	NULL	value < 1	To speed up debugging. Provided as fraction (e.g., 1/10).
verbose	0	0, 1, or 2	The verbose argument is passed to ctStanFit. Higher values print more information during model fit.
WEC	FALSE	FALSE/TRUE	Required if it is intended to estimate the reduction in heterogeneity of effects achieved by introducing study-level moderators. When set to TRUE, *weighted effect coding* of the TIpreds representing the dummies of the primary studies is applied. Returns drift matrices for all primary studies. Should be used once in conjunction with the argument scaleTI = FALSE, and in a second fit with scaleTI = FALSE *plus* additional moderators. The two returned fit-objects then serve as input for the ctmaRedHet function. The first of these two models mimics the ctmaInit function and, like ctmaInit, provides estimates for all primary studies. Fitting this model requires a rather complex internal model setup, and it is therefore highly recommended to compare the estimates with those provided by ctmaInit. When there are large differences in estimates and the *-2ll value* returned by ctmaInit is smaller than the *-2ll value* returned by ctmaFit with WEC = TRUE, the latter results cannot really be trusted.

Argument	Default	Possible Values	Explanation
`ctmaFitList (plotting)`			Informs the plot function that more than a single CoTiMA fit-object should be plotted.
`...`		CoTiMA fit-objects separated by commas	Everything provided as argument within () will be put into a list because the plotting function requires list as arguments. For example, `ctmaFitList(object1, object2)`.

Argument	Default	Possible Values	Explanation
`ctmaGetPub (-)`			Retrieves publication and citation indices for authors from Google Scholar, which could be further processed with `ctmaPub`.
`authorList`	`NULL`	List of vectors with 2 elements	Contains information about authors' names and their Google Scholar https address (or their user ID), e.g., `list( c("Wilmar B.; Schaufeli", "https://scholar.google.de/citations?hl=en&user = w1tHcj4AAAAJ"), c("Maureen; Dollard", "user = J6oH3rgAAAAJ") ) ).` Authors' surnames are separated from given names or initials by a semicolon!
`flush`	`FALSE`	`FALSE/TRUE`	Argument is handed over to scholar R package. If set to `TRUE`, the cache will be cleared, and the data reloaded from Google Scholar. Google Scholar will limit the retrieval of information or even suspend it for a while if the cache is flushed too frequently.
`yearsToExclude`	`NULL`	(vector of) years to exclude	Recommended to leave as `NULL`. Years could be excluded later when using `ctmaPub`.

Argument	Default	Possible Values	Explanation
ctmaInit (EPIC-BiG-Power)			Fits a ctsem model to a list of primary studies prepared by ctmaPrep.
activateRPB	FALSE	FALSE/TRUE	Messages (warning, finished fitting) could be sent to mobile phone if set to TRUE.
activeDirectory	NULL	character string	Specifies the directory where required files are found and saved. Should end with "/". For example, "/Users/GDC/CoTiMA/".
chains	NULL (= 2)	values > 0	The chains argument is passed to ctStanFit and specifies the number of chains to be used for Bayesian estimation.
checkSingle StudyResults	TRUE	FALSE/TRUE	If set to TRUE, displays estimates from single study ctsem models and waits for user input to continue. Useful to check estimates before they are saved.
CINT	0	string vector of names of means of continuous time intercepts	Usually, CoTiMA assumes that standardized variables (correlations) are analyzed, which should result in estimates of manifestMeans (and T0mean) to be 0.0. To facilitate convergence, these parameters are set to 0.0 by default. They can be set free by providing names. Note that this is automatically done if indVarying = "CINT" is specified.
coresToUse	2	value > 0 or < 0	The number of cores (threads) to be used for fitting. If a negative value is provided, the value is subtracted from available cores, else the value sets the number of cores to be used.
CoTiMAStanctArgs	NULL	list of further fitting parameters	All fitting parameters that are allowed in ctStanFit can be specified here, too.
customPar	FALSE	FALSE/TRUE	If set to TRUE some starting values usually used by ctStanFit will be used by CoTiMA specific settings. Not recommended to be used in combination with Bayesian estimation. It was introduced to improve handling of large values used in delta_t*i*. Setting it to FALSE and use scaleTime instead could be a better alternative if estimation problems will nevertheless occur.
diff	NULL	string vector of names of diffusions	Labels for diffusion effects. Have to be either of the character strings of the type "diff_eta1" or "diff_eta2_eta1" (= freely estimated) or values (e.g., 0 for effects to be excluded, which is usually not recommended).
digits	4	value > 0	Rounding used in output.
doPar	1	integer value > 0	Deprecated.
drift	NULL (= all)	vector (!) of row-wise drift matrix elements	Labels for drift effects that should or should not be included. Have to be either of the type *V1toV2* or 0 for effects to be excluded, which is usually not recommended, e.g., c("V1toV1", "V2toV1", 0, "V2toV2").
experimental	FALSE	FALSE/TRUE	
finishsamples	NULL (= 1000)	values > 0	The finishsamples argument is passed to ctStanFit. It defines the number of samples to draw for computation of final results. Larger (e.g., 10.000) values make results more exactly replicable. Larger values are recommended before manuscripts are submitted. Very large values (e.g., 100.000) might be helpful if very small effects (e.g., 0.0002) result from estimation.
fit	TRUE	FALSE/TRUE	When set to FALSE, ctmaInit does not fit the data to the model and returns the data and the ctsem model only, which could be used to make desired adaptations before using ctStanFit.
indVarying	FALSE	FALSE/TRUE	Specifies a random intercept RI model. Works only if all primary studies have 3 or more waves and no missing values (i.e., variables) exist! Note that contrary to ctmaFit indVarying = "CINT" is identical to randomIntercepts = TRUE, and indVarying = TRUE is identical to randomIntercepts = "MANIFEST".

Table continues ...

