

# 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 . . . . .	4
BootParaStdXMY . . . . .	5
BootParaStdXYM . . . . .	6
BootParaStdYMX . . . . .	7
BootParaXMY . . . . .	8
BootParaXYM . . . . .	9
BootParaYMX . . . . .	10
Compress . . . . .	11
DeltaStdXMY . . . . .	12
DeltaStdXYM . . . . .	12
DeltaStdYMX . . . . .	13
DeltaXMY . . . . .	14
DeltaXYM . . . . .	15
DeltaYMX . . . . .	16
FigPlotEffects . . . . .	16
FigScatterPlotCoverage . . . . .	17
FigScatterPlotPower . . . . .	18
FigScatterPlotSeBias . . . . .	19
FigScatterPlotType1 . . . . .	20
FitDy whole . . . . .	21
FitMx . . . . .	21
GenData . . . . .	22
IllustrationBootPara . . . . .	23
IllustrationCompress . . . . .	24
IllustrationFigPlotEffects . . . . .	25
IllustrationFigScatterPlotCoverage . . . . .	25
IllustrationFigScatterPlotPower . . . . .	26
IllustrationFigScatterPlotSeBias . . . . .	27
IllustrationFitDy whole . . . . .	28
IllustrationFitMx . . . . .	28
IllustrationGenData . . . . .	29
IllustrationMCPhiSigma . . . . .	30
IllustrationPrepData . . . . .	30
illustration_dist . . . . .	31
illustration_dist_dt . . . . .	32
illustration_dist_dt_mc . . . . .	32
illustration_dist_mc . . . . .	33
illustration_dist_med . . . . .	33
illustration_dist_med_mc . . . . .	34
illustration_dist_med_std . . . . .	34
illustration_dist_med_std_mc . . . . .	35
illustration_results . . . . .	35
MCStdXMY . . . . .	37
MCStdXYM . . . . .	38
MCStdYMX . . . . .	39
MCXMY . . . . .	40

MCXYM . . . . .	41
MCYMX . . . . .	42
params . . . . .	43
PhiHat . . . . .	43
RandomMeasurement . . . . .	44
results . . . . .	45
Sim . . . . .	46
SimDy whole BootPara . . . . .	47
SimDy whole BootParaStdXMY . . . . .	48
SimDy whole BootParaStdYMX . . . . .	49
SimDy whole BootParaXMY . . . . .	50
SimDy whole BootParaYMX . . . . .	51
SimDy whole DeltaStdXMY . . . . .	52
SimDy whole DeltaStdYMX . . . . .	53
SimDy whole DeltaXMY . . . . .	54
SimDy whole DeltaYMX . . . . .	55
SimDy whole MCStdXMY . . . . .	56
SimDy whole MCStdYMX . . . . .	57
SimDy whole MCXMY . . . . .	58
SimDy whole MCYMX . . . . .	59
SimFitDy whole r . . . . .	60
SimFitMx . . . . .	61
SimFN . . . . .	62
SimGenData . . . . .	62
SimIllustration . . . . .	63
SimIllustrationDy whole rBootPara . . . . .	64
SimIllustrationDy whole rBootParaStdXMY . . . . .	65
SimIllustrationDy whole rBootParaXMY . . . . .	66
SimIllustrationDy whole rDeltaStdXMY . . . . .	67
SimIllustrationDy whole rDeltaXMY . . . . .	68
SimIllustrationDy whole rMCPhiSigma . . . . .	69
SimIllustrationDy whole rMCStdXMY . . . . .	70
SimIllustrationDy whole rMCXMY . . . . .	71
SimIllustrationFitDy whole r . . . . .	72
SimIllustrationFitMx . . . . .	73
SimIllustrationGenData . . . . .	74
SimIllustrationPara . . . . .	75
SimPara . . . . .	76
SimProj . . . . .	76
Sum . . . . .	77
SumDy whole rDeltaStdXMY . . . . .	77
SumDy whole rDeltaStdYMX . . . . .	78
SumDy whole rDeltaXMY . . . . .	79
SumDy whole rDeltaYMX . . . . .	80
SumDy whole rMCStdXMY . . . . .	80
SumDy whole rMCStdYMX . . . . .	81
SumDy whole rMCXMY . . . . .	82
SumDy whole rMCYMX . . . . .	83

