Package ‘tuts’

June 12, 2018

Type Package

Title Time Uncertain Time Series Analysis

Version 0.1.1

Date 2018-06-12

Description Models of time-uncertain time series addressing frequency and non-frequency behavior of continuous and discrete (counting) data.

License GPL (>= 2)

Depends R (>= 3.4.0), rjags

Encoding UTF-8

LazyData true

RoxygenNote 6.0.1

Imports coda, doParallel, foreach, lomb, mcmcplots, parallel, stats, truncnorm

NeedsCompilation no

Author Peter Franke [aut, cre], Andrew Parnell [aut]

Maintainer Peter Franke <peter.franke@ucdconnect.ie>

Repository CRAN

Date/Publication 2018-06-12 20:52:55 UTC

R topics documented:

`JAGS.objects` *JAGS output an internal function used within the tuts package*

Description

`JAGS.objects` an internal function used within the tuts package.

Usage

```
JAGS.objects(JAGS.output)
```

Arguments

`JAGS.output`	An JAGS MCMC Object.

`plot.tuts_ar1` *Plots and visual diagnostics of tuts_ar1 objects*

Description

`plot.tuts_ar1(x,type,...)` generates plots and visual diagnostics of tuts_ar1 objects.

Usage

```
## S3 method for class 'tuts_ar1'
plot(x, type, ...)
```

Arguments

x
: A tuts_ar1 object.

type
: plot type with the following options:
- 'predTUTS' plots one step predictions of the model.
- 'par' plots distributions of parameters of the model.
- 'volatility' plots volatility realizations.
- 'GR' plots Gelman-Rubin diagnostics.
- 'cv' plots 5-fold cross validation.
- 'mcmc' plots diagnostics of MCMC/JAGS objects.

...
: list of optional parameters:
- burn: burn-in parameter ranging from 0 to 0.7 with default value set to 0.
- CI: credible interval ranging from 0.3 to 1 with default value set to 0.95.

Examples

```
# Note: Most of models included in tuts package are computationally intensive. In the example
# below I set parameters to meet CRAN's testing requirement of maximum 5 sec per example.
# A more practical example would contain N=50 in the first line of the code and n.sim=10000.

#1. Import or simulate the data (a simulation is chosen for illustrative purposes):
DATA=simtuts(N=10,Harmonics=c(4,0,0), sin.ampl=c(10,0, 0), cos.ampl=c(0,0,0),
            trend=0,y.sd=2, ti.sd=0.2)
y=DATA$observed$y.obs
ti.mu=DATA$observed$ti.obs.tnorm
ti.sd= rep(0.2, length(ti.mu))

#2. Fit the model:
n.sim=1000
TUAR1=tuar1(y=y,ti.mu=ti.mu,ti.sd=ti.sd,n.sim=n.sim,CV=TRUE,n.cores=2)

#3. Generate plots and diagnostics of the model (optional parameters are listed in brackets):
plot(TUAR1,type='predTUTS')            # One step out of salmple predictions (CI, burn).
plot(TUAR1,type='par', burn=0.4)         # Distributions of parameters (burn).
plot(TUAR1,type='mcmc')                  # MCMC diagnostics.
plot(TUAR1,type='cv', burn=0.4, CI=0.9)   # 5 fold cross validation (CI, burn).
plot(TUAR1,type='GR')                    # Gelman-Rubin diagnostic (CI, burn).
plot(TUAR1,type='volatility')            # Volatility realizaitons.
```

---

plot.tuts_ar1redf *Plots and visual diagnostics of tuts_ar1redf objects*

Description

`plot.tuts_ar1redf` generates plots and visual diagnostics of tuts_ar1redf objects.

Usage

```
## S3 method for class 'tuts_ar1redf'
plot(x, type, ...)
```

Arguments

x	A tuts_tuar1 objects.
type	plot type with the following options: - 'predTUTS' plots one step predictions of the model. - 'par' plots distributions of parameters of the model. - 'tau' plots realizations of time persistence. - 'GR' plots Gelman-Rubin diagnostics. - 'cv' plots 5-fold cross validation. - 'mcmc' plots diagnostics of MCMC/JAGS objects.
...	list of optional parameters: - burn: burn-in parameter ranging from 0 to 0.7 with default value set to 0. - CI: credible interval ranging from 0.3 to 1 with default value set to 0.95.

Examples

```
# Note: Most of models included in tuts package are computationally intensive. In the example
# below I set parameters to meet CRAN's testing requirement of maximum 5 sec per example.
# A more practical example would contain N=50 in the first line of the code and n.sim=10000.

#1. Import or simulate the data (a simulation is chosen for illustrative purposes):
DATA=simtuts(N=10,Harmonics=c(4,0,0), sin.ampl=c(10,0, 0), cos.ampl=c(0,0,0),
            trend=0,y.sd=2, ti.sd=0.2)
y=DATA$observed$y.obs
ti.mu=DATA$observed$ti.obs.tnorm
ti.sd= rep(0.2, length(ti.mu))

#2. Fit the model:
n.sim=1000; n.chains=2
AR1REDF=tuar1redf(y=y,ti.mu=ti.mu,ti.sd=ti.sd,n.sim=n.sim,n.chains=n.chains,CV=TRUE,n.cores=2)

#3. Generate plots and diagnostics of the model (optional parameters are listed in brackets):
plot(AR1REDF,type='predTUTS',burn=0.2,CI=0.99)  # One step out of salmple predictions (CI, burn).
plot(AR1REDF,type='par', burn=0.4)             # Distributions of parameters (burn).
plot(AR1REDF,type='mcmc')                      # MCMC diagnostics.
plot(AR1REDF,type='cv', burn=0.4)              # 5 fold cross validation (CI, burn).
plot(AR1REDF,type='GR', burn=0.4)              # Gelman-Rubin diagnostic (CI, burn).
plot(AR1REDF,type='tau')                       # realizations of persistence of time.
```

plot.tuts_BFS *Plots and visual diagnostics of tuts_BFS objects*

Description

plot.tuts_BFS generates plots and visual diagnostics of tuts_BFS objects.

