

Package ‘csa’

May 8, 2026

Title A Cross-Scale Analysis Tool for Model-Observation Visualization and Integration

Version 0.7.1

Description Integration of Earth system data from various sources is a challenging task. Except for their qualitative heterogeneity, different data records exist for describing similar Earth system process at different spatio-temporal scales. Data inter-comparison and validation are usually performed at a single spatial or temporal scale, which could hamper the identification of potential discrepancies in other scales. 'csa' package offers a simple, yet efficient, graphical method for synthesizing and comparing observed and modelled data across a range of spatio-temporal scales. Instead of focusing at specific scales, such as annual means or original grid resolution, we examine how their statistical properties change across spatio-temporal continuum.

Depends R (>= 3.4.0)

Imports grDevices, stats, ggplot2, data.table, scales, reshape2, moments, Lmoments, foreach, ggpubr, raster, doParallel, parallel

License GPL-2

Encoding UTF-8

LazyData true

URL <https://github.com/imarkonis/csa>

BugReports <https://github.com/imarkonis/csa/issues>

RoxygenNote 7.2.3

Suggests testthat (>= 2.1.0), colorspace

NeedsCompilation no

Author Yannis Markonis [aut, cre],
Christoforos Pappas [aut],
Mijael Vargas [ctb],
Simon Papalexiou [ctb],
Martin Hanel [ctb]

Maintainer Yannis Markonis <imarkonis@gmail.com>

Repository CRAN

Date/Publication 2023-10-24 13:40:11 UTC

Contents

cnrm_nl	2
csa	3
csa.multiplot	4
csa.plot	5
csas	6
dt.to.brick	8
gpm_events	8
gpm_nl	9
knmi_nl	10
ncep_nl	10
rdr_nl	11
Index	12

cnrm_nl	<i>Simulation data (CNRM)</i>
---------	-------------------------------

Description

Model cnrm-cm3; scenario 20c3m; variable pr. 24 h 2.8 degree x 2.8 degree for Holland at daily time step for period 1961-01-01 to 2000-12-31. Spatial Region: 1 grid cell at latitude: 51.625, longitude: 5.625

Usage

```
data(cnrm_nl)
```

Format

An object of class `data.table` (inherits from `data.frame`) with 14610 rows and 2 columns.

Source

[KNMI explorer](#)

Examples

```
str(cnrm_nl)
```

csa *Estimate and print the temporal CSA plot*

Description

The function `csa` computes (and by default plots) the aggregation curve of a given statistic in a single dimension, e.g., time.

Usage

```
csa(
  x,
  stat = "var",
  std = TRUE,
  threshold = 30,
  plot = TRUE,
  fast = FALSE,
  chk = FALSE,
  ...
)
```

Arguments

<code>x</code>	A numeric vector.
<code>stat</code>	The statistic which will be estimated across the cross-scale continuum. Suitable options are: <ul style="list-style-type: none"> • "var" for variance, • "sd" for standard deviation, • "skew" for skewness, • "kurt" for kurtosis, • "l2" for L-scale, • "t2" for coefficient of L-variation, • "t3" for L-skewness, • "t4" for L-kurtosis.
<code>std</code>	logical. If TRUE (the default) the CSA plot is standardized to unit, i.e., zero mean and unit variance in the original time scale.
<code>threshold</code>	numeric. Sample size of the time series at the last aggregated scale.
<code>plot</code>	logical. If TRUE (the default) the CSA plot is printed.
<code>fast</code>	logical. If TRUE the CSA plot is estimated only in logarithmic scale; 1, 2, 3, ..., 10, 20, 30, ..., 100, 200, 300 etc.
<code>chk</code>	logical. If TRUE the number of cores is limited to 2.
<code>...</code>	<code>log_x</code> and <code>log_y</code> (default TRUE) for setting the axes of the CSA plot to logarithmic scale. The argument <code>wn</code> (default FALSE) is used to plot a line presenting the standardized variance of the white noise process. Therefore, it should be used only with <code>stat = "var"</code> and <code>std = T</code> .

