Package ‘SNSeg’

August 26, 2021

Title Self-normalization based change point estimation for time series

Version 1.0.0

Description Implements Self-normaliztion approach in estimating change points for univariate and multivariate time series data.

License MIT + file LICENSE

Encoding UTF-8

LazyData true

Roxygen list(markdown = TRUE)

RoxygenNote 7.1.1

Depends R (>= 3.5.0), stats, utils, graphics

LinkingTo Rcpp

Imports Rcpp, mvtnorm, truncnorm, evd

NeedsCompilation yes

Author Zifeng Zhao [aut, com],
Shubo Sun [com, cre],
Xiaofeng Shao [aut, ths],
Feiyu Jiang [ths]

Maintainer Shubo Sun <shubos2@illinois.edu>

Archs i386, x64

R topics documented:

critical_values_HD 2
critical_values_multi 2
critical_values_single 3
MAR 3
MAR_MTS_Covariance 4
MAR_Variance 4
max_SNsweep 5
SNSeg 6
SNSeg_HD 7
SNSeg_Multi 9
SNSeg_Uni_multi_para 11
SNSeg_Uni_single_para 13

Index **16**

critical_values_HD *Critical Values of Self-normalization for high-dimensional time series (SNHD)*

Description

A dataset containing the critical value of each window size at each confidence level for SNHD.

Usage

```
critical_values_HD
```

Format

A data frame with 6 variables:

`epsilon` window size to estimate change points

`0.9` critical value at confidence level `0.9`

`0.95` critical value at confidence level `0.95`

`0.99` critical value at confidence level `0.99`

`0.995` critical value at confidence level `0.995`

`0.999` critical value at confidence level `0.999`

critical_values_multi *Critical Values of Self-normalization for multi-parameters*

Description

A dataset containing the critical value of each window size at each confidence level for Self-normalization change points estimate based on multi-parameters.

Usage

```
critical_values_multi
```

Format

A data frame with 7 variables:

`epsilon` window size to estimate change points

`p` dimension of the multi-parameters

`0.9` critical value at confidence level `0.9`

`0.95` critical value at confidence level `0.95`

`0.99` critical value at confidence level `0.99`

`0.995` critical value at confidence level `0.995`

`0.999` critical value at confidence level `0.999` critical_values_single   *Critical Values of Self-normalization for single-parameter*

Description

A dataset containing the critical value of each window size at each confidence level for Self-normalization change points estimate based on one single parameter.

Usage

```
critical_values_single
```

Format

A data frame with 6 variables:

- `epsilon` window size to estimate change points
- `0.9` critical value at confidence level `0.9`
- `0.95` critical value at confidence level `0.95`
- `0.99` critical value at confidence level `0.99`
- `0.995` critical value at confidence level `0.995`
- `0.999` critical value at confidence level `0.999`

MAR   *A Funtion to generate a multivariate autoregressive process (MAR) model in time series*

Description

The function `MAR` is used for generating MAR model(s) for examples under `SNSeg_Uni_single_para`, `SNSeg_Uni_multi_para`, and `SNSeg_Multi`.

Usage

```
MAR(n, reptime, rho)
```

Arguments

`n`	the size of time series to be generated
`reptime`	the number of time series to be generated
`rho`	a correlation factor

`MAR_MTS_Covariance` *A Funtion to generate a multivariate autoregressive process (MAR) model in time series (used for testing change points based on multivariate mean, covariance and bivariate correlation)*

Description

The function `MAR_MTS_Covariance` is used for generating MAR model(s) for examples under `SNSeg_Uni_single_para`, `SNSeg_Uni_multi_para`, and `SNSeg_Multi`.

Usage

```
MAR_MTS_Covariance(n, reptime, rho_sets, cp_sets, sigma_cross)
```

Arguments

`n`	the size of time series to be generated
`reptime`	the number of time series to be generated
`rho_sets`	correlation factors for simulation
`cp_sets`	the critical points set to check SN prediction accuracy
`sigma_cross`	a list to generate the covariance structure of the time series

---

`MAR_Variance` *A Funtion to generate a multivariate autoregressive process (MAR) model in time series (used for testing change points based on variance and ACF)*

Description

The function `MAR_Variance` is used for generating MAR model(s) for examples under `SNSeg_Uni_single_para`, `SNSeg_Uni_multi_para`, and `SNSeg_Multi`.

