Package: climodr (via r-universe)

October 27, 2026

Type Package

Title Climate Modeling with Point Data from Climate Stations

Version 1.0.0-9.1.1

Author Alexander Klug [aut, cre, cph], Luise Wraase [aut], Seda Bekar [ctb], Dirk Zeuss [aut]

Maintainer Alexander Klug <kluga@students.uni-marburg.de>

Description An automated and streamlined workflow for predictive climate mapping using climate station data. Works within an environment the user provides a destined path to - otherwise it's tempdir(). Quick and relatively easy creation of resilient and reproducible climate models, predictions and climate maps, shortening the usually long and complicated work of predictive modelling. For more information, please find the provided URL. Many methods in this package are new, but the main method is based on a workflow from Meyer (2019) <doi:10.1016/j.ecolmodel.2019.108815> and Meyer (2022) <doi:10.1038/s41467-022-29838-9> , however, it was generalized and adjusted in the context of this package.

License GPL (>= 3)

Encoding UTF-8

LazyData true

Imports caret, CAST, corrplot, doParallel, dplyr, grDevices, lares, lubridate, magrittr, parallel, stats, stringr, terra, tidyr, utils

Depends R (>= 2.10)

Suggests knitr, rmarkdown, testthat (>= 3.0.0), pls, randomForest, sp

VignetteBuilder knitr

Config/testthat/edition 3

URL https://envima.github.io/climodr/

BugReports https://github.com/envima/climodr/issues

Config/roxygen2/version 8.0.0

RoxygenNote 7.3.3

Roxygen list(markdown = TRUE)

Config/pak/sysreqs libabsl-dev cmake libgdal-dev gdal-bin libgeos-dev libicu-dev libxml2-dev libssl-dev libproj-dev libsqlite3-dev libudunits2-dev

Repository https://envima.r-universe.dev

Date/Publication 2026-08-28 08:29:56 UTC

RemoteUrl https://github.com/envima/climodr

RemoteRef HEAD

RemoteSha 1abf6d2eb8d000c1a6c46b9bbb34c7419ad91ad9

Contents

aggregate_sensor . . . 3
autocorr . . . 4
calc.indices . . . 6
calc.model . . . 7
clim.sample . . . 10
climplot . . . 11
climpred . . . 13
crop.all . . . 17
envi.create . . . 19
ext_vignette . . . 20
extractPredictors . . . 20
fin.csv . . . 21
plot_description . . . 23
prep.csv . . . 24
prepClimateStations . . . 25
prepRasterData . . . 27
proc.csv . . . 29
readClimateData . . . 30
res_area . . . 31
sch_201707 . . . 32
sch_dgm . . . 33
spat.csv . . . 33
splitTime . . . 35
Station_G06 . . . 35
Station_G17 . . . 36
Station_G20 . . . 37
Station_G21 . . . 37
Station_G25 . . . 38
Station_G48 . . . 38
Station_W10 . . . 39
Station_W11 . . . 39
Station_W19 . . . 40
Station_W20 . . . 40

Index **41**

`aggregate_sensor` *Aggregate Sensor*

Description

Description

Usage

```
aggregate_sensor(
  data,
  sensors,
  station_ids,
  time_column,
  min_entries,
  coords = NULL
)
```

Arguments

`data`	climate station data used to be aggregated
`sensors`	character. Column names of your sensor columns.
`station_ids`	character. Column name of your Climate Station IDs.
`time_column`	character. The name of the time column in your climate station data.
`min_entries`	minimum entries a time interval must have to be validly aggregated (e.g. 18 days for a monthly aggregation)
`coords`	vector. If your climate station data already has coordinates, add coordinate column names in this argument (e.g. c("X", "Y"))

Value

data frame with aggregated climate station data

See Also

prepClimateStations

Examples

```
# example code
```

autocorr *Autocorrelation*

Description

Tests the final.csv created with `fin.csv` on autocorrelation to produce reliable models.

