Package ‘manCTMed’

December 15, 2025

Title Continuous Time Mediation

Version 1.0.9

Description Research compendium for the manuscript
Pesigan, I. J. A., Russell, M. A., & Chow, S.-M. (2025).
Inferences and Effect Sizes for Direct, Indirect, and Total Effects
in Continuous-Time Mediation Models.
Psychological Methods.
<doi:10.1037/met0000779>.

URL `https://github.com/jeksterslab/manCTMed`,
`https://jeksterslab.github.io/manCTMed/`, `https://osf.io/qwnmf/`,
`https://doi.org/10.1037/met0000779`

BugReports `https://github.com/jeksterslab/manCTMed/issues`

License MIT + file LICENSE

Encoding UTF-8

LazyData true

LazyDataCompression xz

Roxygen list(markdown = TRUE)

Depends R (>= 4.0.0)

Imports stats, dynr, OpenMx, dynUtils, simStateSpace (== 1.2.9),
bootStateSpace (== 1.0.2), cTMed (== 1.0.6), ggplot2

Suggests knitr, rmarkdown, testthat, DT

Remotes jeksterslab/dynUtils@dc3f47b

RoxygenNote 7.3.3.9000

NeedsCompilation no

Author Ivan Jacob Agaloos Pesigan [aut, cre, cph] (ORCID:
<`https://orcid.org/0000-0003-4818-8420`>),
Michael A. Russell [ctb] (ORCID:
<`https://orcid.org/0000-0002-3956-604X`>),
Sy-Miin Chow [ctb] (ORCID: <`https://orcid.org/0000-0003-1938-027X`>)

Maintainer Ivan Jacob Agaloos Pesigan <`r.jeksterslab@gmail.com`>

Contents

`BootPara` *Parametric Bootstrap*

Description

The function generates simulated datasets based on a fitted model and refits the model to each generated dataset using the `dynr` package.

Usage

```
BootPara(
  fit,
  path,
  prefix,
  taskid,
  B = 1000L,
  ncores = NULL,
  seed = NULL,
  clean = TRUE
)
```

Arguments

`fit`	R object. Output of the `FitDynr()`, `FitMx()`, `IllustrationFitDynr()`, or `IllustrationFitMx()`, functions.
`path`	Path to a directory to store bootstrap samples and estimates.
`prefix`	Character string. Prefix used for the file names for the bootstrap samples and estimates.
`taskid`	Positive integer. Task ID.
`B`	Positive integer. Number of bootstrap samples.
`ncores`	Positive integer. Number of cores to use.
`seed`	Integer. Random seed.
`clean`	Logical. If `clean = TRUE`, delete intermediate files generated by the function.

See Also

Other Confidence Interval Functions: BootParaStdXMY(), BootParaStdXYM(), BootParaStdYMX(), BootParaXMY(), BootParaXYM(), BootParaYMX(), DeltaStdXMY(), DeltaStdXYM(), DeltaStdYMX(), DeltaXMY(), DeltaXYM(), DeltaYMX(), IllustrationBootPara(), MCStdXMY(), MCStdXYM(), MCStdYMX(), MCXMY(), MCXYM(), MCYMX()

Examples

```
## Not run:
set.seed(42)
library(dynr)
sim <- GenData(taskid = 1)
data <- RandomMeasurement(sim)
fit <- FitDynr(data, taskid = 1)
BootPara(
  fit = fit,
  path = getwd(),
  prefix = "pb",
  taskid = 1,
  B = 1000L
)

## End(Not run)
```

BootParaStdXMY *Parametric Bootstrap Confidence Intervals for X-M-Y (Standardized)*

Description

The function generates parametric bootstrap method confidence intervals for the mediation model $X \to M \to Y$ (Standardized).

Usage

```
BootParaStdXMY(boot, theta_hat, delta_t = 1:30, ncores = NULL)
```

Arguments

boot	R object. Output of the BootPara() function.
theta_hat	R object. Output of the ThetaHat() function.
delta_t	Numeric vector. Vector of time intervals.
ncores	Positive integer. Number of cores to use.

See Also

Other Confidence Interval Functions: BootPara(), BootParaStdXYM(), BootParaStdYMX(), BootParaXMY(), BootParaXYM(), BootParaYMX(), DeltaStdXMY(), DeltaStdXYM(), DeltaStdYMX(), DeltaXMY(), DeltaXYM(), DeltaYMX(), IllustrationBootPara(), MCStdXMY(), MCStdXYM(), MCStdYMX(), MCXMY(), MCXYM(), MCYMX()

Examples

```
## Not run:
set.seed(42)
library(dynr)
sim <- GenData(taskid = 1)
data <- RandomMeasurement(sim)
fit <- FitDynr(data, taskid = 1)
boot <- BootPara(
  fit = fit,
  path = getwd(),
  prefix = "pb",
  taskid = 1,
  B = 1000L
)
theta_hat <- ThetaHat(fit)
ci <- BootParaStdXMY(boot = boot, theta_hat = theta_hat)
plot(ci)
plot(ci, type = "bc")

## End(Not run)
```

BootParaStdXYM — *Parametric Bootstrap Confidence Intervals for X-Y-M (Standardized)*

Description

The function generates parametric bootstrap method confidence intervals for the mediation model $X \to Y \to M$ (Standardized).

Usage

```
BootParaStdXYM(boot, theta_hat, delta_t = 1:30, ncores = NULL)
```

Arguments

`boot`	R object. Output of the `BootPara()` function.
`theta_hat`	R object. Output of the `ThetaHat()` function.
`delta_t`	Numeric vector. Vector of time intervals.
`ncores`	Positive integer. Number of cores to use.

See Also

Other Confidence Interval Functions: `BootPara()`, `BootParaStdXMY()`, `BootParaStdYMX()`, `BootParaXMY()`, `BootParaXYM()`, `BootParaYMX()`, `DeltaStdXMY()`, `DeltaStdXYM()`, `DeltaStdYMX()`, `DeltaXMY()`, `DeltaXYM()`, `DeltaYMX()`, `IllustrationBootPara()`, `MCStdXMY()`, `MCStdXYM()`, `MCStdYMX()`, `MCXMY()`, `MCXYM()`, `MCYMX()`

Examples

```
## Not run:
set.seed(42)
library(dynr)
sim <- GenData(taskid = 1)
data <- RandomMeasurement(sim)
fit <- FitDynr(data, taskid = 1)
boot <- BootPara(
  fit = fit,
  path = getwd(),
  prefix = "pb",
  taskid = 1,
  B = 1000L
)
theta_hat <- ThetaHat(fit)
ci <- BootParaStdXYM(boot = boot, theta_hat = theta_hat)
plot(ci)
plot(ci, type = "bc")

## End(Not run)
```

BootParaStdYMX *Parametric Bootstrap Confidence Intervals for Y-M-X (Standardized)*

Description

The function generates parametric bootstrap method confidence intervals for the mediation model $Y \to M \to X$ (Standardized).

Usage

```
BootParaStdYMX(boot, theta_hat, delta_t = 1:30, ncores = NULL)
```

Arguments

`boot`	R object. Output of the `BootPara()` function.
`theta_hat`	R object. Output of the `ThetaHat()` function.
`delta_t`	Numeric vector. Vector of time intervals.
`ncores`	Positive integer. Number of cores to use.

See Also

Other Confidence Interval Functions: `BootPara()`, `BootParaStdXMY()`, `BootParaStdXYM()`, `BootParaXMY()`, `BootParaXYM()`, `BootParaYMX()`, `DeltaStdXMY()`, `DeltaStdXYM()`, `DeltaStdYMX()`, `DeltaXMY()`, `DeltaXYM()`, `DeltaYMX()`, `IllustrationBootPara()`, `MCStdXMY()`, `MCStdXYM()`, `MCStdYMX()`, `MCXMY()`, `MCXYM()`, `MCYMX()`

Examples

```
## Not run:
set.seed(42)
library(dynr)
sim <- GenData(taskid = 1)
data <- RandomMeasurement(sim)
fit <- FitDynr(data, taskid = 1)
boot <- BootPara(
  fit = fit,
  path = getwd(),
  prefix = "pb",
  taskid = 1,
  B = 1000L
)
theta_hat <- ThetaHat(fit)
ci <- BootParaStdYMX(boot = boot, theta_hat = theta_hat)
plot(ci)
plot(ci, type = "bc")

## End(Not run)
```

BootParaXMY — *Parametric Bootstrap Confidence Intervals for X-M-Y*

Description

The function generates parametric bootstrap method confidence intervals for the mediation model $X \to M \to Y$.

Usage

```
BootParaXMY(boot, phi_hat, delta_t = 1:30, ncores = NULL)
```

Arguments

`boot`	R object. Output of the `BootPara()` function.
`phi_hat`	R object. Output of the `PhiHat()` function.
`delta_t`	Numeric vector. Vector of time intervals.
`ncores`	Positive integer. Number of cores to use.

See Also

Other Confidence Interval Functions: `BootPara()`, `BootParaStdXMY()`, `BootParaStdXYM()`, `BootParaStdYMX()`, `BootParaXYM()`, `BootParaYMX()`, `DeltaStdXMY()`, `DeltaStdXYM()`, `DeltaStdYMX()`, `DeltaXMY()`, `DeltaXYM()`, `DeltaYMX()`, `IllustrationBootPara()`, `MCStdXMY()`, `MCStdXYM()`, `MCStdYMX()`, `MCXMY()`, `MCXYM()`, `MCYMX()`

Examples

```
## Not run:
set.seed(42)
library(dynr)
sim <- GenData(taskid = 1)
data <- RandomMeasurement(sim)
fit <- FitDynr(data, taskid = 1)
boot <- BootPara(
  fit = fit,
  path = getwd(),
  prefix = "pb",
  taskid = 1,
  B = 1000L
)
phi_hat <- PhiHat(fit)
ci <- BootParaXMY(boot = boot, phi_hat = phi_hat)
plot(ci)
plot(ci, type = "bc")

## End(Not run)
```

`BootParaXYM` *Parametric Bootstrap Confidence Intervals for X-Y-M*

Description

The function generates parametric bootstrap method confidence intervals for the mediation model $X \rightarrow Y \rightarrow M$.

Usage

```
BootParaXYM(boot, phi_hat, delta_t = 1:30, ncores = NULL)
```

Arguments

`boot`	R object. Output of the `BootPara()` function.
`phi_hat`	R object. Output of the `PhiHat()` function.
`delta_t`	Numeric vector. Vector of time intervals.
`ncores`	Positive integer. Number of cores to use.

See Also

Other Confidence Interval Functions: `BootPara()`, `BootParaStdXMY()`, `BootParaStdXYM()`, `BootParaStdYMX()`, `BootParaXMY()`, `BootParaYMX()`, `DeltaStdXMY()`, `DeltaStdXYM()`, `DeltaStdYMX()`, `DeltaXMY()`, `DeltaXYM()`, `DeltaYMX()`, `IllustrationBootPara()`, `MCStdXMY()`, `MCStdXYM()`, `MCStdYMX()`, `MCXMY()`, `MCXYM()`, `MCYMX()`

Examples

```
## Not run:
set.seed(42)
library(dynr)
sim <- GenData(taskid = 1)
data <- RandomMeasurement(sim)
fit <- FitDynr(data, taskid = 1)
boot <- BootPara(
  fit = fit,
  path = getwd(),
  prefix = "pb",
  taskid = 1,
  B = 1000L
)
phi_hat <- PhiHat(fit)
ci <- BootParaXYM(boot = boot, phi_hat = phi_hat)
plot(ci)
plot(ci, type = "bc")

## End(Not run)
```

BootParaYMX *Parametric Bootstrap Confidence Intervals for Y-M-X*

Description

The function generates parametric bootstrap method confidence intervals for the mediation model $Y \to M \to X$.

Usage

```
BootParaYMX(boot, phi_hat, delta_t = 1:30, ncores = NULL)
```

Arguments

boot	R object. Output of the BootPara() function.
phi_hat	R object. Output of the PhiHat() function.
delta_t	Numeric vector. Vector of time intervals.
ncores	Positive integer. Number of cores to use.