SumFitDynr . . . . .	83
SumIllustration . . . . .	84
SumIllustrationDynrBootParaStdXMY . . . . .	85
SumIllustrationDynrBootParaXMY . . . . .	86
SumIllustrationDynrDeltaStdXMY . . . . .	87
SumIllustrationDynrDeltaXMY . . . . .	88
SumIllustrationDynrMCStdXMY . . . . .	88
SumIllustrationDynrMCXMY . . . . .	89
SumIllustrationFitDynr . . . . .	90
ThetaHat . . . . .	91
<b>Index</b>	<b>92</b>

---

BootPara	<i>Parametric Bootstrap</i>
----------	-----------------------------

---

## 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

<code>fit</code>	R object. Output of the <code>FitDynr()</code> , <code>FitMx()</code> , <code>IllustrationFitDynr()</code> , or <code>IllustrationFitMx()</code> , functions.
<code>path</code>	Path to a directory to store bootstrap samples and estimates.
<code>prefix</code>	Character string. Prefix used for the file names for the bootstrap samples and estimates.
<code>taskid</code>	Positive integer. Task ID.
<code>B</code>	Positive integer. Number of bootstrap samples.
<code>ncores</code>	Positive integer. Number of cores to use.
<code>seed</code>	Integer. Random seed.
<code>clean</code>	Logical. If <code>clean = TRUE</code> , 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(dynrr)
sim <- GenData(taskid = 1)
data <- RandomMeasurement(sim)
fit <- FitDy whole(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  $X \rightarrow M \rightarrow Y$  (Standardized).

**Usage**

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

**Arguments**

boot	R object. Output of the <a href="#">BootPara()</a> function.
theta_hat	R object. Output of the <a href="#">ThetaHat()</a> 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 <- BootParaStdXYM(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  $X \rightarrow Y \rightarrow M$  (Standardized).

## Usage

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

## Arguments

boot	R object. Output of the <a href="#">BootPara()</a> function.
theta_hat	R object. Output of the <a href="#">ThetaHat()</a> 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  $Y \rightarrow M \rightarrow X$  (Standardized).

## Usage

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

## Arguments

boot	R object. Output of the <a href="#">BootPara()</a> function.
theta_hat	R object. Output of the <a href="#">ThetaHat()</a> 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  $X \rightarrow M \rightarrow Y$ .

## Usage

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

## Arguments

boot	R object. Output of the <a href="#">BootPara()</a> function.
phi_hat	R object. Output of the <a href="#">PhiHat()</a> 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 <- BootParaXYM(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 <a href="#">BootPara()</a> function.
phi_hat	R object. Output of the <a href="#">PhiHat()</a> 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 <- BootParaYMX(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  $Y \rightarrow M \rightarrow X$ .

## Usage

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

## Arguments

boot	R object. Output of the <a href="#">BootPara()</a> function.
phi_hat	R object. Output of the <a href="#">PhiHat()</a> 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\(\)](#), [BootParaXYM\(\)](#), [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)  
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\(\)](#)

DeltaStdXYM

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

## Description

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

## Usage

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

## Arguments

- |           |  |
|-----------|--|
| theta_hat | R object. Output of the <a href="#">ThetaHat()</a> function. |
| delta_t   | Numeric vector. Vector of time intervals.                    |

## See Also

Other Confidence Interval Functions: [BootPara\(\)](#), [BootParaStdXYM\(\)](#), [BootParaStdXYM\(\)](#), [BootParaStdYMX\(\)](#), [BootParaXMY\(\)](#), [BootParaXYM\(\)](#), [BootParaYMX\(\)](#), [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 <- FitDydnr(data, taskid = 1)
theta_hat <- ThetaHat(fit)
ci <- DeltaStdXYM(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  $X \rightarrow Y \rightarrow M$  (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 <- FitDy whole(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  $Y \rightarrow M \rightarrow X$  (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\(\)](#), [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)
theta_hat <- ThetaHat(fit)
ci <- DeltaStdYMX(theta_hat)
plot(ci)

## End(Not run)
```

**Description**

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

**Usage**

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

**Arguments**

- |         |  |
|---------|--|
| phi_hat | R object. Output of the <a href="#">PhiHat()</a> 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)
```

DeltaXYM

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

## Description

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

## Usage

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

## Arguments

- |         |  |
|---------|--|
| phi_hat | R object. Output of the <a href="#">PhiHat()</a> 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 <- 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 <a href="#">PhiHat()</a> 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 $x \rightarrow m \rightarrow y$ mediation model. If xmy = FALSE, plot the effects for the $y \rightarrow m \rightarrow x'$ 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  $X \rightarrow M \rightarrow Y$ .