Usage

```
MAR_Variance(reptime, type = "V3")
```

Arguments

`reptime`	the number of time series to be generated
`type`	the type of time series for simulation, which includes V1, V2, V3 , A1, A2 and A3. The V-beginnings are for testing variance, and the A-beginnings are for testing ACF.

max_SNsweep — *Self-normalization test statistics segmentation plot for univariate and multivariate time series*

Description

The function max_SNsweep is a single-parameter change point estimation framework for a multivariate time series using the self-normalized approach.

Usage

```
max_SNsweep(SN_sweep_result, plot_SN = TRUE)
```

Arguments

SN_sweep_result
: a list of matrices containing the SN test statistics from SNSeg_Uni_single_para, SNSeg_Uni_multi_para or SNSeg_Multi

plot_SN
: Only if argument plot_SN is TRUE will return a SN test statistic segmentation plot

Examples

```
# Please run this function before running examples:
exchange_cor_matrix <- function(d, rho){
  tmp <- matrix(rho, d, d)
  diag(tmp) <- 1
  return(tmp)
}

# univariate time series based on single parameter
set.seed(7)
n <- 2000
reptime <- 2
cp_sets <- round(n*c(0,cumsum(c(0.5,0.25)),1))
mean_shift <- c(0.4,0,0.4)
rho <- -0.7
ts <- MAR(n, reptime, rho)
no_seg <- length(cp_sets)-1
for(index in 1:no_seg){ # Mean shift
  tau1 <- cp_sets[index]+1
  tau2 <- cp_sets[index+1]
  ts[tau1:tau2,] <- ts[tau1:tau2,] + mean_shift[index]
}
ts <- ts[,2]

# calculating SN test statistics and change points
result <- SNSeg_Uni_single_para(ts,type = "mean", confidence = 0.9,
grid_size_scale = 0.061, grid_size = NULL, plot_SN = FALSE)
# SN test statistic segmentation plot
max_SNsweep(result$SN_sweep_result, plot_SN = TRUE)

# univariate time series based on multi-parameters
set.seed(7)
```

```
n <- 1000
cp_sets <- c(0,333,667,1000)
no_seg <- length(cp_sets)-1
rho <- 0
# AR time series with no change-point (mean, var)=(0,1)
ts <- MAR(n, 2, rho)*sqrt(1-rho^2)
no_seg <- length(cp_sets)-1
sd_shift <- c(1,1.6,1)
for(index in 1:no_seg){ # Mean shift
  tau1 <- cp_sets[index]+1
  tau2 <- cp_sets[index+1]
  ts[tau1:tau2,] <- ts[tau1:tau2,]*sd_shift[index]
}
d <- 2
ts <- ts[,2]

# calculating SN test statistics and change points
result <- SNSeg_Uni_multi_para(ts, paras_to_test = c(0.8,'mean',"variance",
"acf"), confidence = 0.9, grid_size_scale = 0.05, grid_size = 83, d = 4,
plot_SN = FALSE)
# SN test statistic segmentation plot
max_SNsweep(result$SN_sweep_result, plot_SN = TRUE)

# multivariate time series (only support one single parameter!)
set.seed(10)
d <- 5
n <- 1000
cp_sets <- round(n*c(0,cumsum(c(0.075,0.3,0.05,0.1,0.05)),1))
mean_shift <- c(-3,0,3,0,-3,0)/sqrt(d)
mean_shift <- sign(mean_shift)*ceiling(abs(mean_shift)*10)/10
rho_sets <- 0.5
sigma_cross <- list(exchange_cor_matrix(d,0))
ts <- MAR_MTS_Covariance(n, 2, rho_sets, cp_sets=c(0,n), sigma_cross)
noCP <- length(cp_sets)-2
no_seg <- length(cp_sets)-1
for(rep_index in 1:2){
  for(index in 1:no_seg){ # Mean shift
    tau1 <- cp_sets[index]+1
    tau2 <- cp_sets[index+1]
    ts[[rep_index]][,tau1:tau2] <- ts[[rep_index]][,tau1:tau2] + mean_shift[index]
  }
}
ts <- ts[1][[1]]

# calculating SN test statistics and change points
result <- SNSeg_Multi(ts, type = "mean", confidence = 0.9,
grid_size_scale = 0.079, grid_size = NULL)
# SN test statistic segmentation plot
max_SNsweep(result$SN_sweep_result, plot_SN = TRUE)
```

