Package 'CoSMoS'

May 9, 2019

Type Package

Title Complete Stochastic Modelling Solution

Version 1.0.1

Description A single framework, unifying, extending, and improving a general-purpose modelling strategy, based on the assumption that any process can emerge by transforming a specific 'parent' Gaussian process Papalexiou (2018) <doi:10.1016/j.advwatres.2018.02.013>.

License GPL-3

Depends R (>= 3.6.0), ggplot2, pracma, nloptr, data.table, methods, stats

Encoding UTF-8

LazyData true

RoxygenNote 6.1.1

Suggests testthat, knitr, rmarkdown

VignetteBuilder knitr

NeedsCompilation no

Author Filip Strnad [aut, cre], Simon Michael Papalexiou [aut], Yannis Markonis [ctb]

Maintainer Filip Strnad <strnadf@fzp.czu.cz>

Repository CRAN

Date/Publication 2019-05-09 08:30:53 UTC

R topics documented:

`CoSMoS-package` *CoSMoS: Complete Stochastic Modelling Solution*

Description

CoSMoS is an R package that makes time series generation with desired properties easy. Just choose the characteristics of the time series you want to generate, and it will do the rest.

Details

The generated time series preserve any probability distribution and any linear autocorrelation structure. Users can generate as many and as long time series from processes such as precipitation, wind, temperature, relative humidity etc. It is based on a framework that unified, extended, and improved a modelling strategy that generates time series by transforming "parent" Gaussian time series having specific characteristics (Papalexiou, 2018).

Funding

The package was partly funded by the Global institute for Water Security (GIWS; https://www.usask.ca/water/) and the Global Water Futures (GWF; https://gwf.usask.ca/) program.

Author(s)

Coded and maintained by: Filip Strnad `<strnadf@fzp.czu.cz>`

Conceptual design by: Simon Michael Papalexiou `<sm.papalexiou@usask.ca>`

Tested and documented by: Yannis Markonis `<markonis@fzp.czu.cz>`

References

Papalexiou, S.M., 2018. Unified theory for stochastic modelling of hydroclimatic processes: Preserving marginal distributions, correlation structures, and intermittency. Advances in Water Resources 115, 234-252. (link)

acs *AutoCorrelation Structure*

Description

Provides a parametric function that describes the values of the linear autocorrelation up to desired lags. For more details on the parametric autocorrelation structures see section 3.2 in (Papalexiou 2018)

Usage

```
acs(id, ...)
```

Arguments

id	autocorrelation tructure id
...	other arguments (t as lag and acs parameters)

Examples

```
library(CoSMoS); library(ggplot2)

## specify lag
t <- 0:10

## get the ACS
f <- acs('fgn', t = t, H = .75)
b <- acs('burrXII', t = t, scale = 1, shape1 = .6, shape2 = .4)
w <- acs('weibull', t = t, scale = 2, shape = 0.8)
p <- acs('paretoII', t = t, scale = 3, shape = 0.3)

## visualize the ACS
dta <- data.frame(t, f, b, w, p)

m.dta <- melt(dta, id.vars = 't')

ggplot(m.dta,
       aes(x = t,
           y = value,
           group = variable,
           colour = variable)) +
  geom_point(size = 2.5) +
  geom_line(lwd = 1) +
  scale_color_manual(values = c('steelblue4', 'red4', 'green4', 'darkorange'),
                     labels = c('FGN', 'Burr XII', 'Weibull', 'Pareto II'),
                     name = '') +
  labs(x = bquote(lag ~ tau),
       y = 'Acf') +
```

```
  scale_x_continuous(breaks = t) +
  theme_classic()
```

---

actpnts *AutoCorrelation Transformed Points*

Description

Transforms a gaussian process in order to match a target marginal lowers its autocorrelation values. The actpnts evaluates the corresponding autocorrelations for the given target marginal for a set of gaussian correlations, i.e., it returns (ρ_x, ρ_z) points where $\rho_x and \rho_z$ represent, respectively, the autocorrelations of the target and gaussian process.

