Package 'ezsim'

February 19, 2015

Type Package

Title provide an easy to use framework to conduct simulation

Version 0.5.5

Date 2014-06-25

Author TszKin Julian Chan <ctszkin@gmail.com>

Maintainer TszKin Julian Chan <ctszkin@gmail.com>

Description ezsim provides a handy way to run simulation and examine its result

License GPL (>= 2)

LazyLoad yes

Imports foreach, ggplot2, parallel, digest, plyr, reshape, Jmisc

BugReports TszKin Julian Chan <ctszkin@gmail.com>

NeedsCompilation no

Repository CRAN

Date/Publication 2014-06-26 07:41:04

R topics documented:

`createParDef` *Create a parameterDef Object.*

Description

createParDef creates a new parameterDef object from a list of scalar parameters and a list of other parameters. parameterDef is a short hand of "parameter definition". It defines parameters used by the dgp which is the most important part of a simulation. For each simulation,There is a particular set of parameter. parameterDef allow us to define several parameters for different simulation at once. There are two types of parameter in parameterDef, scalar parameters and other parameters. Scalar parameters must be a scalar. Any vectors or matrix is regarded as a sequence of scalar parameters. For example, n=seq(10,50,10), first simulation takes n=10, second simulation takes n=20 and so on. Other parameters can be anything and it is banker over the scalar parameters. For example, we would like to know how would the sample size affect the variance of the sample mean of normally distributed variable. We can set n=seq(10,50,10), mean=1 and sd=2. (see example)

Usage

```
createParDef(selection = list(), banker = list())
```

Arguments

`selection`	A list of scalar parameters
`banker`	A list of other parameters

Value

A parameterDef object

Author(s)

TszKin Julian Chan <ctszkin@gmail.com>

See Also

`setBanker.parameterDef`,`setSelection.parameterDef`,`evalFunctionOnParameterDef`,`generate.parameterDef`

Examples

```
par_def1<-createParDef(selection=list(mean=1,sd=2,n=seq(10,50,10)))

par_def2<-createParDef()
setSelection(par_def2,mean=1,sd=2,n=seq(10,50,10))

identical(par_def1,par_def2)

evalFunctionOnParameterDef(par_def1, function() rnorm(n,mean,sd) )  # 10 random number
evalFunctionOnParameterDef(par_def1, function() rnorm(n,mean,sd), index=3)  # 30 random number

generate(par_def1)

# More than 1 selection parameters
par_def3<-createParDef(selection=list(sd=2,mean=1:3,n=seq(10,50,10)))

generate(par_def3)
```

`evalFunctionOnParameterDef` *Test Whether a parameterDef Ojbect Work Properly for a dgp.*

Description

several set of parameters is generated from parameterDef. Function `fun` is evaulated under the `index`-th set of parameters and returns its value.

Usage

```
evalFunctionOnParameterDef(x, fun, index = 1, ...)
```

Arguments

`x`	A parameterDef object
`fun`	A function to be evaluated.
`index`	Which set of parameters to use.
`...`	unused

Author(s)

TszKin Julian Chan <ctszkin@gmail.com>

See Also

`parameterDef`

Examples

```
par_def<-createParDef()
par_def<-setSelection(par_def,mean=1,sd=2,n=seq(10,50,10))

evalFunctionOnParameterDef(par_def, function() rnorm(n,mean,sd) )  # 10 random number
evalFunctionOnParameterDef(par_def, function() rnorm(n,mean,sd), index=3)  # 30 random number

## Example 2
par_def <-createParDef()
par_def <- setBanker(par_def,xs=1,b=1)
par_def <- setSelection(par_def,n=seq(20,100,20),es=c(1,10))

dgp<-function(){
x<-rnorm(n,0,xs)
e<-rnorm(n,0,es)
y<-b * x + e
data.frame(y,x)
}
estimator<-function(d){
r<-summary(lm(y~x-1,data=d))
c(b=r$coef[,1],t=(r$coef[,1]-1)/r$coef[,2] )
}

true<-function(){
c(b,(b-1)/(es/sqrt(n)/xs))
}
evalFunctionOnParameterDef(par_def,dgp)
estimator(evalFunctionOnParameterDef(par_def,dgp))
evalFunctionOnParameterDef(par_def,true)
```

`ezsim` *Create an ezsim Object.*

Description

Create an ezsim object from 4 important arguments(dgp,estimator,true,parameter_def).

