Package ‘stxplore’

February 3, 2023

Type Package

Title Exploration of Spatio-Temporal Data

Version 0.1.0

Maintainer Sevvandi Kandanaarachchi <sevvandik@gmail.com>

Description A set of statistical tools for spatio-temporal data exploration.
Includes simple plotting functions, covariance calculations and computations
similar to principal component analysis for spatio-temporal data. Can use
both dataframes and stars objects for all plots and computations. For more
details refer 'Spatio-Temporal Statistics with R' (Christopher K. Wikle,
Andrew Zammit-Mangion, Noel Cressie, 2019, ISBN:9781138711136).

License GPL (>= 3)

Encoding UTF-8

LazyData true

LazyDataCompression gzip

Imports fields, ggmap, ggplot2, ggridges, gridExtra, gstat, lubridate,
magrittr, RColorBrewer, rlang, sp, spacetime, stars, stats,
tidyr

RoxygenNote 7.2.1

Depends R (>= 2.10)

Suggests dplyr, knitr, rmarkdown, ncmeta, units, maps, cubelyr

VignetteBuilder knitr

URL https://sevvandi.github.io/stxplore/

NeedsCompilation no

Author Sevvandi Kandanaarachchi [aut, cre]
(<https://orcid.org/0000-0002-0337-0395>),
Petra Kunhert [aut] (<https://orcid.org/0000-0001-9070-0091>),
Andrew Zammit-Mangion [ctb] (<https://orcid.org/0000-0002-4164-6866>)

Repository CRAN

Date/Publication 2023-02-03 10:10:02 UTC

R topics documented:

`aerosol_australia` *Data from of NASA Earth Observations at https://neo.gsfc.nasa.gov*

Description

Aerosol optical thickness data from December 2019 to December 2020, taken monthly.

Usage

```
aerosol_australia
```

Format

A stars object with x, y and time containing aerosol thickness. Dimensions 70x70x13.

`aerosol_world` *Data from of NASA Earth Observations at https://neo.gsfc.nasa.gov*

Description

Aerosol optical thickness data from December 2019 to December 2020, taken monthly.

Usage

```
aerosol_world
```

Format

A stars object with x, y and time containing aerosol thickness. Dimensions 360x180x13

`cancor_eof` *Performs CCA using Empirical Orthogonal Functions (EOFs) from a lagged dataset*

Description

Performs Canonical Correlation Analysis (CCA) using Empirical Orthogonal Function analysis using in a dataframe or a stars object. The autoplot function can plot the outputs.

The variations are * ‘cancor_eof.data.frame()‘ if the input is a dataframe * ‘cancor_eof.stars()‘ if the input is a stars object * ‘autoplot.cancoreof()‘ to plot the outputs.

Usage

```
cancor_eof(x, lag, n_eof, ...)

## S3 method for class 'data.frame'
cancor_eof(x, lag = 7, n_eof = 10, values_df, ...)

## S3 method for class 'stars'
cancor_eof(x, lag = 7, n_eof = 10, ...)

## S3 method for class 'cancoreof'
autoplot(
  object,
  line_plot = TRUE,
  space_plot = TRUE,
  palette = "Spectral",
  xlab = "Time",
  ...
)
```

Arguments

`x` The dataframe or stars object. If it is a dataframe, then it should have the locations.

`lag` Specifies the lag to be used.

`n_eof` The number of EOFs to be used.

`...` Other arguments currently ignored.

`values_df` For dataframes: the dataframe of dimension `length(times)` x `length(locations)` containing the quantity of interest.

`object` autoplot parameter: the output of the function ‘cancor_eof’.

`line_plot` autoplot parameter: if set to `TRUE`, then the line plot is included.

`space_plot` autoplot parameter: if set to `TRUE`, the space splot is included.

`palette` autoplot parameter: the color palette to use for plotting.

`xlab` autoplot parameter:: he label on the x-axis for the line plot.

Value

A cancoreof object with CCA output, EOF output, original data and cancor object from ‘stats‘.