SNSeg *SNSeg: A package for Self-normalization (SN) on change points estimation in time series.*

Description

The SNuni package provides three important functions: `SNSeg_Uni_single_para`, `SNSeg_Uni_multi_para` and `SNSeg_Multi`. Since SN-based testing approach uses test statistic to make change point estimates, two critical value tables (`critical_values_single` and `critical_values_multi`) were attached.

SNSeg_Uni_single_para

`SNSeg_Uni_single_para` provides change points estimates for a univariate time series based on changes in a single parameter using self-normalized approach.

For the parameters of the user-defined time series, the function `SNSeg_Uni_single_para` offers mean, variance, acf, bivariate correlation and quantiles as avaliale choice. To visualize the changes in time series, users can set "plot_SN = TRUE" to see the SN segmentation plot. The output includes the type of the parameter, the minimal window size to contain a potential change point, and the lag(s) where the change point(s) occur.

SNSeg_Uni_multi_para

`SNSeg_Uni_mul_para` provides change points estimates for a univariate time series based on changes in multi-parameters using self-normalized approach.

Different from `SNSeg_Uni_single_para`, `SNSeg_Uni_multi_para` allows users to place multiple parameters as input. Users can also get the segmentation plot by setting "plot_SN = TRUE". The output is the same as those of `SNSeg_Uni_single_para`.

SNSeg_Multi

`SNSeg_Multi` provides change points estimates for a multivariate time series based on changes in single-parameter using self-normalized approach.

Different from `SNSeg_Uni_single_para` and `SNSeg_Uni_mul_para`, `SNSeg_Multi` does not provide the segmentation plots for users. Users can plot each of the time series by "plot()" and add "abline(v = ...)", of which "..." represents the change point location from `SNSeg_Multi`'s output. The other output of `SNSeg_Multi` is the same as the previous two functions.

critical values table

The package `SNuni` provides two critical values table.

Table `critical_values_single` records critical values of each window size at each confidence level for SN change points estimate based on one single parameter.

Table `critical_values_multi` records critical values of each window size at each confidence level for SN change points estimate based on multi-parameters.

`SNSeg_HD` *Self-normalization change points estimation for high dimensional time series based on changes in multi-means (SNHD).*

Description

The function `SNSeg_HD` is a SNHD change point estimation framework.

Usage

```
SNSeg_HD(ts, confidence = 0.9, grid_size_scale = 0.05, grid_size = NULL)
```

Arguments

`ts`	numeric data of a univariate time series
`confidence`	the confidence level that can be expected to produce a significant Self-normalization test statistic. The avaliable confidence levels are 0.9, 0.95, 0.99, 0.995 and 0.999.
`grid_size_scale`	the window size parameter to determine the mimimum range where an estimated change point for a univariate time series can occur. The grid_size_scale can be selected from the following: 0.05, 0.06, 0.07, 0.08, 0.09, 0.1, 0.11, 0.12, 0.13, 0.14, 0.15, 0.2, 0.25, 0.3, 0.35, 0.4, 0.45 and 0.5.
`grid_size`	the window size to cover an estimated change point, which is calculated from grid_size_scale. The function depends on the value of grid_size instead of grid_size_scale within input, and only when grid_size is NULL will the function use the grid_size_scale in the input.

Value

SNSeg_HD returns the minimum window size to cover a change point, and the lags of which all estimated change points take place.