Usage

```
ezsim(m, estimator, dgp, parameter_def, true_value = NULL,
  display_name = NULL, estimator_parser = NULL, auto_save = 0,
  run = TRUE, run_test = TRUE, use_seed = round(runif(1) * 1e+05),
  use_core = 1, cluster_packages = NULL, cluster = NULL)
```

Arguments

`m`	The number of times you want to simulate.
`estimator`	A function takes the return value of dgp as argument and return estimates of the dgp

dgp	A function defines the data generating process(dgp). It returns an object (usually it is a data.frame) which can be used as argument of estimator. It will be evaluated under an environment generated by a set of parameter generated by parameter_def.
parameter_def	A parameter_def object will be used in this simulation.
true_value	A function defines the true value of estimators(TV). Similar to dgp, but it returns the true value of estimates. It is necessary for computing the bias and rmse of the estimator. The length of its return value must be the same as the lenght of estimator. If it is missing, True Value, Bias and rmse will be NA in the summary of ezsim.
display_name	Display name for the name of parameter and estimator. see plotmath for details.
estimator_parser	A function to parse the return value of estimator function
auto_save	number of auto save during the simulation, default is 0.
run	Whether the simulation will be ran right after the creation.
run_test	Whether to perform a test before the simulation.
use_seed	The seed to be used in the simulation. If use_core=1, set.seed(use_seed) will be called. If use_core=>1 and cluster=NULL, clusterSetRNGStream(cluster,use_seed) will be used. Ignored if use_core=>1 and cluster is provided.
use_core	Number of cpu core to be used.
cluster_packages	Names of the packages to be loaded in the cluster.
cluster	cluster for parallelization. If it is NULL, a cluster with use_code cores will be created automatically (will be removed after the simulation)

Value

An ezsim object.

Author(s)