Usage

```
autocorr(
  envrmt = .GlobalEnv$envrmt,
  max_pvalue = 0.05,
  resp,
  pred,
  plot.corrplot = TRUE,
  corrplot = "coef",
  method = "monthly"
)
```

Arguments

`envrmt`	variable name of your envrmt list created using climodr's `envi.create` function. Default = envrmt.
`max_pvalue`	The maximum p value for a predictor variable in the autocorellation check.
`resp`	numerical. Vector or single input of the columns in the final.csv that contain your sensor data ("response variables"). The function will create one file per variable.
`pred`	numerical. Vector or single input. The columns of your predictor variables, that you want to test for autocorrelation with the response variables.
`plot.corrplot`	logical. Should correlation matrices be plotted?
`corrplot`	character. Vector or single input. If plot.corrplot is true, you can choose the design of the correlation plot. You can choose from "coef", "crossout", "blank". Default is "coef".
`method`	character. Choose the time scale your data is preserved in. Either "annual", "monthly" or "daily".

Value

One .csv file per response variable. These will later be used when `autocorrelation` is set `TRUE` during `calc.model`.

See Also

`calc.model`

Examples

```
#create climodr environment and allow terra-functions to use 70% of RAM
envrmt <- envi.create(proj_path = tempdir(),
                      memfrac = 0.7)

# Load the climodr example data into the current climodr environment
clim.sample(envrmt = envrmt)

#prepare csv-files
prep.csv(envrmt = envrmt,
         method = "proc",
         save_output = TRUE)

#process csv-files
csv_data <- proc.csv(envrmt = envrmt,
                     method = "monthly",
                     rbind = TRUE,
                     save_output = TRUE)

# Crop all raster bands
crop.all(envrmt = envrmt,
         method = "MB_Timeseries",
         overwrite = TRUE)

# Calculate Indices from cropped raster bands
calc.indices(envrmt = envrmt,
             vi = "all",
             bands = c("blue", "green", "red",
                       "nir", "nirb",
                       "re1", "re2", "re3",
                       "swir1", "swir2"),
             overwrite = TRUE)

#extract station coordinates
csv_spat <- spat.csv(envrmt = envrmt,
                     method = "monthly",
                     des_file = "plot_description.csv",
                     save_output = TRUE)


#extract predictor values from raster files
csv_fin <- fin.csv(envrmt = envrmt,
                   method = "monthly",
                   save_output = TRUE)

# Test data for autocorrelation after running fin.csv
autocorr(envrmt = envrmt,
         method = "monthly",
         max_pvalue = 0.05,
         resp = 5,
         pred = c(8:23),
         plot.corrplot = FALSE)
```

`calc.indices` *Calculate spectral indices*

Description

Calculates a set of spectral indices to have more predictor variables available when further modeling.

Usage

```
calc.indices(
  envrmt = .GlobalEnv$envrmt,
  vi = "all",
 bands = c("blue", "green", "red", "nir", "nirb", "re1", "re2", "re3", "swir1", "swir2"),
  overwrite = FALSE
)
```

Arguments

`envrmt`	variable name of your envrmt list created using climodr's `envi.create` function. Default = envrmt.
`vi`	Character. Either "all" or vector containing the preferred spectral indices. See 'Details' for more information.
`bands`	Character. Vector with lenght(bands) = 10. Contains the names of the bands in the Raster Stack. If bands from the *Usage* example vector dont exist, use "NA" in their position. See 'Details' for more information.
`overwrite`	logical. Argument passed down from `terra`-package. Overwrite existing files?

Value

SpatRaster-Stack

See Also

`crop.all`, `fin.csv`

Examples

```
#create climodr environment and allow terra-functions to use 70% of RAM
envrmt <- envi.create(proj_path = tempdir(),
                      memfrac = 0.7)

# Load the climodr example data into the current climodr environment
clim.sample(envrmt = envrmt)

# Crop all raster bands
```

```
crop.all(envrmt = envrmt,
         method = "MB_Timeseries",
         overwrite = TRUE)

# Calculate Indices from cropped raster bands
calc.indices(envrmt = envrmt,
             vi = "all",
             bands = c("blue", "green", "red",
                       "nir", "nirb",
                       "re1", "re2", "re3",
                       "swir1", "swir2"),
             overwrite = TRUE)
```

`calc.model` *Modelling*

Description

Creates Models for each climate value

Usage

```
calc.model(
  envrmt = .GlobalEnv$envrmt,
  climdata = NULL,
  climresp,
  predrows,
  station_ids = "plot",
  time_column = "datetime",
  timespan = NULL,
  classifier = c("rf", "pls", "lm", "glm"),
  seed = NULL,
  test = NULL,
  p = 0.8,
  folds = "all",
  mnote = NULL,
  k = NULL,
  tc_method = "cv",
  metric = "RMSE",
  doParallel = FALSE,
  autocorrelation = FALSE,
  method = NULL,
  ...
)
```