Examples

```
# Dataframe example
data(SSTlonlatshort)
data(SSTdatashort)
cancor_df <- cancor_eof(x = SSTlonlatshort,
           lag = 7,
           n_eof = 8,
           values_df = SSTdatashort)
autoplot(cancor_df)

# Stars example
library(dplyr)
library(stars)
# Create a stars object from a data frame
precip_df <- NOAA_df_1990[NOAA_df_1990$proc == 'Precip', ] %>%
  filter(date >= ”1992-02-01” & date <= ”1992-02-28”)
precip <- precip_df[ ,c('lat', 'lon', 'date', 'z')]
st_precip <- st_as_stars(precip, dims = c(”lon”, ”lat”, ”date”))
cancor_st <- cancor_eof(st_precip)
autoplot(cancor_st, line_plot = TRUE, space_plot = FALSE)
```

canonical_correlation *Computes transformed variables from Canonical Correlation Analysis using a dataframe or a stars object*

Description

Computes Canonical Correlation Analysis (CCA) using 2 datasets. The autoplot function plots the output.

Usage

```
canonical_correlation(x1, x2, ...)

## S3 method for class 'data.frame'
canonical_correlation(x1, x2, ...)

## S3 method for class 'stars'
canonical_correlation(x1, x2, ...)

## S3 method for class 'cancor'
autoplot(object, xlab = "Time", ...)
```

Arguments

x1	The first dataframe or stars object.
x2	The second dataframe or stars objext. The dimensions of both datasets need to be the same.
...	Other arguments currently ignored.
object	For autoplot: the output of the function 'cannonical_correlation'.
xlab	For autoplot: the xlabel to appear on CCA plot.

Value

A canonical correlation object.

Examples

```
# Dataframe example
df1 <- SSTdatashort[1:100, ]
df2 <- SSTdatashort[401:500, ]
ccor <- canonical_correlation(df1, df2)
autoplot(ccor)

# stars example
library(stars)
tif = system.file("tif/olinda_dem_utm25s.tif", package = "stars")
x <- read_stars(tif)
```

```
x1 <- x[[1]][1:50, 1:50]
x2 <- x[[1]][51:100, 1:50]
stx1 <- st_as_stars(x1)
stx2 <- st_as_stars(x2)
canonical_correlation(stx1, stx2)
```

emp_orth_fun *Computes empirical orthogonal functions using a dataframe or a stars object.*

Description

Computes empirical orthogonal functions of the data. Function autoplot can plot the output.

Usage

```
emp_orth_fun(x, ...)

## S3 method for class 'data.frame'
emp_orth_fun(x, values_df, ...)

## S3 method for class 'stars'
emp_orth_fun(x, ...)

## S3 method for class 'emporthfun'
autoplot(
  object,
  EOF_num = 1,
  palette = "Spectral",
  only_EOF = FALSE,
  only_TS = FALSE,
  ...
)
```

Arguments

`x`	The dataframe or stars object. If it is a dataframe, then it should have the locations.
`...`	Other arguments currently ignored.
`values_df`	For dataframes: the dataframe of dimension `length(times)` x `length(locations)` containing the quantity of interest.
`object`	For autoplot: the output of the function ‘emp_orth_fun’.
`EOF_num`	For autoplot: the number of Empirical Orthogonal Functions (EOFs) to plot.
`palette`	The color palette. Default is `Spectral`.
`only_EOF`	For autoplot: if `TRUE`, only the spatial EOF function would be plotted.
`only_TS`	For autoplot: if `TRUE`, only the PC time series would be plotted. If both are set to `FALSE`, both plots would be displayed. Both cannot be set to `TRUE`.

Value

An emporthfun object with temporal PCs and spatial EOFs.