TszKin Julian Chan <ctszkin@gmail.com>

See Also

createParDef setBanker,setSelection summary.ezsim

Examples

```
## Not run:
## Example 1
ezsim_basic<-ezsim(
    m             = 100,
    run           = TRUE,
    display_name  = c(mean_hat="hat(mu)",sd_mean_hat="hat(sigma[hat(mu)])"),
    parameter_def = createParDef(list(n=seq(20,80,20),mu=c(0,2),sigma=c(1,3,5))),
    dgp           = function() rnorm(n,mu,sigma),
```

```
    estimator     = function(x) c(mean_hat = mean(x),
                                  sd_mean_hat=sd(x)/sqrt(length(x)-1)),
    true_value    = function() c(mu, sigma / sqrt(n-1))
)

## Test whether an ezsim object is valid.
## Print the result of the test and dont return the name of estimator.
test(ezsim_basic,print_result=TRUE,return_name=FALSE)

## Summary of an ezsim object
summary(ezsim_basic)

## Summary of a subset of ezsim object
summary(ezsim_basic,subset=list(estimator='mean_hat',n=c(20,40),sigma=c(1,3)))

## More Summary Statistics
summary(ezsim_basic,simple=FALSE,subset=list(estimator='mean_hat',n=c(20,40),sigma=c(1,3)))

## Customize the Summary Statistics
summary(ezsim_basic,stat=c("q25","median","q75"),Q025=quantile(value_of_estimator,0.025),
  Q975=quantile(value_of_estimator,0.975),subset=list(estimator='mean_hat',n=c(20,40),sigma=c(1,3)))

## Plot an ezsim object
plot(ezsim_basic)
## Subet of the Plot
plot(ezsim_basic,subset=list(estimator="sd_mean_hat",mu=0))
plot(ezsim_basic,subset=list(estimator="mean_hat",sigma=3))
## Parameters Priority of the Plot
plot(ezsim_basic,subset=list(estimator="sd_mean_hat",mu=0),parameters_priority=c("sigma","n"))
plot(ezsim_basic,subset=list(estimator="mean_hat",sigma=c(1,3)),parameters_priority="mu")

## Density Plot
plot(ezsim_basic,'density')
plot(ezsim_basic,"density",subset=list(estimator="mean_hat",sigma=3),parameters_priority="n",
  benchmark=dnorm)
plot(ezsim_basic,"density",subset=list(estimator="mean_hat",mu=0),parameters_priority="n" ,
  benchmark=dnorm)

## Plot the summary ezsim
plot(summary(ezsim_basic,c("q25","q75")))
plot(summary(ezsim_basic,c("q25","q75"),subset=list(estimator='mean_hat')))
plot(summary(ezsim_basic,c("median"),subset=list(estimator='sd_mean_hat')))

## Example 2
ezsim_ols<-ezsim(
    m             = 100,
    run           = TRUE,
    display_name  = c(beta_hat='hat(beta)',es='sigma[e]^2',xs='sigma[x]^2',
                      sd_beta_hat='hat(sigma)[hat(beta)]'),
   parameter_def = createParDef(selection=list(xs=c(1,3),beta=c(0,2),n=seq(20,80,20),es=c(1,3))),
    dgp           = function(){
                        x<-rnorm(n,0,xs)
                        e<-rnorm(n,0,es)
```

```
                        y<-beta * x + e
                        data.frame(y,x)
                      },
    estimator     = function(d){
                        r<-summary(lm(y~x-1,data=d))
                        out<-r$coef[1,1:2]
                        names(out)<-c('beta_hat','sd_beta_hat')
                        out
                      },
    true_value    = function() c(beta, es/sqrt(n)/xs)
)
summary(ezsim_ols)
plot(ezsim_ols)
plot(ezsim_ols,subset=list(beta=0))

plot(ezsim_ols,'density')
plot(ezsim_ols,'density',subset=list(es=1,xs=1))


## example 3
ezsim_powerfun<-ezsim(
    run           = TRUE,
    m             = 100,
    parameter_def = createParDef(selection=list(xs=1,n=50,es=c(1,5),b=seq(-1,1,0.1))),
    display_name  = c(b='beta',es='sigma[e]^2',xs='sigma[x]^2'),
    dgp           = function(){
                        x<-rnorm(n,0,xs)
                        e<-rnorm(n,0,es)
                        y<-b * x + e
                        data.frame(y,x)
                      },
    estimator     = function(d){
                        r<-summary(lm(y~x-1,data=d))
                        stat<-r$coef[,1]/r$coef[,2]

                        # test whether b > 0
                        # level of significance : 5%
                        out <- stat > c(qnorm(.95), qt(0.95,df=r$df[2]))
                        names(out)<-c("z-test","t-test")
                        out
                      }
)
plot(ezsim_powerfun,'powerfun')

## End(Not run)
```

---

generate.parameterDef *Generate Parameter*

Description

Generate Parameters from a parameterDef Object. The selection parameters in parameterDef is expanded and concatenated with the banker parameters.

Usage

```
## S3 method for class 'parameterDef'
generate(x,...)
```

Arguments

x	A parameterDef Object
...	unused

Value

other_parameters	A list of other_parameters
scalar_parameters	A data.frame of scalar_parameters
parameter_list	A list of all parameters

Author(s)

TszKin Julian Chan <ctszkin@gmail.com>

See Also

setBanker.parameterDef, setSelection.parameterDef, evalFunctionOnParameterDef, generate.parameterDef

Examples

```
par_def1<-createParDef(selection=list(mean=1,sd=2,n=seq(10,50,10)))
generate(par_def1)
par_def2<-createParDef(selection=list(sd=2,mean=1:3,n=seq(10,50,10)))
generate(par_def2)
```

---

getSelectionName.summary.ezsim

Get Names of selection Parameters.

Description

Get names of selections parameters from an summary.ezsim object.

Usage

```
## S3 method for class 'summary.ezsim'
getSelectionName(x,simple=FALSE,parameters_priority,...)
```

Arguments

x	an summary.ezsim object
simple	If true, return only the name of selection parameters. If False, split the selection into two groups, one with fixed value, one with varied value. Also, subtitle is returned.
parameters_priority	Priority in sorting parameters.
...	unused

Value

Names of selection parameters.

Note

For internal use of ezsim.