See Also

Other Confidence Interval Functions: BootPara(), BootParaStdXMY(), BootParaStdXYM(), BootParaStdYMX(), BootParaXMY(), BootParaXYM(), DeltaStdXMY(), DeltaStdXYM(), DeltaStdYMX(), DeltaXMY(), DeltaXYM(), DeltaYMX(), IllustrationBootPara(), MCStdXMY(), MCStdXYM(), MCStdYMX(), MCXMY(), MCXYM(), MCYMX()

Examples

```
## Not run:
set.seed(42)
library(dynr)
sim <- GenData(taskid = 1)
data <- RandomMeasurement(sim)
fit <- FitDynr(data, taskid = 1)
boot <- BootPara(
  fit = fit,
  path = getwd(),
  prefix = "pb",
  taskid = 1,
  B = 1000L
)
phi_hat <- PhiHat(fit)
ci <- BootParaYMX(boot = boot, phi_hat = phi_hat)
plot(ci)
plot(ci, type = "bc")

## End(Not run)
```

`Compress` *Compress Replication*

Description

Compress Replication

Usage

```
Compress(taskid, repid, output_folder)
```

Arguments

`taskid`	Positive integer. Task ID.
`repid`	Positive integer. Replication ID.
`output_folder`	Character string. Output folder.

Value

The output is saved as an external file in `output_folder`.

Author(s)

Ivan Jacob Agaloos Pesigan

See Also

Other Compression Functions: `IllustrationCompress()`

DeltaStdXMY *Delta Method Confidence Intervals for X-M-Y (Standardized)*

Description

The function generates delta method confidence intervals for the mediation model $X \to M \to Y$ (Standardized).

Usage

```
DeltaStdXMY(theta_hat, delta_t = 1:30)
```

Arguments

theta_hat	R object. Output of the ThetaHat() function.
delta_t	Numeric vector. Vector of time intervals.

See Also

Other Confidence Interval Functions: BootPara(), BootParaStdXMY(), BootParaStdXYM(), BootParaStdYMX(), BootParaXMY(), BootParaXYM(), BootParaYMX(), DeltaStdXYM(), DeltaStdYMX(), DeltaXMY(), DeltaXYM(), DeltaYMX(), IllustrationBootPara(), MCStdXMY(), MCStdXYM(), MCStdYMX(), MCXMY(), MCXYM(), MCYMX()

Examples

```
## Not run:
set.seed(42)
library(dynr)
sim <- GenData(taskid = 1)
data <- RandomMeasurement(sim)
fit <- FitDynr(data, taskid = 1)
theta_hat <- ThetaHat(fit)
ci <- DeltaStdXMY(theta_hat)
plot(ci)

## End(Not run)
```

DeltaStdXYM *Delta Method Confidence Intervals for X-Y-M (Standardized)*

Description

The function generates delta method confidence intervals for the mediation model $X \to Y \to M$ (Standardized).

Usage

```
DeltaStdXYM(theta_hat, delta_t = 1:30)
```

Arguments

`theta_hat`	R object. Output of the `ThetaHat()` function.
`delta_t`	Numeric vector. Vector of time intervals.

See Also

Other Confidence Interval Functions: `BootPara()`, `BootParaStdXMY()`, `BootParaStdXYM()`, `BootParaStdYMX()`, `BootParaXMY()`, `BootParaXYM()`, `BootParaYMX()`, `DeltaStdXMY()`, `DeltaStdYMX()`, `DeltaXMY()`, `DeltaXYM()`, `DeltaYMX()`, `IllustrationBootPara()`, `MCStdXMY()`, `MCStdXYM()`, `MCStdYMX()`, `MCXMY()`, `MCXYM()`, `MCYMX()`

Examples

```
## Not run:
set.seed(42)
library(dynr)
sim <- GenData(taskid = 1)
data <- RandomMeasurement(sim)
fit <- FitDynr(data, taskid = 1)
theta_hat <- ThetaHat(fit)
ci <- DeltaStdXYM(theta_hat)
plot(ci)

## End(Not run)
```

DeltaStdYMX   *Delta Method Confidence Intervals for Y-M-X (Standardized)*

Description

The function generates delta method confidence intervals for the mediation model $Y \to M \to X$ (Standardized).

Usage

```
DeltaStdYMX(theta_hat, delta_t = 1:30)
```

Arguments

`theta_hat`	R object. Output of the `ThetaHat()` function.
`delta_t`	Numeric vector. Vector of time intervals.

See Also

Other Confidence Interval Functions: BootPara(), BootParaStdXMY(), BootParaStdXYM(), BootParaStdYMX(), BootParaXMY(), BootParaXYM(), BootParaYMX(), DeltaStdXMY(), DeltaStdXYM(), DeltaXMY(), DeltaXYM(), DeltaYMX(), IllustrationBootPara(), MCStdXMY(), MCStdXYM(), MCStdYMX(), MCXMY(), MCXYM(), MCYMX()

Examples

```
## Not run:
set.seed(42)
library(dynr)
sim <- GenData(taskid = 1)
data <- RandomMeasurement(sim)
fit <- FitDynr(data, taskid = 1)
theta_hat <- ThetaHat(fit)
ci <- DeltaStdYMX(theta_hat)
plot(ci)

## End(Not run)
```

DeltaXMY *Delta Method Confidence Intervals for X-M-Y*

Description

The function generates delta method confidence intervals for the mediation model $X \to M \to Y$.

Usage

```
DeltaXMY(phi_hat, delta_t = 1:30)
```

Arguments

phi_hat	R object. Output of the PhiHat() function.
delta_t	Numeric vector. Vector of time intervals.

See Also

Other Confidence Interval Functions: BootPara(), BootParaStdXMY(), BootParaStdXYM(), BootParaStdYMX(), BootParaXMY(), BootParaXYM(), BootParaYMX(), DeltaStdXMY(), DeltaStdXYM(), DeltaStdYMX(), DeltaXYM(), DeltaYMX(), IllustrationBootPara(), MCStdXMY(), MCStdXYM(), MCStdYMX(), MCXMY(), MCXYM(), MCYMX()

Examples

```
## Not run:
set.seed(42)
library(dynr)
sim <- GenData(taskid = 1)
data <- RandomMeasurement(sim)
fit <- FitDynr(data, taskid = 1)
phi_hat <- PhiHat(fit)
ci <- DeltaXMY(phi_hat)
plot(ci)

## End(Not run)
```

`DeltaXYM` *Delta Method Confidence Intervals for X-Y-M*

Description

The function generates delta method confidence intervals for the mediation model $X \to Y \to M$.

Usage

```
DeltaXYM(phi_hat, delta_t = 1:30)
```

Arguments

`phi_hat`	R object. Output of the `PhiHat()` function.
`delta_t`	Numeric vector. Vector of time intervals.

See Also

Other Confidence Interval Functions: `BootPara()`, `BootParaStdXMY()`, `BootParaStdXYM()`, `BootParaStdYMX()`, `BootParaXMY()`, `BootParaXYM()`, `BootParaYMX()`, `DeltaStdXMY()`, `DeltaStdXYM()`, `DeltaStdYMX()`, `DeltaXMY()`, `DeltaYMX()`, `IllustrationBootPara()`, `MCStdXMY()`, `MCStdXYM()`, `MCStdYMX()`, `MCXMY()`, `MCXYM()`, `MCYMX()`

Examples

```
## Not run:
set.seed(42)
library(dynr)
sim <- GenData(taskid = 1)
data <- RandomMeasurement(sim)
fit <- FitDynr(data, taskid = 1)
phi_hat <- PhiHat(fit)
ci <- DeltaXYM(phi_hat)
plot(ci)

## End(Not run)
```

DeltaYMX *Delta Method Confidence Intervals for Y-M-X*

Description

The function generates delta method confidence intervals for the mediation model $Y \rightarrow M \rightarrow X$.

Usage

```
DeltaYMX(phi_hat, delta_t = 1:30)
```

Arguments

`phi_hat`	R object. Output of the `PhiHat()` function.
`delta_t`	Numeric vector. Vector of time intervals.

See Also

Other Confidence Interval Functions: `BootPara()`, `BootParaStdXMY()`, `BootParaStdXYM()`, `BootParaStdYMX()`, `BootParaXMY()`, `BootParaXYM()`, `BootParaYMX()`, `DeltaStdXMY()`, `DeltaStdXYM()`, `DeltaStdYMX()`, `DeltaXMY()`, `DeltaXYM()`, `IllustrationBootPara()`, `MCStdXMY()`, `MCStdXYM()`, `MCStdYMX()`, `MCXMY()`, `MCXYM()`, `MCYMX()`

Examples

```
## Not run:
set.seed(42)
library(dynr)
sim <- GenData(taskid = 1)
data <- RandomMeasurement(sim)
fit <- FitDynr(data, taskid = 1)
phi_hat <- PhiHat(fit)
ci <- DeltaYMX(phi_hat)
plot(ci)

## End(Not run)
```

FigPlotEffects *Plot Total, Direct, and Indirect Effects*

Description

Effects for the model $X \rightarrow M \rightarrow Y$.

Usage

```
FigPlotEffects(dynamics = 0, std = FALSE, max_delta_t = 30, xmy = TRUE)
```

Arguments

`dynamics`	Integer. `dynamics` = `0` for original drift matrix, `dynamics` = `-1` for near-neutral dynamics, and `dynamics` = `1` for stronger damping.
`std`	Logical. If `std` = `TRUE`, standardized total, direct, and indirect effects. If `std` = `FALSE`, unstandardized total, direct, and indirect effects.
`max_delta_t`	Numeric. Maximum time interval.
`xmy`	Logical. If `xmy` = `TRUE`, plot the effects for the `x` -> m -> `ymediation model. Ifxmy` = FALSE`, plot the effects for the y` -> m -> x‘ mediation model.

Author(s)

Ivan Jacob Agaloos Pesigan

See Also

Other Figure Functions: `FigScatterPlotCoverage()`, `FigScatterPlotPower()`, `FigScatterPlotSeBias()`, `FigScatterPlotType1()`, `IllustrationFigPlotEffects()`, `IllustrationFigScatterPlotCoverage()`, `IllustrationFigScatterPlotPower()`, `IllustrationFigScatterPlotSeBias()`

Examples

```
FigPlotEffects()
```

`FigScatterPlotCoverage` *Plot Coverage Probabilities*

Description

Coverage probabilities for the model $X \to M \to Y$.

Usage

```
FigScatterPlotCoverage(results, delta_t = NULL, dynamics = 0, std = FALSE)
```

Arguments

`results`	Summary results data frame.
`delta_t`	Vector of time-interval value. If `delta_t` = `NULL`, use all available time-intervals
`dynamics`	Integer. `dynamics` = `0` for original drift matrix, `dynamics` = `-1` for near-neutral dynamics, and `dynamics` = `1` for stronger damping.
`std`	Logical. If `std` = `TRUE`, standardized total, direct, and indirect effects. If `std` = `FALSE`, unstandardized total, direct, and indirect effects.

Author(s)

Ivan Jacob Agaloos Pesigan

See Also

Other Figure Functions: FigPlotEffects(), FigScatterPlotPower(), FigScatterPlotSeBias(), FigScatterPlotType1(), IllustrationFigPlotEffects(), IllustrationFigScatterPlotCoverage(), IllustrationFigScatterPlotPower(), IllustrationFigScatterPlotSeBias()

Examples

```
data(results, package = "manCTMed")
FigScatterPlotCoverage(results)
FigScatterPlotCoverage(results, delta_t = 1:14)
FigScatterPlotCoverage(results, delta_t = 15:30)
```

`FigScatterPlotPower` *Plot Statistical Power*

Description

Statistical power for the model $X \to M \to Y$.

Usage

```
FigScatterPlotPower(results, delta_t = NULL, dynamics = 0, std = FALSE)
```

Arguments

`results`	Summary results data frame.
`delta_t`	Vector of time-interval value. If `delta_t = NULL`, use all available time-intervals
`dynamics`	Integer. `dynamics = 0` for original drift matrix, `dynamics = -1` for near-neutral dynamics, and `dynamics = 1` for stronger damping.
`std`	Logical. If `std = TRUE`, standardized total, direct, and indirect effects. If `std = FALSE`, unstandardized total, direct, and indirect effects.

Author(s)

Ivan Jacob Agaloos Pesigan

See Also

Other Figure Functions: FigPlotEffects(), FigScatterPlotCoverage(), FigScatterPlotSeBias(), FigScatterPlotType1(), IllustrationFigPlotEffects(), IllustrationFigScatterPlotCoverage(), IllustrationFigScatterPlotPower(), IllustrationFigScatterPlotSeBias()

Examples

```
data(results, package = "manCTMed")
FigScatterPlotPower(results)
FigScatterPlotPower(results, delta_t = 1:14)
FigScatterPlotPower(results, delta_t = 15:30)
```

`FigScatterPlotSeBias` *Plot Standard Error Bias*

Description

Standard Error Bias for the model $X \to M \to Y$.