Examples

```
# dataframe example
data(SSTlonlatshort)
data(SSTdatashort)
data(SSTlandmaskshort)
delete_rows <- which(SSTlandmaskshort  ==  1)
SSTdatashort   <- SSTdatashort[-delete_rows, 1:396]
emp1 <- emp_orth_fun(SSTlonlatshort[-delete_rows,  ],
                     SSTdatashort)
autoplot(emp1,
         EOF_num = 1)


# stars example
library(dplyr)
library(stars)
# Create a stars object from a data frame
precip_df <- NOAA_df_1990[NOAA_df_1990$proc == 'Precip', ] %>%
  filter(date >= "1992-02-01" & date <= "1992-02-05")
precip <- precip_df[ ,c('lat', 'lon', 'date', 'z')]
st_precip <- st_as_stars(precip, dims = c("lon", "lat", "date"))
emp <- emp_orth_fun(st_precip)
autoplot(emp, only_TS = TRUE)
```

emp_spatial_cov *Computes empirical spatial covariance using a dataframe or a stars object*

Description

Computes empirical spatial covariance by removing trends and examining residuals. It can compute lag-0 or log-1 empirical covariance either by latitude or longitude. You can split up the spatial domain by latitude or longitude and plot the covariance for each longitudinal/latitudinal strips.

Usage

```
emp_spatial_cov(
  x,
  lat_or_lon_strips = "lon",
  quadratic_time = FALSE,
  quadratic_space = FALSE,
  num_strips = 1,
  lag = 0,
  ...
)

## S3 method for class 'data.frame'
emp_spatial_cov(
  x,
  lat_or_lon_strips = "lon",
  quadratic_time = FALSE,
  quadratic_space = FALSE,
  num_strips = 1,
  lag = 0,
  lat_col,
  lon_col,
  t_col,
  z_col,
  ...
)

## S3 method for class 'stars'
emp_spatial_cov(
  x,
  lat_or_lon_strips = "lon",
  quadratic_time = FALSE,
  quadratic_space = FALSE,
  num_strips = 1,
  lag = 0,
  ...
)

## S3 method for class 'spatialcov'
autoplot(object, xlab = "Latitude", ...)
```

Arguments

`x` A stars object or a dataframe. Arguments differ according to the input type.

`lat_or_lon_strips`
Takes the values `lat` or `lon`. The value `lat` produces latitudinal strips, i.e., covariance plots over longitude for different latitudinal strips. The value `lon` produces longitudinal strips, i.e., covariance plots over latitude for different longitudinal strips.

`quadratic_time` If `TRUE` a linear model with quadratic time is fitted and residuals computed. If `FALSE` the model is fitted with linear space and time coefficients.

`quadratic_space`
If `TRUE` a linear model with quadratic space is fitted and residuals computed. If `FALSE` the model is fitted with linear space and time coefficients.

`num_strips` The number of latitudinal/longitudinal strips to produce. This is used when plotting using autoplot.

`lag` Lag can be either 0 or 1.

`...` Other arguments currently ignored.

`lat_col` For dataframes: the column or the column name giving the latitude. The y coordinate can be used instead of latitude.

`lon_col` For dataframes: the column or the column name giving the longitude. The x coordinate can be used instead of longitude.

`t_col` For dataframes: the time column. Time must be a set of discrete integer values.

`z_col` For dataframes: the The quantity of interest that will be plotted. Eg. temperature.

`object` For autoplot: the output of the function ‘emp_spatial_cov’.

`xlab` For autoplot: the label for x-axis.

Value

A spatialcov object with empirical covariance data organised spatially according to the number of strips and the lagged covariance.

Examples

```
# Dataframe example
library(dplyr)
data(NOAA_df_1990)
Tmax <- filter(NOAA_df_1990,
  proc == "Tmax" &
  month %in% 5:6 &
  year == 1993)
Tmax$t <- Tmax$julian - min(Tmax$julian) + 1
emp_df <- emp_spatial_cov(Tmax,
                lat_col = "lat",
                lon_col = "lon",
                t_col ="t",
                z_col = "z",
                lat_or_lon_strips = "lon",
                num_strips = 4,
                lag = 1)
autoplot(emp_df)

# Stars example
library(stars)
# Create a stars object from a data frame
precip_df <- NOAA_df_1990[NOAA_df_1990$proc == 'Precip', ] %>%
  filter(date >= "1992-02-01" & date <= "1992-02-05")
precip <- precip_df[ ,c('lat', 'lon', 'date', 'z')]
st_precip <- st_as_stars(precip, dims = c("lon", "lat", "date"))
emp_spatial_cov(st_precip)
```