Usage

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

Arguments

x	A tuts_BFS objects.
type	plot type with the following options: - 'periodogram' plots estimates of power spectrum. - 'predTUTS' plots one step predictions of the model. - 'GR' plots Gelman-Rubin diagnostics. - 'cv' plots 5-fold cross validation. - 'mcmc' plots diagnostics of MCMC/JAGS objects. - 'volatility' plots volatility realizations.
...	list of optional parameters: - burn: burn-in parameter ranging from 0 to 0.7 with default value set to 0. - CI: credible interval ranging from 0.3 to 1 with default value set to 0.95.

Examples

```
# Note: Most of models included in tuts package are computationally intensive. In the example
# below I set parameters to meet CRAN's testing requirement of maximum 5 sec per example.
# A more practical example would contain N=50 in the first line of the code and n.sim=10000.

#1. Import or simulate the data (a simulation is chosen for illustrative purposes):
DATA=simtuts(N=5,Harmonics=c(4,0,0), sin.ampl=c(10,0, 0), cos.ampl=c(0,0,0),
            trend=0,y.sd=2, ti.sd=0.2)
y=DATA$observed$y.obs
ti.mu=DATA$observed$ti.obs.tnorm
ti.sd= rep(0.2, length(ti.mu))

#2. Fit the model:
n.sim=10
BFS=tubfs(y=y,ti.mu=ti.mu,ti.sd=ti.sd,freqs='internal',n.sim=n.sim,n.chains=2, CV=TRUE,n.cores=2)

#3. Generate plots and diagnostics of the model (optional parameters are listed in brackets):
plot(BFS,type='periodogram')              # spectral analysis (CI, burn).
plot(BFS,type='predTUTS', CI=0.99)        # One step predictions (CI, burn).
plot(BFS,type='cv')                       # 5 fold cross validation plot (CI, burn).
```

```
plot(BFS,type='GR')                          # Gelman-Rubin diagnostics (CI, burn).
plot(BFS,type='mcmc')                        # mcmc diagnostics.
plot(BFS,type='volatility')                  # Volatility realizaitons.
```

---

plot.tuts_ls *Plots of spectral densities of tuts_ls objects*

Description

`plot.tuts_ls` plots spectra of tuts_ls objects.

Usage

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

Arguments

`x`	A tuts_ls obect.
`...`	optional arguments are not in use in the current version on tuts.

Examples

```
#1. Import or simulate the data (simulation is chosen for illustrative purposes):
DATA=simtuts(N=10,Harmonics=c(4,0,0), sin.ampl=c(10,0, 0), cos.ampl=c(0,0,0),
           trend=0,y.sd=2, ti.sd=0.2)
y=DATA$observed$y.obs
ti.mu=DATA$observed$ti.obs.tnorm
ti.sd= rep(0.2, length(ti.mu))

#2. Run multiple Lomb-Scargle periodograms (optional parameters are listed in brackets):
TULS=tuls(y=y,ti.mu=ti.mu,ti.sd=ti.sd,n.sim=500)     # (ofac, CI).

#3. Plot the Lomb-Scargle periodograms:
plot(TULS)
```

---

plot.tuts_poisBFS *Plots and visual diagnostics of tuts_BFS objects*

Description

`plot.tuts_poisBFS` generates plots and visual diagnostics of tuts_BFS objects.

Usage

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

Arguments

`x`	A tuts_BFS objects.
`type`	plot type with the following options: - 'periodogram' plots estimates of power spectrum. - 'predTUTS' plots one step predictions of the model. - 'GR' plots Gelman-Rubin diagnostics. - 'cv' plots 5-fold cross validation. - 'mcmc' plots diagnostics of MCMC/JAGS objects. - 'lambda' plots lambda realizations.
`...`	list of optional parameters: - burn: burn-in parameter ranging from 0 to 0.7 with default value set to 0. - CI: credible interval ranging from 0.3 to 1 with default value set to 0.95.

Examples

```
# Note: Most of models included in tuts package are computationally intensive. In the example
# below I set parameters to meet CRAN's testing requirement of maximum 5 sec per example.
# A more practical example would contain N=50 in the first line of the code and n.sim=10000.

#1. Import or simulate the data (simulation is chosen for illustrative purposes):
DATA=simtuts(N=7,Harmonics=c(2,0,0), sin.ampl=c(10,0, 0), cos.ampl=c(0,0,0),
           trend=0,y.sd=2, ti.sd=0.2)
y=DATA$observed$y.obs
y=round(y-min(y))
ti.mu=DATA$observed$ti.obs.tnorm
ti.sd= rep(0.2, length(ti.mu))

#2. Fit the model:
n.sim=10
TUPOIS=tupoisbsf(y=y,ti.mu=ti.mu,ti.sd=ti.sd,freqs='internal',n.sim=n.sim,n.chains=2,
                CV=TRUE,n.cores=2)

#3. Generate plots and diagnostics of the model (optional parameters are listed in brackets):
plot(TUPOIS,type='periodogram')             # spectral analysis (CI, burn).
plot(TUPOIS,type='predTUTS', CI=0.99)       # One step predictions (CI, burn).
plot(TUPOIS,type='cv')                      # 5 fold cross validation (CI, burn).
plot(TUPOIS,type='GR')                      # Gelman-Rubin diagnostics (CI, burn).
plot(TUPOIS,type='mcmc')                    # MCMC diagnostics.
plot(TUPOIS,type='lambda')                  # Realizaitons of lambda.
```

---

plot.tuts_poisPN *Plots and visual diagnostics of tuts_poisPN objects*

Description

`plot.tuts_poisPN` generates plots and visual diagnostics of tuts_poisPN objects.

Usage

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

Arguments

x	A tuts_poisPN object.
type	plot type with the following options: - 'predTUTS' plots one step predictions of the model. - 'GR' plots Gelman-Rubin diagnostics. - 'cv' plots 5-fold cross validation. - 'mcmc' plots diagnostics of MCMC/JAGS objects. - 'volatility' plots volatility realizations.
...	list of optional parameters: - burn: burn-in parameter ranging from 0 to 0.7 with default value set to 0. - CI: credible interval ranging from 0.3 to 1 with default value set to 0.95.