Usage

```
actpnts(margdist, margarg, p0 = 0, distbounds = c(-Inf, Inf))
```

Arguments

margdist	target marginal distribution
margarg	list of marginal distribution arguments
p0	probability zero
distbounds	distribution bounds (default set to c(-Inf, Inf))

Examples

```
library(CoSMoS); library(ggplot2)

## here we target to a process that has the Pareto type II marginal distribution
## with scale parameter 1 and shape parameter 0.3
## (note that all parameters have to be named)
dist <- 'paretoII'
distarg <- list(scale = 1, shape = .3)

x <- actpnts(margdist = dist, margarg = distarg, p0 = 0)
x

## you can see the points by using
ggplot(x,
       aes(x = rhox,
           y = rhoz)) +
  geom_point(colour = 'royalblue4', size = 2.5) +
  geom_abline(lty = 5) +
  labs(x = bquote(Autocorrelation ~ rho[x]),
       y = bquote(Gaussian ~ rho[z])) +
  scale_x_continuous(limits = c(0, 1)) +
```

```
  scale_y_continuous(limits = c(0, 1)) +
  theme_classic()
```

---

analyzeTS *The Functions analyzeTS, reportTS, and simulateTS*

Description

Provide a complete set of tools to make time series analysis a piece of cake - analyzeTS() automatically performs seasonal analysis, fits distributions and correlation structures, reportTS provides visualizations of the fitted distributions and correlation structures, and a table with the values of the fitted parameters and basic descriptive statistics, simulateTS automatically takes the results of the analyseTS and generates syntetic ones.

Usage

```
analyzeTS(TS, season = "month", dist = "ggamma", acsID = "weibull",
  norm = "N2", n.points = 30, lag.max = 30, constrain = FALSE)

reportTS(aTS, method = "dist")

simulateTS(aTS, from = NULL, to = NULL)
```

Arguments

TS	time series in format - date, value
season	name of the season (e.g. month, week)
dist	name of the distribution to be fitted
acsID	ID of the autocorrelation structure to be fitted
norm	norm used for distribution fitting - id ('N1', 'N2', 'N3', 'N4')
n.points	number of points to be subsetted from ecdf
lag.max	max lag for the empirical autocorrelation structure
constrain	logical - constrain shape2 parametes for finite tails
aTS	analyzed timeseries
method	report method - 'dist' for distribution fits, 'acs' for ACS fits and 'stat' for basic statistical report
from	starting date/time of the simulation
to	end date/time of the simulation

Details

In practice, we usually want to simulate a natural process using some sampled time series. To generate a synthetic time series with similar characteristics to the observed values, we have to determine marginal distribution, autocorrelation structure and probability zero for each individual month. This can is done by fitting distributions and autocorrelation structures with analyzeTS(). Result can be checked with reportTS(). Syynthetic time series with the same statistical properties can be produced with simulateTS().

Recomended distributions for variables:

- *precipitation*: ggamma (Generalized Gamma), burr### (Burr type)
- *streamflow*: ggamma (Generalized Gamma), burr### (Burr type)
- *relative humidity*: beta
- *temperature*: norm (Normal distribution)

Examples

```
library(CoSMoS)

## Load data included in the package
## (to find out more about the data use ?precip)
data('precip')

## Fit seasonal ACSs and distributions to the data
a <- analyzeTS(precip)

reportTS(a, 'dist') ## show seasonal distribution fit
reportTS(a, 'acs') ## show seasonal ACS fit
reportTS(a, 'stat') ## display basic descriptive statisctics

#####################################
## 'duplicate' analyzed time series ##
sim <- simulateTS(a)

## plot the result
precip[, id := 'observed']
sim[, id := 'simulated']

dta <- rbind(precip, sim)

ggplot(dta) +
  geom_line(aes(x = date, y = value)) +
  facet_wrap(~id, ncol = 1) +
  theme_classic()

#############################################
## or simulate timeseries of different length ##
sim <- simulateTS(a,
                  from = as.POSIXct('1978-12-01 00:00:00'),
                  to = as.POSIXct('2008-12-01 00:00:00'))
```

```
## and plot the result
precip[, id := 'observed']
sim[, id := 'simulated']

dta <- rbind(precip, sim)

ggplot(dta) +
  geom_line(aes(x = date, y = value)) +
  facet_wrap(~id, ncol = 1) +
  theme_classic()
```

BurrIII *Burr Type III distribution*

Description

Provides density, distribution function, quantile function, random value generation, and raw moments of order *r* for the Burr Type III distribution.