hovmoller *Computes the data structure for the Hovmoller plots*

Description

This function creates the data structure for Hovmoller plots for either latitude or longitude. This function can take either a stars object or a dataframe. Input arguments differ for each case. The function autoplot can plot this object.

Usage

```
hovmoller(x, lat_or_lon = "lat", xlen = NULL, ...)

## S3 method for class 'data.frame'
hovmoller(
  x,
  lat_or_lon = "lat",
  xlen = NULL,
  lat_or_lon_col,
  t_col,
  z_col,
  ...
)

## S3 method for class 'stars'
hovmoller(x, lat_or_lon = "lat", xlen = NULL, ...)

## S3 method for class 'hovmoller'
autoplot(
  object,
  ylab = "Day",
  xlab = NULL,
  title = "",
  palette = "Spectral",
  legend_title = "z",
  ...
)
```

Arguments

`x` A stars object or a dataframe. Arguments differ according to the input type.

`lat_or_lon` Needs to be either `lat` or `lon`. `lat` plots the latitudinal Hovmoller plat, while `lon` plots the longitudinal Hovmoller plot.

`xlen` The length of the xaxis for latitude/longitude.

`...` Other arguments currently ignored.

lat_or_lon_col For dataframes: the column or the column name corresponding to the latitude/longitude.

t_col For dataframes: the time column. Time must be a set of discrete integer values.

z_col For dataframes: the The quantity of interest that will be plotted. Eg. temperature.

object For autoplot: the output of the function ‘hovmoller’.

ylab The y label.

xlab The x label.

title The graph title.

palette The color palette. Default is Spectral.

legend_title The title for the legend.

Value

An object of hovmoller class containing the original data and the Hovmoller data.

Examples

```
# dataframe examples
library(dplyr)
data(NOAA_df_1990)
Tmax <- filter(NOAA_df_1990,
  proc == "Tmax" &
  month %in% 5:9 &
  year == 1993 &
  id < 4000)
Tmax$t <- Tmax$julian - min(Tmax$julian) + 1
hov <- hovmoller(lat_or_lon = "lat",
          x = Tmax,
          lat_or_lon_col = 'lat',
          t_col = 't',
          z_col = 'z')
autoplot(hov)

# stars examples
library(stars)
prec_file = system.file("nc/test_stageiv_xyt.nc", package = "stars")
prec <- read_ncdf(prec_file)
prec2 <- prec %>% slice(time, 1:5)
hov <- hovmoller(prec2)
hov
```

`locs` *The locations used in the NOAA dataset.*

Description

This dataset is included in the STRbook R package.

Usage

```
locs
```

Format

A data frame with 328 rows and 3 variables:

id Location is

lat Latitude

lon Longitude ...

`NOAA_df_1990` *National oceanic and atmospheric administration (NOAA) data from 1990 to 1993*

Description

A dataset containing the precipitation, maximum and minimum temperatures taken from the STRbook R package.

Usage

```
NOAA_df_1990
```

Format

A data frame with 53940 rows and 10 variables:

julian Day in Julian time

year The year

month The month

day The day

id The location id

z The value

proc The type of observation

lat Latitude

lon Longitude

date The date ...

ridgeline *Ridgeline plots grouped by an attribute using a dataframe as an input.*

Description

Plots ridgeline plots grouped by latitude/longitude or time. This function can take either a stars object or a dataframe. Input arguments differ for each case.