Arguments

envrmt	variable name of your envrmt list created using climodr's envi.create function. Default = envrmt.
climresp	numeric. Vector or single input. Should contain all column's in the tabular data that contain response variables.
predrows	numeric. Vector or single input. Should contain the rows where all the predictor values are stored in.
station_ids	character. Name of your Station_ID Column. Default = "plot"
time_column	character. Name of your datetime-column. Default = "datetime"
timespan	numeric. Vector or single input. Should contain all dates to be modeled. The dates should be in format %Y-%m-%d (default when created with climodr functions).
classifier	vector or character. Model variants to be used. Supported models: Random Forest = "rf", Partial-Least-Squares = "pls", Neural Networks = "nnet", Linear Regression = "lm" or generalized boosted regression = "gbm".
seed	integer. Seed to reproduce the same model over and over.
test	Vector. Either vector with row numbers of stations to be split for testing or object created by caret::createDataPartition().
p	numeric. Between 0 and 1. Percentage of data used for cross validation. Default = 0.8
folds	character. Vector or single input. Either folding over location only "LLO", over time only "LTO", or over both "LLTO". Use "all" to use all possibilitys.
mnote	character. Model note for special modifications used. Default: "normal"
k	integer. When 'fold' = "LLO" or "LTO". Set k to the number of unique spatial or temporal units. Leave out to use preset values.
tc_method	character. Method for train control function from caret package. Default = "cv".
metric	character. See train.
doParallel	logical. Parallelization accelerates the modelling process. Warning: Your PC will slow down drastically. Make sure to not run any other heavy processes during this.
autocorrelation	logical. Should autocorrelating data in the predictor variables be excluded from the model run? Only works if autocorr has been executed beforehand.
method	character. Time period of your desired model. Default: "monthly"
...	arguments passed down from other functions.

Value

data frame.

See Also

autocorr

Examples

```
#create climodr environment and allow terra-functions to use 70% of RAM
envrmt <- envi.create(proj_path = tempdir(),
                      memfrac = 0.7)

# Load the climodr example data into the current climodr environment
clim.sample(envrmt = envrmt)

#prepare csv-files
prep.csv(envrmt = envrmt,
         method = "proc",
         save_output = TRUE)

#process csv-files
csv_data <- proc.csv(envrmt = envrmt,
                     method = "monthly",
                     rbind = TRUE,
                     save_output = TRUE)

# Crop all raster bands
crop.all(envrmt = envrmt,
         method = "MB_Timeseries",
         overwrite = TRUE)

# Calculate Indices from cropped raster bands
calc.indices(envrmt = envrmt,
             vi = "all",
             bands = c("blue", "green", "red",
                       "nir", "nirb",
                       "re1", "re2", "re3",
                       "swir1", "swir2"),
             overwrite = TRUE)

#extract station coordinates
csv_spat <- spat.csv(envrmt = envrmt,
                     method = "monthly",
                     des_file = "plot_description.csv",
                     save_output = TRUE)


#extract predictor values from raster files
csv_fin <- fin.csv(envrmt = envrmt,
                   method = "monthly",
                   save_output = TRUE)

# Test data for autocorrelation after running fin.csv
autocorr(envrmt = envrmt,
         method = "monthly",
         resp = 5,
         pred = c(8:23),
         plot.corrplot = FALSE)
```

```
# Create 36 different models (12 months x 3 classifiers) for every month in 2017
calc.model(envrmt = envrmt,
           method = "monthly",
           timespan = c(2017),
           climresp = 5,
           classifier = c("rf",
                          "pls",
                          "lm"),
           seed = 707,
           p = 0.8,
           folds = "LLO",
           mnote = "normal",
           predrows = c(8:23),
           tc_method = "cv",
           metric = "RMSE",
           autocorrelation = TRUE,
           doParallel = FALSE)
```

`clim.sample` *Load in Example Data*

Description

Climodr comes with a full set of example data. But since this package runs primarily with data, that is not linked to the global environment, but saved in local folders build via `envi.create`, one can't just load example data. This function will load all the example data used in the vignette into your climodr environment. This way you can run all the code from the vignette.

Usage

```
clim.sample(envrmt = .GlobalEnv$envrmt)
```

Arguments

`envrmt`	variable name of your envrmt list created using climodr's `envi.create` function. Default = envrmt.