`grid_size` the minimal window size to detect a potential change point

`SN_sweep_result` a list of matrices to record the test statistics, the location of the estimated change points, and the range of the window set to contain each change point

`est_cp` the estimated change points for the given time series

`confidence` the confidence level for SN tests

`critical_value` the critical value for SN change points estimation

Examples

```
n <- 600
p <- 100
nocp <- 5
cp_sets <- round(seq(0,nocp+1,1)/(nocp+1)*n)
num_entry <- 5
kappa <- sqrt(4/5) # Wang et al(2020)
mean_shift <- rep(c(0,kappa),100)[1:(length(cp_sets)-1)]
set.seed(1)
ts <- matrix(rnorm(n*p,0,1),n,p)
no_seg <- length(cp_sets)-1
for(index in 1:no_seg){ # Mean shift
  tau1 <- cp_sets[index]+1
  tau2 <- cp_sets[index+1]
 ts[tau1:tau2,1:num_entry] <- ts[tau1:tau2,1:num_entry] + mean_shift[index] # sparse change
}
# grid_size undefined
SNSeg_HD(ts, confidence = 0.9, grid_size_scale = 0.05,
         grid_size = NULL)
# grid_size defined
SNSeg_HD(ts, confidence = 0.9, grid_size_scale  = 0.05,
         grid_size = 52)
```

SNSeg_Multi — *Self-normalization change points estimation for multivariate time series based on single parameter changes*

Description

The function SNSeg_Uni is a single-parameter change point estimation framework for a multivariate time series using the self-normalized approach.

Usage

```
SNSeg_Multi(
  ts,
  type = "mean",
  confidence = 0.9,
  grid_size = NULL,
  grid_size_scale = 0.05
)
```

Arguments

ts	numeric data of a multivariate time series
type	the type of parameters of time series data that SN depends, which includes mean and covariance.
confidence	the confidence level that can be expected to produce a significant Self-normalization test statistic
grid_size	the window size to cover an estimated change point, which is calculated from grid_size_scale. The function depends on the value of grid_size instead of grid_size_scale within input, and only when grid_size is NULL will the function use the grid_size_scale in the input.
grid_size_scale	the window size parameter to determine the mimimum range where an estimated change point for a time series can occur

Value

SNSeg_Multi returns the type of the multivariate time series, the minimum window size to cover a change point, and the lags of which all estimated change points take place.

type the type of parameter being used for self-normalization

grid_size the minimal window size to detect a potential change point

SN_sweep_result a list of matrices to record the test statistics, the location of the estimated change points, and the range of the window set to contain each change point

est_cp the estimated change points for the given time series

confidence the confidence level for SN tests

critical_value the critical value for SN change points estimation

Examples

```
# Please run this function before running examples:
exchange_cor_matrix <- function(d, rho){
  tmp <- matrix(rho, d, d)
  diag(tmp) <- 1
  return(tmp)
}

# SN Segmentation for Multivariate Mean
library(mvtnorm)
set.seed(10)
d <- 5
n <- 1000
cp_sets <- round(n*c(0,cumsum(c(0.075,0.3,0.05,0.1,0.05)),1))
mean_shift <- c(-3,0,3,0,-3,0)/sqrt(d)
mean_shift <- sign(mean_shift)*ceiling(abs(mean_shift)*10)/10
rho_sets <- 0.5
sigma_cross <- list(exchange_cor_matrix(d,0))
ts <- MAR_MTS_Covariance(n, 2, rho_sets, cp_sets=c(0,n), sigma_cross)
noCP <- length(cp_sets)-2
no_seg <- length(cp_sets)-1
for(rep_index in 1:2){
  for(index in 1:no_seg){ # Mean shift
    tau1 <- cp_sets[index]+1
    tau2 <- cp_sets[index+1]
    ts[[rep_index]][,tau1:tau2] <- ts[[rep_index]][,tau1:tau2] + mean_shift[index]
  }
}
ts <- ts[1][[1]]

# grid_size undefined
SNSeg_Multi(ts, type = "mean", confidence = 0.95, grid_size_scale = 0.079,
            grid_size = NULL)
# grid_size defined
SNSeg_Multi(ts, type = "mean", confidence = 0.99, grid_size_scale = 0.05,
            grid_size = 65)

# SN Segmentation for Multivariate Covariance
library(mvtnorm)
set.seed(10)
reptime <- 2
d <- 4
n <- 1000
sigma_cross <- list(exchange_cor_matrix(d,0.2), 2*exchange_cor_matrix(d,0.5),
                    4*exchange_cor_matrix(d,0.5))
rho_sets <- c(0.3,0.3,0.3)
mean_shift <- c(0,0,0) # with mean change
cp_sets <- round(c(0,n/3,2*n/3,n))
# 2-dimensional AR time series with change-point in bivariate correlation
ts <- MAR_MTS_Covariance(n, reptime, rho_sets, cp_sets, sigma_cross)
noCP <- length(cp_sets)-2
no_seg <- length(cp_sets)-1
for(rep_index in 1:reptime){
  for(index in 1:no_seg){ # Mean shift
    tau1 <- cp_sets[index]+1
    tau2 <- cp_sets[index+1]
```

```
    ts[[rep_index]][,tau1:tau2] <- ts[[rep_index]][,tau1:tau2] + mean_shift[index]
  }
}
ts <- ts[[1]]
SNSeg_Multi(ts, type = "covariance", confidence = 0.9, grid_size_scale = 0.05,
            grid_size = NULL)
SNSeg_Multi(ts, type = "covariance", confidence = 0.9, grid_size_scale = 0.05,
            grid_size = 81)
```