Argument	Default	Possible Values	Explanation
ctmaInit (EPIC-BiG-Power)			*... continued*
indVaryingT0	TRUE	FALSE/TRUE	Deprecated.
iter	NULL (= 1000)	values > 0	The iter argument is passed to ctStanFit. It specifies the number of iterations used for Bayesian estimation, half of which will be devoted to warmup.
lambda	NULL (= identity matrix)	n.latent × n.manifest matrix	Matrix with pattern of fixed (= 1) or free (any string) loadings.
loadSingleStudyModelFit	c()	string vector with filename followed by the numbers of studies for which the fit is saved	Load the fit of single study ctsem models, e.g., loadSingleStudyModelFit = c("myModel", 1, 4, 5, 6:100). This is useful, e.g., if primary studies are added to the pool of primary studies. Only the added studies will be fitted, the previously fitted models are loaded, and all is then stored in the fit-object created with ctmaInit.
manifestMeans	0	string vector of names of means of T0 latent variables	Usually, CoTiMA assumes that standardized variables (correlations) are analyzed, which should result in estimates of manifestMeans (and T0mean) to be 0.0. To facilitate convergence, these parameters are set to 0.0 by default. They can be set free by providing names. Note that this is automatically done if indVarying = TRUE is specified.
manifestVars	NULL	0 or n.manifest × n.manifest matrix with values or strings	Lower triangular matrix with error(co-)variances of manifest indicators. Usually, CoTiMA assumes that a single indicator is used per latent. This typically requires assuming error variances to be 0.0. Alternatively, they can be assigned a particular value, e.g., 1-Cronbach's alpha. In cases where many waves of observation are available, the error variance of single manifest indicators could be estimated, too. This is achieved by assigning labels.
n.latent	NULL	value > 0	Number of latent variables.
n.manifest	0 (= n.latent)	value ≥ n.latent	Number of manifest variables.
optimize	TRUE	FALSE/TRUE	The optimize argument is passed to ctStanFit. If FALSE, Bayesian estimations is used. The chosen sampler is conditional on the priors argument. Note that this works differently than the optimise argument of ctStanFit.
primaryStudies	NULL	list	A list created with ctmaPrep that contains all information (e.g., empcovi, delta_ti, sampleSizei etc.) relevant for ctmaInit and subsequent analyses.
priors	FALSE	FALSE/TRUE	Replaces previously used and now deprecated nopriors argument. Consequences of TRUE or FALSE are conditional on the optimize argument. optimize = TRUE & priors = FALSE implies maximum likelihood estimation, optimize = TRUE & priors = TRUE implies maximum a posteriori estimation, optimize = FALSE & priors = TRUE implies Bayesian estimation using NUTS (No U-Turn Sampler)..

Table continues ...

Argument	Default	Possible Values	Explanation
`ctmaInit (EPIC-BiG-Power)`			*... continued*
`randomIntercepts`	`FALSE`	`FALSE/TRUE`	Specifies a random intercept RI model. Works only if all primary studies have 3 or more waves without missing values (i.e., variables) exist! Note that contrary to `ctmaFit, indVarying = "CINT"` is identical to `randomIntercepts = TRUE`, and `indVarying = TRUE` is identical to `randomIntercepts = "MANIFEST"`. The difference between the arguments `randomIntercepts` and `indVarying` is the internal model setup (as processes vs. intercepts), which should yield identical fit. However, sometimes one of the two models is numerically easier to fit, and the fit with the smaller *-2ll value* should be preferred.
`saveRawData`	`list()`	list	A list with required information to save generated pseudo raw data. Might be useful for methodological research questions. For example: `list("saveRawData$studyNumbers = c(1: 20), "saveRawData$fileName" = "pseudoRaw", "saveRawData$row.names" = FALSE, "saveRawData$col.names" = TRUE, "saveRawData$sep" = " ", "saveRawData$dec" = ".")`
`sameInitialTimes`	`FALSE`	`FALSE/TRUE`	Useful argument when raw data are used. When set to `FALSE` (default), all initial times points with fully missing data are eliminated from raw data frames, which is prevented when set to `TRUE`. Could be useful when it is desired to interpret continuous time intercepts as growth factors with the same initial starting point (e.g., school enrolment).
`saveSingleStudyModelFit`	`c()`	vector with filename followed by the numbers of studies for which the fit is saved	Save the fit of single study ctsem models (could save a lot of time afterwards if the fit is loaded, e.g., `saveSingleStudyModelFit = c("myModel", 1, 4, 5, 6:100)`
`scaleTI`	`NULL` (= `TRUE`)	`FALSE/TRUE`	Whether or not the to standardize the TI predictors that represent the primary study dummies.
`scaleTime`	`NULL` (= `FALSE`)	value > 0	Whether or not the time scale used for `delta_ti` should be changed. For example, `scaleTime = 1/12` could change the time scale from months to years. It is usually recommended to avoid `delta_ti` larger than 6.
`silentOverwrite`	`FALSE`	`FALSE/TRUE`	Whether or not to prompt user preventing undesired overwriting of existing single study fit files (requested via `saveSingleStudyModelFit`).
`T0means`	0	string vector of names of means of T0 latent variables	Usually, CoTiMA assumes that standardized variables (correlations) are analyzed, which should result in estimates of `T0mean` (and `manifestMeans`) to be 0.0. To facilitate convergence, these parameters are set to 0.0 by default. They can be set free by providing names.
`T0var`	`"auto"`	0 or n.manifest × n.manifest matrix with values or strings	Could be used to specify the Time 0 covariance matrix by setting certain elements free or fixing them to particular values. Usually not recommended.
`useSV`	`FALSE`	`FALSE/TRUE`	If set to `TRUE` provided starting values will be used.
`verbose`	`0`	0, 1, or 2	The `verbose` argument is passed to `ctStanFit`. Higher values print more information during model fit.

Argument	Default	Possible Values	Explanation
ctmaLCS (-)			Takes a CoTiMA fit-object or CTSEM fit and transforms estimates into those estimates typically reported when fitting (dual) latent change score (LCS) models. The fit-object has to include random intercepts (either randomly varying ct intercepts obtained with indVarying = "CINT" or with or randomly varying manifest means obtained with indVarying = TRUE). The function could also be used to transform estimates produced with indVarying = TRUE into estimates that would be obtained with indVarying = "CINT".
CoTiMAFit	NULL	CoTiMA fit-object (or ctsem fit object)	Object to which all CoTiMA fit-object has been assigned to (i.e., what has been returned by ctmaFit). The ctmaLCS function also takes fit-objects delivered by CTSEM.
undoTimeScaling	TRUE	FALSE/TRUE or any value	Undoes possible time scaling achieve in ctmaInit or ctmaFit by setting the argument scaleTime. When a number is provided instead of a logical argument, the number is used to multiply the obtained effects (e.g., when time is scaled in days in a study, and scaleTime = 30.5 is used, the returned effect corresponds to 1-month intervals).
activateRPB	FALSE	FALSE/TRUE	Messages (warning, finished fitting) could be sent to mobile phone if set to TRUE.
digits	4	integer value > -1	Rounding used in output.