Value

If `plot = TRUE`, the `csa` returns a list containing:

- `values`: Matrix of the timeseries values for the selected `stat` at each scale.
- `plot`: Plot of scale versus `stat` as a *ggplot* object.

If `plot = FALSE`, then it returns only the matrix of the timeseries values for the selected `stat` at each scale.

References

Markonis et al., A cross-scale analysis framework for model/data comparison and integration, Geoscientific Model Development, Submitted.

Examples

```
## Not run:
csa(rnorm(1000), wn = TRUE)
data(gpm_n1, knmi_n1, rdr_n1, ncep_n1, cnrm_n1, gpm_events)
csa(knmi_n1$prcp, threshold = 10, fast = TRUE, chk = TRUE)

csa(gpm_n1$prcp, stat = "skew", std = FALSE, log_x = FALSE, log_y = FALSE,
smooth = TRUE, chk = TRUE)

gpm_skew <- csa(gpm_n1$prcp, stat = "skew", std = FALSE, log_x = FALSE, log_y = FALSE,
smooth = TRUE, plot = FALSE, chk = TRUE)
rdr_skew <- csa(rdr_n1$prcp, stat = "skew", std = FALSE, log_x = FALSE, log_y = FALSE,
smooth = TRUE, plot = FALSE, chk = TRUE)
csa.multiplot(rbind(data.frame(gpm_skew, dataset = "gpm"), data.frame(rdr_skew,
dataset = "rdr")), log_x = FALSE, log_y = FALSE, smooth = TRUE)

set_1 <- data.frame(csa(gpm_n1$prcp, plot = FALSE, fast = TRUE), dataset = "gpm")
set_2 <- data.frame(csa(rdr_n1$prcp, plot = FALSE, fast = TRUE), dataset = "radar")
set_3 <- data.frame(csa(knmi_n1$prcp, plot = FALSE, fast = TRUE), dataset = "station")
set_4 <- data.frame(csa(ncep_n1$prcp, plot = FALSE, fast = TRUE), dataset = "ncep")
set_5 <- data.frame(csa(cnrm_n1$prcp, plot = FALSE, fast = TRUE), dataset = "cnrm")
csa.multiplot(rbind(set_1, set_2, set_3, set_4, set_5))

## End(Not run)
```

csa.multiplot

Multiple CSA plotting

Description

Function for plotting multiple CSA curves in a single plot.

Usage

```
csa.multiplot(df, log_x = TRUE, log_y = TRUE, wn = FALSE, smooth = FALSE)
```

Arguments

df	A matrix or data.frame composed of three columns; scale for the temporal or spatial scale; value for the estimate of a given statistic (e.g., variance) at the given aggregated scale and variable for defining the corresponding dataset.
log_x	logical. If TRUE (the default) the x axis of the CSA plot is set to the logarithmic scale.
log_y	logical. If TRUE (the default) the y axis of the CSA plot is set to the logarithmic scale.
wn	logical. The argument wn (default FALSE) is used to plot a line presenting the standardized variance of the white noise process. Therefore, it should be used only with stat = "var" and std = T in the csa/csas functions.
smooth	logical. If TRUE (the default) the aggregation curves are smoothed (loess function).

Value

The CSA plot as a ggplot object.

Examples

```
aa <- rnorm(1000)
csa_aa <- data.frame(csa(aa, plot = FALSE, chk = TRUE), variable = 'wn')
bb <- as.numeric(arima.sim(n = 1000, list(ar = c(0.8897, -0.4858), ma = c(-0.2279, 0.2488))))
csa_bb <- data.frame(csa(bb, plot = FALSE, chk = TRUE), variable = 'arma(2, 2)')
csa.multiplot(rbind(csa_aa, csa_bb), wn = TRUE)
csa.multiplot(rbind(csa_aa, csa_bb), wn = TRUE, smooth = TRUE)
```

csa.plot

CSA curve plotting

Description

Function for plotting single CSA curves.