Examples

```
# Note: Most of models included in tuts package are computationally intensive. In the example
# below I set parameters to meet CRAN's testing requirement of maximum 5 sec per example.
# A more practical example would contain N=50 in the first line of the code and n.sim=10000.

#1. Import or simulate the data (a simulation is chosen for illustrative purposes):
DATA=simtuts(N=10,Harmonics=c(4,0,0), sin.ampl=c(10,0, 0), cos.ampl=c(0,0,0),
            trend=0,y.sd=2, ti.sd=0.2)
y=DATA$observed$y.obs
y=round(y-min(y))
ti.mu=DATA$observed$ti.obs.tnorm
ti.sd= rep(0.2, length(ti.mu))

#2. Fit the model:
polyorder=2
n.sim=1000
PPN=tupoispn(y=y,ti.mu=ti.mu,ti.sd=ti.sd,polyorder=polyorder,n.sim=n.sim,CV=TRUE,n.cores=2)

#3. Generate plots and diagnostics of the model (optional parameters are listed in brackets):
plot(PPN,type='predTUTS',CI=0.95)   # One step out of salmple predictions (CI, burn).
plot(PPN,type='cv',burn=0.3)        # 5 fold cross-validation (CI, burn).
plot(PPN,type='GR',CI=0.95)         # Gelman-Rubin diagnostic (CI).
plot(PPN,type='mcmc')               # MCMC diagnostics.
plot(PPN,type='lambda')             # Volatility realizaitons.
```

plot.tuts_polyn *Plots and visual diagnostics of tuts_polyn objects*

Description

plot.tuts_polyn generates plots and visual diagnostics of tuts_polyn objects.

Usage

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

Arguments

x	A tuts_polyn object.
type	plot type with the following options: - 'predTUTS' plots one step predictions of the model. - 'GR' plots Gelman-Rubin diagnostics. - 'cv' plots 5-fold cross validation. - 'mcmc' plots diagnostics of MCMC/JAGS objects. - 'volatility' plots volatility realizations.
...	list of optional parameters: - burn: burn-in parameter ranging from 0 to 0.7 with default value set to 0. - CI: credible interval ranging from 0.3 to 1 with default value set to 0.95.

Examples

```
# Note: Most of models included in tuts package are computationally intensive. In the example
# below I set parameters to meet CRAN's testing requirement of maximum 5 sec per example.
# A more practical example would contain N=50 in the first line of the code and n.sim=10000.

#1. Import or simulate the data (simulation is chosen for illustrative purposes):
DATA=simtuts(N=10,Harmonics=c(4,0,0), sin.ampl=c(10,0, 0), cos.ampl=c(0,0,0),
            trend=0,y.sd=2, ti.sd=0.2)
y=DATA$observed$y.obs
ti.mu=DATA$observed$ti.obs.tnorm
ti.sd= rep(0.2, length(ti.mu))

#2. Fit the model:
polyorder=2
n.sim=1000
PN=tupolyn(y=y,ti.mu=ti.mu,ti.sd=ti.sd,polyorder=polyorder,n.sim=n.sim,CV=TRUE,n.cores=2)

#3. Generate plots and diagnostics of the model (optional parameters are listed in brackets):
plot(PN,type='predTUTS',CI=0.95)        # One step out of salmple predictions (CI, burn).
plot(PN,type='cv',burn=0.3)               # 5 fold cross-validation (CI, burn).
plot(PN,type='GR',CI=0.95)                # Gelman-Rubin diagnostic (CI).
```

```
plot(PN,type='mcmc')                    # MCMC diagnostics.
plot(PN,type='volatility')              # Volatility realizaitons.
```

---

simtuts *Generating time-uncertain time series*

Description

`simtuts` function generates time-uncertain time series. It returns two data frames containing simulation of an actual process and its observations.
The actual process consists of a sum of a constant, a linear trend, and three sine and three cosine functions, and its observations are normally distributed *y.obs~N(y.act, y.sd)*.
Timing of simulated processes is modeled as *t.act~U(0,N)* and sorted in the ascending order. Observations of timings are modeled in two ways:

1. Normally distributed timing *t.obs.norm~N(ti.act,ti.sd)*, sorted from the smallest to the largest value to ensure non-overlapping feature of observations,
2. Timing simulated with truncated normal distribution t.obs.tnorm~N(ti.act,ti.sd,....).

Note: variability of timing can be substantially greater when the normal distribution is chosen, the truncated distribution utilizes enforced limits applied in the midpoints of the actual timing.

Usage

```
simtuts(N, Harmonics, sin.ampl, cos.ampl, trend = 0, y.sd, ti.sd)
```

Arguments

`N`	A number of observations.
`Harmonics`	A vector of three harmonics, typically integers.
`sin.ampl`	A vector of three amplitudes of the sine terms.
`cos.ampl`	vector of three amplitudes of the cosine terms.
`trend`	A constant trend.
`y.sd`	A standard deviation of observations.
`ti.sd`	A standard deviation of estimates of timing.

References

https://en.wikipedia.org/wiki/Truncated_normal_distribution

Examples

```
# 1. Generate actual and observed time series as a sum of 2 sine functions:
DATA=simtuts(N=50,Harmonics=c(10,20,0), sin.ampl=c(10,10, 0), cos.ampl=c(0,0,0),trend=0,
y.sd=2, ti.sd=0.3)
```

summary.tuts_ar1 — *Prints summary tables of tuts_ar1 objects*

Description

summary.tuts_ar1 prints summary tables of tuts_ar1 objects

Usage

```
## S3 method for class 'tuts_ar1'
summary(object, ...)
```

Arguments

object	A tuts_ar1 object.
...	list of optional parameters: - burn: burn-in parameter ranging from 0 to 0.7 with default value set to 0. - CI: credible interval ranging from 0.3 to 1 with default value set to 0.95.