### 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  $X \rightarrow M \rightarrow Y$ .

**Usage**

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

**Arguments**

results	Summary results data frame.
delta_t	Vector of time-interval value. If <code>delta_t = NULL</code> , use all available time-intervals
dynamics	Integer. <code>dynamics = 0</code> for original drift matrix, <code>dynamics = -1</code> for near-neutral dynamics, and <code>dynamics = 1</code> for stronger damping.
std	Logical. If <code>std = TRUE</code> , standardized total, direct, and indirect effects. If <code>std = FALSE</code> , 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  $X \rightarrow M \rightarrow Y$ .

## Usage

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

## Arguments

results	Summary results data frame.
delta_t	Vector of time-interval value. If <code>delta_t = NULL</code> , use all available time-intervals
dynamics	Integer. <code>dynamics = 0</code> for original drift matrix, <code>dynamics = -1</code> for near-neutral dynamics, and <code>dynamics = 1</code> for stronger damping.
std	Logical. If <code>std = TRUE</code> , standardized total, direct, and indirect effects. If <code>std = FALSE</code> , 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  $Y \rightarrow M \rightarrow X$ .

### Usage

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

### Arguments

<code>results</code>	Summary results data frame.
<code>delta_t</code>	Vector of time-interval value. If <code>delta_t</code> = <code>NULL</code> , use all available time-intervals
<code>dynamics</code>	Integer. <code>dynamics</code> = 0 for original drift matrix, <code>dynamics</code> = -1 for near-neutral dynamics, and <code>dynamics</code> = 1 for stronger damping.
<code>std</code>	Logical. If <code>std</code> = <code>TRUE</code> , standardized total, direct, and indirect effects. If <code>std</code> = <code>FALSE</code> , 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 <a href="#">RandomMeasurement()</a> 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

<code>fit</code>	R object. Fitted CT-VAR model.
<code>path</code>	Path to a directory to store bootstrap samples and estimates.
<code>prefix</code>	Character string. Prefix used for the file names for the bootstrap samples and estimates.
<code>taskid</code>	Positive integer. Task ID.
<code>B</code>	Positive integer. Number of bootstrap samples.
<code>ncores</code>	Positive integer. Number of cores to use.
<code>seed</code>	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\(\)](#), [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  $X \rightarrow M \rightarrow Y$ .

**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  $X \rightarrow M \rightarrow Y$ .

**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\(\)](#), [IllustrationFigScatterPlot\(\)](#),  
[IllustrationFigScatterPlotSeBias\(\)](#)

**Examples**

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

**IllustrationFigScatterPlotPower**  
*Illustration Plot Statistical Power*

**Description**

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

**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\(\)](#), [IllustrationFigScatterPlot\(\)](#),  
[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\(\)](#), [IllustrationFigScatterPlot\(\)](#),  
[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 <a href="#">IllustrationPrepData()</a> 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 <a href="#">IllustrationPrepData()</a> 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 the 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

## Description

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

## Usage

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

## Arguments

theta_hat	R object. Output of the <a href="#">ThetaHat()</a> 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 <- FitDyN(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 <a href="#">ThetaHat()</a> 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 <- FitDyN(data, taskid = 1)
theta_hat <- ThetaHat(fit)
ci <- MCStdXYM(theta_hat, seed = 42)
plot(ci)

## End(Not run)
```

## Description

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

## Usage

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

## Arguments

theta_hat	R object. Output of the <a href="#">ThetaHat()</a> 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 <- FitDyN(data, taskid = 1)
theta_hat <- ThetaHat(fit)
ci <- MCStdYMX(phi_hat, seed = 42)
plot(ci)

## End(Not run)
```

**Description**

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

**Usage**

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

**Arguments**

phi_hat	R object. Output of the <a href="#">PhiHat()</a> 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 <- FitDyN(data, taskid = 1)
phi_hat <- PhiHat(fit)
ci <- MCXMY(phi_hat, seed = 42)
plot(ci)

## End(Not run)
```

## Description

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

## Usage

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

## Arguments

phi_hat	R object. Output of the <a href="#">PhiHat()</a> 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 <- FitDy whole(data, taskid = 1)
phi_hat <- PhiHat(fit)
ci <- MCXYM(phi_hat, seed = 42)
plot(ci)

## End(Not run)
```

## 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 <a href="#">PhiHat()</a> 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 <- FitDyN(data, taskid = 1)
phi_hat <- PhiHat(fit)
ci <- MCYMX(phi_hat, seed = 42)
plot(ci)