Usage

```
dburrIII(x, scale, shape1, shape2, log = FALSE)

pburrIII(q, scale, shape1, shape2, lower.tail = TRUE, log.p = FALSE)

qburrIII(p, scale, shape1, shape2, lower.tail = TRUE, log.p = FALSE)

rburrIII(n, scale, shape1, shape2)

mburrIII(r, scale, shape1, shape2)
```

Arguments

x, q	vector of quantiles.
scale, shape1, shape2	scale and shape parameters; the shape arguments cannot be a vectors (must have length one).
log, log.p	logical; if TRUE, probabilities p are given as log(p).
lower.tail	logical; if TRUE (default), probabilities are $P[X \leq x]$ otherwise, $P[X > x]$.
p	vector of probabilities.
n	number of observations. If `length(n) > 1`, the length is taken to be the number required.
r	raw moment order

Examples

```
## plot the density

ggplot(data.frame(x = c(1, 15)),
       aes(x)) +
  stat_function(fun = dburrIII,
                args = list(scale = 5,
                            shape1 = .25,
                            shape2 = .75),
                colour = 'royalblue4') +
  labs(x = '',
       y = 'Density') +
  theme_classic()
```

BurrXII *Burr Type XII distribution*

Description

Provides density, distribution function, quantile function, random value generation, and raw moments of order *r* for the Burr Type XII distribution.

Usage

```
dburrXII(x, scale, shape1, shape2, log = FALSE)

pburrXII(q, scale, shape1, shape2, lower.tail = TRUE, log.p = FALSE)

qburrXII(p, scale, shape1, shape2, lower.tail = TRUE, log.p = FALSE)

rburrXII(n, scale, shape1, shape2)

mburrXII(r, scale, shape1, shape2)
```

Arguments

x, q	vector of quantiles.
scale, shape1, shape2	scale and shape parameters; the shape arguments cannot be a vectors (must have length one).
log, log.p	logical; if TRUE, probabilities p are given as log(p).
lower.tail	logical; if TRUE (default), probabilities are $P[X \leq x]$ otherwise, $P[X > x]$.
p	vector of probabilities.
n	number of observations. If length(n) > 1, the length is taken to be the number required.
r	raw moment order

Examples

```
## plot the density

ggplot(data.frame(x = c(0, 10)),
       aes(x)) +
  stat_function(fun = dburrXII,
                args = list(scale = 5,
                            shape1 = .25,
                            shape2 = .75),
                colour = 'royalblue4') +
  labs(x = '',
       y = 'Density') +
  theme_classic()
```

---

checkTS *Check generated timeseries*

Description

Compares generated time series sample statistics with the theoretically expected values.

Usage

```
checkTS(TS, distbounds = c(-Inf, Inf))
```

Arguments

TS	generated timeseries
distbounds	distribution bounds (default set to c(-Inf, Inf))

Examples

```
library(CoSMoS)

## check your generated timeseries
x <- generateTS(margdist = 'burrXII',
                margarg = list(scale = 1,
                               shape1 = .75,
                               shape2 = .25),
                acsvalue = acs(id = 'weibull',
                               t = 0:30,
                               scale = 10,
                               shape = .75),
                n = 1000, p = 30, p0 = .5, TSn = 5)

checkTS(x)
```

fitactf *Fit the AutoCorrelation Transformation Function*

Description

Fits the ACTF (Autocorrelation Transformation Function) to the estimated points (ρ_x, ρ_z) using nls

Usage

```
fitactf(actpnts, discrete = FALSE)
```

Arguments

actpnts	estimated ACT points
discrete	logical - is the marginal distribution discrete?

Examples

```
library(CoSMoS)

## choose the marginal distribution as Pareto type II with corresponding parameters
dist <- 'paretoII'
distarg <- list(scale = 1, shape = .3)

## estimate rho 'x' and 'z' points using ACTI
p <- actpnts(margdist = dist, margarg = distarg, p0 = 0)

## fit ACTF
fit <- fitactf(p)

## plot the result
plot(fit, method = 'base')
```

fitDist *Distribution fitting*

Description

Uses Nelder-Mead simplex algorithm to minimize fitting norms.