Value

Multiple files used by the climodr vignette

Examples

```
#create climodr environment and allow terra-functions to use 70% of RAM
envrmt <- envi.create(proj_path = tempdir(),
                      memfrac = 0.7)

# Load the climodr example data into the current climodr environment
clim.sample(envrmt = envrmt)
```

climplot *Create Maps using the* terra *package graphic parameters*

Description

Plot results of climodr into maps. Right now maps are created using the terra package. The maps created are very basic. Will be updated to run with tidyterra in future.

Usage

```
climplot(
  envrmt = .GlobalEnv$envrmt,
  mnote,
  sensor,
  aoa = FALSE,
  mapcolors = rev(grDevices::terrain.colors(50)),
  scale_position = "bottomleft",
  north_position = "topright"
)
```

Arguments

envrmt	variable name of your envrmt list created using climodr's `envi.create` function. Default = envrmt.
mnote	character. The modelnote you want to create maps of.
sensor	character. The sensor you want to create maps for.
aoa	logical. Do you want the area of applicability to be added to your map?
mapcolors	The color pallete you want to use for the map. Default is `rev(grDevices::terrain.colors(50))`
scale_position	character. Graphical parameter. The relative positiion of the Scale for the map. See `terra::plot` for more details.
north_position	character. Graphical parameter. The relative positiion of the Scale for the map. See `terra::plot` for more details.

Value

Maps in PNG-Format to your harddrive.

See Also

`terra::plot`

Examples

```
#create climodr environment and allow terra-functions to use 70% of RAM
envrmt <- envi.create(proj_path = tempdir(),
                      memfrac = 0.7)

# Load the climodr example data into the current climodr environment
clim.sample(envrmt = envrmt)

#prepare csv-files
prep.csv(envrmt = envrmt,
         method = "proc",
         save_output = TRUE)

#process csv-files
csv_data <- proc.csv(envrmt = envrmt,
                     method = "monthly",
                     rbind = TRUE,
                     save_output = TRUE)

# Crop all raster bands
crop.all(envrmt = envrmt,
         method = "MB_Timeseries",
         overwrite = TRUE)

# Calculate Indices from cropped raster bands
calc.indices(envrmt = envrmt,
             vi = "all",
             bands = c("blue", "green", "red",
                       "nir", "nirb",
                       "re1", "re2", "re3",
                       "swir1", "swir2"),
             overwrite = TRUE)

#extract station coordinates
csv_spat <- spat.csv(envrmt = envrmt,
                     method = "monthly",
                     des_file = "plot_description.csv",
                     save_output = TRUE)


#extract predictor values from raster files
csv_fin <- fin.csv(envrmt = envrmt,
                   method = "monthly",
                   save_output = TRUE)

# Test data for autocorrelation after running fin.csv
autocorr(envrmt = envrmt,
         method = "monthly",
         resp = 5,
         pred = c(8:23),
         plot.corrplot = FALSE)
```

```
# Create 36 different models (12 months x 3 classifiers) for every month in 2017
calc.model(envrmt = envrmt,
           method = "monthly",
           timespan = c(2017),
           climresp = 5,
           classifier = c("rf",
                          "pls",
                          "lm"),
           seed = 707,
           p = 0.8,
           folds = "LLO",
           mnote = "normal",
           predrows = c(8:23),
           tc_method = "cv",
           metric = "RMSE",
           autocorrelation = TRUE,
           doParallel = FALSE)

# Make predictions
climpred(envrmt = envrmt,
         method = "monthly",
         metric = "Nrmse",
         mnote = "normal",
         AOA = TRUE)

# Create a Temperature Map from the vignette model
climplot(envrmt = envrmt,
         mnote = "normal",
         sensor = "Ta_200",
         aoa = TRUE,
         mapcolors = rev(heat.colors(50)),
         scale_position = "bottomleft",
         north_position = "topright")
```

`climpred` *Predict sensor data area wide*

Description

Use the models created using `calc.model` to predict the modeled data onto a full spatial raster scene.

Use the models created using `calc.model` to predict the modeled data onto a full spatial raster scene.