## End(Not run)
```

---

params	<i>Simulation Parameters</i>
--------	------------------------------

---

### 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	<i>Estimated Drift Matrix</i>
--------	-------------------------------

---

### 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	<i>Simulation Results</i>
---------	---------------------------

---

## 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  $X \rightarrow M \rightarrow Y$ . If FALSE, the mediation model is  $Y \rightarrow M \rightarrow X$ .
- 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 <code>manCTMed:::SimSuffix()</code> .
overwrite	Logical. Overwrite existing output in <code>output_folder</code> .
integrity	Logical. If <code>integrity = TRUE</code> , check for the output file integrity when <code>overwrite = FALSE</code> .
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 <code>manCTMed:::SimSuffix()</code> .
overwrite	Logical. Overwrite existing output in <code>output_folder</code> .
integrity	Logical. If <code>integrity = TRUE</code> , check for the output file integrity when <code>overwrite = FALSE</code> .
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 <code>manCTMed:::SimSuffix()</code> .
overwrite	Logical. Overwrite existing output in <code>output_folder</code> .
integrity	Logical. If <code>integrity = TRUE</code> , check for the output file integrity when <code>overwrite = FALSE</code> .
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 <code>manCTMed:::SimSuffix()</code> .
overwrite	Logical. Overwrite existing output in <code>output_folder</code> .
integrity	Logical. If <code>integrity = TRUE</code> , check for the output file integrity when <code>overwrite = FALSE</code> .
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 <code>manCTMed:::SimSuffix()</code> .
overwrite	Logical. Overwrite existing output in <code>output_folder</code> .
integrity	Logical. If <code>integrity = TRUE</code> , check for the output file integrity when <code>overwrite = FALSE</code> .
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 <code>manCTMed:::SimSuffix()</code> .
overwrite	Logical. Overwrite existing output in <code>output_folder</code> .
integrity	Logical. If <code>integrity = TRUE</code> , check for the output file integrity when <code>overwrite = FALSE</code> .
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 <code>manCTMed:::SimSuffix()</code> .
overwrite	Logical. Overwrite existing output in <code>output_folder</code> .
integrity	Logical. If <code>integrity = TRUE</code> , check for the output file integrity when <code>overwrite = FALSE</code> .
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 <code>manCTMed:::SimSuffix()</code> .
overwrite	Logical. Overwrite existing output in <code>output_folder</code> .
integrity	Logical. If <code>integrity = TRUE</code> , check for the output file integrity when <code>overwrite = FALSE</code> .
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 <code>manCTMed:::SimSuffix()</code> .
overwrite	Logical. Overwrite existing output in <code>output_folder</code> .
integrity	Logical. If <code>integrity = TRUE</code> , check for the output file integrity when <code>overwrite = FALSE</code> .
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 <code>manCTMed:::SimSuffix()</code> .
overwrite	Logical. Overwrite existing output in <code>output_folder</code> .
integrity	Logical. If <code>integrity = TRUE</code> , check for the output file integrity when <code>overwrite = FALSE</code> .