Argument	Default	Possible Values	Explanation
ctmaOptimizeFit (EP**IC**-BiG-Power)			Repeated initial fitting (i.e., applies `ctmaInit`) to a primary study or repeated full fitting (i.e., applies `ctmaFit`) several times to capitalize on chance for obtaining a hard-to-find optimal fit. Certain fitting parameters can be specified to randomly vary across fitting attempts.
`activateRPB`	`FALSE`	`FALSE/TRUE`	Messages (warning, finished fitting) could be sent to mobile phone if set to `TRUE`.
`activeDirectory`	`NULL`	character string	Specifies the directory where required files are found and saved. Should end with `"/"`. For example, `"/Users/GDC/CoTiMA/"`.
`coresToUse`	`2`	value > 0 or < 0	The number of cores (threads) to be used for fitting. If a negative value is provided, the value is subtracted from available cores, else the value sets the number of cores to be used.
`CoTiMAStanctArgs`	`NULL`	list of further fitting parameters	All fitting parameters that are allowed in `ctStanFit` can be specified here, too.
`ctmaFitFit`	`NULL`	CoTiMA fit-object created with `ctmaFit`	The CoTiMA Full fit-object of which the fit should be improved.
`ctmaInitFit`	`NULL`	CoTiMA fit-object created with `ctmaInit`	The CoTiMA Init fit-object that was used when creating the CoTiMA Full fit-object.
`customPar`	`NULL`	`FALSE/TRUE`	If set to `TRUE` some starting values usually used by `ctStanFit` will be used by CoTiMA specific settings. Not recommended to be used in combination with Bayesian estimation. It was introduced to improve handling of large values used in `delta_ti`. Setting it to `FALSE` and use `scaleTime` instead could be a better alternative if estimation problems will nevertheless occur.
`finishsamples`	`NULL` (= 1000)	values > 0	The `finishsamples` argument is passed to `ctStanFit`. It specifies the number of samples to draw for final results computation. Larger (e.g., 10.000) values make results more exactly replicable. Larger values are recommended before manuscripts are submitted. Very large values (e.g., 100.000) might be helpful if very small effects (e.g., 0.0002) result from estimation.
`Iter`	5000	values > 0	Number of iterations allowed.
`primaryStudies`	`NULL`	list()	A list created with `ctmaPrep` that contains all information (e.g., `empcov`*i*, `delta_ti`, `sampleSize`*i* etc.). Relevant for `ctmaInit` only. In a typical workflow, one would create new list with `ctmaPrep` that only contains the primary studies that were previously identified to yield improper fits.
`problemStudy`	`NULL`	value > 0	Number of the study (not the position) in the primaryStudies list that should be re-fitted.
`randomPar`	`FALSE`	`FALSE/TRUE`	If set to `TRUE`, it overrides the `customPar` argument used to create the CoTiMA Full fit-object. Instead, `customPar` varies randomly between `TRUE` and `FALSE` during the number of fit attempts specified with `reFits`.
`randomScaleTI`	`FALSE`	`FALSE/TRUE`	If set to `TRUE`, randomly varies randomly between `TRUE` and `FALSE` for scaling the time independent predictors (TIpreds) representing primary studies. Only relevant if `ctmaOptimizeFit` is used to re-fit primary studies (i.e., when poor fit was obtained with `ctmaFit`).
`randomScaleTime`	`c(1,1)`	a pair of positive values	From a uniform distribution within the provided lower and upper limits a value (rounded to 2 decimals) is drawn for each refit attempt.
`reFits`	`NULL`	value > 0	How many reFits should be done.
`saveModelFits`	`FALSE`	`FALSE/TRUE`	Saves each fitted model into the activeDirectory using the name `"saveModelFits`*i*`.rds"`, with *i* representing the number of the current fit attempt. Could be useful if out-of-range estimates interrupt the `refit` attempts.
`scaleTI`	`TRUE`	`FALSE/TRUE`	Scales the dummy coded (0, 1) time independent predictors (TIpreds) representing the k – 1 primary study. May help to achieve improved model convergence and better model fit.

Table continues ...

Argument	Default	Possible Values	Explanation
`ctmaOptimizeFit (EPIC-BiG-Power)`			*... continued*
`shuffleStudyList`	`FALSE`	`FALSE/TRUE`	Usually, the last primary study in the `primaryStudyList` is the reference study, for which no dummy variable exists. Sometimes, changing the reference study makes fitting easier, and the argument `shuffleStudyList` randomly shuffles the order of the primary studies (and save the current order in the returned fit file for replication).
`verbose`	`0`	0, 1, or 2	The `verbose` argument is passed to `ctStanFit`. Higher values print more information during model fit.

Argument	Default	Possible Values	Explanation
ctmaPlotCtsemMod (-)			Auxiliary function to plot moderator effects estimated with the ctsem package using ctStanFit.
ctStanFitObject	NULL	CTSEM (ctStanFit) fit-object	Fit-object created with ctStanFit, in which a single moderator (= time independent predictor; TIpred) moderates drift effects.
fitSummary	NULL	Summary object of CTSEM (ctStanFit) fit-object	Object containing the summary of the fit-object. Saves time by preventing repeatedly applying the summary function on the fit-object.
activeDirectory	NULL	path to directory	Specifies the directory where required files are found and saved. Should end with "/". For example, "/Users/GDC/Co-TiMA/".
Tipred.pos	1	values > 0	The time-independent predictor (TIpred) that represents the moderator. Could be more than one in case dummy variables made from of categorical moderators (e.g., Tipred.pos = c(3,4)).
saveFilePrefix	"Moderator Plot "	character string	Prefix used for saving plots.
scaleTime	1	value > 0	Factor to increase or decrease the time scale (e.g., 1/12 if estimates were based on yearly intervals and figure should show monthly intervals).
mod.sd.to.plot	-1:1	vector	The standard deviation values of the moderator, for which the drift effects are plotted. The argument is ignored if the moderator is represented by dummy variables made from a categorical variable.
digits	4	value > 0	Rounding used in output.
timeUnit	"not specified"	character string	Label for the x-axis.
timeRange	NULL	vector with 3 values: c(xMin, xMax, stepwidth)	The range across which discrete time effects are plotted, e.g., c(10, 20, .01) would plot effects from 10 units of time to 20 using steps of .01. Note that a stepwidth < 1 could be specified to obtain more fine-grained figures.
yLimitsForEffects	values slightly exceeding min and max empirical effect sizes	vector with 2 values: c(yMin, yMax)	The min and max values for the y-axis. Setting explicit values could be better than relying on the automatically determined range, for example, to ensure identical y-axis across a larger set of plots.
mod.type	"cont"	"cont" or "cat"	The type of moderator.
no.mod.cats	NULL	value > 0	Need to be specified if type = "cat". The number of categories should usually be equal the number of dummy variables used to represent the categorical moderator + 1.
n.x.labels	NULL	value > 0	How many values to be used for indicating time points on the x-axis (0 is automatically added and should not be counted).
plot	TRUE	FALSE/TRUE	Plots figures if set to TRUE (default) otherwise only return moderated drift matrices.
plot.xMin	0	value ≧ 0	Smallest x value (time interval) to plot.
plot.xMax	NULL	value > 0	Largest x value (time interval) to plot.
plot.yMin	-1	Value	Smallest y value (cross effect) to plot.
plot.yMax	1	Value	Largest y value (cross effect) to plot.
plot..type	"l"	Any letter that can be used to represent the type of plot in R	Two dots (..) should be used. Points, lines, both etc.
plot.lty	1	Any letter that can be used to represent the type of lines in R	solid, dotted, dashed etc. lines.