Usage

```
ridgeline(
  x,
  num_grps = 10,
  xlab = "Value",
  ylab = "Group Intervals",
  title = "",
  legend_title = "z",
  ...
)

## S3 method for class 'data.frame'
ridgeline(
  x,
  num_grps = 10,
  xlab = "Value",
  ylab = "Group Intervals",
  title = "",
  legend_title = "z",
  group_col,
  z_col,
  ...
)

## S3 method for class 'stars'
ridgeline(
  x,
  num_grps = 10,
  xlab = "Value",
  ylab = "Group Intervals",
  title = "",
  legend_title = "z",
  group_dim,
  ...
)
```

Arguments

x	A stars object or a dataframe. Arguments differ according to the input type.

`num_grps` The number of levels for the ridgeline plot.

`xlab` The x label.

`ylab` The y label.

`title` The graph title.

`legend_title` The title for the legend.

`...` Other arguments currently ignored.

`group_col` For dataframes: the column name of the group column.

`z_col` For dataframes: the The quantity of interest that will be plotted. Eg. temperature.

`group_dim` For stars objects: the dimension for the grouping variable.

Value

A ggplot object.

Examples

```
# Dataframe example
library(dplyr)
data(NOAA_df_1990)
TmaxJan <- filter(NOAA_df_1990,
                 proc == ”Tmax” &
                 year == 1993 &
                 month == 1)
ridgeline(TmaxJan,
      group_col = 'lat',
      z_col = 'z',
      xlab = 'Maximum Temperature',
      ylab = 'Latitude Intervals')

# stars examples
library(stars)
library(units)

# stars Example 1
tif = system.file(”tif/olinda_dem_utm25s.tif”, package = ”stars”)
x <- read_stars(tif)
dim(x)
ridgeline(x, group_dim = 1)
ridgeline(x, group_dim = 2)


# stars Example 2
tif = system.file(”tif/lc.tif”, package = ”stars”)
x <- read_stars(tif)
ridgeline(x, group_dim = 1)
ridgeline(x, group_dim = 2)
```

semivariogram *Computes the semi-variogram using a dataframe or a stars object.*

Description

Computes the semi-variogram from a stars or a dataframe. Input arguments differ for each case. Function autoplot can plot the output.

When the input is a dataframe, the locations, time and the quantity of interest needs to be given. When the input is a stars object, a 3 dimensional stars object needs to be given as input with the first 2 dimensions being spatial and the third being time.

Usage

```
semivariogram(
  x,
  latitude_linear = TRUE,
  longitude_linear = TRUE,
  missing_value = -9999,
  width = 80,
  cutoff = 1000,
  tlagmax = 6,
  ...
)

## S3 method for class 'data.frame'
semivariogram(
  x,
  latitude_linear = TRUE,
  longitude_linear = TRUE,
  missing_value = -9999,
  width = 80,
  cutoff = 1000,
  tlagmax = 6,
  times_df,
  values_df,
  ...
)

## S3 method for class 'stars'
semivariogram(
  x,
  latitude_linear = TRUE,
  longitude_linear = TRUE,
  missing_value = -9999,
  width = 80,
  cutoff = 1000,
  tlagmax = 6,
  ...
)

## S3 method for class 'semivariogramobj'
autoplot(object, ...)
```

Arguments

`x` The dataframe or stars object. If it is a dataframe, then it should have the locations.

`latitude_linear` If `TRUE` a linear model is fitted with latitude as a covariate is fitted.

`longitude_linear` If `TRUE` a linear model is fitted with longitude as a covariate is fitted.

`missing_value` If a certain value such as -9999 denotes the missing values for given locations and times.

`width` A parameter to the `gstat::variogram` function. The width of the distance intervals to be considered.

`cutoff` A parameter to the `gstat::variogram` function. The spatial separation distance.

`tlagmax` A parameter to the `gstat::variogram` function. The maximum time lag.

`...` Other arguments that need to be used for datafames or currently ignored.

`times_df` For dataframes: the dataframe containing the dates in `Date` format.

`values_df` For dataframes: the dataframe of dimension `length(times)` x `length(locations)` containing the quantity of interest.

`object` For autoplot: the output from the semivariogram function.