## 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 <code>manCTMed:::SimSuffix()</code> .
overwrite	Logical. Overwrite existing output in <code>output_folder</code> .
integrity	Logical. If <code>integrity = TRUE</code> , check for the output file integrity when <code>overwrite = FALSE</code> .

## 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	<i>Simulation File Name</i>
-------	-----------------------------

**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	<i>Simulation Replication - GenData</i>
------------	---

**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 <code>output_folder</code> .  |
| integrity     | Logical. If <code>integrity</code> = TRUE, check for the output file integrity when <code>overwrite</code> = 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**

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

<code>delta_t</code>	Numeric vector. Vector of time intervals.
<code>R</code>	Positive integer. Number of Monte Carlo replications.
<code>B</code>	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**

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

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

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

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

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

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

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

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

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

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

**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 <code>manCTMed:::SimSuffix()</code> .
overwrite	Logical. Overwrite existing output in <code>output_folder</code> .
integrity	Logical. If <code>integrity = TRUE</code> , check for the output file integrity when <code>overwrite = FALSE</code> .

**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**

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

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

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

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

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

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

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

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

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

<code>taskid</code>	Positive integer. Task ID.
<code>reps</code>	Positive integer. Number of replications.
<code>output_folder</code>	Character string. Output folder.
<code>overwrite</code>	Logical. Overwrite existing output in <code>output_folder</code> .
<code>integrity</code>	Logical. If <code>integrity</code> = TRUE, check for the output file integrity when <code>overwrite</code> = 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	<i>Estimated Drift Matrix and Process Noise</i>
----------	---

---

## 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

- \* **Compression Functions**
  - Compress, 11
  - IllustrationCompress, 24
- \* **Confidence Interval Functions**
  - BootPara, 4
  - BootParaStdXMY, 5
  - BootParaStdXYM, 6
  - BootParaStdYMX, 7
  - BootParaXMY, 8
  - BootParaXYM, 9
  - BootParaYMX, 10
  - DeltaStdXMY, 12
  - DeltaStdXYM, 12
  - DeltaStdYMX, 13
  - DeltaXMY, 14
  - DeltaXYM, 15
  - DeltaYMX, 16
  - IllustrationBootPara, 23
  - MCStdXMY, 37
  - MCStdXYM, 38
  - MCStdYMX, 39
  - MCXMY, 40
  - MCXYM, 41
  - MCYMX, 42
- \* **Data Generation Functions**
  - GenData, 22
  - IllustrationGenData, 29
  - RandomMeasurement, 44
- \* **Figure Functions**