Author(s)

TszKin Julian Chan <ctszkin@gmail.com>

See Also

getSelectionName.ezsim keywords internal

Examples

```
## Not run:
ezsim_basic<-ezsim(
    m             = 100,
    run           = TRUE,
    display_name  = c(mean_hat="hat(mu)",sd_mean_hat="hat(sigma[hat(mu)])"),
    parameter_def = createParDef(list(n=seq(20,80,20),mu=c(0,2),sigma=c(1,3,5))),
    dgp           = function() rnorm(n,mu,sigma),
    estimator     = function(x) c(mean_hat = mean(x),
                                  sd_mean_hat=sd(x)/sqrt(length(x)-1)),
    true_value    = function() c(mu, sigma / sqrt(n-1))
)

getSelectionName(ezsim_basic)
getSelectionName(summary(ezsim_basic))

## End(Not run)
```

merge.ezsim *Merge two ezsim objects*

Description

Merge two ezsim objects. Either `m` or `parameter_def` of two ezsim objects must be the same. If `parameter_def` are the same, the merging is regarded as increasing the number of simulation `m`. If the `parameter_def` are different and `m` are the same, the merging is regarded as extending the `parameter_def`. Notice that, use_seed will not be merged!

Usage

```
## S3 method for class 'ezsim'
merge(x, y, ...)
```

Arguments

`x`	A ezsim to merge with
`y`	A ezsim to merge with
`...`	unused

Author(s)

TszKin Julian Chan <ctszkin@gmail.com>

See Also

`ezsim`

merge.parameterDef *Merge two parameterDef objects*

Description

Merge two parameterDef objects. 'others' of two parameterDef objects must be the same. 'scalars' of two parameterDef objects must have same name and the value must not overlap.

Usage

```
## S3 method for class 'parameterDef'
merge(x, y, ...)
```

Arguments

x	A parameterDef to merge with
y	A parameterDef to merge with
...	unused

Author(s)

TszKin Julian Chan <ctszkin@gmail.com>

See Also

createParDef, createParDef

---

plot.ezsim *Plot an ezsim Object*

Description

There are 3 different modes to plot an ezsim object. 'summary', 'density' and 'powerfun' plot the summary statistics,density function and power function of an ezsim object respectively.

'summary': The y-variable of the plot are summary statistics of the estimator. Two confidence bounds will be shaded in the plot. 25% and 75% percentile will form a 50% confidence bound. Similarly, 2.5% and 97.5% percentile will form a 95% confidence bound. Each plot have only one estimator. The scalars parameter has the longest length will be the x-variable of the plot. The rest of the selection parameters will be become the facets of the plot (see **ggplot2**).

density : Density plot of the estimator. Each plot have only one estimator. selection parameter will appear as different colour and in different facets.

powerfun : Plot the power function of test(s). Estimators have to be a test (value = 1 if rejecting the null hypothesis, value = 0 if fail to reject the null hypothesis) banker parameters will not be shown in the graph.

Usage

```
## S3 method for class 'ezsim'
plot(x, type = c("summary", "density", "powerfun"), subset,
  parameters_priority, return_print = FALSE, ylab, title, pdf_option,
  null_hypothesis, benchmark, ...)
```

Arguments

x	An ezsim object
type	Type of plot
subset	subset of estimators or parameters. See subset.ezsim for details.
parameters_priority	Display priority of parameter. If any parameter is missing here, they will be sorted by length.
return_print	If TRUE, return a list of ggplot2 object. If FALSE(default), all of the plot will be printed out.
ylab	Label of y-axis
title	Title of the plot
pdf_option	A list of option pass to pdf. If it is not missing, the plot will export to a pdf file
null_hypothesis	Null hypothesis of the test. For type=='powerfun' only.
benchmark	Benchmark distribution. For type=='density' only.
...	unused

Value

Optional: a list of ggplot2 object

Author(s)