Value

A semivariogram object with a gstat variogram and the original data.

Examples

```
# Dataframe example
library(dplyr)
data(locs)
data(Times)
data(Tmax)
temp_part <- with(Times, paste(year, month, day, sep = ”-”))
temp_part <- data.frame(date = as.Date(temp_part)[913:923])
Tmax <- Tmax[913:923, ]
semidf <- semivariogram(locs,
        temp_part,
        Tmax,
        latitude_linear = FALSE,
        longitude_linear = FALSE,
        missing_value = -9999,
        width = 50,
        cutoff = 1000,
        tlagmax = 7
)
autoplot(semidf)

# Stars example
library(stars)
# Create a stars object from a data frame
precip_df <- NOAA_df_1990[NOAA_df_1990$proc == 'Precip', ] %>%
  filter(date >= "1992-02-01" & date <= "1992-02-05")
precip <- precip_df[ ,c('lat', 'lon', 'date', 'z')]
st_precip <- st_as_stars(precip, dims = c("lon", "lat", "date"))
semist <- semivariogram(st_precip)
autoplot(semist)
```

spatial_means *Computes spatial empirical means using a dataframe or a stars object*

Description

This function computes spatial empirical means by latitude and longitude averaged over time. This function can take either a stars object or a dataframe. Input arguments differ for each case. The autoplot function can plot this object.

The variations are * 'spatial_means.data.frame()' if the input is a dataframe * 'spatial_means.stars()' if the input is a stars object * 'autoplot.spatialmeans()' to plot the outputs.

Usage

```
spatial_means(x, ...)

## S3 method for class 'data.frame'
spatial_means(x, lat_col, lon_col, t_col, z_col, ...)

## S3 method for class 'stars'
spatial_means(x, ...)

## S3 method for class 'spatialmeans'
autoplot(
  object,
  ylab = "Mean Value",
  xlab1 = "Latitude",
  xlab2 = "Longitude",
  title = "Spatial Empirical Means",
  ...
)
```

Arguments

`x` A stars object or a dataframe. Arguments differ according to the input type.

`...` Other arguments currently ignored.

`lat_col` For dataframes: the column or the column name giving the latitude. The y coordinate can be used instead of latitude.

`lon_col` For dataframes: the column or the column name giving the longitude. The x coordinate can be used instead of longitude.

`t_col` For dataframes: the time column. Time must be a set of discrete integer values.

`z_col` For dataframes: the The quantity of interest that will be plotted. Eg. temperature.

`object` For autoplot: the output from the 'spatial_means' function.

`ylab` For autoplot: the ylabel.

`xlab1` For autoplot: The xlabel for the first plot.

`xlab2` For autuoplot: The xlabel for the second plot.

`title` The graph title.

Value

A spatialmeans object contaiing spatial averages and the original data.

Examples

```
# dataframe example
data(NOAA_df_1990)
library(dplyr)
Tmax <- filter(NOAA_df_1990,                       # subset the data
              proc == "Tmax" &                    # extract max temperature
                month %in% 5:9 &                  # May to July
                year == 1993)                     # year 1993
Tmax$t <- Tmax$julian - min(Tmax$julian) + 1     # create a new time variable starting at 1
sp_df <- spatial_means(Tmax,
       lat_col = "lat",
       lon_col = "lon",
       t_col = "t",
       z_col = "z")
autoplot(sp_df)

# stars examples
library(stars)
tif = system.file("tif/olinda_dem_utm25s.tif", package = "stars")
x <- read_stars(tif)
sp_means <- spatial_means(x)
autoplot(sp_means)
```

`spatial_snapshots` *Plots spatial snapshots of data through time using a dataframe or a stars object.*

Description

This function can take either a stars object or a dataframe. Input arguments differ for each case.

For dataframes, usage involves latitude and longitude. However, x and y coordinates can be given instead of longitude and latitude. If x and y are given instead of longitude and latitude, the country borders will not be shown.