Usage

```
climpred(
  envrmt = .GlobalEnv$envrmt,
```

```
  rasterdata = NULL,
  input = "Single",
  metric = "accuracy",
  mnote,
  AOA = TRUE,
  method = NULL
)

climpred(
  envrmt = .GlobalEnv$envrmt,
  rasterdata = NULL,
  input = "Single",
  metric = "accuracy",
  mnote,
  AOA = TRUE,
  method = NULL
)
```

Arguments

envrmt	variable name of your envrmt list created using climodr's `envi.create` function. Default = envrmt.
metric	Character. Which perfomance metric should be used to determine the best model? One of "accuracy", "Nrmse" or "Rsqrd". Default = "accuracy".
mnote	Character. Model note to filter models for the fitting model run.
AOA	Logical. Should the Area of Applicability be calculated additional to the models?
method	Character. Either "daily", monthly" or "annual". Also depends on the available data.

Value

Multiple models.rds stored in the /workflow/models folder.

Multiple models.rds stored in the /workflow/models folder.

See Also

`autocorr`, `predict`

`autocorr`, `predict`

Examples

```
#create climodr environment and allow terra-functions to use 70% of RAM
envrmt <- envi.create(proj_path = tempdir(),
                      memfrac = 0.7)

# Load the climodr example data into the current climodr environment
clim.sample(envrmt = envrmt)
```

```
#prepare csv-files
prep.csv(envrmt = envrmt,
         method = "proc",
         save_output = TRUE)

#process csv-files
csv_data <- proc.csv(envrmt = envrmt,
                     method = "monthly",
                     rbind = TRUE,
                     save_output = TRUE)

# Crop all raster bands
crop.all(envrmt = envrmt,
         method = "MB_Timeseries",
         overwrite = TRUE)

# Calculate Indices from cropped raster bands
calc.indices(envrmt = envrmt,
             vi = "all",
             bands = c("blue", "green", "red",
                       "nir", "nirb",
                       "re1", "re2", "re3",
                       "swir1", "swir2"),
             overwrite = TRUE)

#extract station coordinates
csv_spat <- spat.csv(envrmt = envrmt,
                     method = "monthly",
                     des_file = "plot_description.csv",
                     save_output = TRUE)


#extract predictor values from raster files
csv_fin <- fin.csv(envrmt = envrmt,
                   method = "monthly",
                   save_output = TRUE)

# Test data for autocorrelation after running fin.csv
autocorr(envrmt = envrmt,
         method = "monthly",
         resp = 5,
         pred = c(8:23),
         plot.corrplot = FALSE)

# Create 36 different models (12 months x 3 classifiers) for every month in 2017
calc.model(envrmt = envrmt,
           method = "monthly",
           timespan = c(2017),
           climresp = 5,
           classifier = c("rf",
                          "pls",
                          "lm"),
```

```
            seed = 707,
            p = 0.8,
            folds = "LLO",
            mnote = "normal",
            predrows = c(8:23),
            tc_method = "cv",
            metric = "RMSE",
            autocorrelation = TRUE,
            doParallel = FALSE)

# Make predictions
results <- climpred(envrmt = envrmt,
                    method = "monthly",
                    metric = "Nrmse",
                    mnote = "normal",
                    AOA = TRUE)

results$Prediction
results$Validation



#create climodr environment and allow terra-functions to use 70% of RAM
envrmt <- envi.create(proj_path = tempdir(),
                      memfrac = 0.7)

# Load the climodr example data into the current climodr environment
clim.sample(envrmt = envrmt)

#prepare csv-files
prep.csv(envrmt = envrmt,
         method = "proc",
         save_output = TRUE)

#process csv-files
csv_data <- proc.csv(envrmt = envrmt,
                     method = "monthly",
                     rbind = TRUE,
                     save_output = TRUE)

# Crop all raster bands
crop.all(envrmt = envrmt,
         method = "MB_Timeseries",
         overwrite = TRUE)

# Calculate Indices from cropped raster bands
calc.indices(envrmt = envrmt,
             vi = "all",
             bands = c("blue", "green", "red",
                       "nir", "nirb",
                       "re1", "re2", "re3",
                       "swir1", "swir2"),
             overwrite = TRUE)
```

```
#extract station coordinates
csv_spat <- spat.csv(envrmt = envrmt,
                     method = "monthly",
                     des_file = "plot_description.csv",
                     save_output = TRUE)


#extract predictor values from raster files
csv_fin <- fin.csv(envrmt = envrmt,
                   method = "monthly",
                   save_output = TRUE)

# Test data for autocorrelation after running fin.csv
autocorr(envrmt = envrmt,
         method = "monthly",
         resp = 5,
         pred = c(8:23),
         plot.corrplot = FALSE)

# Create 36 different models (12 months x 3 classifiers) for every month in 2017
calc.model(envrmt = envrmt,
           method = "monthly",
           timespan = c(2017),
           climresp = 5,
           classifier = c("rf",
                          "pls",
                          "lm"),
           seed = 707,
           p = 0.8,
           folds = "LLO",
           mnote = "normal",
           predrows = c(8:23),
           tc_method = "cv",
           metric = "RMSE",
           autocorrelation = TRUE,
           doParallel = FALSE)

# Make predictions
results <- climpred(envrmt = envrmt,
                    method = "monthly",
                    metric = "Nrmse",
                    mnote = "normal",
                    AOA = TRUE)

results$Prediction
results$Validation
```