TszKin Julian Chan <ctszkin@gmail.com>

See Also

ezsim,summary.ezsim, plot.summary.ezsim,

Examples

```
## Not run:
## example 1
ezsim_basic<-ezsim(
    m             = 100,
    run           = TRUE,
    display_name  = c(mean_hat="hat(mu)",sd_mean_hat="hat(sigma[hat(mu)])"),
    parameter_def = createParDef(list(n=seq(20,80,20),mu=c(0,2),sigma=c(1,3,5))),
    dgp           = function() rnorm(n,mu,sigma),
    estimator     = function(x) c(mean_hat = mean(x),
                                  sd_mean_hat=sd(x)/sqrt(length(x)-1)),
    true_value    = function() c(mu, sigma / sqrt(n-1))
)
## Plot an ezsim object
plot(ezsim_basic)
## Subet of the Plot
plot(ezsim_basic,subset=list(estimator="sd_mean_hat",mu=0))
```

```
plot(ezsim_basic,subset=list(estimator="mean_hat",sigma=3))
## Parameters Priority of the Plot
plot(ezsim_basic,subset=list(estimator="sd_mean_hat",mu=0),parameters_priority=c("sigma","n"))
plot(ezsim_basic,subset=list(estimator="mean_hat",sigma=c(1,3)),parameters_priority="mu")

## Density Plot
plot(ezsim_basic,'density')
plot(ezsim_basic,"density",subset=list(estimator="mean_hat",sigma=3),parameters_priority="n",
   benchmark=dnorm)
plot(ezsim_basic,"density",subset=list(estimator="mean_hat",mu=0),parameters_priority="n" ,
   benchmark=dnorm)

## example 2
ezsim_ols<-ezsim(
    m             = 100,
    run           = TRUE,
    display_name  = c(beta_hat='hat(beta)',es='sigma[e]^2',xs='sigma[x]^2',
                      sd_beta_hat='hat(sigma)[hat(beta)]'),
   parameter_def = createParDef(selection=list(xs=c(1,3),beta=c(0,2),n=seq(20,80,20),es=c(1,3))),
    dgp           = function(){
                        x<-rnorm(n,0,xs)
                        e<-rnorm(n,0,es)
                        y<-beta * x + e
                        data.frame(y,x)
                      },
    estimator     = function(d){
                        r<-summary(lm(y~x-1,data=d))
                        out<-r$coef[1,1:2]
                        names(out)<-c('beta_hat','sd_beta_hat')
                        out
                      },
    true_value    = function() c(beta, es/sqrt(n)/xs)
)
plot(ezsim_ols)
plot(ezsim_ols,subset=list(beta=0))

plot(ezsim_ols,'density')
plot(ezsim_ols,'density',subset=list(es=1,xs=1))


## example 3
ezsim_powerfun<-ezsim(
    run           = TRUE,
    m             = 100,
    parameter_def = createParDef(selection=list(xs=1,n=50,es=c(1,5),b=seq(-1,1,0.1))),
    display_name  = c(b='beta',es='sigma[e]^2',xs='sigma[x]^2'),
    dgp           = function(){
                        x<-rnorm(n,0,xs)
                        e<-rnorm(n,0,es)
                        y<-b * x + e
                        data.frame(y,x)
                      },
    estimator     = function(d){
```

```
                        r<-summary(lm(y~x-1,data=d))
                        stat<-r$coef[,1]/r$coef[,2]

                        # test whether b > 0
                        # level of significance : 5%
                        out <- stat > c(qnorm(.95), qt(0.95,df=r$df[2]))
                        names(out)<-c("z-test","t-test")
                        out
                    }
)
plot(ezsim_powerfun,'powerfun')

## End(Not run)
```

`plot.summary.ezsim` *Plot an summary.ezsim Object*

Description

Plot the summary statistics for several estimators in the same plot. Summary statistics abd estimators are separated by colour and linetype. The longest scalars parameter will be the x-variable of the plot. The rest of the scalars parameters will be become the facets of the plot (see **ggplot2**).Banker parameters will not be shown in the graph.

Usage

```
## S3 method for class 'summary.ezsim'
plot(x, parameters_priority,
  ylab = "Summary Statistics", title, pdf_option, return_print, ...)
```

Arguments

`x`	An summary.ezsim Object
`parameters_priority`	Display priority of parameter. Any missed parameters will be sorted by length.
`ylab`	Label of y-axis
`title`	Title of the plot
`return_print`	If TRUE, return a list of ggplot2 object. If FALSE(default), all of the plot will be printed out.
`pdf_option`	A list of option pass to `pdf`. If it is not missing, the plot will export to a pdf file
`...`	unused

Value

Optional: a ggplot2 object

Author(s)