Table continues ...

Argument	Default	Possible Values	Explanation
`ctmaPlotCtsemMod (-)`			*... continued*
`plot.col`	`"grey"`	Any color names or color number that can be used to represent the type of lines in R	Line color.
`plot.lwd`	`1.5`	value > 0	Line width.
`dot.plot.type`	`"b"`	Any letter that can be used to represent the type of plot in R	Sets the type of symbol used to show the moderator values along their trajectories.
`dot.plot.col`	`"black"`	Any color names or color number that can be used to represent the type of lines in R	Sets the color of the symbol used to show the moderator values.
`dot.plot.lwd`	`.5`	value > 0	Sets the size of the symbol used to show the moderator values.
`dot.plot.lty`	`3`	Any letter that can be used to represent the type of lines in R	solid, dotted, dashed etc. lines.
`dot.plot.pch`	`16`	Any integer that can be used to represent the type of lines in R	Sets the shape of the symbol used to show the moderator values.
`dot.plot.cex`	`3`	value > 0	Magnifier for the symbol used to show the moderator values.

Argument	Default	Possible Values	Explanation
`ctmaPower (EPIC-BiG-`**`Power`**`)`			Performs analysis of post hoc statistical power and required sample sizes to achieve a desired level of statistical power.
`ctmaInitFit`	`NULL`	CoTiMA fit-object	Object to which all single ctsem fits of primary studies has been assigned to (i.e., what has been returned by `ctmaInit`).
`activeDirectory`	`NULL`	path to directory	Specifies the directory where required files are found and saved. Should end with `"/"`. For example, `"/Users/GDC/CoTiMA/"`.
`statisticalPower`	`c()`	vector	Vector of requested statistical power values, e.g. c(.95, .80).
`failSafeN`	`NULL`	value > 0	A sample size used to determine across which time intervals expected effects become not significant. If not specified, the average sample size of primary studies will be used.
`failSafeP`	`NULL`	value between 0 and 1	A *p*-value used to determine across which time intervals expected effects become not significant. If not specified .01 will be used.
`timeRange`	`c(0, 1.5*maxDelta, 1)`	vector with 3 values	Specifies the time range across which statistical power etc. will be computed. A vector with 3 values: starting point, end point, step width, e.g., `c(0, 50, 1)`. When `timeRange` is not specified, `maxDelta` is determined by taking the largest delta found in the primary studies.
`useMBESS`	`FALSE`	`FALSE/TRUE`	If set to `TRUE`, the `MBESS` package is used to calculate statistical power (slower). Otherwise, use the internal CoTiMA function (faster).
`coresToUse`	`1`	value > 0 or < 0	The number of cores (threads) to be used for fitting. If a negative value is provided, the value is subtracted from available cores, else the value sets the number of cores to be used.
`digits`	`4`	value > 0	Rounding used in output.
`indVarying`	`FALSE`	`FALSE/TRUE`	Specifies a random (manifest) intercept model. Works only if all primary studies have 3 or more waves and no missing values (i.e., variables) exist.
`activateRPB`	`FALSE`	`FALSE/TRUE`	Messages (warning, finished fitting) could be sent to mobile phone if set to `TRUE`.
`silentOverwrite`	`FALSE`	`FALSE/TRUE`	Whether or not to prompt user preventing undesired overwriting of existing fit files (requested via `saveAllInvFit` or `saveAllInvWOSingFit`).
`loadAllInvFit`	`c()`	character string	Load the fit of a CoTiMA model with all effects invariant across primary studies, e.g., `loadAllInvFit = c("myAllInvariantModel")`.
`saveAllInvFit`	`c()`	character string	Save the fit of a CoTiMA model with all effects invariant across primary studies, e.g., `saveAllInvFit = c("myAllInvariantModel")`.
`loadAllInvWOSingFit`	`c()`	character string	not yet operational.
`saveAllInvWOSingFit`	`c()`	character string	not yet operational.
`skipScaling`	`TRUE`	`FALSE/TRUE`	If set to `FALSE`, combined raw data are standardized again. Although pseudo raw data for each primary study have variance = 1.0, this is not the case if they are combined into the single data set that is used to compute the model with all effects being invariant. This is because variance is computed with denominator *N* - 1. Could be corrected by setting `skipScaling = FALSE`, but usually has little practical consequences.
`useSampleFraction`	`NULL`	value between 0 and 100	Analyze only a fraction of the overall sample. Could help speeding up debugging. Provided as percent (e.g., `useSampleFraction = 30` uses 30% of the overall sample size).
`optimize`	`TRUE`	`FALSE/TRUE`	The optimize argument is passed to `ctStanFit`. If set to `FALSE`, Bayesian estimations is used. The chosen sampler is conditional on the priors argument. Note that this works differently than the `optimise` argument of `ctStanFit`.
`nopriors`	`TRUE`	`FALSE/TRUE`	Deprecated.

Table continues ...