Usage

```
FigScatterPlotSeBias(results, delta_t = NULL, dynamics = 0, std = FALSE)
```

Arguments

`results`	Summary results data frame.
`delta_t`	Vector of time-interval value. If `delta_t` = `NULL`, use all available time-intervals
`dynamics`	Integer. `dynamics` = `0` for original drift matrix, `dynamics` = `-1` for near-neutral dynamics, and `dynamics` = `1` for stronger damping.
`std`	Logical. If `std` = `TRUE`, standardized total, direct, and indirect effects. If `std` = `FALSE`, unstandardized total, direct, and indirect effects.

Author(s)

Ivan Jacob Agaloos Pesigan

See Also

Other Figure Functions: `FigPlotEffects()`, `FigScatterPlotCoverage()`, `FigScatterPlotPower()`, `FigScatterPlotType1()`, `IllustrationFigPlotEffects()`, `IllustrationFigScatterPlotCoverage()`, `IllustrationFigScatterPlotPower()`, `IllustrationFigScatterPlotSeBias()`

Examples

```
data(results, package = "manCTMed")
FigScatterPlotSeBias(results)
```

FigScatterPlotType1 *Plot Type I Error*

Description

Type I error for the model $Y \to M \to X$.

Usage

```
FigScatterPlotType1(results, delta_t = NULL, dynamics = 0, std = FALSE)
```

Arguments

`results`	Summary results data frame.
`delta_t`	Vector of time-interval value. If `delta_t` = `NULL`, use all available time-intervals
`dynamics`	Integer. `dynamics` = `0` for original drift matrix, `dynamics` = `-1` for near-neutral dynamics, and `dynamics` = `1` for stronger damping.
`std`	Logical. If `std` = `TRUE`, standardized total, direct, and indirect effects. If `std` = `FALSE`, unstandardized total, direct, and indirect effects.

Author(s)

Ivan Jacob Agaloos Pesigan

See Also

Other Figure Functions: `FigPlotEffects()`, `FigScatterPlotCoverage()`, `FigScatterPlotPower()`, `FigScatterPlotSeBias()`, `IllustrationFigPlotEffects()`, `IllustrationFigScatterPlotCoverage()`, `IllustrationFigScatterPlotPower()`, `IllustrationFigScatterPlotSeBias()`

Examples

```
data(results, package = "manCTMed")
FigScatterPlotType1(results)
FigScatterPlotType1(results, delta_t = 1:14)
FigScatterPlotType1(results, delta_t = 15:30)
```

FitDynr *Fit the Model using the dynr Package*

Description

The function fits the model using the dynr::dynr package.

Usage

```
FitDynr(data, taskid)
```

Arguments

`data`	R object. Output of the `RandomMeasurement()` function.
`taskid`	Positive integer. Task ID.

See Also

Other Model Fitting Functions: `FitMx()`, `IllustrationFitDynr()`, `IllustrationFitMx()`, `IllustrationMCPhiSigma()`
`IllustrationPrepData()`, `PhiHat()`, `ThetaHat()`

Examples

```
## Not run:
set.seed(42)
library(dynr)
sim <- GenData(taskid = 1)
data <- RandomMeasurement(sim)
fit <- FitDynr(data, taskid = 1)
summary(fit)

## End(Not run)
```

FitMx *Fit the Model using the OpenMx Package*

Description

The function fits the model using the OpenMx::OpenMx package.

Usage

```
FitMx(data, taskid)
```

Arguments

`data`	R object. Output of the `RandomMeasurement()` function.
`taskid`	Positive integer. Task ID.

See Also

Other Model Fitting Functions: `FitDynr()`, `IllustrationFitDynr()`, `IllustrationFitMx()`, `IllustrationMCPhiSigma()`, `IllustrationPrepData()`, `PhiHat()`, `ThetaHat()`

Examples

```
## Not run:
set.seed(42)
library(OpenMx)
sim <- GenData(taskid = 1)
data <- RandomMeasurement(sim)
fit <- FitMx(data, taskid = 1)
summary(fit)

## End(Not run)
```

`GenData` *Simulate Data*

Description

The function simulates data using the `simStateSpace::SimSSMOUFixed()` function.

Usage

```
GenData(taskid)
```

Arguments

`taskid`	Positive integer. Task ID.

See Also

Other Data Generation Functions: `IllustrationGenData()`, `RandomMeasurement()`

Examples

```
## Not run:
set.seed(42)
sim <- GenData(taskid = 1)
plot(sim)

## End(Not run)
```

IllustrationBootPara *Parametric Bootstrap (Illustration)*

Description

The function generates simulated datasets based on a fitted model and refits the model to each generated dataset using the dynr package.

Usage

```
IllustrationBootPara(
  fit,
  path,
  prefix,
  taskid,
  B = 1000L,
  ncores = NULL,
  seed = NULL
)
```

Arguments

fit	R object. Fitted CT-VAR model.
path	Path to a directory to store bootstrap samples and estimates.
prefix	Character string. Prefix used for the file names for the bootstrap samples and estimates.
taskid	Positive integer. Task ID.
B	Positive integer. Number of bootstrap samples.
ncores	Positive integer. Number of cores to use.
seed	Integer. Random seed.

See Also

Other Confidence Interval Functions: BootPara(), BootParaStdXMY(), BootParaStdXYM(), BootParaStdYMX(), BootParaXMY(), BootParaXYM(), BootParaYMX(), DeltaStdXMY(), DeltaStdXYM(), DeltaStdYMX(), DeltaXMY(), DeltaXYM(), DeltaYMX(), MCStdXMY(), MCStdXYM(), MCStdYMX(), MCXMY(), MCXYM(), MCYMX()

Examples

```
## Not run:
library(dynr)
sim <- IllustrationGenData(seed = 42)
data <- IllustrationPrepData(sim)
fit <- IllustrationFitDynr(data)
summary(fit)
```

```
IllustrationBootPara(
  fit = fit,
  path = getwd(),
  prefix = "pb",
  taskid = 1,
  B = 1000L,
  seed = 42
)

## End(Not run)
```

`IllustrationCompress` *Compress Replication (Illustration)*

Description

Compress Replication (Illustration)

Usage

```
IllustrationCompress(taskid, repid, output_folder)
```

Arguments

`taskid`	Positive integer. Task ID.
`repid`	Positive integer. Replication ID.
`output_folder`	Character string. Output folder.

Value

The output is saved as an external file in `output_folder`.

Author(s)

Ivan Jacob Agaloos Pesigan

See Also

Other Compression Functions: `Compress()`

`IllustrationFigPlotEffects` *Plot Total, Direct, and Indirect Effects (Illustration)*

Description

Effects for the model $X \to M \to Y$.

Usage

```
IllustrationFigPlotEffects(std = FALSE, max_delta_t = 30)
```

Arguments

`std`	Logical. If `std = TRUE`, standardized total, direct, and indirect effects. If `std = FALSE`, unstandardized total, direct, and indirect effects.
`max_delta_t`	Numeric. Maximum time interval.

Author(s)

Ivan Jacob Agaloos Pesigan

See Also

Other Figure Functions: `FigPlotEffects()`, `FigScatterPlotCoverage()`, `FigScatterPlotPower()`, `FigScatterPlotSeBias()`, `FigScatterPlotType1()`, `IllustrationFigScatterPlotCoverage()`, `IllustrationFigScatterPlotPower()`, `IllustrationFigScatterPlotSeBias()`

Examples

```
IllustrationFigPlotEffects(std = FALSE)
IllustrationFigPlotEffects(std = TRUE)
```

`IllustrationFigScatterPlotCoverage` *Illustration Plot Coverage Probabilities*

Description

Coverage probabilities for the model $X \to M \to Y$.

Usage

```
IllustrationFigScatterPlotCoverage(illustration_results)
```

Arguments

`illustration_results`
Summary results data frame.

Author(s)

Ivan Jacob Agaloos Pesigan

See Also

Other Figure Functions: `FigPlotEffects()`, `FigScatterPlotCoverage()`, `FigScatterPlotPower()`, `FigScatterPlotSeBias()`, `FigScatterPlotType1()`, `IllustrationFigPlotEffects()`, `IllustrationFigScatterPlo`
`IllustrationFigScatterPlotSeBias()`

Examples

```
data(illustration_results, package = "manCTMed")
IllustrationFigScatterPlotCoverage(illustration_results)
```

IllustrationFigScatterPlotPower
Illustration Plot Statistical Power

Description

Statistical Power for the model $X \to M \to Y$.

Usage

```
IllustrationFigScatterPlotPower(illustration_results)
```

Arguments

`illustration_results`
Summary results data frame.

Author(s)

Ivan Jacob Agaloos Pesigan

See Also

Other Figure Functions: `FigPlotEffects()`, `FigScatterPlotCoverage()`, `FigScatterPlotPower()`, `FigScatterPlotSeBias()`, `FigScatterPlotType1()`, `IllustrationFigPlotEffects()`, `IllustrationFigScatterPlo`
`IllustrationFigScatterPlotSeBias()`

Examples

```
data(illustration_results, package = "manCTMed")
IllustrationFigScatterPlotPower(illustration_results)
```

IllustrationFigScatterPlotSeBias
Illustration Plot Standard Error Bias

Description

Standard Error Bias for the model $X \rightarrow M \rightarrow Y$.

Usage

```
IllustrationFigScatterPlotSeBias(illustration_results)
```

Arguments

`illustration_results`	Summary results data frame.

Author(s)

Ivan Jacob Agaloos Pesigan

See Also

Other Figure Functions: `FigPlotEffects()`, `FigScatterPlotCoverage()`, `FigScatterPlotPower()`, `FigScatterPlotSeBias()`, `FigScatterPlotType1()`, `IllustrationFigPlotEffects()`, `IllustrationFigScatterPlo`
`IllustrationFigScatterPlotPower()`

Examples

```
data(illustration_results, package = "manCTMed")
IllustrationFigScatterPlotSeBias(illustration_results)
```

IllustrationFitDynr *Fit the Model using the dynr Package (Illustration)*

Description

The function fits the model using the dynr::dynr package.

Usage

```
IllustrationFitDynr(data)
```

Arguments

`data`	R object. Output of the `IllustrationPrepData()` function.

See Also

Other Model Fitting Functions: `FitDynr()`, `FitMx()`, `IllustrationFitMx()`, `IllustrationMCPhiSigma()`, `IllustrationPrepData()`, `PhiHat()`, `ThetaHat()`

Examples

```
## Not run:
library(dynr)
sim <- IllustrationGenData(seed = 42)
data <- IllustrationPrepData(sim)
fit <- IllustrationFitDynr(data)
summary(fit)

## End(Not run)
```

IllustrationFitMx *Fit the Model using the OpenMx Package (Illustration)*

Description

The function fits the model using the OpenMx::OpenMx package.

Usage

```
IllustrationFitMx(data)
```

Arguments

`data`	R object. Output of the `IllustrationPrepData()` function.

See Also

Other Model Fitting Functions: `FitDynr()`, `FitMx()`, `IllustrationFitDynr()`, `IllustrationMCPhiSigma()`, `IllustrationPrepData()`, `PhiHat()`, `ThetaHat()`

Examples

```
## Not run:
library(OpenMx)
sim <- IllustrationGenData(seed = 42)
data <- IllustrationPrepData(sim)
fit <- IllustrationFitMx(data)
summary(fit)

## End(Not run)
```

`IllustrationGenData` *Simulate Data (Illustration)*

Description

The function simulates data using the `simStateSpace::SimSSMOUFixed()` function.

Usage

```
IllustrationGenData(seed = NULL, n = 133, m = 101, delta_t_gen = 0.1)
```

Arguments

`seed`	Integer. Random seed.
`n`	Positive integer. Sample size.
`m`	Positive integer. Measurement occasions.
`delta_t_gen`	Numeric. Time interval used to generate data.

See Also

Other Data Generation Functions: `GenData()`, `RandomMeasurement()`

Examples

```
## Not run:
sim <- IllustrationGenData(seed = 42)
plot(sim)

## End(Not run)
```

IllustrationMCPhiSigma *Generate a Sampling Distribution of Drift Matrices and Process Noise Covariance Matrices (Illustration)*

Description

The function generates a sampling distribution of drift matrices and process noise covariance matrices using te Monte Carlo method.