Usage

```
fitDist(data, dist, n.points, norm, constrain)
```

Arguments

data	value to be fitted
dist	name of the distribution to be fitted
n.points	number of points to be subsetted from ecdf
norm	norm used for distribution fitting - id ('N1', 'N2', 'N3', 'N4')
constrain	logical - constrain shape2 parametes for finite tails

Examples

```
x <- fitDist(rnorm(1000), 'norm', 30, 'N1', FALSE)
x
```

---

generateTS *Generate timeseries*

Description

Generates timeseries with given properties, just provide (1) the target marginal distribution and its parameters, (2) the target autocorrelation structure or individual autocorrelation values up to a desired lag, and (3) the probablility zero if you wish to simulate an intermittent process.

Usage

```
generateTS(n, margdist, margarg, p = NULL, p0 = 0, TSn = 1,
  distbounds = c(-Inf, Inf), acsvalue = NULL)
```

Arguments

n	number of values
margdist	target marginal distribution
margarg	list of marginal distribution arguments
p	integer - model order (if NULL - limits maximum model order according to auto-correlation structure values)
p0	probability zero
TSn	number of timeseries to be generated
distbounds	distribution bounds (default set to c(-Inf, Inf))
acsvalue	target auto-correlation structure (from lag 0)

Details

A step-by-step guide:

- First define the target marginal (margdist), that is, the probability distribution of the generated data. For example set margdist = 'ggamma' if you wish to generate data following the Generalized Gamma distribution, margidst = 'burrXII' for Burr type XII distribution etc. For a full list of the distributions we support see the help vignette: `vignette('vignette', package = 'CoSMoS')`. In general, the package supports all build-in distribution functions of R and of other packages.
- Define the parameters' values (margarg) of the distribution you selected. For example the Generalized Gamma has one scale and two shape parameters so set the desired value, e.g., margarg = list(scale = 2, shape1 = 0.9, shape2 = 0.8). Note distributions might have different number of parameters and different type of parameters (location, scale, shape). To see the parameters of each distribution we support, see the help vignette: `vignette('vignette', package = 'CoSMoS')`.
- If you wish your time series to be intermittent (e.g., precipitation), then define the probability zero. For example, set p0 = 0.9, if you wish your generated data to have 90% of zero values (dry days).
- Define your linear autocorrelations.
  - You can supply specific lag autocorrelations starting from lag 0 and up to a desired lag, e.g., acs = c(1, 0.9, 0.8, 0.7); this will generate a process with lag1, 2 and 3 autocorrelations equal with 0.9, 0.8 and 0.7.
  - Alternatively, you can use a parametric autocorrelation structure (see section 3.2 in Papalexiou 2018). We support the following autocorrelation structures (acs) weibull, paretoII, fgn and burrXII. See also acs examples.
- Define the order to the autoregressive model p. For example if you aim to preserve the first 10 lag autocorrelations then just set p = 10. Otherwise set it p = NULL and the model will decide the value of p in order to preserve the whole autocorrelation structure.
- Lastly just define the time series length, e.g., n = 1000 and number of time series you wish to generate, e.g., TSn = 10.

Play around with the following given examples which will make the whole process a piece of cake.

Examples

```
library(CoSMoS)

## Case1:
## You wish to generate 3 time series of size 1000 each
## that follow the Generalized Gamma distribution with parameters
## scale = 1, shape1 = 0.8, shape2 = 0.8
## and autocorrelation structure the ParetoII
## with parameters scale = 1 and shape = .75
x <- generateTS(margdist = 'ggamma',
                margarg = list(scale = 1,
                               shape1 = .8,
                               shape2 = .8),
                acsvalue = acs(id = 'paretoII',
                               t = 0:30,
```

```
                                scale = 1,
                                shape = .75),
                n = 1000,
                p = 30,
                TSn = 3)

## see the results
plot(x)



## Case2:
## You wish to generate time series the same distribution
## and autocorrelations as is Case1 but intermittent
## with probability zero equal to 90%
y <- generateTS(margdist = 'ggamma',
                margarg = list(scale = 1,
                                shape1 = .8,
                                shape2 = .8),
                acsvalue = acs(id = 'paretoII',
                                t = 0:30,
                                scale = 1,
                                shape = .75),
                p0 = .9,
                n = 1000,
                p = 30,
                TSn = 3)

## see the results
plot(y)

## Case3:
## You wish to generate a time series of size 1000
## that follows the Beta distribution
## (e.g., relative humidity ranging from 0 to 1)
## with parameters shape1 = 0.8, shape2 = 0.8, is defined from 0 to 1
## and autocorrelation structure the ParetoII
## with parameters scale = 1 and shape = .75
z <- generateTS(margdist = 'beta',
                margarg = list(shape1 = .6,
                                shape2 = .8),
                distbounds = c(0, 1),
                acsvalue = acs(id = 'paretoII',
                                t = 0:30,
                                scale = 1,
                                shape = .75),
                n = 1000,
                p = 20)

## see the results
plot(z)

## Case4:
```

```
## Same in previous case but now you provide specific
## autocorrelation values for the first three lags,
## ie.., lag 1 to 3 equal to 0.9, 0.8 and 0.7

z <- generateTS(margdist = 'beta',
                margarg = list(shape1 = .6,
                               shape2 = .8),
                distbounds = c(0, 1),
                acsvalue = c(1, .9, .8, .7),
                n = 1000,
                p = TRUE)

## see the results
plot(z)
```