Examples

```
# Note: Most of models included in tuts package are computationally intensive. In the example
# below I set parameters to meet CRAN's testing requirement of maximum 5 sec per example.
# A more practical example would contain N=50 in the first line of the code and n.sim=10000.

#1. Import or simulate the data (a simulation is chosen for illustrative purposes):
DATA=simtuts(N=10,Harmonics=c(4,0,0), sin.ampl=c(10,0, 0), cos.ampl=c(0,0,0),
            trend=0,y.sd=2, ti.sd=0.2)
y=DATA$observed$y.obs
ti.mu=DATA$observed$ti.obs.tnorm
ti.sd= rep(0.2, length(ti.mu))

#2. Fit the model:
n.sim=1000
TUAR1=tuar1(y=y,ti.mu=ti.mu,ti.sd=ti.sd,n.sim=n.sim,CV=FALSE)

#3. Generate summary results (optional parameters are listed in brackets):
summary(TUAR1)                            # Summary results (CI, burn).
```

summary.tuts_ar1redf — *Prints summary tables of tuts_ar1redf objects*

Description

summary.tuts_ar1redf prints summary tables of tuts_ar1redf objects.

Usage

```
## S3 method for class 'tuts_ar1redf'
summary(object, ...)
```

Arguments

object	A tuts_ar1redf object.
...	list of optional parameters: - burn: burn-in parameter ranging from 0 to 0.7 with default value set to 0. - CI: credible interval ranging from 0.3 to 1 with default value set to 0.95.

Examples

```
# Note: Most of models included in tuts package are computationally intensive. In the example
# below I set parameters to meet CRAN's testing requirement of maximum 5 sec per example.
# A more practical example would contain N=50 in the first line of the code and n.sim=10000.

#1. Import or simulate the data (a simulation is chosen for illustrative purposes):
DATA=simtuts(N=10,Harmonics=c(4,0,0), sin.ampl=c(10,0, 0), cos.ampl=c(0,0,0),
            trend=0,y.sd=2, ti.sd=0.2)
y=DATA$observed$y.obs
ti.mu=DATA$observed$ti.obs.tnorm
ti.sd= rep(0.2, length(ti.mu))

#2. Fit the model:
n.sim=1000; n.chains=2
AR1REDF=tuar1redf(y=y,ti.mu=ti.mu,ti.sd=ti.sd,n.sim=n.sim,n.chains=n.chains, CV=FALSE)

#3. Generate summary results (optional parameters are listed in brackets):
summary(AR1REDF)                                # Summary results (CI, burn).
```

---

summary.tuts_BFS *Prints summary tables of tuts_BFS objects*

Description

`summary.tuts_BFS` prints summary tables of tuts_BFS objects

Usage

```
## S3 method for class 'tuts_BFS'
summary(object, ...)
```

Arguments

object	A tuts_BFS object.
...	A list of optional parameters: - burn: burn-in parameter ranging from 0 to 0.7, the default value is 0. - CI: confidence interval, the default value is set to 0.99.

Examples

```
# Note: Most of models included in tuts package are computationally intensive. In the example
# below I set parameters to meet CRAN's testing requirement of maximum 5 sec per example.
# A more practical example would contain N=50 in the first line of the code and n.sim=10000.

#1. Import or simulate the data (a simulation is chosen for illustrative purposes):
DATA=simtuts(N=8,Harmonics=c(4,0,0), sin.ampl=c(10,0, 0), cos.ampl=c(0,0,0),
            trend=0,y.sd=2, ti.sd=0.2)
y=DATA$observed$y.obs
ti.mu=DATA$observed$ti.obs.tnorm
ti.sd= rep(0.2, length(ti.mu))

#2. Fit the model:
n.sim=10
BFS=tubfs(y=y,ti.mu=ti.mu,ti.sd=ti.sd,freqs='internal',n.sim=n.sim,n.chains=2, CV=FALSE)

#3. Generate summary results (optional parameters are listed in brackets):
summary(BFS)                                # Summary results (CI, burn).
summary(BFS,burn=0.2)                       # Results after 20% of burn-in (CI).
```

`summary.tuts_ls` *Function returns a list of frequencies having significant power estimates*

Description

`summary.tuts_ls` returns a list of frequencies exceeding confidence intervals.

Usage

```
## S3 method for class 'tuts_ls'
summary(object, ...)
```

Arguments

`object`	A tuts_ls obect.
`...`	optional arguments, not in use in the current version on tuts.

Examples

```
#1. Import or simulate the data (simulation is chosen for illustrative purposes):
DATA=simtuts(N=10,Harmonics=c(4,0,0), sin.ampl=c(10,0, 0), cos.ampl=c(0,0,0),
            trend=0,y.sd=2, ti.sd=0.2)
y=DATA$observed$y.obs
ti.mu=DATA$observed$ti.obs.tnorm
ti.sd= rep(0.2, length(ti.mu))

#2. Run multiple Lomb-Scargle periodograms (optional parameters are listed in brackets):
TULS=tuls(y=y,ti.mu=ti.mu,ti.sd=ti.sd,n.sim=500)     # (ofac, CI).
```

```
#3. Obtain list of frequencies for which spectral power exceeds confidence interval:
summary(TULS)
```

`summary.tuts_poisBFS` *Summary tables of tuts_poisBFS objects*

Description

`summary.tuts_poisBFS` prints summary tables of tuts_poisBFS objects.

Usage

```
## S3 method for class 'tuts_poisBFS'
summary(object, ...)
```

Arguments

`object`	A tuts_poisBFS object.
`...`	A list of optional parameters: - burn: burn-in parameter ranging from 0 to 0.7, the default value is 0. - CI: confidence interval, the default value is set to 0.99.

Examples

```
# Note: Most of models included in tuts package are computationally intensive. In the example
# below I set parameters to meet CRAN's testing requirement of maximum 5 sec per example.
# A more practical example would contain N=50 in the first line of the code and n.sim=10000.

 #1. Import or simulate the data (simulation is chosen for illustrative purposes):
DATA=simtuts(N=7,Harmonics=c(4,0,0), sin.ampl=c(10,0, 0), cos.ampl=c(0,0,0),
            trend=0,y.sd=2, ti.sd=0.2)
y=DATA$observed$y.obs
y=round(y-min(y))
ti.mu=DATA$observed$ti.obs.tnorm
ti.sd= rep(0.2, length(ti.mu))

#2. Fit the model:
n.sim=10
TUPOIS=tupoisbsf(y=y,ti.mu=ti.mu,ti.sd=ti.sd,freqs='internal',n.sim=n.sim,n.chains=2, CV=FALSE)

#3. Generate summary results (optional parameters are listed in brackets):
summary(TUPOIS)                              # Summary results (CI, burn).
summary(TUPOIS,burn=0.2)                     # Results after 20% of burn-in (CI).
```

summary.tuts_poisPN *Prints summary tables of tuts_poisPN objects*

Description

`summary.tuts_poisPN` prints summary tables of tuts_poisPN objects.