Usage

```
IllustrationMCPhiSigma(fit, R = 20000L, seed = NULL)
```

Arguments

fit	R object. Fitted CT-VAR model.
R	Positive integer. Number of Monte Carlo replications.
seed	Integer. Random seed.

See Also

Other Model Fitting Functions: `FitDynr()`, `FitMx()`, `IllustrationFitDynr()`, `IllustrationFitMx()`, `IllustrationPrepData()`, `PhiHat()`, `ThetaHat()`

Examples

```
## Not run:
library(dynr)
sim <- IllustrationGenData(seed = 42)
data <- IllustrationPrepData(sim)
fit <- IllustrationFitDynr(data)
IllustrationMCPhiSigma(fit, seed = 42)

## End(Not run)
```

IllustrationPrepData *Prepare Data Before Model Fitting (Illustration)*

Description

The function converts the output of `IllustrationGenData()` into a data frame.

Usage

```
IllustrationPrepData(sim)
```

Arguments

`sim`	R object. Output of the `IllustrationGenData()` function.

See Also

Other Model Fitting Functions: `FitDynr()`, `FitMx()`, `IllustrationFitDynr()`, `IllustrationFitMx()`, `IllustrationMCPhiSigma()`, `PhiHat()`, `ThetaHat()`

Examples

```
## Not run:
sim <- IllustrationGenData(seed = 42)
data <- IllustrationPrepData(sim)
head(data)
dim(data)

## End(Not run)
```

`illustration_dist` *Illustration Sampling Distribution*

Description

Illustration Sampling Distribution

Usage

```
data(illustration_dist)
```

Format

A matrix with 1000 rows and 27 columns:

phi_xx Elements of the drift matrix.

sigma_xx Elements of the process noise covariance matrix.

theta_xx Elements of the measurement error covariance matrix.

mu0_x Elements of the initial condition mean vector.

sigma0_xx Elements of the initial condition covariance matrix.

Author(s)

Ivan Jacob Agaloos Pesigan

`illustration_dist_dt` *Illustration Sampling Distribution Discrete Time - Time Interval of 1*

Description

Illustration Sampling Distribution Discrete Time - Time Interval of 1

Usage

```
data(illustration_dist_dt)
```

Format

A matrix with 1000 rows and 15 columns:

beta_xx Elements of the matrix of lagged coefficients.

psi_xx Elements of the process noise covariance matrix.

Author(s)

Ivan Jacob Agaloos Pesigan

`illustration_dist_dt_mc` *Illustration Sampling Distribution Discrete Time - Time Interval of 1 (Monte Carlo Method)*

Description

Illustration Sampling Distribution Discrete Time - Time Interval of 1 (Monte Carlo Method)

Usage

```
data(illustration_dist_dt_mc)
```

Format

A matrix with 20000 rows and 15 columns:

beta_xx Elements of the matrix of lagged coefficients.

psi_xx Elements of the process noise covariance matrix.

Author(s)

Ivan Jacob Agaloos Pesigan

`illustration_dist_mc` *Illustration Sampling Distribution (Monte Carlo Method)*

Description

Illustration Sampling Distribution (Monte Carlo Method)

Usage

```
data(illustration_dist_mc)
```

Format

A matrix with 20000 rows and 15 columns:

phi_xx Elements of the drift matrix.

sigma_xx Elements of the process noise covariance matrix.

Author(s)

Ivan Jacob Agaloos Pesigan

`illustration_dist_med` *Illustration Sampling Distribution Total, Direct, and Indirect Effects - Time Interval of 1*

Description

Illustration Sampling Distribution Total, Direct, and Indirect Effects - Time Interval of 1

Usage

```
data(illustration_dist_med)
```

Format

A matrix with 1000 rows and 27 columns:

total Total effect.

direct Direct effect.

indirect Indirect effect.

Author(s)

Ivan Jacob Agaloos Pesigan

`illustration_dist_med_mc`

Illustration Sampling Distribution Total, Direct, and Indirect Effects - Time Interval of 1 (Monte Carlo Method)

Description

Illustration Sampling Distribution Total, Direct, and Indirect Effects - Time Interval of 1 (Monte Carlo Method)

Usage

```
data(illustration_dist_med_mc)
```

Format

A matrix with 20000 rows and 27 columns:

total Total effect.

direct Direct effect.

indirect Indirect effect.

Author(s)

Ivan Jacob Agaloos Pesigan

`illustration_dist_med_std`

Illustration Sampling Distribution Standardized Total, Direct, and Indirect Effects - Time Interval of 1

Description

Illustration Sampling Distribution Standardized Total, Direct, and Indirect Effects - Time Interval of 1

Usage

```
data(illustration_dist_med_std)
```

Format

A matrix with 1000 rows and 27 columns:

total Total effect.

direct Direct effect.

indirect Indirect effect.

Author(s)

Ivan Jacob Agaloos Pesigan

`illustration_dist_med_std_mc` *Illustration Sampling Distribution Standardized Total, Direct, and Indirect Effects - Time Interval of 1 (Monte Carlo Method)*

Description

Illustration Sampling Distribution Standardized Total, Direct, and Indirect Effects - Time Interval of 1 (Monte Carlo Method)

Usage

```
data(illustration_dist_med_std_mc)
```

Format

A matrix with 20000 rows and 27 columns:

total Total effect.

direct Direct effect.

indirect Indirect effect.

Author(s)

Ivan Jacob Agaloos Pesigan

`illustration_results` *Illustration Small Scale Simulation Results*

Description

Illustration Small Scale Simulation Results

Usage

```
data(illustration_results)
```

Format

A with 22 columns:

taskid Task ID.

replications Number of replications.

effect Total, direct, or indirect effect.

interval Time interval.

parameter Population parameter.

method Method used to generate confidence intervals.

xmy Logical. `TRUE` for x to m to y path.

std Logical. `TRUE` for standardized. `FALSE` for unstandardized.

est Mean parameter estimate.

se Mean standard error.

z Mean z statistic.

p Mean p-value.

R Number of Monte Carlo or bootstrap replications.

ll Mean lower limit of the 95% confidence interval.

ul Mean upper limit of the 95% confidence interval.

sig Proportion of statistically significant results.

zero_hit Proportion of replications where the confidence intervals included zero.

theta_hit Proportion of replications where the confidence intervals included the population parameter.

sq_error Mean squared error.

se_bias Bias in standard error estimate.

coverage Coverage probability.

power Statistical power.

Author(s)

Ivan Jacob Agaloos Pesigan

MCStdXMY — *Monte Carlo Method Confidence Intervals for X-M-Y (Standardized)*

Description

The function generates Monte Carlo method confidence intervals for the mediation model $X \to M \to Y$ (Standardized).

Usage

```
MCStdXMY(theta_hat, delta_t = 1:30, R = 20000L, seed = NULL)
```

Arguments

theta_hat	R object. Output of the `ThetaHat()` function.
delta_t	Numeric vector. Vector of time intervals.
R	Positive integer. Number of Monte Carlo replications.
seed	Integer. Random seed.

See Also

Other Confidence Interval Functions: `BootPara()`, `BootParaStdXMY()`, `BootParaStdXYM()`, `BootParaStdYMX()`, `BootParaXMY()`, `BootParaXYM()`, `BootParaYMX()`, `DeltaStdXMY()`, `DeltaStdXYM()`, `DeltaStdYMX()`, `DeltaXMY()`, `DeltaXYM()`, `DeltaYMX()`, `IllustrationBootPara()`, `MCStdXYM()`, `MCStdYMX()`, `MCXMY()`, `MCXYM()`, `MCYMX()`

Examples

```
## Not run:
set.seed(42)
library(dynr)
sim <- GenData(taskid = 1)
data <- RandomMeasurement(sim)
fit <- FitDynr(data, taskid = 1)
theta_hat <- ThetaHat(fit)
ci <- MCStdXMY(theta_hat, seed = 42)
plot(ci)

## End(Not run)
```

MCStdXYM *Monte Carlo Method Confidence Intervals for X-Y-M (Standardized)*

Description

The function generates Monte Carlo method confidence intervals for the mediation model $X \rightarrow Y \rightarrow M$ (Standardized).

Usage

```
MCStdXYM(theta_hat, delta_t = 1:30, R = 20000L, seed = NULL)
```

Arguments

`theta_hat`	R object. Output of the `ThetaHat()` function.
`delta_t`	Numeric vector. Vector of time intervals.
`R`	Positive integer. Number of Monte Carlo replications.
`seed`	Integer. Random seed.

See Also

Other Confidence Interval Functions: `BootPara()`, `BootParaStdXMY()`, `BootParaStdXYM()`, `BootParaStdYMX()`, `BootParaXMY()`, `BootParaXYM()`, `BootParaYMX()`, `DeltaStdXMY()`, `DeltaStdXYM()`, `DeltaStdYMX()`, `DeltaXMY()`, `DeltaXYM()`, `DeltaYMX()`, `IllustrationBootPara()`, `MCStdXMY()`, `MCStdYMX()`, `MCXMY()`, `MCXYM()`, `MCYMX()`

Examples

```
## Not run:
set.seed(42)
library(dynr)
sim <- GenData(taskid = 1)
data <- RandomMeasurement(sim)
fit <- FitDynr(data, taskid = 1)
theta_hat <- ThetaHat(fit)
ci <- MCStdXYM(theta_hat, seed = 42)
plot(ci)

## End(Not run)
```

MCStdYMX *Monte Carlo Method Confidence Intervals for Y-M-X (Standardized)*

Description

The function generates Monte Carlo method confidence intervals for the mediation model $Y \to M \to X$ (Standardized).

Usage

```
MCStdYMX(theta_hat, delta_t = 1:30, R = 20000L, seed = NULL)
```

Arguments

`theta_hat`	R object. Output of the `ThetaHat()` function.
`delta_t`	Numeric vector. Vector of time intervals.
`R`	Positive integer. Number of Monte Carlo replications.
`seed`	Integer. Random seed.

See Also

Other Confidence Interval Functions: `BootPara()`, `BootParaStdXMY()`, `BootParaStdXYM()`, `BootParaStdYMX()`, `BootParaXMY()`, `BootParaXYM()`, `BootParaYMX()`, `DeltaStdXMY()`, `DeltaStdXYM()`, `DeltaStdYMX()`, `DeltaXMY()`, `DeltaXYM()`, `DeltaYMX()`, `IllustrationBootPara()`, `MCStdXMY()`, `MCStdXYM()`, `MCXMY()`, `MCXYM()`, `MCYMX()`

Examples

```
## Not run:
set.seed(42)
library(dynr)
sim <- GenData(taskid = 1)
data <- RandomMeasurement(sim)
fit <- FitDynr(data, taskid = 1)
theta_hat <- ThetaHat(fit)
ci <- MCStdYMX(phi_hat, seed = 42)
plot(ci)

## End(Not run)
```

MCXMY — *Monte Carlo Method Confidence Intervals for X-M-Y*

Description

The function generates Monte Carlo method confidence intervals for the mediation model $X \to M \to Y$.

Usage

```
MCXMY(phi_hat, delta_t = 1:30, R = 20000L, seed = NULL)
```

Arguments

`phi_hat`	R object. Output of the `PhiHat()` function.
`delta_t`	Numeric vector. Vector of time intervals.
`R`	Positive integer. Number of Monte Carlo replications.
`seed`	Integer. Random seed.

See Also

Other Confidence Interval Functions: `BootPara()`, `BootParaStdXMY()`, `BootParaStdXYM()`, `BootParaStdYMX()`, `BootParaXMY()`, `BootParaXYM()`, `BootParaYMX()`, `DeltaStdXMY()`, `DeltaStdXYM()`, `DeltaStdYMX()`, `DeltaXMY()`, `DeltaXYM()`, `DeltaYMX()`, `IllustrationBootPara()`, `MCStdXMY()`, `MCStdXYM()`, `MCStdYMX()`, `MCXYM()`, `MCYMX()`

Examples

```
## Not run:
set.seed(42)
library(dynr)
sim <- GenData(taskid = 1)
data <- RandomMeasurement(sim)
fit <- FitDynr(data, taskid = 1)
phi_hat <- PhiHat(fit)
ci <- MCXMY(phi_hat, seed = 42)
plot(ci)

## End(Not run)
```

MCXYM — *Monte Carlo Method Confidence Intervals for X-Y-M*

Description

The function generates Monte Carlo method confidence intervals for the mediation model $X \to Y \to M$.