  - FigPlotEffects, 16
  - FigScatterPlotCoverage, 17
  - FigScatterPlotPower, 18
  - FigScatterPlotSeBias, 19
  - FigScatterPlotType1, 20
  - IllustrationFigPlotEffects, 25
  - IllustrationFigScatterPlotCoverage, 25
  - IllustrationFigScatterPlotPower, 26
  - IllustrationFigScatterPlotSeBias, 27
- \* **Model Fitting Functions**
  - FitDynr, 21
  - FitMx, 21
  - IllustrationFitDynr, 28
  - IllustrationFitMx, 28
  - IllustrationMCPhiSigma, 30
  - IllustrationPrepData, 30
  - PhiHat, 43
  - ThetaHat, 91
- \* **ci**
  - BootPara, 4
  - BootParaStdXMY, 5
  - BootParaStdXYM, 6
  - BootParaStdYMX, 7
  - BootParaXMY, 8
  - BootParaXYM, 9
  - BootParaYMX, 10
  - DeltaStdXMY, 12
  - DeltaStdXYM, 12
  - DeltaStdYMX, 13
  - DeltaXMY, 14
  - DeltaXYM, 15
  - DeltaYMX, 16
  - IllustrationBootPara, 23
  - MCStdXMY, 37
  - MCStdXYM, 38
  - MCStdYMX, 39
  - MCXMY, 40
  - MCXYM, 41
  - MCYMX, 42
  - PhiHat, 43
  - SimDynrBootPara, 47
  - SimDynrBootParaStdXMY, 48
  - SimDynrBootParaStdYMX, 49
  - SimDynrBootParaXMY, 50
  - SimDynrBootParaYMX, 51

SimDy whole page

- \* **SimDynrDeltaStdXMY**, 52
- \* **SimDynrDeltaStdYMX**, 53
- \* **SimDynrDeltaXMY**, 54
- \* **SimDynrDeltaYMX**, 55
- \* **SimDynrMCStdXMY**, 56
- \* **SimDynrMCStdYMX**, 57
- \* **SimDynrMCXMY**, 58
- \* **SimDynrMCYMX**, 59
- \* **SimIllustrationDynrBootPara**, 64
- \* **SimIllustrationDynrBootParaStdXMY**, 65
- \* **SimIllustrationDynrBootParaXMY**, 66
- \* **SimIllustrationDynrDeltaStdXMY**, 67
- \* **SimIllustrationDynrDeltaXMY**, 68
- \* **SimIllustrationDynrMCPhiSigma**, 69
- \* **SimIllustrationDynrMCStdXMY**, 70
- \* **SimIllustrationDynrMCXMY**, 71
- \* **ThetaHat**, 91

\* **compress**

- \* **Compress**, 11
- \* **IllustrationCompress**, 24

\* **data**

- \* **illustration\_dist**, 31
- \* **illustration\_dist\_dt**, 32
- \* **illustration\_dist\_dt\_mc**, 32
- \* **illustration\_dist\_mc**, 33
- \* **illustration\_dist\_med**, 33
- \* **illustration\_dist\_med\_mc**, 34
- \* **illustration\_dist\_med\_std**, 34
- \* **illustration\_dist\_med\_std\_mc**, 35
- \* **illustration\_results**, 35
- \* **params**, 43
- \* **results**, 45

\* **figure**

- \* **FigPlotEffects**, 16
- \* **FigScatterPlotCoverage**, 17
- \* **FigScatterPlotPower**, 18
- \* **FigScatterPlotSeBias**, 19
- \* **FigScatterPlotType1**, 20
- \* **IllustrationFigPlotEffects**, 25
- \* **IllustrationFigScatterPlotCoverage**, 25
- \* **IllustrationFigScatterPlotPower**, 26
- \* **IllustrationFigScatterPlotSeBias**, 27

\* **fit**

- \* **FitDynr**, 21

\* **FitMx**, 21

\* **IllustrationFitDynr**, 28

\* **IllustrationFitMx**, 28

\* **SimFitDynr**, 60

\* **SimFitMx**, 61

\* **SimIllustrationFitDynr**, 72

\* **SimIllustrationFitMx**, 73

\* **gendata**

- \* **GenData**, 22
- \* **IllustrationGenData**, 29
- \* **IllustrationPrepData**, 30
- \* **RandomMeasurement**, 44
- \* **SimGenData**, 62
- \* **SimIllustrationGenData**, 74

\* **illustration**

- \* **illustration\_dist**, 31
- \* **illustration\_dist\_dt**, 32
- \* **illustration\_dist\_dt\_mc**, 32
- \* **illustration\_dist\_mc**, 33
- \* **illustration\_dist\_med**, 33
- \* **illustration\_dist\_med\_mc**, 34
- \* **illustration\_dist\_med\_std**, 34
- \* **illustration\_dist\_med\_std\_mc**, 35
- \* **illustration\_results**, 35
- \* **IllustrationBootPara**, 23
- \* **IllustrationFigScatterPlotCoverage**, 25
- \* **IllustrationFigScatterPlotPower**, 26
- \* **IllustrationFigScatterPlotSeBias**, 27
- \* **IllustrationFitDynr**, 28
- \* **IllustrationFitMx**, 28
- \* **IllustrationGenData**, 29
- \* **IllustrationMCPhiSigma**, 30