Usage

```
spatial_snapshots(
  x,
  xlab = "x",
  ylab = "y",
  title = "",
  palette = "Spectral",
  legend_title = "z",
  ...
)

## S3 method for class 'data.frame'
spatial_snapshots(
  x,
  xlab = "Longitude",
  ylab = "Latitude",
  title = "",
  palette = "Spectral",
  legend_title = "z",
  lat_col,
  lon_col,
  t_col,
  z_col,
  ifxy = FALSE,
  ...
)

## S3 method for class 'stars'
spatial_snapshots(
  x,
  xlab = "x",
  ylab = "y",
  title = "",
  palette = "Spectral",
  legend_title = "z",
  ...
)
```

Arguments

x	A stars object or a dataframe. Arguments differ according to the input type.
xlab	The x label.
ylab	The y label.
title	The graph title.
palette	The color palette. Default is `Spectral`.
legend_title	The title for the legend.
...	Other arguments currently ignored.
lat_col	For dataframes: the column or the column name giving the latitude. The y coordinate can be used instead of latitude.
lon_col	For dataframes: the column or the column name giving the longitude. The x coordinate can be used instead of longitude.
t_col	For dataframes: the time column. Time must be a set of discrete integer values.
z_col	For dataframes: the The quantity of interest that will be plotted. Eg. temperature.
ifxy	For dataframes: if `TRUE` then the country borders are not drawn as longitude and latitude are unknown.

Value

A ggplot object.

Examples

```
library(dplyr)
# Dataframe example
data(NOAA_df_1990)
Tmax <- filter(NOAA_df_1990,
  proc == "Tmax" &
  month == 5 &
  year == 1993 &
  id < 4000)
Tmax$t <- Tmax$julian - min(Tmax$julian) + 1
Tmax_days <- subset(Tmax, t %in% c(1, 15))
spatial_snapshots(Tmax_days,
  lat_col = 'lat',
  lon_col = 'lon',
  t_col = 't',
  z_col = 'z',
  title = "Maximum Temperature for 2 days ")

# stars example
```

```
library(stars)
tif = system.file("tif/L7_ETMs.tif", package = "stars")
x <- read_stars(tif)
x2 <- x %>% slice(band, 1:2)
spatial_snapshots(x2)
```

`SSTdatashort` *The data from of the Sea Surface Temperature (SST) dataset. A subset of the original dataset is used.*

Description

The original dataset is included in the STRbook R package.

Usage

```
SSTdatashort
```

Format

A dataframe with 500 rows and 396 columns.

`SSTlandmaskshort` *The land mask for the Sea Surface Temperature (SST) dataset. A subset of the original dataset is used.*

Description

The original dataset is included in the STRbook R package.

Usage

```
SSTlandmaskshort
```

Format

A dataframe with 500 rows and 1 column.

mask A value of 1 is given if the location covers land. ...

SSTlonlatshort *The locations of the Sea Surface Temperatures (SST) dataset. A subset of the original dataset is used.*

Description

The original dataset is included in the STRbook R package.

Usage

```
SSTlonlatshort
```

Format

A data frame with 500 rows and 2 variables:

lon Longitude

lat Latitude ...

temporal_means *Computes temporal empirical means using a dataframe or a stars object.*

Description

This function computes temporal empirical means averaged per time unit. This function can take either a stars object or a dataframe. Input arguments differ for each case. The function autoplot plots the output.

Usage

```
temporal_means(x, ...)

## S3 method for class 'data.frame'
temporal_means(x, t_col, z_col, id_col, ...)

## S3 method for class 'stars'
temporal_means(x, ...)

## S3 method for class 'temporalmeans'
autoplot(
  object,
  ylab = "Value",
  xlab = "Time",
  legend_title = "",
  title = "Temporal Empirical Means",
  ...
)
```

Arguments

`x`	A stars object or a dataframe. Arguments differ according to the input type.
`...`	Other arguments currently ignored.
`t_col`	For dataframes: the time column. Time must be a set of discrete integer values.
`z_col`	For dataframes: the The quantity of interest that will be plotted. Eg. temperature.
`id_col`	The column of the location id.
`object`	For autoplot: the output of the function ‘temporal_means’.
`ylab`	The y label.
`xlab`	The x label.
`legend_title`	For autoplot: the title for the legend.
`title`	The graph title.