TszKin Julian Chan <ctszkin@gmail.com>

See Also

ezsim,summary.ezsim

Examples

```
## Not run:
ezsim_basic<-ezsim(
    m             = 100,
    run           = TRUE,
    display_name  = c(mean_hat="hat(mu)",sd_mean_hat="hat(sigma[hat(mu)])"),
    parameter_def = createParDef(list(n=seq(20,80,20),mu=c(0,2),sigma=c(1,3,5))),
    dgp           = function() rnorm(n,mu,sigma),
    estimator     = function(x) c(mean_hat = mean(x),
                                  sd_mean_hat=sd(x)/sqrt(length(x)-1)),
    true_value    = function() c(mu, sigma / sqrt(n-1))
)
## Plot the summary ezsim
plot(summary(ezsim_basic,c("q25","q75")))
plot(summary(ezsim_basic,c("q25","q75"),subset=list(estimator='mean_hat')))
plot(summary(ezsim_basic,c("median"),subset=list(estimator='sd_mean_hat')))

## End(Not run)
```

---

print.ezsim *Print an ezsim Object.*

Description

Print an ezsim Object. See ezsim for details.

Usage

```
## S3 method for class 'ezsim'
print(x, ...)
```

Arguments

x	An ezsim Object
...	unused

Author(s)

TszKin Julian Chan <ctszkin@gmail.com>

See Also

ezsim

`print.parameterDef` *Print a parameterDef Object.*

Description

Print a parameterDef Object in the console

Usage

```
## S3 method for class 'parameterDef'
print(x, ...)
```

Arguments

x	A parameterDef Object
...	unused

Author(s)

TszKin Julian Chan <ctszkin@gmail.com>

See Also

createParDef

`print.parameters` *Print a parameters Object.*

Description

Print a parameters Object in the console

Usage

```
## S3 method for class 'parameters'
print(x, ...)
```

Arguments

x	A parameters Object
...	unused

Author(s)

TszKin Julian Chan <ctszkin@gmail.com>

See Also

parameterDef

---

print.summary.ezsim *Print a summary.ezsim Object.*

Description

Print a summary.ezsim Object in the console. See summary.ezsim for details

Usage

```
## S3 method for class 'summary.ezsim'
print(x,digits=4,...)
```

Arguments

x	A summary.ezsim Object
digits	Number of digits the data will be rounded to.
...	unused

Author(s)

TszKin Julian Chan <ctszkin@gmail.com>

See Also

summary.ezsim

---

run.ezsim *Run the Simulation*

Description

Run the Simulation of an ezsim object. The simulation result is store into the ezsim object in the argument directly, reassignment is not needed.

Usage

```
## S3 method for class 'ezsim'
run(x, ...)
```

Arguments

x	An ezsim object
...	not used

Author(s)

TszKin Julian Chan <ctszkin@gmail.com>

Examples

```
## Not run:
ezsim_basic<-ezsim(
    m             = 100,
    run           = FALSE,
    run_test      = TRUE,
    display_name  = c(mean_hat="hat(mu)",sd_mean_hat="hat(sigma[hat(mu)])"),
    parameter_def = createParDef(list(n=seq(20,80,20),mu=c(0,2),sigma=c(1,3,5))),
    dgp           = function() rnorm(n,mu,sigma),
    estimator     = function(x) c(mean_hat = mean(x),
                                  sd_mean_hat=sd(x)/sqrt(length(x)-1)),
    true_value    = function() c(mu, sigma / sqrt(n-1))
)
run(ezsim_basic)

## End(Not run)
```

---

setBanker.parameterDef

Set a parameterDef Object.

Description

setBanker sets the scalar parameters of a parameterDef object. setBanker are "call by reference", so assignment is not needed to update the parameterDef object. In other words, they will overwrite the value of its argument(parameterDef object). parameterDef is a short hand of "parameter definition". It defines parameters used by the dgp which is the most important part of a simulation. For each simulation,There is a particular set of parameter. parameterDef allow us to define several parameters for different simulation at once. There are two types of parameter in parameterDef, scalar parameters and other parameters. Scalar parameters must be a scalar. Any vectors or matrix is regarded as a sequence of scalar parameters. For example, n=seq(10,50,10), first simulation takes n=10, second simulation takes n=20 and so on. Other parameters can be anything and it is banker over the scalar parameters. For example, we would like to know how would the sample size affect the variance of the sample mean of normally distributed variable. We can set n=seq(10,50,10), mean=1 and sd=2. (see example)