`SNSeg_Uni_multi_para` *Self-normalization change point estimates for univariate time series based on multi-parameters' changes*

Description

The function `SNSeg_Uni_multi_para` is a multi-parameter change point estimation framework for a univariate time series using the self-normalized approach.

Usage

```
SNSeg_Uni_multi_para(
  ts,
  paras_to_test = c(0.9, 0.95),
  confidence = 0.9,
  grid_size_scale = 0.05,
  grid_size = NULL,
  d = 2,
  plot_SN = FALSE
)
```

Arguments

`ts`	numeric data of a univariate time series
`paras_to_test`	the multi-parameters of ts to be measured and tested. The type(s) of input for paras_to_test includes "mean", "variance", "acf", and all numeric value of quantile(s) between 0 and 1.
`confidence`	the confidence level that can be expected to produce a significant Self-normalization test statistic
`grid_size_scale`	the window size parameter to determine the mimimum range where an estimated change point for a time series can occur
`grid_size`	the window size to cover an estimated change point, which is calculated from grid_size_scale. The function depends on the value of grid_size instead of grid_size_scale within input, and only when grid_size is NULL will the function use the grid_size_scale in the input.
`d`	the dimension of paras_to_test
`plot_SN`	Only if argument plot_SN is TRUE will return a SN segmentation plot

Value

SNSeg_Uni_multi_para returns the type of the multivariate time series, the minimum window size to cover a change point, and the lags where all estimated change points take place.

parameter the type of parameter being used for self-normalization

grid_size the minimal window size to detect a potential change point

SN_sweep_result a list of matrices to record the test statistics, the location of the estimated change points, and the range of the window set to contain each change point