- \* **IllustrationPrepData**, 30
- \* **SimIllustration**, 63
- \* **SimIllustrationDynrBootPara**, 64
- \* **SimIllustrationDynrBootParaStdXMY**, 65
- \* **SimIllustrationDynrBootParaXMY**, 66
- \* **SimIllustrationDynrDeltaStdXMY**, 67
- \* **SimIllustrationDynrDeltaXMY**, 68
- \* **SimIllustrationDynrMCPhiSigma**, 69
- \* **SimIllustrationDynrMCStdXMY**, 70
- \* **SimIllustrationDynrMCXMY**, 71
- \* **SimIllustrationFitDynr**, 72
- \* **SimIllustrationFitMx**, 73

- SimIllustrationGenData, 74
- SimIllustrationPara, 75
- SumIllustration, 84
- SumIllustrationDynrBootParaStdXMY, 85
- SumIllustrationDynrBootParaXMY, 86
- SumIllustrationDynrDeltaStdXMY, 87
- SumIllustrationDynrDeltaXMY, 88
- SumIllustrationDynrMCStdXMY, 88
- SumIllustrationDynrMCXMY, 89
- SumIllustrationFitDynr, 90
- \* **manCTMed**
  - BootPara, 4
  - BootParaStdXMY, 5
  - BootParaStdXYM, 6
  - BootParaStdYMX, 7
  - BootParaXMY, 8
  - BootParaXYM, 9
  - BootParaYMX, 10
  - Compress, 11
  - DeltaStdXMY, 12
  - DeltaStdXYM, 12
  - DeltaStdYMX, 13
  - DeltaXMY, 14
  - DeltaXYM, 15
  - DeltaYMX, 16
  - FigPlotEffects, 16
  - FigScatterPlotCoverage, 17
  - FigScatterPlotPower, 18
  - FigScatterPlotSeBias, 19
  - FigScatterPlotType1, 20
  - FitDynr, 21
  - FitMx, 21
  - GenData, 22
  - IllustrationBootPara, 23
  - IllustrationCompress, 24
  - IllustrationFigPlotEffects, 25
  - IllustrationFigScatterPlotCoverage, 25
  - IllustrationFigScatterPlotPower, 26
  - IllustrationFigScatterPlotSeBias, 27
  - IllustrationFitDynr, 28
  - IllustrationFitMx, 28
  - IllustrationGenData, 29
  - IllustrationMCPhiSigma, 30
  - IllustrationPrepData, 30
  - MCStdXMY, 37
  - MCStdXYM, 38
  - MCStdYMX, 39
  - MCXMY, 40
  - MCXYM, 41
  - MCYMX, 42
  - PhiHat, 43
  - RandomMeasurement, 44
  - Sim, 46
  - SimDynrBootPara, 47
  - SimDynrBootParaStdXMY, 48
  - SimDynrBootParaStdYMX, 49
  - SimDynrBootParaXMY, 50
  - SimDynrBootParaYMX, 51
  - SimDynrDeltaStdXMY, 52
  - SimDynrDeltaStdYMX, 53
  - SimDynrDeltaXMY, 54
  - SimDynrDeltaYMX, 55
  - SimDynrMCStdXMY, 56
  - SimDynrMCStdYMX, 57
  - SimDynrMCXMY, 58
  - SimDynrMCYMX, 59
  - SimFitDynr, 60
  - SimFitMx, 61
  - SimFN, 62
  - SimGenData, 62
  - SimIllustration, 63
  - SimIllustrationDynrBootPara, 64
  - SimIllustrationDynrBootParaStdXMY, 65
  - SimIllustrationDynrBootParaXMY, 66
  - SimIllustrationDynrDeltaStdXMY, 67
  - SimIllustrationDynrDeltaXMY, 68
  - SimIllustrationDynrMCPhiSigma, 69
  - SimIllustrationDynrMCStdXMY, 70
  - SimIllustrationDynrMCXMY, 71
  - SimIllustrationFitDynr, 72
  - SimIllustrationFitMx, 73
  - SimIllustrationGenData, 74
  - SimIllustrationPara, 75
  - SimPara, 76
  - SimProj, 76
  - Sum, 77
  - SumDynrDeltaStdXMY, 77
  - SumDynrDeltaStdYMX, 78
  - SumDynrDeltaXMY, 79
  - SumDynrDeltaYMX, 80
  - SumDynrMCStdXMY, 80

SumDynrMCStdYMX, 81  
 SumDynrMCXMY, 82  
 SumDynrMCYMX, 83  
 SumFitDynr, 83  
 SumIllustration, 84  
 SumIllustrationDynrBootParaStdXMY,  
     85  
 SumIllustrationDynrBootParaXMY, 86  
 SumIllustrationDynrDeltaStdXMY, 87  
 SumIllustrationDynrDeltaXMY, 88  
 SumIllustrationDynrMCStdXMY, 88  
 SumIllustrationDynrMCXMY, 89  
 SumIllustrationFitDynr, 90  
 ThetaHat, 91  
**\* parameters**  
 params, 43  
 results, 45  
**\* simulation**  
 Sim, 46  
 SimDynrBootPara, 47  
 SimDynrBootParaStdXMY, 48  
 SimDynrBootParaStdYMX, 49  
 SimDynrBootParaXMY, 50  
 SimDynrBootParaYMX, 51  
 SimDynrDeltaStdXMY, 52  
 SimDynrDeltaStdYMX, 53  
 SimDynrDeltaXMY, 54  
 SimDynrDeltaYMX, 55  
 SimDynrMCStdXMY, 56  
 SimDynrMCStdYMX, 57  
 SimDynrMCXMY, 58  
 SimDynrMCYMX, 59  
 SimFitDynr, 60  
 SimFitMx, 61  