Argument	Default	Possible Values	Explanation
ctmaPower (EPIC-BiG-**Power**)			*... continued*
priors	FALSE	FALSE/TRUE	Replaces previously used and now deprecated nopriors argument. Consequences of TRUE or FALSE are conditional on the optimize argument. optimize = TRUE & priors = FALSE implies maximum likelihood estimation, optimize = TRUE & priors = TRUE implies maximum a posteriori estimation, optimize = FALSE & priors = TRUE implies Bayesian estimation using NUTS (No U-Turn Sampler).
finishsamples	NULL (= 1000)	values > 0	The finishsamples argument is passed to ctStanFit. It specifies the number of samples to draw for final results computation. Larger (e.g., 10.000) values make results more exactly replicable. Larger values are recommended before manuscripts are submitted. Very large values (e.g., 100.000) might be helpful if very small effects (e.g., 0.0002) result from estimation.
iter	NULL (= 1000)	values > 0	The iter argument is passed to ctStanFit. It specifies the number of iterations used for Bayesian estimation, half of which will be devoted to warmup.
chains	NULL (= 2)	values > 0	The chains argument is passed to ctStanFit and defines the number of chains to be used for Bayesian estimation.
verbose	NULL	0, 1, or 2	The verbose argument is passed to ctStanFit. Higher values print more information during model fit.
customPar	FALSE	FALSE/TRUE	If set to TRUE some starting values usually used by ctStanFit will be used by CoTiMA specific settings. Not recommended to be used in combination with Bayesian estimation. It was introduced to improve handling of large values used in delta_t*i*. Setting it to FALSE and use scaleTime instead could be a better alternative if estimation problems will nevertheless occur.
scaleTime	NULL		This argument can facilitate model convergence. It internally changes the time scale assigned to delta_t*i*. For example, scaleTime = 1/12 could change the time scale from months to years. It is usually recommended to avoid delta_ti larger than 6.

Argument	Default	Possible Values	Explanation
`ctmaPub (-)`			Augments `ctmaGetPub`. Returns NEPP (= the number of studies published by the authors of the primary studies supplied *until* the year when the primary study was published), NEPPRecency (like NEPP, but limited to the number of years before the publication as specified with the `recency` argument). It also returns the *Meaning of NEPP* and *Meaning of NEPPRecency*, which explain what `"number"` exactly means (e.g., could be the mean of the sum of each author's publication, or the sum of the maximum publications per year of the authors).
`getPubObj`	`NULL`	Object created with `ctmaPubGet`	Publication (and citation) information of authors.
`primaryStudyList`	`NULL`	list of primary studies	A list of primary studies compiled with `ctmaPrep` that contains a subset of studies included in `ctmaInit`. Useful to exclude studies without the need to use `ctmaInit` again.
`yearsToExclude`	`0`	(vector of) years to exclude	Years to be excluded from computations. For example, the current year might be excluded because publication information might not be very reliable. Early years (e.g., 1900-1960) might be excluded because they would cause invalid publications (sometimes this happens in Google Scholar).
`targetYear`	`NULL` (= publication year)	value > 0	If left `NULL`, all publications before the year of the authors' publication count.
`recency`	`5`	value > 0	`ctmaPub` computes two indices. For the first one (NEPP), all years before `targetYear` count. For the second one (NEPPrecency), the years between `targetYear` and `targetYear - recency` count.
`indFUN`	`"sum"`	any of: `"mean"`, `"sum"`, `"max"`, `"min"`, `"var"`	Specifies the function used to aggregate an individual author's publication numbers, e.g., `"sum"` (recommended) computes the sum of an author's publication before `targetYear`, and `"var"` computes the variance of the number of publications for an author's first year of publication to `targetYear`.
`colFUN`	`"mean"`	any of: `"mean"`, `"sum"`, `"max"`, `"min"`, `"var"`	Specifies the function used to aggregate a group of authors (collective) publication numbers, e.g., `"mean"` computes the mean of all authors' publication scores (created with `indFUN`, e.g., the sums) before `targetYear`, and `"max"` takes largest of all authors' publication scores (created with `indFUN`, e.g., the sums) to `targetYear`.
`addAsMod`	`FALSE`	`FALSE/TRUE`	Currently disabled.

Argument	Default	Possible Values	Explanation
ctmaPrep (E**P**IC-BiG-Power)			Combines information of primary studies into a list-object and returns this list.
selectedStudies	NULL	vector with integers	Vector of primary study numbers (numeric values with no leading 0; e.g., "2" but not "02").
excludedElements	NULL	vector with integers	Could be used to exclude some predefined objects from the results reported. Note that some predefined objects are strongly defined; they have to be used in a special way because they are actually used in subsequent analyses. Some other objects could be used at the researcher's convenience (information is just collected). Strongly predefined objects are delta_t (should be of the type c(NA, NA) in cases when raw data are provided, with the number of NAs corresponding to the number of time intervals), sampleSize (single number), pairwiseN (matrix of pairwise N; could be used if correlation matrix is based on pairwise N), empcov (correlation matrix), moderator (vector of numbers; could be continuous or categorical), alphas (vector of reliability estimates of the variables of a primary study), startValues (vector of start values), rawData (information about file name and structure of raw data), empMeans (means for variables; usually 0), and empVars (variances for variables; usually 1.0). Weakly predefined objects are studyNumber (intended as a special number used for the outputs of subsequently fitted Co-TiMA models), source (intended as vector of authors' names and publication year), ageM value intended for indicating the mean age of participants in a primary study, malePercent (intended as value indicating the percentage of male participants in a primary study), occupation (intended as vector of character strings representing the occupations of participants in a primary study), country (intended as single character string representing the country in which a primary study was conducted), and targetVariables (intended as vector of character strings representing information about the variables used).
activeDirectory	NULL	path to directory	Specifies the directory where required files are found and saved. Should end with "/". For example, "/Users/GDC/Co-TiMA/".
addElements	NULL	vector of character strings	Could be used to add user-defined objects that are handled as the weakly predefined objects. The major purpose is to collect information a researcher regards as important, e.g., c("Important", "Interesting").
digits	4	value ≥ 0	Rounding used in output.
moderatorLabels	NULL	vector of character strings	Vector of names used to label moderators in the output e.g., c("Mod1", "Control").
moderatorValues	NULL	list of vectors	List of vector of names (assignments) used to label moderators in the output e.g., list(c("1 = Emotional Exhaustion", "2 = Exhaustion"), "continuous").
newRawDataDirectory	NULL	path to directory	Change paths for all raw data files. The original directory is included in the rawDatai objects (as part of the file name), which could be replaced by newRawDataDirectory.
summary	TRUE	FALSE/TRUE	Requests summary table and xlsx workbook in return object. Could be set to FALSE to avoid reporting errors.
ctmaPrepObject	NULL	Object created with ctmaPrep	Object previously created with ctmaPrep, from which studies should be excluded. Only works in combination with the argument excludedStudies.
excludedStudies	NULL	vector of integers	Studies to be excluded from a previously created ctmaPrep object.