Usage

```
csa.plot(x, log_x = TRUE, log_y = TRUE, smooth = FALSE, wn = FALSE)
```

Arguments

x	A matrix or data.frame composed of two columns; scale for the temporal or spatial scale and value for the estimate of a given statistic (e.g., variance) at the given aggregated scale.
log_x	logical. If TRUE (the default) the x axis of the CSA plot is set to the logarithmic scale.

log_y	logical. If TRUE (the default) the y axis of the CSA plot is set to the logarithmic scale.
smooth	logical. If TRUE (the default) the aggregation curves are smoothed (loess function).
wn	logical. The argument wn (default FALSE) is used to plot a line presenting the standardized variance of the white noise process. Therefore, it should be used only with stat = "var" and std = T in the csa/csas functions.

Value

The CSA plot as a ggplot object.

Examples

```
## Not run:
aa <- rnorm(1000)
csa_aa <- csa(aa, chk = TRUE, plot = FALSE)
csa.plot(csa_aa)

## End(Not run)
```

csas

Estimate and print the spatial CSA plot

Description

The function `csa` computes (and by default plots) the aggregation curve of a given statistic in two dimensions, e.g., space.

Usage

```
csas(
  x,
  stat = "var",
  std = TRUE,
  plot = TRUE,
  threshold = 30,
  chk = FALSE,
  ...
)
```

Arguments

- | | |
|------|---|
| x | A raster or brick object. |
| stat | The statistic which will be estimated across the cross-scale continuum. Suitable options are: <ul style="list-style-type: none"> "var" for variance, |

	<ul style="list-style-type: none"> • "sd" for standard deviation, • "skew" for skewness, • "kurt" for kurtosis, • "l2" for L-scale, • "t2" for coefficient of L-variation, • "t3" for L-skewness, • "t4" for L-kurtosis.
std	logical. If TRUE (the default) the CSA plot is standardized to unit, i.e., zero mean and unit variance in the original time scale.
plot	logical. If TRUE (the default) the CSA plot is printed
threshold	numeric. Sample size of the time series at the last aggregated scale.
chk	logical. If TRUE the number of cores is limited to 2.
...	log_x and log_y (default TRUE) for setting the axes of the CSA plot to logarithmic scale. The argument wn (default FALSE) is used to plot a line presenting the standardized variance of the white noise process. Therefore, it should be used only with stat = "var" and std = T.

Value

If plot = TRUE, the csa returns a list containing:

- values: Matrix of the timeseries values for the selected stat at each scale.
- plot: Plot of scale versus stat as a *ggplot* object.

If plot = FALSE, then it returns only the matrix of the timeseries values for the selected stat at each scale.

References

Markonis et al., A cross-scale analysis framework for model/data comparison and integration, Geoscientific Model Development, Submitted.

Examples

```
## Not run:
data(gpm_events)
event_dates <- format(gpm_events[, unique(time)], "%d-%m-%Y")
gpm_events_brick <- dt.to.brick(gpm_events, var_name = "prcp")
plot(gpm_events_brick, col = rev(colorspace::sequential_hcl(40)),
     main = event_dates)
csas(gpm_events_brick, chk = TRUE)

gpm_sp_scale <- csas(gpm_events_brick, plot = FALSE, chk = TRUE)
gpm_sp_scale[, variable := factor(variable, labels = event_dates)]
csa.multiplot(gpm_sp_scale, smooth = TRUE, log_x = FALSE, log_y = FALSE)

## End(Not run)
```

dt.to.brick	<i>Transform data.table to brick</i>
-------------	--------------------------------------

Description

The function `dt.to.brick` transforms a `data.table` object to brick (raster) format

Usage

```
dt.to.brick(dt, var_name)
```

Arguments

<code>dt</code>	The data table object to be transformed. It must be in a four-column format, with the coordinate columns named as "lat" & "lon" and time values as "time".
<code>var_name</code>	The name (chr) of the column in the data table (dt) which holds the values of the variable, e.g., "temperature".