Usage

```
MCXYM(phi_hat, delta_t = 1:30, R = 20000L, seed = NULL)
```

Arguments

`phi_hat`	R object. Output of the `PhiHat()` function.
`delta_t`	Numeric vector. Vector of time intervals.
`R`	Positive integer. Number of Monte Carlo replications.
`seed`	Integer. Random seed.

See Also

Other Confidence Interval Functions: `BootPara()`, `BootParaStdXMY()`, `BootParaStdXYM()`, `BootParaStdYMX()`, `BootParaXMY()`, `BootParaXYM()`, `BootParaYMX()`, `DeltaStdXMY()`, `DeltaStdXYM()`, `DeltaStdYMX()`, `DeltaXMY()`, `DeltaXYM()`, `DeltaYMX()`, `IllustrationBootPara()`, `MCStdXMY()`, `MCStdXYM()`, `MCStdYMX()`, `MCXMY()`, `MCYMX()`

Examples

```
## Not run:
set.seed(42)
library(dynr)
sim <- GenData(taskid = 1)
data <- RandomMeasurement(sim)
fit <- FitDynr(data, taskid = 1)
phi_hat <- PhiHat(fit)
ci <- MCXYM(phi_hat, seed = 42)
plot(ci)

## End(Not run)
```

MCYMX *Monte Carlo Method Confidence Intervals for Y-M-X*

Description

The function generates Monte Carlo method confidence intervals for the mediation model $Y \rightarrow M \rightarrow X$.

Usage

```
MCYMX(phi_hat, delta_t = 1:30, R = 20000L, seed = NULL)
```

Arguments

phi_hat	R object. Output of the PhiHat() function.
delta_t	Numeric vector. Vector of time intervals.
R	Positive integer. Number of Monte Carlo replications.
seed	Integer. Random seed.

See Also

Other Confidence Interval Functions: BootPara(), BootParaStdXMY(), BootParaStdXYM(), BootParaStdYMX(), BootParaXMY(), BootParaXYM(), BootParaYMX(), DeltaStdXMY(), DeltaStdXYM(), DeltaStdYMX(), DeltaXMY(), DeltaXYM(), DeltaYMX(), IllustrationBootPara(), MCStdXMY(), MCStdXYM(), MCStdYMX(), MCXMY(), MCXYM()

Examples

```
## Not run:
set.seed(42)
library(dynr)
sim <- GenData(taskid = 1)
data <- RandomMeasurement(sim)
fit <- FitDynr(data, taskid = 1)
phi_hat <- PhiHat(fit)
ci <- MCYMX(phi_hat, seed = 42)
plot(ci)

## End(Not run)
```

`params` *Simulation Parameters*

Description

Simulation Parameters

Usage

```
data(params)
```

Format

A dataframe with 30 rows and 3 columns:

taskid Simulation Task ID.

n Sample size.

dynamics Dynamics. `0` for original drift matrix, `-1` for near-neutral dynamics, and `1` for stronger damping.

Author(s)

Ivan Jacob Agaloos Pesigan

`PhiHat` *Estimated Drift Matrix*

Description

The function extracts the estimated drift matrix from the fitted model.

Usage

```
PhiHat(fit)
```

Arguments

`fit`	R object. Output of the `FitDynr()`, `FitMx()`, `IllustrationFitDynr()`, or `IllustrationFitMx()`, functions.

See Also

Other Model Fitting Functions: `FitDynr()`, `FitMx()`, `IllustrationFitDynr()`, `IllustrationFitMx()`, `IllustrationMCPhiSigma()`, `IllustrationPrepData()`, `ThetaHat()`

Examples

```
## Not run:
set.seed(42)
library(dynr)
sim <- GenData(n = 50)
data <- RandomMeasurement(sim)
fit <- FitDynr(data)
PhiHat(fit)

## End(Not run)
```

`RandomMeasurement` *Simulate Random Measurement*

Description

The function randomly selects 100 observations from the generated data and replaces the unselected observations with `NA`.

Usage

```
RandomMeasurement(sim)
```

Arguments

`sim`	R object. Output of the `GenData()` function.

See Also

Other Data Generation Functions: `GenData()`, `IllustrationGenData()`

Examples

```
## Not run:
set.seed(42)
sim <- GenData(taskid = 1)
RandomMeasurement(sim)

## End(Not run)
```

`results` *Simulation Results*

Description

Simulation Results

Usage

```
data(results)
```

Format

A dataframe with 24 columns:

taskid Task ID.

replications Number of replications.

effect Total, direct, or indirect effect.

interval Time interval.

dynamics Dynamics. `0` for original drift matrix, `-1` for near-neutral dynamics, and `1` for stronger damping.

parameter Population parameter.

method Method used to generate confidence intervals.

xmy If `TRUE`, the mediation model is $X \to M \to Y$. If `FALSE`, the mediation model is $Y \to M \to X$.

std If `TRUE`, standardized total, direct, and indirect effects. If `FALSE`, unstandardized total, direct, and indirect effects.

n Sample size.

est Mean parameter estimate.

se Mean standard error.

z Mean z statistic.

p Mean p-value.

R Number of Monte Carlo replications.

ll Mean lower limit of the 95% confidence interval.

ul Mean upper limit of the 95% confidence interval.

sig Proportion of statistically significant results.

zero_hit Proportion of replications where the confidence intervals contained zero.

theta_hit Proportion of replications where the confidence intervals contained the population parameter.

sq_error Mean squared error.

se_bias Bias in standard error estimate.

coverage Coverage probability.

power Statistical power.

Author(s)

Ivan Jacob Agaloos Pesigan

`Sim` *Simulation Replication*

Description

Simulation Replication

Usage

```
Sim(
  taskid,
  repid,
  output_folder,
  overwrite,
  integrity,
  seed,
  ci,
  pb,
  delta_t,
  R,
  B
)
```

Arguments

`taskid`	Positive integer. Task ID.
`repid`	Positive integer. Replication ID.
`output_folder`	Character string. Output folder.
`overwrite`	Logical. Overwrite existing output in `output_folder`.
`integrity`	Logical. If `integrity = TRUE`, check for the output file integrity when `overwrite = FALSE`.
`seed`	Integer. Random seed.
`ci`	Logical. Run simulations for confidence intervals.
`pb`	Logical. Run simulations for parametric bootstrap confidence intervals.
`delta_t`	Numeric vector. Vector of time intervals.
`R`	Positive integer. Number of Monte Carlo replications.
`B`	Positive integer. Number of bootstrap samples.

Value

The output is saved as an external file in `output_folder`.

Author(s)

Ivan Jacob Agaloos Pesigan

`SimDynrBootPara` *Simulation Replication - BootPara*

Description

Simulation Replication - BootPara

Usage

```
SimDynrBootPara(
  taskid,
  repid,
  output_folder,
  seed,
  suffix,
  overwrite,
  integrity,
  B,
  ncores = NULL
)
```

Arguments

`taskid`	Positive integer. Task ID.
`repid`	Positive integer. Replication ID.
`output_folder`	Character string. Output folder.
`seed`	Integer. Random seed.
`suffix`	Character string. Output of `manCTMed:::.SimSuffix()`.
`overwrite`	Logical. Overwrite existing output in `output_folder`.
`integrity`	Logical. If `integrity = TRUE`, check for the output file integrity when `overwrite = FALSE`.
`B`	Positive integer. Number of bootstrap samples.
`ncores`	Positive integer. Number of cores to use.

Details

This function is executed via the `Sim` function.

Value

The output is saved as an external file in `output_folder`.

Author(s)

Ivan Jacob Agaloos Pesigan

SimDynrBootParaStdXMY *Simulation Replication - BootParaStdXMY*

Description

Simulation Replication - BootParaStdXMY

Usage

```
SimDynrBootParaStdXMY(
  taskid,
  repid,
  output_folder,
  seed,
  suffix,
  overwrite,
  integrity,
  delta_t
)
```

Arguments

`taskid`	Positive integer. Task ID.
`repid`	Positive integer. Replication ID.
`output_folder`	Character string. Output folder.
`seed`	Integer. Random seed.
`suffix`	Character string. Output of `manCTMed:::.SimSuffix()`.
`overwrite`	Logical. Overwrite existing output in `output_folder`.
`integrity`	Logical. If `integrity = TRUE`, check for the output file integrity when `overwrite = FALSE`.
`delta_t`	Numeric vector. Vector of time intervals.

Details

This function is executed via the `Sim` function.

Value

The output is saved as an external file in `output_folder`.

Author(s)

Ivan Jacob Agaloos Pesigan

SimDynrBootParaStdYMX *Simulation Replication - BootParaStdYMX*

Description

Simulation Replication - BootParaStdYMX

Usage

```
SimDynrBootParaStdYMX(
  taskid,
  repid,
  output_folder,
  seed,
  suffix,
  overwrite,
  integrity,
  delta_t
)
```

Arguments

`taskid`	Positive integer. Task ID.
`repid`	Positive integer. Replication ID.
`output_folder`	Character string. Output folder.
`seed`	Integer. Random seed.
`suffix`	Character string. Output of `manCTMed:::.SimSuffix()`.
`overwrite`	Logical. Overwrite existing output in `output_folder`.
`integrity`	Logical. If `integrity = TRUE`, check for the output file integrity when `overwrite = FALSE`.
`delta_t`	Numeric vector. Vector of time intervals.

Details

This function is executed via the `Sim` function.

Value

The output is saved as an external file in `output_folder`.

Author(s)

Ivan Jacob Agaloos Pesigan

`SimDynrBootParaXMY` *Simulation Replication - BootParaXMY*

Description

Simulation Replication - BootParaXMY

Usage

```
SimDynrBootParaXMY(
  taskid,
  repid,
  output_folder,
  seed,
  suffix,
  overwrite,
  integrity,
  delta_t
)
```

Arguments

`taskid`	Positive integer. Task ID.
`repid`	Positive integer. Replication ID.
`output_folder`	Character string. Output folder.
`seed`	Integer. Random seed.
`suffix`	Character string. Output of `manCTMed:::.SimSuffix()`.
`overwrite`	Logical. Overwrite existing output in `output_folder`.
`integrity`	Logical. If `integrity = TRUE`, check for the output file integrity when `overwrite = FALSE`.
`delta_t`	Numeric vector. Vector of time intervals.

Details

This function is executed via the `Sim` function.

Value

The output is saved as an external file in `output_folder`.

Author(s)

Ivan Jacob Agaloos Pesigan

`SimDynrBootParaYMX` *Simulation Replication - BootParaYMX*

Description

Simulation Replication - BootParaYMX

Usage

```
SimDynrBootParaYMX(
  taskid,
  repid,
  output_folder,
  seed,
  suffix,
  overwrite,
  integrity,
  delta_t
)
```

Arguments

`taskid`	Positive integer. Task ID.
`repid`	Positive integer. Replication ID.
`output_folder`	Character string. Output folder.
`seed`	Integer. Random seed.
`suffix`	Character string. Output of `manCTMed:::.SimSuffix()`.
`overwrite`	Logical. Overwrite existing output in `output_folder`.
`integrity`	Logical. If `integrity = TRUE`, check for the output file integrity when `overwrite = FALSE`.
`delta_t`	Numeric vector. Vector of time intervals.

Details

This function is executed via the `Sim` function.

Value

The output is saved as an external file in `output_folder`.

Author(s)

Ivan Jacob Agaloos Pesigan

SimDynrDeltaStdXMY *Simulation Replication - DynrDeltaStdXMY*

Description

Simulation Replication - DynrDeltaStdXMY

Usage

```
SimDynrDeltaStdXMY(
  taskid,
  repid,
  output_folder,
  seed,
  suffix,
  overwrite,
  integrity,
  delta_t
)
```

Arguments

`taskid`	Positive integer. Task ID.
`repid`	Positive integer. Replication ID.
`output_folder`	Character string. Output folder.
`seed`	Integer. Random seed.
`suffix`	Character string. Output of `manCTMed:::.SimSuffix()`.
`overwrite`	Logical. Overwrite existing output in `output_folder`.
`integrity`	Logical. If `integrity = TRUE`, check for the output file integrity when `overwrite = FALSE`.
`delta_t`	Numeric vector. Vector of time intervals.

Details

This function is executed via the `Sim` function.

Value

The output is saved as an external file in `output_folder`.