Value

An object of class temporalmeans containing the averages and the original data in two dataframes.

Examples

```
# dataframe example
data(NOAA_df_1990)
library(dplyr)
Tmax <- filter(NOAA_df_1990,                      # subset the data
              proc == "Tmax" &                   # extract max temperature
                month %in% 5:9 &                 # May to July
                year == 1993)                    # year 1993
Tmax$t <- Tmax$julian - min(Tmax$julian) + 1    # create a new time variable starting at 1
tem <- temporal_means(Tmax,
       t_col = 'date',
       z_col = 'z',
       id_col = 'id')
autoplot(tem)

# stars example
library(stars)
library(dplyr)
library(units)
# Example
prec_file = system.file("nc/test_stageiv_xyt.nc", package = "stars")
prec <- read_ncdf(prec_file)
temporal_means(prec)
```

temporal_snapshots *Plots temporal snapshots of data for specific spatial locations using a dataframe or a stars object.*

Description

This function plots temporal snapshos for specific spatial locations. The location id sample need to be given as a function argument.

Usage

```
temporal_snapshots(x, xlab = "x", ylab = "y", title = "", ...)

## S3 method for class 'data.frame'
temporal_snapshots(
  x,
  xlab = "Time",
  ylab = "Value",
  title = "",
  t_col,
  z_col,
  id_col,
  id_sample,
  ...
)

## S3 method for class 'stars'
temporal_snapshots(
  x,
  xlab = "Time",
  ylab = "Value",
  title = "",
  xvals,
  yvals,
  precision = 0,
  ...
)
```

Arguments

`x`	A stars object or a dataframe. Arguments differ according to the input type.
`xlab`	The x label.
`ylab`	The y label.
`title`	The graph title.
`...`	Other arguments currently ignored.

`t_col` For dataframes: the time column. Time must be a set of discrete integer values.

`z_col` For dataframes: the The quantity of interest that will be plotted. Eg. temperature.

`id_col` The column of the location id.

`id_sample` The sample of location ids to be plotted

`xvals` For stars objects: the set of xvalues to plot.

`yvals` For stars objects: the set of yvalues to plot. These two lengths need to be the same.

`precision` For stars objects: set to 0, if the given values are compared with the integer values in the stars object.

Value

A ggplot.

Examples

```
# Dataframe example
library(dplyr)
data(NOAA_df_1990)
Tmax <- filter(NOAA_df_1990,
             proc == "Tmax" &
             month %in% 5:9 &
             year == 1993)
Tmax_ID <- unique(Tmax$id)
Tmax$t <- Tmax$julian - min(Tmax$julian) + 1
ids <- sample(Tmax_ID, 10)
temporal_snapshots(Tmax,
                  t_col = 't',
                  z_col = 'z',
                  id_col = 'id',
                  id_sample = ids)


# stars example
library(stars)
tif = system.file("tif/L7_ETMs.tif", package = "stars")
x <- read_stars(tif)
xvals <- c(288876.0,289047.0)
yvals <- c(9120405, 9120006)
temporal_snapshots(x,
                  xvals = xvals,
                  yvals = yvals)
```

`Times` *The time period in which the NOAA dataset was recorded. This spans from January 1990 to December 1993.*

Description

This dataset is included in the STRbook R package.

Usage

```
Times
```

Format

A data frame with 1461 rows and 4 variables:

julian Day in Julian time

year The year

month The month

day The day ...

`Tmax` *The maximum temperature values used in the NOAA dataset in a wide dataframe format.*

Description

This dataset is included in the STRbook R package.

Usage

```
Tmax
```

Format

A data frame with 1461 rows and columns having maximum temperature for times and locations in data locs and Times.

Index