Usage

```
## S3 method for class 'parameterDef'
setBanker(x,...)
```

Arguments

x	A parameterDef object
...	Variables to be added to a parameterDef object

Value

A parameterDef object

Author(s)

TszKin Julian Chan <ctszkin@gmail.com>

See Also

setSelection.parameterDef,createParDef,evalFunctionOnParameterDef,generate.parameterDef

Examples

```
par_def1<-createParDef(selection=list(mean=1,sd=2,n=seq(10,50,10)))

par_def2<-createParDef()
setSelection(par_def2,mean=1,sd=2,n=seq(10,50,10))

identical(par_def1,par_def2)

evalFunctionOnParameterDef(par_def1, function() rnorm(n,mean,sd) )  # 10 random number
evalFunctionOnParameterDef(par_def1, function() rnorm(n,mean,sd), index=3)  # 30 random number

generate(par_def1)

# More than 1 selection parameters
par_def3<-createParDef(selection=list(sd=2,mean=1:3,n=seq(10,50,10)))

generate(par_def3)

#
par_def4<-createParDef(selection=list(mean=1,sd=2,n=seq(10,50,10)))
setBanker(par_def4,some_matrix=matrix(1:4,nrow=2),some_vector=1:6)
par_def4
generate(par_def4)
```

---

setSelection.parameterDef
Set a parameterDef Object.

Description

setSelection sets the banker parameters of a parameterDef object. setSelection are "call by reference", so assignment is not needed to update the parameterDef object. In other words, they will overwrite the value of its argument(parameterDef object). parameterDef is a short hand of "parameter definition". It defines parameters used by the dgp which is the most important part of a simulation. For each simulation,There is a particular set of parameter. parameterDef allow us to define several parameters for different simulation at once. There are two types of parameter in parameterDef, scalar parameters and other parameters. Scalar parameters must be a scalar. Any vectors or matrix is regarded as a sequence of scalar parameters. For example, n=seq(10,50,10), first simulation takes n=10, second simulation takes n=20 and so on. Other parameters can be anything and it is banker over the scalar parameters. For example, we would like to know how would the sample size affect the variance of the sample mean of normally distributed variable. We can set n=seq(10,50,10), mean=1 and sd=2. (see example)#'

Usage

```
## S3 method for class 'parameterDef'
setSelection(x,...)
```

Arguments

x	A parameterDef object
...	Variables to be added to a parameterDef object

Value

A parameterDef object

Author(s)

TszKin Julian Chan <ctszkin@gmail.com>

See Also

setSelection.parameterDef,createParDef,evalFunctionOnParameterDef,generate.parameterDef

Examples

```
par_def1<-createParDef(selection=list(mean=1,sd=2,n=seq(10,50,10)))

par_def2<-createParDef()
setSelection(par_def2,mean=1,sd=2,n=seq(10,50,10))

identical(par_def1,par_def2)

evalFunctionOnParameterDef(par_def1, function() rnorm(n,mean,sd) )  # 10 random number
evalFunctionOnParameterDef(par_def1, function() rnorm(n,mean,sd), index=3)  # 30 random number

generate(par_def1)
```

```
# More than 1 selection parameters
par_def3<-createParDef(selection=list(sd=2,mean=1:3,n=seq(10,50,10)))

generate(par_def3)
```

`subset.ezsim` *Return of the Simulation*

Description

Return a subset of the simulation result of an ezsim object.

Usage

```
## S3 method for class 'ezsim'
subset(x, subset, ...)
```

Arguments

`x`	An ezsim Object
`subset`	A list contains the subset of estimators and parameters. To select a subset of estimators: `list(estimator=c('name of estimator1','name of estimator2'))`. To select a subset of parameters: `list(mean=1:3,sd=4:5)`. Or both.
`...`	unused

Value

sim of ezsim

Note

For internal use of ezsim.