Author(s)

Ivan Jacob Agaloos Pesigan

`SimDynrDeltaStdYMX` *Simulation Replication - DynrDeltaStdYMX*

Description

Simulation Replication - DynrDeltaStdYMX

Usage

```
SimDynrDeltaStdYMX(
  taskid,
  repid,
  output_folder,
  seed,
  suffix,
  overwrite,
  integrity,
  delta_t
)
```

Arguments

`taskid`	Positive integer. Task ID.
`repid`	Positive integer. Replication ID.
`output_folder`	Character string. Output folder.
`seed`	Integer. Random seed.
`suffix`	Character string. Output of `manCTMed:::.SimSuffix()`.
`overwrite`	Logical. Overwrite existing output in `output_folder`.
`integrity`	Logical. If `integrity = TRUE`, check for the output file integrity when `overwrite = FALSE`.
`delta_t`	Numeric vector. Vector of time intervals.

Details

This function is executed via the `Sim` function.

Value

The output is saved as an external file in `output_folder`.

Author(s)

Ivan Jacob Agaloos Pesigan

`SimDynrDeltaXMY` *Simulation Replication - DynrDeltaXMY*

Description

Simulation Replication - DynrDeltaXMY

Usage

```
SimDynrDeltaXMY(
  taskid,
  repid,
  output_folder,
  seed,
  suffix,
  overwrite,
  integrity,
  delta_t
)
```

Arguments

`taskid`	Positive integer. Task ID.
`repid`	Positive integer. Replication ID.
`output_folder`	Character string. Output folder.
`seed`	Integer. Random seed.
`suffix`	Character string. Output of `manCTMed:::.SimSuffix()`.
`overwrite`	Logical. Overwrite existing output in `output_folder`.
`integrity`	Logical. If `integrity = TRUE`, check for the output file integrity when `overwrite = FALSE`.
`delta_t`	Numeric vector. Vector of time intervals.

Details

This function is executed via the `Sim` function.

Value

The output is saved as an external file in `output_folder`.

Author(s)

Ivan Jacob Agaloos Pesigan

SimDynrDeltaYMX *Simulation Replication - DynrDeltaYMX*

Description

Simulation Replication - DynrDeltaYMX

Usage

```
SimDynrDeltaYMX(
  taskid,
  repid,
  output_folder,
  seed,
  suffix,
  overwrite,
  integrity,
  delta_t
)
```

Arguments

`taskid`	Positive integer. Task ID.
`repid`	Positive integer. Replication ID.
`output_folder`	Character string. Output folder.
`seed`	Integer. Random seed.
`suffix`	Character string. Output of `manCTMed:::.SimSuffix()`.
`overwrite`	Logical. Overwrite existing output in `output_folder`.
`integrity`	Logical. If `integrity = TRUE`, check for the output file integrity when `overwrite = FALSE`.
`delta_t`	Numeric vector. Vector of time intervals.

Details

This function is executed via the `Sim` function.

Value

The output is saved as an external file in `output_folder`.

Author(s)

Ivan Jacob Agaloos Pesigan

SimDynrMCStdXMY *Simulation Replication - DynrMCStdXMY*

Description

Simulation Replication - DynrMCStdXMY

Usage

```
SimDynrMCStdXMY(
  taskid,
  repid,
  output_folder,
  seed,
  suffix,
  overwrite,
  integrity,
  delta_t,
  R
)
```

Arguments

`taskid`	Positive integer. Task ID.
`repid`	Positive integer. Replication ID.
`output_folder`	Character string. Output folder.
`seed`	Integer. Random seed.
`suffix`	Character string. Output of `manCTMed:::.SimSuffix()`.
`overwrite`	Logical. Overwrite existing output in `output_folder`.
`integrity`	Logical. If `integrity = TRUE`, check for the output file integrity when `overwrite = FALSE`.
`delta_t`	Numeric vector. Vector of time intervals.
`R`	Positive integer. Number of Monte Carlo replications.

Details

This function is executed via the `Sim` function.

Value

The output is saved as an external file in `output_folder`.

Author(s)

Ivan Jacob Agaloos Pesigan

`SimDynrMCStdYMX` *Simulation Replication - DynrMCStdYMX*

Description

Simulation Replication - DynrMCStdYMX

Usage

```
SimDynrMCStdYMX(
  taskid,
  repid,
  output_folder,
  seed,
  suffix,
  overwrite,
  integrity,
  delta_t,
  R
)
```

Arguments

`taskid`	Positive integer. Task ID.
`repid`	Positive integer. Replication ID.
`output_folder`	Character string. Output folder.
`seed`	Integer. Random seed.
`suffix`	Character string. Output of `manCTMed:::.SimSuffix()`.
`overwrite`	Logical. Overwrite existing output in `output_folder`.
`integrity`	Logical. If `integrity = TRUE`, check for the output file integrity when `overwrite = FALSE`.
`delta_t`	Numeric vector. Vector of time intervals.
`R`	Positive integer. Number of Monte Carlo replications.

Details

This function is executed via the `Sim` function.

Value

The output is saved as an external file in `output_folder`.

Author(s)

Ivan Jacob Agaloos Pesigan

SimDynrMCXMY *Simulation Replication - DynrMCXMY*

Description

Simulation Replication - DynrMCXMY

Usage

```
SimDynrMCXMY(
  taskid,
  repid,
  output_folder,
  seed,
  suffix,
  overwrite,
  integrity,
  delta_t,
  R
)
```

Arguments

`taskid`	Positive integer. Task ID.
`repid`	Positive integer. Replication ID.
`output_folder`	Character string. Output folder.
`seed`	Integer. Random seed.
`suffix`	Character string. Output of `manCTMed:::.SimSuffix()`.
`overwrite`	Logical. Overwrite existing output in `output_folder`.
`integrity`	Logical. If `integrity = TRUE`, check for the output file integrity when `overwrite = FALSE`.
`delta_t`	Numeric vector. Vector of time intervals.
`R`	Positive integer. Number of Monte Carlo replications.

Details

This function is executed via the `Sim` function.

Value

The output is saved as an external file in `output_folder`.

Author(s)

Ivan Jacob Agaloos Pesigan

SimDynrMCYMX — *Simulation Replication - DynrMCYMX*

Description

Simulation Replication - DynrMCYMX

Usage

```
SimDynrMCYMX(
  taskid,
  repid,
  output_folder,
  seed,
  suffix,
  overwrite,
  integrity,
  delta_t,
  R
)
```

Arguments

`taskid`	Positive integer. Task ID.
`repid`	Positive integer. Replication ID.
`output_folder`	Character string. Output folder.
`seed`	Integer. Random seed.
`suffix`	Character string. Output of `manCTMed:::.SimSuffix()`.
`overwrite`	Logical. Overwrite existing output in `output_folder`.
`integrity`	Logical. If `integrity = TRUE`, check for the output file integrity when `overwrite = FALSE`.
`delta_t`	Numeric vector. Vector of time intervals.
`R`	Positive integer. Number of Monte Carlo replications.

Details

This function is executed via the `Sim` function.

Value

The output is saved as an external file in `output_folder`.

Author(s)

Ivan Jacob Agaloos Pesigan

`SimFitDynr` *Simulation Replication - FitDynr*

Description

Simulation Replication - FitDynr

Usage

```
SimFitDynr(taskid, repid, output_folder, seed, suffix, overwrite, integrity)
```

Arguments

`taskid`	Positive integer. Task ID.
`repid`	Positive integer. Replication ID.
`output_folder`	Character string. Output folder.
`seed`	Integer. Random seed.
`suffix`	Character string. Output of `manCTMed:::.SimSuffix()`.
`overwrite`	Logical. Overwrite existing output in `output_folder`.
`integrity`	Logical. If `integrity = TRUE`, check for the output file integrity when `overwrite = FALSE`.

Details

This function is executed via the `Sim` function.

Value

The output is saved as an external file in `output_folder`.

Author(s)

Ivan Jacob Agaloos Pesigan

`SimFitMx` *Simulation Replication - FitMx*

Description

Simulation Replication - FitMx

Usage

```
SimFitMx(taskid, repid, output_folder, seed, suffix, overwrite, integrity)
```

Arguments

`taskid`	Positive integer. Task ID.
`repid`	Positive integer. Replication ID.
`output_folder`	Character string. Output folder.
`seed`	Integer. Random seed.
`suffix`	Character string. Output of `manCTMed:::.SimSuffix()`.
`overwrite`	Logical. Overwrite existing output in `output_folder`.
`integrity`	Logical. If `integrity = TRUE`, check for the output file integrity when `overwrite = FALSE`.

Details

This function is executed via the `Sim` function.

Value

The output is saved as an external file in `output_folder`.

Author(s)

Ivan Jacob Agaloos Pesigan

`SimFN` *Simulation File Name*

Description

Simulation File Name

Usage

```
SimFN(output_type, output_folder, suffix)
```

Arguments

`output_type`	Character string. Output type.
`output_folder`	Character string. Output folder.
`suffix`	Character string. Output of `manCTMed:::.SimSuffix()`.

Value

Returns a character string file name with the `output_folder` in the OS-specific format.

`SimGenData` *Simulation Replication - GenData*

Description

Simulation Replication - GenData

Usage

```
SimGenData(taskid, repid, output_folder, seed, suffix, overwrite, integrity)
```

Arguments

`taskid`	Positive integer. Task ID.
`repid`	Positive integer. Replication ID.
`output_folder`	Character string. Output folder.
`seed`	Integer. Random seed.
`suffix`	Character string. Output of `manCTMed:::.SimSuffix()`.
`overwrite`	Logical. Overwrite existing output in `output_folder`.
`integrity`	Logical. If `integrity = TRUE`, check for the output file integrity when `overwrite = FALSE`.

Details

This function is executed via the `Sim` function.

Value

The output is saved as an external file in `output_folder`.

Author(s)

Ivan Jacob Agaloos Pesigan

`SimIllustration` *Simulation Replication (Illustration)*

Description

Simulation Replication (Illustration)

Usage

```
SimIllustration(
  taskid,
  repid,
  output_folder,
  overwrite,
  integrity,
  seed,
  ci,
  pb,
  delta_t,
  R,
  B
)
```

Arguments

`taskid`	Positive integer. Task ID.
`repid`	Positive integer. Replication ID.
`output_folder`	Character string. Output folder.
`overwrite`	Logical. Overwrite existing output in `output_folder`.
`integrity`	Logical. If `integrity = TRUE`, check for the output file integrity when `overwrite = FALSE`.
`seed`	Integer. Random seed.
`ci`	Logical. Run simulations for confidence intervals.
`pb`	Logical. Run simulations for parametric bootstrap confidence intervals.

`delta_t`	Numeric vector. Vector of time intervals.
`R`	Positive integer. Number of Monte Carlo replications.
`B`	Positive integer. Number of bootstrap samples.

Value

The output is saved as an external file in `output_folder`.

Author(s)

Ivan Jacob Agaloos Pesigan

`SimIllustrationDynrBootPara`
Simulation Replication - BootPara

Description

Simulation Replication - BootPara

Usage

```
SimIllustrationDynrBootPara(
  taskid,
  repid,
  output_folder,
  seed,
  suffix,
  overwrite,
  integrity,
  B,
  ncores = NULL
)
```

Arguments

`taskid`	Positive integer. Task ID.
`repid`	Positive integer. Replication ID.
`output_folder`	Character string. Output folder.
`seed`	Integer. Random seed.
`suffix`	Character string. Output of `manCTMed:::.SimSuffix()`.
`overwrite`	Logical. Overwrite existing output in `output_folder`.
`integrity`	Logical. If `integrity = TRUE`, check for the output file integrity when `overwrite = FALSE`.
`B`	Positive integer. Number of bootstrap samples.
`ncores`	Positive integer. Number of cores to use.

Details

This function is executed via the `IllustrationSim` function.

Value

The output is saved as an external file in `output_folder`.

Author(s)

Ivan Jacob Agaloos Pesigan

`SimIllustrationDynrBootParaStdXMY`
Simulation Replication - BootParaStdXMY

Description

Simulation Replication - BootParaStdXMY

Usage

```
SimIllustrationDynrBootParaStdXMY(
  taskid,
  repid,
  output_folder,
  seed,
  suffix,
  overwrite,
  integrity,
  delta_t
)
```

Arguments

`taskid`	Positive integer. Task ID.
`repid`	Positive integer. Replication ID.
`output_folder`	Character string. Output folder.
`seed`	Integer. Random seed.
`suffix`	Character string. Output of `manCTMed:::.SimSuffix()`.
`overwrite`	Logical. Overwrite existing output in `output_folder`.
`integrity`	Logical. If `integrity = TRUE`, check for the output file integrity when `overwrite = FALSE`.
`delta_t`	Numeric vector. Vector of time intervals.