Argument	Default	Possible Values	Explanation

ctmaRedHet (EPIC-**BiG**-Power)			Computes the reduction in heterogeneity of drift effects after moderators are added. The function takes two CoTiMA fit-objects as arguments. Both models should mimic the `ctmaInit` fit function by using `ctmaFit` with the argument `WEC = TRUE`. The first model should yield exactly the same fit (*-2ll*) as the `ctmaInit` fit because it is algebraically identical (all drift effects are moderated by dummy variables representing the primary studies). Estimation problems may occur, however (see explanation of the `WEC` argument of `ctmaFit`). The second model includes additional moderator variables. Mimicking `ctmaInit` by using `ctmaFit` is necessary because moderators are study-level variables that cannot be modelled if each primary study is fitted separately as in `ctmaInit`. For example,
			`fit1 <- ctmaFit(ctmaInitFit = fitObject, WEC = TRUE))`
			and
			`fit2 <- ctmaFit(ctmaInitFit = fitObject, WEC = TRUE, mod.numer = 1, mod.type = "cont")`
			and then
			`results <- ctmaRedHet(ctmaFitObject = fit1, ctmaFitObjectMod = fit2)`
			`summary(results)`
			Interpreting (reduction in) heterogeneity of single continuous time drift coefficients could be challenged because they operate *in concert*. This is different for discrete time drift coefficients, and the argument `dt` could be used to request (reduction of) heterogeneity across a set of different time intervals.

`activateRPB`	`FALSE`	`FALSE/TRUE`	Messages (warning, finished fitting) could be sent to mobile phone if set to `TRUE`.
`activeDirectory`	`NULL`	character string	Specifies the directory where required files are found and saved. Should end with `"/"`. For example, `"/Users/GDC/CoTiMA/"`.
`ctmaFitObject`	`NULL`	CoTiMA fit-object	A CoTiMA fit-object created with `ctmaFit` using the argument `WEC = TRUE`.
`ctmaFitObjectMod`	`NULL`	CoTiMA fit-object	A CoTiMA fit-object created with `ctmaFit` using the argument `WEC = TRUE` and at least one moderating effect.
`digits`	`4`	value > 0	Rounding used in output.
`dt`	`NULL`	vector of values > 0	Could be used to request (reduction of) heterogeneity across a set of different time intervals.
`undoTimeScaling`	`TRUE`	`FALSE/TRUE` or any value	Undoes possible time scaling achieved in `ctmaFit` by setting the argument `scaleTime`. When a number is provided instead of a logical argument, the number is used to multiply the obtained effects (e.g., when time is scaled in days in a study, and `scaleTime = 30.5` is used, the returned effect corresponds to 1-month intervals).

Argument	Default	Possible Values	Explanation
ctmaShapeRawData (E**P**IC-BiG-Power)			Transform raw data into the form required by ctsem or CoTiMA.
dataFrame	NULL	R data frame	An (typically pre-processed) R object containing data. For example, using the *foreign* R-package a SPSS data file can be imported using: "tmpData <- data.frame(read.spss(X.sav"), use.value.labels = FALSE))", by which time stamps are translated into number of seconds since October 14, 1582 (i.e., the internal SPSS format). The time scale can be change with the argument scaleTime (see below).
id	NULL	character string	The identifier (i.e., variable label) of subjects if data are in long format.
inputDataFrame Format	NULL	"wide" or "long"	Specifies if the dataFrame object contains data in wide or long format.
inputTimeFormat	"time"	"time" or "delta"	Whether time points ("time") or time intervals ("delta") are included in the dataFrame object.
missingValues	NA	single value	Value that indicates missing values in the dataFrame object.
n.manifest	NULL	single value > 0	Number of process variables in the dataFrame object (i.e., excluding possible moderators (for CoTiMA) or time dependent (TDpreds) and time independent (TIpreds; for ctsem) per time point. Possibly 2 for CoTiMA in most instances (i.e., a bivariate model).
manifest.per. latent	NULL		The number of manifest variables per latent factor. In most instances there probably is only 1 manifest per latent, but e.g. c(2,3,1) also possible for 6 manifest variables loading on 3 latent factors.
Tpoints	NULL	single value > 0	Number of time points in the dataFrame object.
allInput VariablesNames	NULL	vector of character strings	Vector of all process variable names, time dependent predictor names, time independent predictor names, and names of times/deltas. Only required if the dataFrame does not have column names. Used to identify the variables that should be selected later.
orderInput VariablesNames	NULL	"names" or "time"	When "names" is specified, the process variables are expected to be in the order X1, X2, X3, Y1, Y2, X3 etc. When "time" is specified, the expected order is X1, Y1, X2, Y2, etc. For ctsem/CoTiMA, the output file will order them by time.
targetInput VariablesNames	NULL	vector of character strings	The process variables in the dataFrame that should be used (in "names" or in "times" order as specified with the argument orderInputVariablesNames). This is used to delete variables from the data frame that are not required.
targetInput TDpredNames	NULL	vector of character strings	Not important for CoTiMA, but perhaps for fitting ctsem models. The vector of character strings should contain the actual TDpreds labels, e.g., 3, or 6, or 9, ... names if Tpoints = 3. Each of the 3, 6, etc. represents one TDpred. One typically does not have TD predictors in a CoTiMA.
targetInp TIpredNames	NULL	vector of character strings	Time independent (TI) predictor names in dataFrame. One typically does not have TI predictors in CoTiMA except it uses raw data, where TIpreds are available for individual cases. (in case data are prepared for CoTiMA, TIpreds could be present as moderator variables, which have to be specified using ctmaPrep) , In case data are prepared for ctsem, having TIpred is feasible, and the vector of character strings should contain the actual TIpred labels in the dataFrame object.
targetTime VariablesNames	NULL	vector of character strings	The labels of the time variables and time variables in dataFrame that should be used, e.g., c("time2", "time4") or c("dT0", "dT1").

Table continues ...