est_cp the estimated change points for the given time series

confidence the confidence level for SN tests

critical_value the critical value for SN change points estimation

Examples

```
set.seed(7)
n <- 1000
cp_sets <- c(0,333,667,1000)
no_seg <- length(cp_sets)-1
rho <- 0
# AR time series with no change-point (mean, var)=(0,1)
ts <- MAR(n, 2, rho)*sqrt(1-rho^2)
no_seg <- length(cp_sets)-1
sd_shift <- c(1,1.6,1)
for(index in 1:no_seg){ # Mean shift
  tau1 <- cp_sets[index]+1
  tau2 <- cp_sets[index+1]
  ts[tau1:tau2,] <- ts[tau1:tau2,]*sd_shift[index]
}
d <- 2
ts <- ts[,2]

# 90th and 95th quantile with grid_size undefined
SNSeg_Uni_multi_para(ts, paras_to_test = c(0.9,0.95),
                     confidence = 0.9, grid_size_scale = 0.05, grid_size = NULL,
                     d = 2, plot_SN = FALSE)

# 90th quantile and the variance with grid_size undefined
SNSeg_Uni_multi_para(ts, paras_to_test = c(0.9,'variance'),
                     confidence = 0.95, grid_size_scale = 0.078, grid_size = NULL,
                     d = 2, plot_SN = FALSE)

# 90th quantile, variance and acf with grid_size undefined
SNSeg_Uni_multi_para(ts, paras_to_test = c(0.9,'variance',"acf"),
                     confidence = 0.9, grid_size_scale = 0.064, grid_size = NULL,
                     d = 3, plot_SN = TRUE)

# 60th quantile, mean, variance and acf with grid_size defined
SNSeg_Uni_multi_para(ts, paras_to_test = c(0.6,'mean',"variance","acf"),
                     confidence = 0.9, grid_size_scale = 0.05, grid_size = 83,
                     d = 4, plot_SN = TRUE)
```

SNSeg_Uni_single_para *Self-normalization change points estimation for univariate time series based on one single parameter changes*

Description

The function SNSeg_Uni_single_para is a single-parameter change point estimation framework for a univariate time series using the self-normalized approach.

Usage

```
SNSeg_Uni_single_para(
  ts,
  type = "mean",
  confidence = 0.9,
  quantile_level = 0.1,
  grid_size_scale = 0.05,
  grid_size = NULL,
  plot_SN = FALSE
)
```

Arguments

ts	numeric data of a univariate time series
type	the type of parameters of time series data that SN depends, which includes mean, variance, acf, bivcor and quantile.
confidence	the confidence level that can be expected to produce a significant Self-normalization test statistic. The avaliable confidence levels are 0.9, 0.95, 0.99, 0.995 and 0.999.
quantile_level	the quantile level of the given time series
grid_size_scale	the window size parameter to determine the mimimum range where an estimated change point for a univariate time series can occur. The grid_size_scale can be selected from the following: 0.05, 0.06, 0.07, 0.08, 0.09, 0.1, 0.11, 0.12, 0.13, 0.14, 0.15, 0.2, 0.25, 0.3, 0.35, 0.4, 0.45 and 0.5.
grid_size	the window size to cover an estimated change point, which is calculated from grid_size_scale. The function depends on the value of grid_size instead of grid_size_scale within input, and only when grid_size is NULL will the function use the grid_size_scale in the input.
plot_SN	Only if argument plot_SN is TRUE will return an SN segmentation plot

Value

SNSeg_Uni_single_para returns the type of the univariate time series, the minimum window size to cover a change point, and the lags of which all estimated change points take place.

`type` the type of parameter being used for self-normalization

`grid_size` the minimal window size to detect a potential change point

`SN_sweep_result` a list of matrices to record the test statistics, the location of the estimated change points, and the range of the window set to contain each change point `est_cp` the estimated change points for the given time series