GGamma *Generalized gamma distribution*

Description

Provides density, distribution function, quantile function, random value generation, and raw moments of order *r* for the generalized gamma distribution.

Usage

```
dggamma(x, scale, shape1, shape2, log = FALSE)

pggamma(q, scale, shape1, shape2, lower.tail = TRUE, log.p = FALSE)

qggamma(p, scale, shape1, shape2, lower.tail = TRUE, log.p = FALSE)

rggamma(n, scale, shape1, shape2)

mggamma(r, scale, shape1, shape2)
```

Arguments

x, q	vector of quantiles.
scale, shape1, shape2	scale and shape parameters; the shape arguments cannot be a vectors (must have length one).
log, log.p	logical; if TRUE, probabilities p are given as log(p).
lower.tail	logical; if TRUE (default), probabilities are $P[X \leq x]$ otherwise, $P[X > x]$.
p	vector of probabilities.

n	number of observations. If `length(n) > 1`, the length is taken to be the number required.
r	raw moment order

Examples

```
## plot the density

ggplot(data.frame(x = c(0, 20)),
       aes(x)) +
  stat_function(fun = dggamma,
                args = list(scale = 5,
                            shape1 = .25,
                            shape2 = .75),
                colour = 'royalblue4') +
  labs(x = '',
       y = 'Density') +
  theme_classic()
```

`moments` *Numerical estimation of moments*

Description

Uses numerical integration to caclulate the theoretical raw or central moments of the specified distribution

Usage

```
moments(dist, distarg, p0 = 0, raw = T, central = T, coef = T,
  distbounds = c(-Inf, Inf), order = 1:4)
```

Arguments

dist	distribution
distarg	list of distribution arguments
p0	probability zero
raw	logical - calculate raw moments?
central	logical - calculate central moments?
coef	logical - calculate coeffitients (coeffitient of variation, skweness and kurtosis)?
distbounds	distribution bounds (default set to c(-Inf, Inf))
order	vector of integers - raw moment orders

Examples

```
library(CoSMoS)

## Normal Distribution
moments('norm', list(mean = 2, sd = 1))

## Pareto type II
scale <- 1
shape <- .2

moments(dist = 'paretoII',
        distarg = list(shape = shape,
                       scale = scale))
```

`ParetoII` *Pareto type II distribution*

Description

Provides density, distribution function, quantile function, random value generation and raw moments of order *r* for the Pareto type II distribution.

Usage

```
dparetoII(x, scale, shape, log = FALSE)

pparetoII(q, scale, shape, lower.tail = TRUE, log.p = FALSE)

qparetoII(p, scale, shape, lower.tail = TRUE, log.p = FALSE)

rparetoII(n, scale, shape)

mparetoII(r, scale, shape)
```

Arguments

`x, q`	vector of quantiles.
`scale, shape`	scale and shape parameters; the shape argument cannot be a vector (must have length one).
`log, log.p`	logical; if TRUE, probabilities p are given as log(p).
`lower.tail`	logical; if TRUE (default), probabilities are $P[X \leq x]$ otherwise, $P[X > x]$.
`p`	vector of probabilities.
`n`	number of observations. If `length(n) > 1`, the length is taken to be the number required.
`r`	raw moment order

Examples

```
## plot the density

ggplot(data.frame(x = c(0, 20)),
       aes(x)) +
  stat_function(fun = dparetoII,
                args = list(scale = 1,
                            shape = .3),
                colour = 'royalblue4') +
  labs(x = '',
       y = 'Density') +
  theme_classic()
```