Argument	Default	Possible Values	Explanation
ctmaShapeRawData (E**P**IC-BiG-Power)			*... continued*
outputData FrameFormat	"long"	"long" or "wide"	The output format of the returned R object. Should be usually "wide" for CoTiMA and "long" for ctsem.
output VariablesNames	"Y"	vector of character strings	The default value "Y" will result in variable labels Y1_T0, Y2_T0, Y3_T0 etc. with numbers representing the n.manifest variables. When, e.g., n.manifest = 3 one could also specify, e.g., c("X", "Y", "Z"), which will result in Y_T0, X_T0, Z_T0 etc.
output TDpredNames	NULL	vector of character strings	Not really important for CoTiMA, but perhaps for fitting ctsem models. The default value "TD" will result in TDpred labels TD1_T0, TD2_T0, TD3_T0 etc. with numbers representing the number of TDpred. When, e.g., three TD labels per time point are specified with the argument targetInputTDpredNames, one could also specify, e.g., c("A", "B", "C"), which will result in A_T0, B_T0, C_T0 etc.
output TIpredNames	NULL	vector of character strings	Not really important for CoTiMA, but perhaps for fitting ctsem models. The default value "TI" will result in TIpred labels TI1, TI2, TI3 etc. with numbers representing the number of TDpred.
outputTime VariablesNames	"time"	character string	Not really important for CoTiMA, but perhaps for fitting ctsem models. The default value "time" will result in time variables labeled time0, time1, time2 etc.
outputTimeFormat	"time"	"time" or "delta"	Whether time is stored in absolute time or deltas (time intervals). Note the CoTiMA requires "delta" (and "wide" format), whereas ctsem requires "time" (and "long" format).
scaleTime	1	any positive value	Scales time in the returned data frame by a scalar that is used to multiply the time variable. Typically used to rescale primary study time to the time scale use in other primary studies. For example, scaleTime = 1/(60 x 60 x 24 x 365.25) rescales time provided in seconds (frequent case when imported from SPSS) into years (60sec x 60min x 24hrs x 365.25 days incl. leap years).
minInterval	0.0001	single value > .00001	Set to smaller values than any possible observed measurement interval, but larger than 0.0001. The value is used for indicating unavailable time interval information (caused by missing values) because NA is technically not possible for time intervals.
minTolDelta	NULL	single value > 0	The shortest time interval to be tolerated. Could be useful to eliminate invalid data, e.g., because primary researchers coded time wrongly or participants filled in invalid values to time questions or did not adhere to the research protocol. For example, assuming time is coded in months in a study that was supposed to have approximate 12-month (1-year) intervals, a value of 6 would delete values at a time point that was closer than 6 months to the preceding time point. Note that minTolDelta applies to the time intervals after the scaleTime argument has applied (i.e., scaleTime may need adaptation for each primary study, but minTolDelta does not).
maxTolDelta	NULL	single value > 0	The longest time interval to be tolerated. Could be useful to eliminate invalid data, e.g., because primary researchers coded time wrongly or participants filled in invalid values to time questions or did not adhere to the research protocol. For example, assuming time is coded in months in a study that was supposed to sample 6 times within a 6-month (1/2-year) time frame, a value of 6 would delete values at all (!!) time points that were farer away from any other one than 6 months. Note that maxTolDelta applies to the time intervals after the scaleTime argument has applied (i.e., scaleTime may need adaptation for each primary study, but maxTolDelta does not).

Table continues ...

Argument	Default	Possible Values	Explanation
`ctmaShapeRawData (EPIC-BiG-Power)`			*... continued*
`negTolDelta`	`FALSE`	`FALSE/TRUE`	When the default setting (`FALSE`) is used, cases that have at least one negative delta are excluded (may indicate *unreliable responding*). Use the argument `minTolDelta` to delete certain variables only.
`min.val.n.Vars`	`1`	any integer value between 0 and number of manifest variables	Specifies the minimum no. of valid variables (i.e., non-NA) that has to be available per participant. Default = 1 (retains cases with only 1 valid variable), 0 would retain cases will all variables missing (not very useful). Retaining participants who provide a single valid variable is technically possible, but these participants contribute to the estimation of the variance/mean of this variable only. Since variance/mean are 1/0 in most CoTiMA applications, this is not very informative but at the cost of additional computational burden. Setting `min.val.n.Vars = 2` is recommended.
`min.val.Tpoints`	`1`	any integer value between 1 and number of available time points	Minimum no. of valid time points (i.e. time points where `min.val.n.Vars` is met). Default = 1 retains participants with full set of valid variables at least at one single time point (which will become T0). Setting `min.val.Tpoints = 2` or higher values retains participants which provide longitudinal information. Since T0 covariances are usually not too interesting, `min.val.Tpoints = 2` may be more reasonable than the default = 1.
`standardization`	`"none"`	Any of `"none"`, `"withinTimeA"`, `"withinTimeB"`, `"withinColumn"`, `"withinPerson"`, or `"overall"`	Different ways to standardize raw data are possible (`"withinTimeA"` standardizes within time points and deletes cases with missing T0 data. `"withinTimeB"` standardizes within time points and does not delete cases, and in subsequent ctsem or CoTiMA applications the user is advised to use the argument `sameInitialTimes = TRUE`. This argument probably handles raw data in the way primary studies do when computing pairwise correlation matrices. `"withinColumn"` standardizes variables after arranging them in wide format. Since time points without valid data are eliminated for each case (i.e., values are moved from later to earlier time points), this makes the difference to `"withinTimeA"` and `"withinTimeB"`. `"withinPerson"` standardizes all variables within person. `"overall"` standardizes all variables across all persons.

Argument	Default	Possible Values	Explanation
`ctmaSV (EPIC-BiG-Power)`			Computes new start values and returns an augmented list of primary studies that has *inits* elements containing these start values. This list can then be used for the `primaryStudies` argument in subsequent `ctmaInit` applications. Starting values are obtained by using lavaan to fit discrete time SEM to the primary studies provided. The discrete time estimates are then transformed into their continuous time counterparts and some specific transformations are applied (required by ctsem) before returned. In case of models with 3 or more waves of data, continuous time effects are computed for each interval and then averaged.
`ctmaInitFit`	`NULL`	CoTiMA fit-object	CoTiMA fit-object created with `ctmaInit`.
`activeDirectory`	`NULL`	path to directory	Specifies the directory where required files are found and saved. Should end with `"/"`. For example, `"/Users/GDC/CoTiMA/"`.
`coresToUse`	`1`	value > 0 or < 0	The number of cores (threads) to be used for fitting. If a negative value is provided, the value is subtracted from available cores, else the value sets the number of cores to be used.
`primaryStudies`	`NULL`	CoTiMA fit-object created with `ctmaPrep`	In cases in which the CoTiMA fit-object assigned to `ctmaInitFit` (possibly old fit files) does not contain the `primaryStudies` object created with `ctmaPrep` it could be added by assigning it to the `primaryStudies` argument.
`replaceSV`	`TRUE`	`FALSE/TRUE`	The computed starting values could either replace exiting starting values in the returned list of primary studies or save them as an additional list element `inits`.