Usage

```
## S3 method for class 'tuts_poisPN'
summary(object, ...)
```

Arguments

`object`	A tuts_poisPN object.
`...`	list of optional parameters. The list contains burn-in parameter ranging from 0 to 0.5, with the default value burn=0, and the credible interval parameter ranging between 0.5 and 1, with the default CI=0.99.

Examples

```
# Note: Most of models included in tuts package are computationally intensive. In the example
# below I set parameters to meet CRAN's testing requirement of maximum 5 sec per example.
# A more practical example would contain N=50 in the first line of the code and n.sim=10000.

#1. Import or simulate the data (a simulation is chosen for illustrative purposes):
DATA=simtuts(N=10,Harmonics=c(4,0,0), sin.ampl=c(10,0, 0), cos.ampl=c(0,0,0),
           trend=0,y.sd=2, ti.sd=0.2)
y=DATA$observed$y.obs
y=round(y-min(y))
ti.mu=DATA$observed$ti.obs.tnorm
ti.sd= rep(0.2, length(ti.mu))

#2. Fit the model:
polyorder=2
n.sim=1000
PPN=tupoispn(y=y,ti.mu=ti.mu,ti.sd=ti.sd,polyorder=polyorder,n.sim=n.sim,CV=FALSE)

#3. Generate summary results (optional parameters are listed in brackets):
summary(PPN)                          # Summary results (burn, CI).
```

summary.tuts_polyn *Prints summary tables of tuts_polyn objects*

Description

summary.tuts_polyn prints summary tables of tuts_polyn objects.

Usage

```
## S3 method for class 'tuts_polyn'
summary(object, ...)
```

Arguments

object	A tuts_polyn object.
...	A list of optional parameters: - burn: burn-in parameter ranging from 0 to 0.7, the default value is 0. - CI: confidence interval, the default value is set to 0.99.

Examples

```
# Note: Most of models included in tuts package are computationally intensive. In the example
# below I set parameters to meet CRAN's testing requirement of maximum 5 sec per example.
# A more practical example would contain N=50 in the first line of the code and n.sim=10000.

#1. Import or simulate the data (simulation is chosen for illustrative purposes):
DATA=simtuts(N=10,Harmonics=c(4,0,0), sin.ampl=c(10,0, 0), cos.ampl=c(0,0,0),
           trend=0,y.sd=2, ti.sd=0.2)
y=DATA$observed$y.obs
ti.mu=DATA$observed$ti.obs.tnorm
ti.sd= rep(0.2, length(ti.mu))

#2. Fit the model:
polyorder=2
n.sim=1000
PN=tupolyn(y=y,ti.mu=ti.mu,ti.sd=ti.sd,polyorder=polyorder,n.sim=n.sim, CV=FALSE)

#3. Generate summary results (optional parameters are listed in brackets):
summary(PN)                            # Summary results (burn, CI).
```

summary.tuts_wrap *Model comaprison of multiple time-uncertain models based on DIC criterion*

Description

summary.tuts_wrap function returns DIC criteria of multiple models contained in tuts_wrap objects.

Usage

```
## S3 method for class 'tuts_wrap'
summary(object, ...)
```

Arguments

object	A tuwrap object.
...	optional arguments, not in use in the current version on tuts.

Examples

```
# Note: Most of models included in tuts package are computationally intensive. In the example
# below I set parameters to meet CRAN's testing requirement of maximum 5 sec per example.
# A more practical example would contain N=50 in the first line of the code and n.sim=10000.

#1. Import or simulate the data (a simulation is chosen for illustrative purposes):
DATA=simtuts(N=5,Harmonics=c(4,0,0), sin.ampl=c(10,0, 0), cos.ampl=c(0,0,0),
           trend=0,y.sd=2, ti.sd=0.2)
y=DATA$observed$y.obs
ti.mu=DATA$observed$ti.obs.tnorm
ti.sd= rep(0.2, length(ti.mu))

#2. Fit the models:
n.sim=100
WRAP=tuwrap(y=y,ti.mu=ti.mu,ti.sd=ti.sd,n.sim=n.sim)

#3. Generate summary results:
summary(WRAP)
```

---

tuar1 *Time-uncertain AR(1) model*

Description

tuar1 estimates unbiased parameters of time-uncertain AR(1) model.

Usage

```
tuar1(y, ti.mu, ti.sd, n.sim, CV = FALSE, ...)
```

Arguments

y	A vector of observations.
ti.mu	A vector of estimates of timing of observations.
ti.sd	A vector of standard deviations of timing.
n.sim	A number of simulations.

CV	TRUE/FALSE cross-validation indicator.
...	list of optional parameters: - n.chains: number of MCMC chains, the default number of chains is set to 2. - n.cores: number of cores used in cross-validation. No value or 'MAX' applies all the available cores in computation. - Thin: thinning factor, the default values is set to 4.

Examples

```
# Note: Most of models included in tuts package are computationally intensive. In the example
# below I set parameters to meet CRAN's testing requirement of maximum 5 sec per example.
# A more practical example would contain N=50 in the first line of the code and n.sim=10000.

#1. Import or simulate the data (a simulation is chosen for illustrative purposes):
DATA=simtuts(N=10,Harmonics=c(4,0,0), sin.ampl=c(10,0, 0), cos.ampl=c(0,0,0),
           trend=0,y.sd=2, ti.sd=0.2)
y=DATA$observed$y.obs
ti.mu=DATA$observed$ti.obs.tnorm
ti.sd= rep(0.2, length(ti.mu))

#2. Fit the model:
n.sim=1000
TUAR1=tuar1(y=y,ti.mu=ti.mu,ti.sd=ti.sd,n.sim=n.sim,CV=TRUE,n.cores=2)

#3. Generate summary results (optional parameters are listed in brackets):
summary(TUAR1)                          # Summary results (CI, burn).

#4. Generate plots and diagnostics of the model (optional parameters are listed in brackets):
plot(TUAR1,type='predTUTS')           # One step out of salmple predictions (CI, burn).
plot(TUAR1,type='par', burn=0.4)        # Distributions of parameters (burn).
plot(TUAR1,type='mcmc')                 # MCMC diagnostics.
plot(TUAR1,type='cv', burn=0.4, CI=0.9)   # 5 fold cross validation (CI, burn).
plot(TUAR1,type='GR')                   # Gelman-Rubin diagnostic (CI, burn).
plot(TUAR1,type='volatility')           # Volatility realizaitons.
```

---

tuar1redf *Time-uncertain time-persistence AR(1) model*

Description

tuar1redf estimates parameters of the AR(1) model specified in *"Climate Time Series Analysis"* by M.Mudelsee. We modify the model to account for time-uncertainty.