Value

dt as a brick object.

Examples

```
## Not run:
aa <- expand.grid(lat = seq(40, 50, 1),
                 lon = seq(20, 30, 1),
                 time = seq(1900, 2000, 1))
aa$anomaly = rnorm(nrow(aa))
aa <- brick(dt.to.brick(aa, "anomaly"))

## End(Not run)
```

gpm_events	<i>GPM-IMERG precipitation events over 10 mm/day</i>
------------	--

Description

GPM IMERG Final Precipitation L3 1 day 0.1 degree x 0.1 degree for Holland at daily time step for period 2014-03-12 to 2018-05-15. Spatial averaged over: latitude: 50.75, 53.55, longitude: 3.45, 7.15

Usage

```
data(gpm_events)
```

Format

An object of class `data.table` (inherits from `data.frame`) with 6612 rows and 6 columns.

Source

[KNMI explorer](#)

Examples

```
str(gpm_events)
```

<code>gpm_n1</code>	<i>Satellite data (GPM-IMERG)</i>
---------------------	-----------------------------------

Description

GPM IMERG Final Precipitation L3 1 day 0.1 degree x 0.1 degree for Holland at daily time step for period 2014-03-12 to 2018-05-15. Spatial averaged over: latitude: 50.75, 53.55, longitude: 3.45, 7.15

Usage

```
data(gpm_n1)
```

Format

An object of class `data.table` (inherits from `data.frame`) with 1526 rows and 2 columns.

Source

[KNMI explorer](#)

Examples

```
str(gpm_n1)
```

knmi_nl	<i>Station data (KNMI)</i>
---------	----------------------------

Description

240 homogenized stations 1951-now. 24 h point data for Holland at daily time step for period 1950-12-31 to 2018-04-29. Spatial Region: latitude: 50.78, 53.48, longitude: 3.4, 7.11

Usage

```
data(knmi_nl)
```

Format

An object of class `data.table` (inherits from `data.frame`) with 24592 rows and 2 columns.

Source

[KNMI explorer](#)

Examples

```
str(knmi_nl)
```

ncep_nl	<i>Reanalysis data (NCEP/NCAR)</i>
---------	------------------------------------

Description

NMC reanalysis 24 h 2.5 degree x 2.5 degree for Holland at daily time step for period 1948-01-01 to 2018-06-05. Spatial Region: 1 grid cell at latitude: 52.38, longitude: 5.625

Usage

```
data(ncep_nl)
```

Format

An object of class `data.table` (inherits from `data.frame`) with 25601 rows and 2 columns.

Source

[KNMI explorer](#)

Examples

```
str(ncep_nl)
```

rdr_nl	<i>Radar data (KNMI)</i>
--------	--------------------------

Description

RAD_NL25_RAC_MFBS_24H_NC 24 h 1 km x 1 km for Holland at daily time step for period 2014-03-11 to 2018-03-30. Spatial Region: latitude: 50.76, 53.56, longitude: 3.37, 7.22

Usage

```
data(rdr_nl)
```

Format

An object of class `data.table` (inherits from `data.frame`) with 1472 rows and 2 columns.

Source

[KNMI explorer](#)

Examples

```
str(rdr_nl)
```

Index

* datasets

- cnrm_nl, 2
- gpm_events, 8
- gpm_nl, 9
- knmi_nl, 10
- ncep_nl, 10
- rdr_nl, 11

- cnrm_nl, 2
- csa, 3
- csa.multiplot, 4
- csa.plot, 5
- csas, 6

- dt.to.brick, 8

- gpm_events, 8
- gpm_nl, 9

- knmi_nl, 10

- ncep_nl, 10

- rdr_nl, 11