The plot function, which is described next, works slightly different than other CoTiMA functions. Like all other CoTiMA functions, some arguments could be used as always. However, in addition, it is important to note that several plotting parameters (`"fitAddSpecs"`) have to be assigned to the CoTiMA fit-object before plotting it, rather than providing plotting parameters as arguments to the plot function (e.g., `CoTiMAInitFitObject$xMax <- 200`). This is because the arguments have different effects conditional on the type of fit-object. The number of plotting parameters that can be changed in this way is still limited; we are working on extensions. Further, user-defined plotting parameters differ for fit-objects created with `ctmaBiG` versus `ctmaInit` and `ctmaFit`). Finally, if problems with plot are encountered, we recommend trying `ctmaPlot` instead of `plot`.

Argument	Default	Possible Values	Explanation
plot/ctmaPlot			Generates plots.
ctmaFitObject	NULL	CoTiMA fit object	A CoTiMA fit-object created by ctmaInit, ctmaFit, or ctmaBiG.
activeDirectory	NULL	path to directory	Specifies the directory where required files are found and saved. Should end with "/". For example, "/Users/GDC/CoTiMA/".
saveFilePrefix	"ctmaPlot"	vector of character strings	Labels for the generated plot, which might be automatically augmented by further information (e.g., "ctmaplot V1toV1.png").
activateRPB	FALSE	FALSE/TRUE	Messages (warning, finished fitting) could be sent to mobile phone if set to TRUE.
plotCrossEffects	TRUE	FALSE/TRUE	Affects plotting of ctmaInit or ctmaFit fit-objects only. Plotting of discrete time cross effects can be suspended.
plotAutoEffects	TRUE	FALSE/TRUE	Affects plotting of ctmaInit or ctmaFit fit-objects only. Plotting of discrete time auto effects can be suspended.
timeUnit	"timeUnit (not specified)"	vector of character strings	Affects plotting of ctmaInit or ctmaFit fit-objects only. Label used for the x-axis of discrete time plots.
timeRange	c() = 0 to 1.5 times the longest interval used in primary studies	vector with 3 values: c(xMin, xMax, stepwidth)	Affects plotting of ctmaInit or ctmaFit fit-objects only. The range across which discrete time effects are plotted, e.g., c(10, 20, .01) would plot effects from 10 units of time to 20 using steps of .01. Note that a stepwidth < 1 could be specified to obtain more fine-grained figures.
yLimits ForEffects	values slightly exceeding min and max empirical effect sizes	vector with 2 values: c(yMin, yMax)	Affects plotting of ctmaInit or ctmaFit fit-objects only. The min and max values for the y-axis. Setting explicit values could be better than relying on the automatically determined range, for example, to ensure identical y-axis across a larger set of plots.
mod.values	-2:2	vector with numbers	Affects plotting of ctmaFit fit-objects only. The moderator values for which plots of continuous moderators should be generated. Corresponds to the standard deviations below and above the mean value if the continuous moderator was standardized with scaleMod = TRUE. Does not affect plotting of categorical moderators.
mod.number	1	value > 0	The number of the moderator effect that is plotted if more than a single moderator is included in the fit-object created with ctmaFit. Note that mod.number does not select the dummies used for categorical moderators (all dummy effects are plotted); rather it refers to the 1st, 2nd etc. continuous or categorical moderator of an analysis.
aggregateLabel	"" (= nothing)	character(s)	Affects plotting of ctmaFit fit-objects only. Symbol to be attached to the discrete time plot of a ctmaFit fit-object. In the case of ctmaInit fit-objects, each study is usually identified in the plot with a dot inside which the study number is shown. In the case of a ctmaFit fit-object with aggregated effects, one could use a symbol, e.g., aggregateLabel = "Σ".
xLabels	NULL	vector with numbers	Affects plotting of ctmaFit fit-objects only. The numbers indicating the time intervals on the x-axis are usually determined automatically. They could also be directly defined, and the values provided are equally distributed across the time range used to plot the discrete time effects, e.g., c(1, 3, 5, 7, 9).
undoTimeScaling	TRUE	FALSE/TRUE	Undoes possible time scaling achieve in ctmaInit or ctmaFit by setting the argument scaleTime.

In the following tables, named list elements rather than function arguments are shown. These list elements have to be explicitly assigned to fit-objects to change the output of the plot function.

Named List Element	Default	Possible Values	Explanation
`fitAddSpecs for ctmaBias fit-objects`			
`CoTiMAFit$xMin`	`0`	value ≥ 0	Internally, the x-axis ranges from 0 to 300 (the values shown in the plot are irrelevant). Setting xMin > 0 creates a plot where the left part is left out. For example, if one wants to leave out the first quarter (i.e. 0 to 300/4 = 0 to 75) you could set `CoTiMAFit$xMin <- 75`. If one wants extra space on the right-hand side of the plot, one could lift xMax to values larger than 300, e.g., `CoTiMAFit$xMax <- 400`.
`CoTiMAFit$xMax`	`300`	value ≥ 0	See above.

Named List Element	Default	Possible Values	Explanation
`fitAddSpecs for ctmaInit & ctmaFit fit-objects`			
`CoTiMAFit$col`	`"grey"` & `"black"` for `ctmaInit` & `ctmaFit` fit-objects, respectively.	R-type color code	Defines the color of the curve showing the discrete time affects across time, e.g., `CoTiMAFit$col <- "red"`
`CoTiMAFit$lwd`	`1.5` & `2.5` for `ctmaInit` & `ctmaFit` fit-objects, respectively.	value ≥ 0	Defines the line width of the curve showing the discrete time affects across time, e.g., `CoTiMAFit$lwd <- 4`
`CoTiMAFit$lty`	`1` & `2` for `ctmaInit` & `ctmaFit` fit-objects, respectively.	R-type integer value ≥ 0	Defines the line type of the curve showing the discrete time affects across time, e.g., `CoTiMAFit$lty <- 1`, with, e.g., 1 = solid, 2 = dashed, and 3 = dotted.
`CoTiMAFit$xMin`	min of `time Range`	integer values	Limits the displayed range of the discrete time effects plotted. Should only be used in combination with the `xLabels` argument. Overrides other setting if multiple fit-objects are supplied. This is still experimental; it is recommended to set `timeRange` instead.
`CoTiMAFit$xMax`	max of `time Range`	integer values	Limits the displayed range of the discrete time effects plotted. Should only be used in combination with the `xLabels` argument. Overrides other setting if multiple fit-objects are supplied. This is still experimental; it is recommended to set `timeRange` instead.
`CoTiMA Fit$dot.type`	`"b"`	R-type characters	Type of the plot. Use `"p"` for points, `"l"` for lines, `"b"` for both, `"c"` for the lines part alone of `"b"` etc.
... to be continued			