`plot.acti` *AutoCorrelation Transformation Function visualisation*

Description

Visualizes the autocorrelation tranformation integral (there are two possible methods for plotting - base graphics and ggplot2 package)

Usage

```
## S3 method for class 'acti'
plot(x, ...)
```

Arguments

`x`	fitactf result object
`...`	other arguments

Examples

```
library(CoSMoS)

## choose the marginal distribution as Pareto type II with corresponding parameters
dist <- 'paretoII'
distarg <- list(scale = 1, shape = .3)

## estimate rho 'x' and 'z' points using ACTI
p <- actpnts(margdist = dist, margarg = distarg, p0 = 0)

## fit ACTF
fit <- fitactf(p)

## plot the results
```

```
plot(fit)
plot(fit, main = 'Pareto type II distribution \nautocorrelation tranformation')
```

`plot.checkTS` *Plot method for check results*

Description

Plot method for check results

Usage

```
## S3 method for class 'checkTS'
plot(x, ...)
```

Arguments

`x`	check result
`...`	other args

Examples

```
library(CoSMoS)

## check your generated timeseries
x <- generateTS(margdist = 'burrXII',
                margarg = list(scale = 1,
                               shape1 = .75,
                               shape2 = .15),
                acsvalue = acs(id = 'weibull',
                               t = 0:30,
                               scale = 10,
                               shape = .75),
                n = 10000, p = 30, p0 = .25, TSn = 100)

chck <- checkTS(x)

plot(chck)
```

plot.cosmosts *Plot generated Timeseries*

Description

Visualizes Timeseries generated by the package CoSMoS

Usage

```
## S3 method for class 'cosmosts'
plot(x, ...)
```

Arguments

x	fitactf result object
...	other arguments

Examples

```
library(CoSMoS)

## generate TS
ts <- generateTS(margdist = 'ggamma',
                 margarg = list(scale = 1,
                                shape1 = .8,
                                shape2 = .8),
                 acsvalue = acs(id = 'paretoII',
                                t = 0:30,
                               scale = 1,
                                shape = .75),
                 n = 1000,
                 p = 30,
                 TSn = 2)

## plot the TS
plot(ts)
```

precip *Hourly station precipitation data*

Description

Station details

- Name: Philadelphia International Airport
- Network ID: COOP:366889
- Latitude/Longitude: 39.87327°, -75.22678°
- Elevation: 3m

Usage

```
precip
```

Format

A data.table with 79633 rows and 2 variables:

date POSIXct format date/time

value precipitation totals

Details

more details can be found here.

Source

The National Oceanic and Atmospheric Administration (NOAA)

---

regenerateTS *Bulk Timeseries generation*

Description

Resamples given Timeseries

Usage

```
regenerateTS(ts, TSn = 1)
```

Arguments

ts	generated timeseries using ARp
TSn	number of timeseries to be (re)generated

Details

You have used the generateTS function and you wish to generate more time series. Instead of re-running the generateTS you can use the regenerateTS Timeseries that used all the parameters calculated by the generateTS function and thus it is faster.

Examples

```
library(CoSMoS)

## define marginal distribution and arguments with target autocorrelation structure
x <- generateTS(margdist = 'burrXII',
                margarg = list(scale = 1,
                               shape1 = .75,
                               shape2 = .25),
                acsvalue = acs(id = 'weibull',
                               t = 0:30,
                               scale = 10,
                               shape = .75),
                n = 1000, p = 30, p0 = .5, TSn = 3)

## generate new values with same parameters
r <- regenerateTS(x)

plot(r)
```

`sample.moments` *Estimation of sample moments*

Description

Estimation of sample moments

Usage

```
sample.moments(x, na.rm = FALSE, raw = T, central = T, coef = T,
  order = 1:4)
```

Arguments

`x`	a numeric vector of values
`na.rm`	a logical value indicating whether NA values should be stripped before the computation proceeds
`raw`	logical - calculate raw moments?
`central`	logical - calculate central moments?
`coef`	logical - calculate coeffitients (coeffitient of variation, skweness and kurtosis)?
`order`	vector of integers - raw moment orders

Examples

```
library(CoSMoS)

x <- rnorm(1000)
sample.moments(x)

y <- rparetoII(1000, 10, .1)
sample.moments(y)
```

Index