Author(s)

TszKin Julian Chan <ctszkin@gmail.com>

See Also

ezsim

Examples

```
## Not run:
ezsim_basic<-ezsim(
    m             = 100,
    run           = TRUE,
    display_name  = c(mean_hat="hat(mu)",sd_mean_hat="hat(sigma[hat(mu)])"),
    parameter_def = createParDef(list(n=seq(20,80,20),mu=c(0,2),sigma=c(1,3,5))),
    dgp           = function() rnorm(n,mu,sigma),
    estimator     = function(x) c(mean_hat = mean(x),
                                  sd_mean_hat=sd(x)/sqrt(length(x)-1)),
    true_value    = function() c(mu, sigma / sqrt(n-1))
)
subset(ezsim_basic,subset=list(estimator='mean_hat',mu=0,n=c(20,40)))

## End(Not run)
```

`summary.ezsim` *Summarize an ezsim Object*

Description

A quick summary to the simulation. Summary statistics included mean, true value (tv), bias, bias percentage (mean/tv-1), sd, rmse (root mean square error), min, q25 (first quarter), median, q75 (third quarter), max, p value of jb-test. See `ezsim` and `plot.summary.ezsim` for details and examples.

Usage

```
## S3 method for class 'ezsim'
summary(object,stat=c('mean','tv','bias',
'biaspercentage','sd','rmse','min','q25','median',
'q75','max','jb_test'),simple=TRUE,subset,...)
```

Arguments

`object`	An ezsim object
`stat`	Some preset summary statistics. Included, `c('mean','tv','bias','biaspercentage','sd','rmse',`
`simple`	If True, shows only mean, true value, bias, sd and rmse of the estimator. If False, shows all statistics in stat.
`subset`	subset of estimators or parameters. See `subset.ezsim` for details.
`...`	Furhter summary statistics. Given in the form stat_name=stat. For example, Mean=mean

Value

A summary.ezsim object

Author(s)

TszKin Julian Chan <ctszkin@gmail.com>

See Also

ezsim, plot.summary.ezsim, getSelectionName.summary.ezsim

Examples

```
## Not run:
ezsim_basic<-ezsim(
    m             = 100,
    run           = TRUE,
    display_name  = c(mean_hat="hat(mu)",sd_mean_hat="hat(sigma[hat(mu)])"),
    parameter_def = createParDef(list(n=seq(20,80,20),mu=c(0,2),sigma=c(1,3,5))),
    dgp           = function() rnorm(n,mu,sigma),
    estimator     = function(x) c(mean_hat = mean(x),
                                  sd_mean_hat=sd(x)/sqrt(length(x)-1)),
    true_value    = function() c(mu, sigma / sqrt(n-1))
)



## Summary of an ezsim object
summary(ezsim_basic)

## Summary of a subset of ezsim object
summary(ezsim_basic,subset=list(estimator='mean_hat',n=c(20,40),sigma=c(1,3)))

## More Summary Statistics
summary(ezsim_basic,simple=FALSE,subset=list(estimator='mean_hat',n=c(20,40),sigma=c(1,3)))

## Customize the Summary Statistics
summary(ezsim_basic,stat=c("q25","median","q75"),Q025=quantile(value_of_estimator,0.025),
  Q975=quantile(value_of_estimator,0.975),subset=list(estimator='mean_hat',n=c(20,40),sigma=c(1,3)))

## End(Not run)
```

---

test.ezsim *Perform a Test for an ezsim Object*

Description

For each set of parameters, the simulation is ran once to obtain the value of estimator and true value to make sure everything in ezsim is properly defined. The test results will be shown in the console. The test will be ran automatically when you create an ezsim object.

Usage

```
## S3 method for class 'ezsim'
test(x, return_name = TRUE, print_result = FALSE, ...)
```

Arguments

x	An ezsim Object
return_name	Whehter to return the name of estimator
print_result	Whehter to print the return
...	unused

Value

Optional: names of estimator.

Author(s)

TszKin Julian Chan <ctszkin@gmail.com>

See Also

ezsim

Examples

```
## Not run:
ezsim_basic<-ezsim(
    m             = 100,
    run           = FALSE,
    run_test      = FALSE,
    display_name  = c(mean_hat="hat(mu)",sd_mean_hat="hat(sigma[hat(mu)])"),
    parameter_def = createParDef(list(n=seq(20,80,20),mu=c(0,2),sigma=c(1,3,5))),
    dgp           = function() rnorm(n,mu,sigma),
    estimator     = function(x) c(mean_hat = mean(x),
                                  sd_mean_hat=sd(x)/sqrt(length(x)-1)),
    true_value    = function() c(mu, sigma / sqrt(n-1))
)

test(ezsim_basic,print_result=TRUE)

## End(Not run)
```

Index