 SimFN, 62  
 SimGenData, 62  
 SimPara, 76  
 SimProj, 76  
 Sum, 77  
 SumDynrDeltaStdXMY, 77  
 SumDynrDeltaStdYMX, 78  
 SumDynrDeltaXMY, 79  
 SumDynrDeltaYMX, 80  
 SumDynrMCStdXMY, 80  
 SumDynrMCStdYMX, 81  
 SumDynrMCXMY, 82  
 SumDynrMCYMX, 83  
 SumFitDynr, 83  
**\* summary**  
 Sum, 77  
 SumDynrDeltaStdXMY, 77  
 SumDynrDeltaStdYMX, 78  
 SumDynrDeltaXMY, 79  
 SumDynrDeltaYMX, 80  
 SumDynrMCStdXMY, 80  
 SumDynrMCStdYMX, 81  
 SumDynrMCXMY, 82  
 SumDynrMCYMX, 83  
 SumFitDynr, 83  
 SumIllustration, 84  
 SumIllustrationDynrBootParaStdXMY,  
     85  
 SumIllustrationDynrBootParaXMY, 86  
 SumIllustrationDynrDeltaStdXMY, 87  
 SumIllustrationDynrDeltaXMY, 88  
 SumIllustrationDynrMCStdXMY, 88  
 SumIllustrationDynrMCXMY, 89  
 SumIllustrationFitDynr, 90  
 BootPara, 4, 5–10, 12–16, 23, 37–42  
 BootPara(), 5–10  
 BootParaStdXMY, 5, 5–10, 12–16, 23, 37–42  
 BootParaStdXYM, 5, 6, 7–10, 12–16, 23, 37–42  
 BootParaStdYMX, 5, 6, 7, 8–10, 12–16, 23,  
     37–42  
 BootParaXMY, 5–7, 8, 9, 10, 12–16, 23, 37–42  
 BootParaXYM, 5–8, 9, 10, 12–16, 23, 37–42  
 BootParaYMX, 5–9, 10, 12–16, 23, 37–42  
 Compress, 11, 24  
 DeltaStdXMY, 5–10, 12, 13–16, 23, 37–42  
 DeltaStdXYM, 5–10, 12, 12, 14–16, 23, 37–42  
 DeltaStdYMX, 5–10, 12, 13, 13–16, 23, 37–42  
 DeltaXMY, 5–10, 12, 13, 14, 14–16, 23, 37–42  
 DeltaXYM, 5–10, 12–14, 15, 16, 23, 37–42  
 DeltaYMX, 5–10, 12–15, 16, 23, 37–42  
 dynr::dynr, 21, 28  
 FigPlotEffects, 16, 18–20, 25–27  
 FigScatterPlotCoverage, 17, 17–20, 25–27  
 FigScatterPlotPower, 17, 18, 18–20, 25–27  
 FigScatterPlotSeBias, 17, 18, 19, 20,  
     25–27  
 FigScatterPlotType1, 17–19, 20, 25–27  
 FitDynr, 21, 22, 28–31, 43, 91  
 FitDynr(), 4, 43, 91

FitMx, 21, 21, 28–31, 43, 91  
 FitMx(), 4, 43, 91  
  
 GenData, 22, 29, 44  
 GenData(), 44  
  
 illustration\_dist, 31  
 illustration\_dist\_dt, 32  
 illustration\_dist\_dt\_mc, 32  
 illustration\_dist\_mc, 33  
 illustration\_dist\_med, 33  
 illustration\_dist\_med\_mc, 34  
 illustration\_dist\_med\_std, 34  
 illustration\_dist\_med\_std\_mc, 35  
 illustration\_results, 35  
 IllustrationBootPara, 5–10, 12–16, 23, 37–42  
 IllustrationCompress, 11, 24  
 IllustrationFigPlotEffects, 17–20, 25, 26, 27  
 IllustrationFigScatterPlotCoverage, 17–20, 25, 25–27  
 IllustrationFigScatterPlotPower, 17–20, 25, 26, 26, 27  
 IllustrationFigScatterPlotSeBias, 17–20, 25, 26, 27  
 IllustrationFitDynr, 21, 22, 28, 29–31, 43, 91  
 IllustrationFitDynr(), 4, 43, 91  
 IllustrationFitMx, 21, 22, 28, 28, 30, 31, 43, 91  
 IllustrationFitMx(), 4, 43, 91  
 IllustrationGenData, 22, 29, 44  
 IllustrationGenData(), 30, 31  
 IllustrationMCPhiSigma, 21, 22, 28, 29, 30, 31, 43, 91  
 IllustrationPrepData, 21, 22, 28, 29, 30, 30, 43, 91  
 IllustrationPrepData(), 28  
  
 MCStdXMY, 5–10, 12–16, 23, 37, 38–42  
 MCStdXYM, 5–10, 12–16, 23, 37, 38, 39–42  
 MCStdYMX, 5–10, 12–16, 23, 37, 38, 39, 40–42  
 MCXYM, 5–10, 12–16, 23, 37–39, 40, 41, 42  
 MCYMX, 5–10, 12–16, 23, 37–41, 42  
  
 OpenMx::OpenMx, 21, 28  
  
 params, 43

PhiHat, 21, 22, 28–31, 43, 91  
 PhiHat(), 8–10, 14–16, 40–42  
  
 RandomMeasurement, 22, 29, 44  
 RandomMeasurement(), 21, 22  
 results, 45

Sim, 46  
 SimDy whole page

SumDynrMCYMX, 83  
SumFitDynr, 83  
SumIllustration, 84  
SumIllustrationDynrBootParaStdXMY, 85  
SumIllustrationDynrBootParaXMY, 86  
SumIllustrationDynrDeltaStdXMY, 87  
SumIllustrationDynrDeltaXMY, 88  
SumIllustrationDynrMCStdXMY, 88  
SumIllustrationDynrMCXMY, 89  
SumIllustrationFitDynr, 90

ThetaHat, 21, 22, 28–31, 43, 91  
ThetaHat(), 5–7, 12, 13, 37–39