Usage

```
tuar1redf(y, ti.mu, ti.sd, n.sim, n.chains = 2, CV = FALSE, ...)
```

Arguments

`y`	A vector of observations.
`ti.mu`	A vector of estimates of timing of observations.
`ti.sd`	A vector of standard deviations of timing.
`n.sim`	A number of simulations.
`n.chains`	A number of chains.
`CV`	TRUE/FALSE cross-validation indicator.
`...`	list of optional parameters: - n.chains: number of MCMC chains, the default number of chains is set to 2. - Thin: thinning factor, the default values is set to 4. - n.cores: number of cores used in cross-validation. No value or 'MAX' applies all the available cores in computation.

Details

Note: tuar1redf model estimates autocorrelation parameters with a certain bias unmitigated after the correction described in *"Climate Time Series Analysis"* by M.Mudelsee. In addition the model is not suitable for time series series generated with negative values of autocorrelation parameters. In contrast, the function tuar1 generates unbiased estimates of the parameters, and is not limited to positive values of parameters.
We include this model to provide suppodt for justificaiton of results obtained with the REDFIT method.

Examples

```
# Note: Most of models included in tuts package are computationally intensive. In the example
# below I set parameters to meet CRAN's testing requirement of maximum 5 sec per example.
# A more practical example would contain N=50 in the first line of the code and n.sim=10000.

#1. Import or simulate the data (a simulation is chosen for illustrative purposes):
DATA=simtuts(N=10,Harmonics=c(4,0,0), sin.ampl=c(10,0, 0), cos.ampl=c(0,0,0),
            trend=0,y.sd=2, ti.sd=0.2)
y=DATA$observed$y.obs
ti.mu=DATA$observed$ti.obs.tnorm
ti.sd= rep(0.2, length(ti.mu))

#2. Fit the model:
n.sim=1000
n.chains=2
AR1REDF=tuar1redf(y=y,ti.mu=ti.mu,ti.sd=ti.sd,n.sim=n.sim,n.chains=n.chains, CV=TRUE,n.cores=2)

#3. Generate summary results (optional parameters are listed in brackets):
summary(AR1REDF)                                # Summary results (CI, burn).

#4. Generate plots and diagnostics of the model (optional parameters are listed in brackets):
plot(AR1REDF,type='predTUTS',burn=0.2,CI=0.99)  # One step out of salmple predictions (CI, burn).
plot(AR1REDF,type='par', burn=0.4)              # Distributions of parameters (burn).
```

```
plot(AR1REDF,type='mcmc')                  # MCMC diagnostics.
plot(AR1REDF,type='cv', burn=0.4)          # 5 fold cross validation (CI, burn).
plot(AR1REDF,type='GR', burn=0.4)          # Gelman-Rubin diagnostic (CI, burn).
plot(AR1REDF,type='tau')                   # realizations of persistence of time.
```

`tubfs` *Bayesian Frequency Selection of time-uncertain data sets*

Description

`tubfs` performs spectral analysis of time-uncertain time series using the Bayesian Frequency Selection method described in the paper Frequency selection in paleoclimate time series: A model-based approach incorporating possible time uncertainty by P. Franke, Prof B. Huntley, Dr A. Parnell.

Usage

```
tubfs(y, ti.mu, ti.sd, n.sim, CV = FALSE, ...)
```

Arguments

`y`	A vector of observations.
`ti.mu`	A vector of estimates/observed timings of observations.
`ti.sd`	A vector of standard deviations of timings.
`n.sim`	A number of simulations.
`CV`	TRUE/FALSE cross-validation indicator.
`...`	optional arguments: - n.chains: number of MCMC chains, the default number of chains is set to 2. - Thin: thinning factor, the default values is set to 4. - m: maximum number of significant frequencies in the data, the default value is set to 5. - polyorder: the polynomial regression component, the default odrer is set to 3. - freqs: set to a positive integer k returns a vector of k equally spaced frequencies in the Nyquist range. freqs can be provided as a vector of custom frequencies of interest. Set to 'internal' (the default value) generates a vector of equally spaced frequencies in the Nyquist range. - n.cores: number of cores used in cross-validation. No value or 'MAX' applies all the available cores in computation.

Examples

```
# Note: Most of models included in tuts package are computationally intensive. In the example
# below I set parameters to meet CRAN's testing requirement of maximum 5 sec per example.
# A more practical example would contain N=50 in the first line of the code and n.sim=10000.

#1. Import or simulate the data (a simulation is chosen for illustrative purposes):
DATA=simtuts(N=8,Harmonics=c(4,0,0), sin.ampl=c(10,0, 0), cos.ampl=c(0,0,0),
```

```
               trend=0,y.sd=2, ti.sd=0.2)
y=DATA$observed$y.obs
ti.mu=DATA$observed$ti.obs.tnorm
ti.sd= rep(0.2, length(ti.mu))

#2. Fit the model:
n.sim=10
BFS=tubfs(y=y,ti.mu=ti.mu,ti.sd=ti.sd,freqs='internal',n.sim=n.sim,n.chains=2,CV=TRUE,n.cores=2)

#3. Generate summary results (optional parameters are listed in brackets):
summary(BFS)                              # Summary results (CI, burn).
summary(BFS,burn=0.2)                     # Results after 20% of burn-in (CI).

#4. Generate plots and diagnostics of the model (optional parameters are listed in brackets):
plot(BFS,type='periodogram')              # spectral analysis (CI, burn).
plot(BFS,type='predTUTS', CI=0.99)        # One step predictions (CI, burn).
plot(BFS,type='cv')                       # 5 fold cross validation plot (CI, burn).
plot(BFS,type='GR')                       # Gelman-Rubin diagnostics (CI, burn).
plot(BFS,type='mcmc')                     # mcmc diagnostics.
plot(BFS,type='volatility')               # Volatility realizaitons.
```

`tuls` *Spectral analysis of time-uncertain time series using Lomb-Scargle method*

Description

`tuls` computes multiple power estimates using the Lomb-Scargle algorithm and simulated realizations of uncorrelated timings of observations. Timings are simulated with normal distribution *ti~N(ti.mu,ti.sd)*, and sorted in ascending order to ensure non-overlapping feature of observations.