Details

This function is executed via the `IllustrationSim` function.

Value

The output is saved as an external file in `output_folder`.

Author(s)

Ivan Jacob Agaloos Pesigan

`SimIllustrationDynrBootParaXMY` *Simulation Replication - BootParaXMY*

Description

Simulation Replication - BootParaXMY

Usage

```
SimIllustrationDynrBootParaXMY(
  taskid,
  repid,
  output_folder,
  seed,
  suffix,
  overwrite,
  integrity,
  delta_t
)
```

Arguments

`taskid`	Positive integer. Task ID.
`repid`	Positive integer. Replication ID.
`output_folder`	Character string. Output folder.
`seed`	Integer. Random seed.
`suffix`	Character string. Output of `manCTMed:::.SimSuffix()`.
`overwrite`	Logical. Overwrite existing output in `output_folder`.
`integrity`	Logical. If `integrity = TRUE`, check for the output file integrity when `overwrite = FALSE`.
`delta_t`	Numeric vector. Vector of time intervals.

Details

This function is executed via the `IllustrationSim` function.

Value

The output is saved as an external file in `output_folder`.

Author(s)

Ivan Jacob Agaloos Pesigan

`SimIllustrationDynrDeltaStdXMY`
Simulation Replication - Illustration (DynrDeltaStdXMY)

Description

Simulation Replication - Illustration (DynrDeltaStdXMY)

Usage

```
SimIllustrationDynrDeltaStdXMY(
  taskid,
  repid,
  output_folder,
  seed,
  suffix,
  overwrite,
  integrity,
  delta_t
)
```

Arguments

`taskid`	Positive integer. Task ID.
`repid`	Positive integer. Replication ID.
`output_folder`	Character string. Output folder.
`seed`	Integer. Random seed.
`suffix`	Character string. Output of `manCTMed:::.SimSuffix()`.
`overwrite`	Logical. Overwrite existing output in `output_folder`.
`integrity`	Logical. If `integrity = TRUE`, check for the output file integrity when `overwrite = FALSE`.
`delta_t`	Numeric vector. Vector of time intervals.

Details

This function is executed via the `IllustrationSim` function.

Value

The output is saved as an external file in `output_folder`.

Author(s)

Ivan Jacob Agaloos Pesigan

`SimIllustrationDynrDeltaXMY` *Simulation Replication - Illustration (DynrDeltaXMY)*

Description

Simulation Replication - Illustration (DynrDeltaXMY)

Usage

```
SimIllustrationDynrDeltaXMY(
  taskid,
  repid,
  output_folder,
  seed,
  suffix,
  overwrite,
  integrity,
  delta_t
)
```

Arguments

`taskid`	Positive integer. Task ID.
`repid`	Positive integer. Replication ID.
`output_folder`	Character string. Output folder.
`seed`	Integer. Random seed.
`suffix`	Character string. Output of `manCTMed:::.SimSuffix()`.
`overwrite`	Logical. Overwrite existing output in `output_folder`.
`integrity`	Logical. If `integrity = TRUE`, check for the output file integrity when `overwrite = FALSE`.
`delta_t`	Numeric vector. Vector of time intervals.

Details

This function is executed via the `IllustrationSim` function.

Value

The output is saved as an external file in `output_folder`.

Author(s)

Ivan Jacob Agaloos Pesigan

`SimIllustrationDynrMCPhiSigma`
Simulation Replication - Illustration (MCPhiSigma)

Description

Simulation Replication - Illustration (MCPhiSigma)

Usage

```
SimIllustrationDynrMCPhiSigma(
  taskid,
  repid,
  output_folder,
  seed,
  suffix,
  overwrite,
  integrity,
  R
)
```

Arguments

`taskid`	Positive integer. Task ID.
`repid`	Positive integer. Replication ID.
`output_folder`	Character string. Output folder.
`seed`	Integer. Random seed.
`suffix`	Character string. Output of `manCTMed:::.SimSuffix()`.
`overwrite`	Logical. Overwrite existing output in `output_folder`.
`integrity`	Logical. If `integrity = TRUE`, check for the output file integrity when `overwrite = FALSE`.
`R`	Positive integer. Number of Monte Carlo replications.

Details

This function is executed via the `IllustrationSim` function.

Value

The output is saved as an external file in `output_folder`.

Author(s)

Ivan Jacob Agaloos Pesigan

`SimIllustrationDynrMCStdXMY` *Simulation Replication - Illustration (DynrMCStdXMY)*

Description

Simulation Replication - Illustration (DynrMCStdXMY)

Usage

```
SimIllustrationDynrMCStdXMY(
  taskid,
  repid,
  output_folder,
  seed,
  suffix,
  overwrite,
  integrity,
  delta_t,
  R
)
```

Arguments

`taskid`	Positive integer. Task ID.
`repid`	Positive integer. Replication ID.
`output_folder`	Character string. Output folder.
`seed`	Integer. Random seed.
`suffix`	Character string. Output of `manCTMed:::.SimSuffix()`.
`overwrite`	Logical. Overwrite existing output in `output_folder`.
`integrity`	Logical. If `integrity = TRUE`, check for the output file integrity when `overwrite = FALSE`.
`delta_t`	Numeric vector. Vector of time intervals.
`R`	Positive integer. Number of Monte Carlo replications.

Details

This function is executed via the `IllustrationSim` function.

Value

The output is saved as an external file in `output_folder`.

Author(s)

Ivan Jacob Agaloos Pesigan

`SimIllustrationDynrMCXMY`
Simulation Replication - Illustration (DynrMCXMY)

Description

Simulation Replication - Illustration (DynrMCXMY)

Usage

```
SimIllustrationDynrMCXMY(
  taskid,
  repid,
  output_folder,
  seed,
  suffix,
  overwrite,
  integrity,
  delta_t,
  R
)
```

Arguments

`taskid`	Positive integer. Task ID.
`repid`	Positive integer. Replication ID.
`output_folder`	Character string. Output folder.
`seed`	Integer. Random seed.
`suffix`	Character string. Output of `manCTMed:::.SimSuffix()`.
`overwrite`	Logical. Overwrite existing output in `output_folder`.
`integrity`	Logical. If `integrity = TRUE`, check for the output file integrity when `overwrite = FALSE`.
`delta_t`	Numeric vector. Vector of time intervals.
`R`	Positive integer. Number of Monte Carlo replications.

Details

This function is executed via the `IllustrationSim` function.

Value

The output is saved as an external file in `output_folder`.

Author(s)

Ivan Jacob Agaloos Pesigan

`SimIllustrationFitDynr` *Simulation Replication - IllustrationFitDynr*

Description

Simulation Replication - IllustrationFitDynr

Usage

```
SimIllustrationFitDynr(
  taskid,
  repid,
  output_folder,
  seed,
  suffix,
  overwrite,
  integrity
)
```

Arguments

`taskid`	Positive integer. Task ID.
`repid`	Positive integer. Replication ID.
`output_folder`	Character string. Output folder.
`seed`	Integer. Random seed.
`suffix`	Character string. Output of `manCTMed:::.SimSuffix()`.
`overwrite`	Logical. Overwrite existing output in `output_folder`.
`integrity`	Logical. If `integrity = TRUE`, check for the output file integrity when `overwrite = FALSE`.

Details

This function is executed via the `IllustrationSim` function.

Value

The output is saved as an external file in `output_folder`.

Author(s)

Ivan Jacob Agaloos Pesigan

`SimIllustrationFitMx` *Simulation Replication - IllustrationFitMx*

Description

Simulation Replication - IllustrationFitMx

Usage

```
SimIllustrationFitMx(
  taskid,
  repid,
  output_folder,
  seed,
  suffix,
  overwrite,
  integrity
)
```

Arguments

`taskid`	Positive integer. Task ID.
`repid`	Positive integer. Replication ID.
`output_folder`	Character string. Output folder.
`seed`	Integer. Random seed.
`suffix`	Character string. Output of `manCTMed:::.SimSuffix()`.
`overwrite`	Logical. Overwrite existing output in `output_folder`.
`integrity`	Logical. If `integrity = TRUE`, check for the output file integrity when `overwrite = FALSE`.

Details

This function is executed via the `IllustrationSim` function.

Value

The output is saved as an external file in `output_folder`.

Author(s)

Ivan Jacob Agaloos Pesigan

`SimIllustrationGenData` *Simulation Replication - IllustrationGenData*

Description

Simulation Replication - IllustrationGenData

Usage

```
SimIllustrationGenData(
  taskid,
  repid,
  output_folder,
  seed,
  suffix,
  overwrite,
  integrity
)
```

Arguments

`taskid`	Positive integer. Task ID.
`repid`	Positive integer. Replication ID.
`output_folder`	Character string. Output folder.
`seed`	Integer. Random seed.
`suffix`	Character string. Output of `manCTMed:::.SimSuffix()`.
`overwrite`	Logical. Overwrite existing output in `output_folder`.
`integrity`	Logical. If `integrity = TRUE`, check for the output file integrity when `overwrite = FALSE`.

Details

This function is executed via the `IllustrationSim` function.

Value

The output is saved as an external file in `output_folder`.

Author(s)

Ivan Jacob Agaloos Pesigan

`SimIllustrationPara` *Simulation Replication Parametric Bootstrap (Parallel)*

Description

Simulation Replication Parametric Bootstrap (Parallel)

Usage

```
SimIllustrationPara(
  taskid,
  repid,
  output_folder,
  overwrite,
  integrity,
  seed,
  B
)
```

Arguments

`taskid`	Positive integer. Task ID.
`repid`	Positive integer. Replication ID.
`output_folder`	Character string. Output folder.
`overwrite`	Logical. Overwrite existing output in `output_folder`.
`integrity`	Logical. If `integrity = TRUE`, check for the output file integrity when `overwrite = FALSE`.
`seed`	Integer. Random seed.
`B`	Positive integer. Number of bootstrap samples.

Value

The output is saved as an external file in `output_folder`.

Author(s)

Ivan Jacob Agaloos Pesigan

`SimPara` *Simulation Replication Parametric Bootstrap (Parallel)*

Description

Simulation Replication Parametric Bootstrap (Parallel)

Usage

```
SimPara(taskid, repid, output_folder, overwrite, integrity, seed, B)
```

Arguments

`taskid`	Positive integer. Task ID.
`repid`	Positive integer. Replication ID.
`output_folder`	Character string. Output folder.
`overwrite`	Logical. Overwrite existing output in `output_folder`.
`integrity`	Logical. If `integrity = TRUE`, check for the output file integrity when `overwrite = FALSE`.
`seed`	Integer. Random seed.
`B`	Positive integer. Number of bootstrap samples.

Value

The output is saved as an external file in `output_folder`.

Author(s)

Ivan Jacob Agaloos Pesigan

`SimProj` *Simulation Project Name*

Description

Simulation Project Name

Usage

```
SimProj()
```

Value

Returns the project name as a character string.

Author(s)

Ivan Jacob Agaloos Pesigan

`Sum` *Summary*

Description

Summary

Usage

```
Sum(taskid, reps, output_folder, overwrite, integrity)
```

Arguments

`taskid`	Positive integer. Task ID.
`reps`	Positive integer. Number of replications.
`output_folder`	Character string. Output folder.
`overwrite`	Logical. Overwrite existing output in `output_folder`.
`integrity`	Logical. If `integrity = TRUE`, check for the output file integrity when `overwrite = FALSE`.

Value

The output is saved as an external file in `output_folder`.

Author(s)

Ivan Jacob Agaloos Pesigan

`SumDynrDeltaStdXMY` *Summary (DynrDeltaStdXMY)*

Description

Summary (DynrDeltaStdXMY)

Usage

```
SumDynrDeltaStdXMY(taskid, reps, output_folder, overwrite, integrity)
```

Arguments

`taskid`	Positive integer. Task ID.
`reps`	Positive integer. Number of replications.
`output_folder`	Character string. Output folder.
`overwrite`	Logical. Overwrite existing output in `output_folder`.
`integrity`	Logical. If `integrity = TRUE`, check for the output file integrity when `overwrite = FALSE`.

Details

This function is executed via the `Sum` function.

Value

The output is saved as an external file in `output_folder`.