`confidence` the confidence level for SN tests

`critical_value` the critical value for SN change points estimation

Examples

```
# Please run the following function before running examples:
mix_GauGPD <- function(u,p,trunc_r,gpd_scale,gpd_shape){
  indicator <- u<p
  rv <- rep(0, length(u))
  rv[indicator>0] <- qtruncnorm(u[indicator>0]/p,a=-Inf,b=trunc_r)
 rv[indicator<=0] <- qgpd((u[indicator<=0]-p)/(1-p),loc=trunc_r,scale=gpd_scale,shape=gpd_shape)
  return(rv)
}

# SN Segmentation for Mean
set.seed(7)
n <- 2000
reptime <- 2
cp_sets <- round(n*c(0,cumsum(c(0.5,0.25)),1))
mean_shift <- c(0.4,0,0.4)
rho <- -0.7
ts <- MAR(n, reptime, rho)
no_seg <- length(cp_sets)-1
for(index in 1:no_seg){ # Mean shift
  tau1 <- cp_sets[index]+1
  tau2 <- cp_sets[index+1]
  ts[tau1:tau2,] <- ts[tau1:tau2,] + mean_shift[index]
}
ts <- ts[,2]
# grid_size undefined
SNSeg_Uni_single_para(ts,type = "mean", confidence = 0.9,
       grid_size_scale = 0.05, grid_size = NULL, plot_SN = FALSE)
# grid_size defined
SNSeg_Uni_single_para(ts,type = "mean", confidence = 0.9,
       grid_size_scale = 0.05, grid_size = 116, plot_SN = FALSE)

# SN Segmentation for Variance
set.seed(7)
ts <- MAR_Variance(2, "V1")
ts <- ts[,2]
# grid_size undefined
SNSeg_Uni_single_para(ts,type = "variance", confidence = 0.9,
       grid_size_scale = 0.067, grid_size = NULL, plot_SN = FALSE)
# grid_size defined
SNSeg_Uni_single_para(ts,type = "variance", confidence = 0.9,
       grid_size_scale = 0.05, grid_size = 67, plot_SN = FALSE)

# SN Segmentation for ACF
set.seed(7)
ts <- MAR_Variance(2, "A3")
ts <- ts[,2]
# grid_size undefined
SNSeg_Uni_single_para(ts,type = "acf", confidence = 0.9,
       grid_size_scale = 0.05, grid_size = NULL, plot_SN = FALSE)
# grid_size defined
```

```
SNSeg_Uni_single_para(ts,type = "acf", confidence = 0.9,
       grid_size_scale = 0.05, grid_size = 92, plot_SN = FALSE)

# SN Segmentation for bivariate correlation
library(mvtnorm)
set.seed(7)
n <- 1000
sigma_cross <- list(4*matrix(c(1,0.8,0.8,1), nrow=2),
      matrix(c(1,0.2,0.2,1), nrow=2), matrix(c(1,0.8,0.8,1), nrow=2))
cp_sets <- round(c(0,n/3,2*n/3,n))
noCP <- length(cp_sets)-2
rho_sets <- rep(0.5, noCP+1)
ts <- MAR_MTS_Covariance(n, 2, rho_sets, cp_sets, sigma_cross)
ts <- ts[1][[1]]
# grid_size undefined
SNSeg_Uni_single_para(ts,type = "bivcor", confidence = 0.9,
        grid_size_scale = 0.05, grid_size = NULL, plot_SN = TRUE)
# grid_size defined
SNSeg_Uni_single_para(ts,type = "bivcor", confidence = 0.9,
        grid_size_scale = 0.05, grid_size = 77, plot_SN = TRUE)

# SN Segmentation for quantile
library(truncnorm)
library(evd)
set.seed(7)
n <- 1000
cp_sets <- c(0,n/2,n)
noCP <- length(cp_sets)-2
reptime <- 2
rho <- 0.2
# AR time series with no change-point (mean, var)=(0,1)
ts <- MAR(n, reptime, rho)*sqrt(1-rho^2)
trunc_r <- 0
p <- pnorm(trunc_r)
gpd_scale <- 2
gpd_shape <- 0.125
for(ts_index in 1:reptime){
  ts[(cp_sets[2]+1):n, ts_index] <- mix_GauGPD(pnorm(ts[(cp_sets[2]+1):n, ts_index]),
                                     p,trunc_r,gpd_scale,gpd_shape)
}
ts <- ts[,2]
SNSeg_Uni_single_para(ts,type = "quantile", confidence = 0.9, quantile_level = 0.9,
        grid_size_scale = 0.066, grid_size = NULL, plot_SN = TRUE)
SNSeg_Uni_single_para(ts,type = "quantile", confidence = 0.9, quantile_level = 0.8,
        grid_size_scale = 0.05, grid_size = 102, plot_SN = TRUE)
```

Index

∗ **datasets**
critical_values_HD, 2
critical_values_multi, 2
critical_values_single, 3

critical_values_HD, 2
critical_values_multi, 2
critical_values_single, 3

MAR, 3
MAR_MTS_Covariance, 4
MAR_Variance, 4
max_SNsweep, 5

SNSeg, 6
SNSeg_HD, 7
SNSeg_Multi, 9
SNSeg_Uni_multi_para, 11
SNSeg_Uni_single_para, 13