Usage

```
tuls(y, ti.mu, ti.sd, n.sim = 1000, ...)
```

Arguments

`y`	A vector of observations.
`ti.mu`	A vector of estimates of timings of observations.
`ti.sd`	A vector of standard deviations of timings.
`n.sim`	A number of simulations.
`...`	list of optional parameters: - oversamlping parameter: the default value of ofac=4. - confidence interval: the default value is CI=0.99. - number of simulations: the default vale set to n.sim=1000.

References

https://en.wikipedia.org/wiki/Least-squares_spectral_analysis

See Also

https://CRAN.R-project.org/package=Bchron

Examples

```
#1. Import or simulate the data (a simulation is chosen for illustrative purposes):
DATA=simtuts(N=50,Harmonics=c(4,0,0), sin.ampl=c(10,0, 0), cos.ampl=c(0,0,0),
            trend=0,y.sd=2, ti.sd=0.2)
y=DATA$observed$y.obs
ti.mu=DATA$observed$ti.obs.tnorm
ti.sd= rep(0.2, length(ti.mu))

#2. Run multiple Lomb-Scargle periodograms (optional parameters are listed in brackets):
TULS=tuls(y=y,ti.mu=ti.mu,ti.sd=ti.sd,n.sim=500)     # (ofac, CI).

#3. Plot the Lomb-Scargle periodograms:
plot(TULS)

#4. Obtain list of frequencies for which spectral power exceeds confidence interval:
summary(TULS)
```

---

tupoisbsf — *Time uncertain Poisson regression with the Bayesian Frequency Selection method*

Description

tupoisbsf performs spectral analysis of time-uncertain time series of count data using bayesian frequency selection method.

Usage

```
tupoisbsf(y, ti.mu, ti.sd, n.sim, CV = FALSE, ...)
```

Arguments

y	A vector of observations.
ti.mu	A vector of estimates/observed timings of observations.
ti.sd	A vector of standard deviations of timings.
n.sim	A number of simulations.
CV	TRUE/FALSE cross-validation indicator.
...	optional arguments: - n.chains: number of MCMC chains, the default number of chains is set to 2. - Thin: thinning factor, the default values is set to 4. - m: maximum number of significant frequencies in the data, the default value is set to 5. - polyorder: the polynomial regression component, the default odrer is set to 3.

- freqs: set to a positive integer k returns a vector of k equally spaced frequencies in the Nyquist range. freqs can be provided as a vector of custom frequencies of interest. Set to 'internal' (the default value) generates a vector of equally spaced frequencies in the Nyquist range. - n.cores: number of cores used in cross-validation. No value or 'MAX' applies all the available cores in computation.

Examples

```
# Note: Most of models included in tuts package are computationally intensive. In the example
# below I set parameters to meet CRAN's testing requirement of maximum 5 sec per example.
# A more practical example would contain N=50 in the first line of the code and n.sim=10000.

#1. Import or simulate the data (simulation is chosen for illustrative purposes):
DATA=simtuts(N=7,Harmonics=c(4,0,0), sin.ampl=c(10,0, 0), cos.ampl=c(0,0,0),
           trend=0,y.sd=2, ti.sd=0.2)
y=DATA$observed$y.obs
y=round(y-min(y))
ti.mu=DATA$observed$ti.obs.tnorm
ti.sd= rep(0.2, length(ti.mu))

#2. Fit the model:
n.sim=10
TUPOIS=tupoisbsf(y=y,ti.mu=ti.mu,ti.sd=ti.sd,freqs='internal',n.sim=n.sim,n.chains=2,
                CV=TRUE,n.cores=2)

#3. Generate summary results (optional parameters are listed in brackets):
summary(TUPOIS)                              # Summary results (CI, burn).
summary(TUPOIS,burn=0.2)                     # Results after 20% of burn-in (CI).

#4. Generate plots and diagnostics of the model (optional parameters are listed in brackets):
plot(TUPOIS,type='periodogram')              # spectral analysis (CI, burn).
plot(TUPOIS,type='predTUTS', CI=0.99)        # One step predictions (CI, burn).
plot(TUPOIS,type='cv')                       # 5 fold cross validation (CI, burn).
plot(TUPOIS,type='GR')                       # Gelman-Rubin diagnostics (CI, burn).
plot(TUPOIS,type='mcmc')                     # MCMC diagnostics.
plot(TUPOIS,type='lambda')                   # Realizaitons of lambda.
```

---

tupoispn *Time-uncertain polynomial regression of counting time series*

Description

`tupoispn` performs estimation of parameters of Poisson N-th order polynomial regression of time-uncertain time series.

Usage

```
tupoispn(y, ti.mu, ti.sd, n.sim, polyorder = 3, CV = FALSE, ...)
```

Arguments

y	A vector of observations.
ti.mu	A vector of estimates of timing of observations.
ti.sd	A vector of standard deviations of timing.
n.sim	A number of simulations.
polyorder	Order of the polynomial regression.
CV	cross-validation indicator.
...	optional arguments: - n.chains: number of MCMC chains, the default number of chains is set to 2. - Thin: thinning factor, the default values is set to 4. - polyorder: order of the polynomial regression, the default odrer is set to 3. - n.cores: number of cores used in cross-validation. No value or 'MAX' applies all the available cores in computation.