Author(s)

Ivan Jacob Agaloos Pesigan

`SumDynrDeltaStdYMX` *Summary (DynrDeltaStdYMX)*

Description

Summary (DynrDeltaStdYMX)

Usage

```
SumDynrDeltaStdYMX(taskid, reps, output_folder, overwrite, integrity)
```

Arguments

`taskid`	Positive integer. Task ID.
`reps`	Positive integer. Number of replications.
`output_folder`	Character string. Output folder.
`overwrite`	Logical. Overwrite existing output in `output_folder`.
`integrity`	Logical. If `integrity = TRUE`, check for the output file integrity when `overwrite = FALSE`.

Details

This function is executed via the `Sum` function.

Value

The output is saved as an external file in `output_folder`.

Author(s)

Ivan Jacob Agaloos Pesigan

`SumDynrDeltaXMY` *Summary (DynrDeltaXMY)*

Description

Summary (DynrDeltaXMY)

Usage

```
SumDynrDeltaXMY(taskid, reps, output_folder, overwrite, integrity)
```

Arguments

`taskid`	Positive integer. Task ID.
`reps`	Positive integer. Number of replications.
`output_folder`	Character string. Output folder.
`overwrite`	Logical. Overwrite existing output in `output_folder`.
`integrity`	Logical. If `integrity = TRUE`, check for the output file integrity when `overwrite = FALSE`.

Details

This function is executed via the `Sum` function.

Value

The output is saved as an external file in `output_folder`.

Author(s)

Ivan Jacob Agaloos Pesigan

SumDynrDeltaYMX *Summary (DynrDeltaYMX)*

Description

Summary (DynrDeltaYMX)

Usage

```
SumDynrDeltaYMX(taskid, reps, output_folder, overwrite, integrity)
```

Arguments

`taskid`	Positive integer. Task ID.
`reps`	Positive integer. Number of replications.
`output_folder`	Character string. Output folder.
`overwrite`	Logical. Overwrite existing output in `output_folder`.
`integrity`	Logical. If `integrity = TRUE`, check for the output file integrity when `overwrite = FALSE`.

Details

This function is executed via the `Sum` function.

Value

The output is saved as an external file in `output_folder`.

Author(s)

Ivan Jacob Agaloos Pesigan

SumDynrMCStdXMY *Summary (DynrMCStdXMY)*

Description

Summary (DynrMCStdXMY)

Usage

```
SumDynrMCStdXMY(taskid, reps, output_folder, overwrite, integrity)
```

Arguments

`taskid`	Positive integer. Task ID.
`reps`	Positive integer. Number of replications.
`output_folder`	Character string. Output folder.
`overwrite`	Logical. Overwrite existing output in `output_folder`.
`integrity`	Logical. If `integrity = TRUE`, check for the output file integrity when `overwrite = FALSE`.

Details

This function is executed via the `Sum` function.

Value

The output is saved as an external file in `output_folder`.

Author(s)

Ivan Jacob Agaloos Pesigan

`SumDynrMCStdYMX` *Summary (DynrMCStdYMX)*

Description

Summary (DynrMCStdYMX)

Usage

```
SumDynrMCStdYMX(taskid, reps, output_folder, overwrite, integrity)
```

Arguments

`taskid`	Positive integer. Task ID.
`reps`	Positive integer. Number of replications.
`output_folder`	Character string. Output folder.
`overwrite`	Logical. Overwrite existing output in `output_folder`.
`integrity`	Logical. If `integrity = TRUE`, check for the output file integrity when `overwrite = FALSE`.

Details

This function is executed via the `Sum` function.

Value

The output is saved as an external file in `output_folder`.

Author(s)

Ivan Jacob Agaloos Pesigan

`SumDynrMCXMY` *Summary (DynrMCXMY)*

Description

Summary (DynrMCXMY)

Usage

```
SumDynrMCXMY(taskid, reps, output_folder, overwrite, integrity)
```

Arguments

`taskid`	Positive integer. Task ID.
`reps`	Positive integer. Number of replications.
`output_folder`	Character string. Output folder.
`overwrite`	Logical. Overwrite existing output in `output_folder`.
`integrity`	Logical. If `integrity = TRUE`, check for the output file integrity when `overwrite = FALSE`.

Details

This function is executed via the `Sum` function.

Value

The output is saved as an external file in `output_folder`.

Author(s)

Ivan Jacob Agaloos Pesigan

`SumDynrMCYMX` *Summary (DynrMCYMX)*

Description

Summary (DynrMCYMX)

Usage

```
SumDynrMCYMX(taskid, reps, output_folder, overwrite, integrity)
```

Arguments

`taskid`	Positive integer. Task ID.
`reps`	Positive integer. Number of replications.
`output_folder`	Character string. Output folder.
`overwrite`	Logical. Overwrite existing output in `output_folder`.
`integrity`	Logical. If `integrity = TRUE`, check for the output file integrity when `overwrite = FALSE`.

Details

This function is executed via the `Sum` function.

Value

The output is saved as an external file in `output_folder`.

Author(s)

Ivan Jacob Agaloos Pesigan

`SumFitDynr` *Summary (FitDynr)*

Description

Summary (FitDynr)

Usage

```
SumFitDynr(taskid, reps, output_folder, overwrite, integrity)
```

Arguments

`taskid`	Positive integer. Task ID.
`reps`	Positive integer. Number of replications.
`output_folder`	Character string. Output folder.
`overwrite`	Logical. Overwrite existing output in `output_folder`.
`integrity`	Logical. If `integrity = TRUE`, check for the output file integrity when `overwrite = FALSE`.

Details

This function is executed via the `Sum` function.

Value

The output is saved as an external file in `output_folder`.

Author(s)

Ivan Jacob Agaloos Pesigan

`SumIllustration` *Summary (Illustration)*

Description

Summary (Illustration)

Usage

```
SumIllustration(taskid, reps, output_folder, overwrite, integrity)
```

Arguments

`taskid`	Positive integer. Task ID.
`reps`	Positive integer. Number of replications.
`output_folder`	Character string. Output folder.
`overwrite`	Logical. Overwrite existing output in `output_folder`.
`integrity`	Logical. If `integrity = TRUE`, check for the output file integrity when `overwrite = FALSE`.

Value

The output is saved as an external file in `output_folder`.

Author(s)

Ivan Jacob Agaloos Pesigan

SumIllustrationDynrBootParaStdXMY

Summary - Illustration (DynrBootParaStdXMY)

Description

Summary - Illustration (DynrBootParaStdXMY)

Usage

```
SumIllustrationDynrBootParaStdXMY(
  taskid,
  reps,
  output_folder,
  overwrite,
  integrity,
  type = "pc"
)
```

Arguments

`taskid`	Positive integer. Task ID.
`reps`	Positive integer. Number of replications.
`output_folder`	Character string. Output folder.
`overwrite`	Logical. Overwrite existing output in `output_folder`.
`integrity`	Logical. If `integrity = TRUE`, check for the output file integrity when `overwrite = FALSE`.
`type`	Character string. Confidence interval type.

Details

This function is executed via the `IllustrationSum` function.

Value

The output is saved as an external file in `output_folder`.

Author(s)

Ivan Jacob Agaloos Pesigan

SumIllustrationDynrBootParaXMY
Summary - Illustration (DynrBootParaXMY)

Description

Summary - Illustration (DynrBootParaXMY)

Usage

```
SumIllustrationDynrBootParaXMY(
  taskid,
  reps,
  output_folder,
  overwrite,
  integrity,
  type = "pc"
)
```

Arguments

`taskid`	Positive integer. Task ID.
`reps`	Positive integer. Number of replications.
`output_folder`	Character string. Output folder.
`overwrite`	Logical. Overwrite existing output in `output_folder`.
`integrity`	Logical. If `integrity = TRUE`, check for the output file integrity when `overwrite = FALSE`.
`type`	Character string. Confidence interval type.

Details

This function is executed via the `IllustrationSum` function.

Value

The output is saved as an external file in `output_folder`.

Author(s)

Ivan Jacob Agaloos Pesigan

SumIllustrationDynrDeltaStdXMY

Summary - Illustration (DynrDeltaStdXMY)

Description

Summary - Illustration (DynrDeltaStdXMY)

Usage

```
SumIllustrationDynrDeltaStdXMY(
  taskid,
  reps,
  output_folder,
  overwrite,
  integrity
)
```

Arguments

`taskid`	Positive integer. Task ID.
`reps`	Positive integer. Number of replications.
`output_folder`	Character string. Output folder.
`overwrite`	Logical. Overwrite existing output in `output_folder`.
`integrity`	Logical. If `integrity = TRUE`, check for the output file integrity when `overwrite = FALSE`.

Details

This function is executed via the `IllustrationSum` function.

Value

The output is saved as an external file in `output_folder`.

Author(s)

Ivan Jacob Agaloos Pesigan

SumIllustrationDynrDeltaXMY
Summary - Illustration (DynrDeltaXMY)

Description

Summary - Illustration (DynrDeltaXMY)

Usage

```
SumIllustrationDynrDeltaXMY(taskid, reps, output_folder, overwrite, integrity)
```

Arguments

`taskid`	Positive integer. Task ID.
`reps`	Positive integer. Number of replications.
`output_folder`	Character string. Output folder.
`overwrite`	Logical. Overwrite existing output in `output_folder`.
`integrity`	Logical. If `integrity = TRUE`, check for the output file integrity when `overwrite = FALSE`.

Details

This function is executed via the `IllustrationSum` function.

Value

The output is saved as an external file in `output_folder`.

Author(s)

Ivan Jacob Agaloos Pesigan

SumIllustrationDynrMCStdXMY
Summary - Illustration (DynrMCStdXMY)

Description

Summary - Illustration (DynrMCStdXMY)

Usage

```
SumIllustrationDynrMCStdXMY(taskid, reps, output_folder, overwrite, integrity)
```

Arguments

`taskid`	Positive integer. Task ID.
`reps`	Positive integer. Number of replications.
`output_folder`	Character string. Output folder.
`overwrite`	Logical. Overwrite existing output in `output_folder`.
`integrity`	Logical. If `integrity = TRUE`, check for the output file integrity when `overwrite = FALSE`.

Details

This function is executed via the `IllustrationSum` function.

Value

The output is saved as an external file in `output_folder`.

Author(s)

Ivan Jacob Agaloos Pesigan

`SumIllustrationDynrMCXMY` *Summary - Illustration (DynrMCXMY)*

Description

Summary - Illustration (DynrMCXMY)

Usage

```
SumIllustrationDynrMCXMY(taskid, reps, output_folder, overwrite, integrity)
```

Arguments

`taskid`	Positive integer. Task ID.
`reps`	Positive integer. Number of replications.
`output_folder`	Character string. Output folder.
`overwrite`	Logical. Overwrite existing output in `output_folder`.
`integrity`	Logical. If `integrity = TRUE`, check for the output file integrity when `overwrite = FALSE`.

Details

This function is executed via the `IllustrationSum` function.

Value

The output is saved as an external file in `output_folder`.

Author(s)

Ivan Jacob Agaloos Pesigan

`SumIllustrationFitDynr` *Summary - Illustration (FitDynr)*

Description

Summary - Illustration (FitDynr)

Usage

```
SumIllustrationFitDynr(taskid, reps, output_folder, overwrite, integrity)
```

Arguments

`taskid`	Positive integer. Task ID.
`reps`	Positive integer. Number of replications.
`output_folder`	Character string. Output folder.
`overwrite`	Logical. Overwrite existing output in `output_folder`.
`integrity`	Logical. If `integrity = TRUE`, check for the output file integrity when `overwrite = FALSE`.

Details

This function is executed via the `IllustrationSum` function.

Value

The output is saved as an external file in `output_folder`.

Author(s)

Ivan Jacob Agaloos Pesigan

ThetaHat — *Estimated Drift Matrix and Process Noise*

Description

The function extracts the estimated drift matrix and process noise from the fitted model.

Usage

```
ThetaHat(fit)
```

Arguments

`fit`	R object. Output of the `FitDynr()`, `FitMx()`, `IllustrationFitDynr()`, or `IllustrationFitMx()`, functions.

See Also

Other Model Fitting Functions: `FitDynr()`, `FitMx()`, `IllustrationFitDynr()`, `IllustrationFitMx()`, `IllustrationMCPhiSigma()`, `IllustrationPrepData()`, `PhiHat()`

Examples

```
## Not run:
set.seed(42)
library(dynr)
sim <- GenData(n = 50)
data <- RandomMeasurement(sim)
fit <- FitDynr(data)
ThetaHat(fit)

## End(Not run)
```

Index