Examples

```
# Note: Most of models included in tuts package are computationally intensive. In the example
# below I set parameters to meet CRAN's testing requirement of maximum 5 sec per example.
# A more practical example would contain N=50 in the first line of the code and n.sim=10000.

#1. Import or simulate the data (a simulation is chosen for illustrative purposes):
DATA=simtuts(N=10,Harmonics=c(4,0,0), sin.ampl=c(10,0, 0), cos.ampl=c(0,0,0),
            trend=0,y.sd=2, ti.sd=0.2)
y=DATA$observed$y.obs
y=round(y-min(y))
ti.mu=DATA$observed$ti.obs.tnorm
ti.sd= rep(0.2, length(ti.mu))

#2. Fit the model:
polyorder=2
n.sim=1000
PPN=tupoispn(y=y,ti.mu=ti.mu,ti.sd=ti.sd,polyorder=polyorder,n.sim=n.sim,CV=TRUE,n.cores=2)

#3. Generate summary results (optional parameters are listed in brackets):
summary(PPN)                           # Summary results (burn, CI).

#4. Generate plots and diagnostics of the model (optional parameters are listed in brackets):
plot(PPN,type='predTUTS',CI=0.95)   # One step out of salmple predictions (CI, burn).
plot(PPN,type='cv',burn=0.3)        # 5 fold cross-validation (CI, burn).
plot(PPN,type='GR',CI=0.95)         # Gelman-Rubin diagnostic (CI).
plot(PPN,type='mcmc')               # MCMC diagnostics.
plot(PPN,type='lambda')             # Volatility realizaitons.
```

tupolyn — *Time-uncertain polynomial regression*

Description

tupolyn performs estimation of parameters of N-th order polynomial regression of time-uncertain time series.

Usage

```
tupolyn(y, ti.mu, ti.sd, n.sim, polyorder, CV = FALSE, ...)
```

Arguments

y	A vector of observations.
ti.mu	A vector of estimates of timing of observations.
ti.sd	A vector of standard deviations of timing.
n.sim	A number of simulations.
polyorder	Order of the polynomial regression.
CV	TRUE/FALSE cross-validation indicator.
...	optional arguments: - n.chains: number of MCMC chains, the default number of chains is set to 2. - Thin: thinning factor, the default values is set to 4. - n.cores: number of cores used in cross-validation. No value or 'MAX' applies all the available cores in computation.

Examples

```
# Note: Most of models included in tuts package are computationally intensive. In the example
# below I set parameters to meet CRAN's testing requirement of maximum 5 sec per example.
# A more practical example would contain N=50 in the first line of the code and n.sim=10000.

#1. Import or simulate the data (a simulation is chosen for illustrative purposes):
DATA=simtuts(N=10,Harmonics=c(4,0,0), sin.ampl=c(10,0, 0), cos.ampl=c(0,0,0),
           trend=0,y.sd=2, ti.sd=0.2)
y=DATA$observed$y.obs
ti.mu=DATA$observed$ti.obs.tnorm
ti.sd= rep(0.2, length(ti.mu))

#2. Fit the model:
polyorder=2
n.sim=1000
PN=tupolyn(y=y,ti.mu=ti.mu,ti.sd=ti.sd,polyorder=polyorder,n.sim=n.sim,CV=TRUE,n.cores=2)

#3. Generate summary results (optional parameters are listed in brackets):
summary(PN)                             # Summary results (burn, CI).
```

```
#4. Generate plots and diagnostics of the model (optional parameters are listed in brackets):
plot(PN,type='predTUTS',CI=0.95)        # One step out of salmple predictions (CI, burn).
plot(PN,type='cv',burn=0.3)               # 5 fold cross-validation (CI, burn).
plot(PN,type='GR',CI=0.95)                # Gelman-Rubin diagnostic (CI).
plot(PN,type='mcmc')                      # MCMC diagnostics.
plot(PN,type='volatility')                # Volatility realizaitons.
```

---

tuwrap *Wrapper of the models contained in the tuts package*

Description

tuwrap tuwrap compares results obtained from fitting multiple models of time-uncertain time series.

Usage

```
tuwrap(y, ti.mu, ti.sd, n.sim, ...)
```

Arguments

y	A vector of observations.
ti.mu	A vector of estimates/observed timings of observations.
ti.sd	A vector of standard deviations of timings.
n.sim	A number of simulations.
...	optional arguments: - CV: TRUE/FALSE cross-validation indicator, the default value is set to FALSE. - n.chains: number of MCMC chains, the default number of chains is set to 2. - Thin: thinning factor, the default values is set to 4. - m: maximum number of significant frequencies in the data, the default value is set to 5. - polyorder: the polynomial regression component, the default odrer is set to 3. - freqs: set to a positive integer k returns a vector of k equally spaced frequencies in the Nyquist range. freqs can be provided as a vector of custom frequencies of interest. Set to 'internal' (the default value) generates a vector of equally spaced frequencies in the Nyquist range. - n.cores: number of cores used in cross-validation. No value or 'MAX' applies all the available cores in computation.

Examples

```
# Note: Most of models included in tuts package are computationally intensive. In the example
# below I set parameters to meet CRAN's testing requirement of maximum 5 sec per example.
# A more practical example would contain N=50 in the first line of the code and n.sim=10000.

#1. Import or simulate the data (a simulation is chosen for illustrative purposes):
```

```
DATA=simtuts(N=5,Harmonics=c(4,0,0), sin.ampl=c(10,0, 0), cos.ampl=c(0,0,0),
            trend=0,y.sd=2, ti.sd=0.2)
y=DATA$observed$y.obs
ti.mu=DATA$observed$ti.obs.tnorm
ti.sd= rep(0.2, length(ti.mu))

#2. Fit the models:
n.sim=100
WRAP=tuwrap(y=y,ti.mu=ti.mu,ti.sd=ti.sd,n.sim=n.sim,CV=FALSE,n.cores=2)

#3. Generate summary results:
summary(WRAP)

# Note: Accessing individual summaries, diagnostics and plots is presented in manuals
#       of models contained in the wrapper.SOme examples:

plot(WRAP$BFS,type='periodogram')
summary(WRAP$BFS,CI=0.99)
```

Index