`crop.all` *Cropping tiff data*

Description

Crops input data to the extent size and reprojects them into project Coordinate reference system.

Usage

```
crop.all(
  envrmt = .GlobalEnv$envrmt,
  method = "MB_Timeseries",
  crs = NULL,
  ext = NULL,
  overwrite = FALSE,
  ...
)
```

Arguments

envrmt	variable name of your envrmt list created using climodr's envi.create function. Default = envrmt.
method	character. Use "MB_Timeseries" for now. More methods are planned and will be added in future.
crs	Coordinate reference system Used to crop all images in folder_path. If crs it will automatically reprojected into this one. Default: crs of smallest Extent.
ext	SpatRaster, SpatVector or SpatExtent. Extent all data is cropped into. Default: Smallest Extent in folder_path.
overwrite	logical. Should existing files with the same filename be overwritten? Default = FALSE
...	arguments passed down from other functions.

Value

SpatRaster-Stack. Also saved to /workflow/rworkflow

See Also

fin.csv, calc.indices

Examples

```
#create climodr environment and allow terra-functions to use 70% of RAM
envrmt <- envi.create(proj_path = tempdir(),
                      memfrac = 0.7)

# Load the climodr example data into the current climodr environment
clim.sample(envrmt = envrmt)

# Crop all raster bands
crop.all(envrmt = envrmt,
         method = "MB_Timeseries",
```

```
            overwrite = TRUE)
```

---

envi.create *Create climodr environment*

Description

Creates an environment climodr will use during the calculation process. A list is returned with all paths to all folders. After creating the environment, all necessary data should be stored into the depending Input sub-folders. There is also an additional temp-folder, where temporary data is stored, which can be deleted after not being used anymore.

Usage

```
envi.create(proj_path = tempdir(), memfrac = NULL, ...)
```

Arguments

proj_path	character. Path to project directory. Climodr will work exclusively in this folder and create all project folders in here.
memfrac	numeric. Value between 0 and 0.9. The fraction of RAM that may be used by the terra package
...	arguments passed down from other functions.

Value

list. Contains all paths to each folder in the project directory. Necessary for climodr to operate its functions.

Examples

```
# create climodr environment and allow terra-functions to use 70% of RAM
envrmt <- envi.create(proj_path = tempdir(),
                      memfrac = 0.7)
```

ext_vignette *Extent file for vignette*

Description

A vector file containing the shape and extent of the Area used in the vignette

Usage

```
ext_vignette
```

Format

ext_vignette:
class SpatVector
geometry polygons
dimensions 1, 1 (geometries, attributes)
extent 805737, 812824, 5890352, 5896005 (xmin, xmax, ymin, ymax)
coord. ref. WGS 84 / UTM zone 32N (EPSG:32632)
values 1
Spat Vector

Source

Randomly created in (QGIS)https://www.qgis.org/download/thank-you/.

extractPredictors *Extraction of Raster data at climate stations*

Description

Extract the raster values of all raster layers from a scene at the station coordinates at each time stamp. The extracted data will be attached to the station data so there is a .csv-file with coordinates, sensor data (response values) and extracted raster data (predictor values). The data is ready to be used for modelling.

Usage

```
extractPredictors(
  envrmt = .GlobalEnv$envrmt,
  climdata,
  rasters,
  unit,
  input = "Single",
  assign_raster_dates = "floor",
  save_output = FALSE
)
```

Arguments

envrmt	variable name of your envrmt list created using climodr's envi.create function. Default = envrmt.
climdata	Data frame. Climate station produced by spat.csv. Or produced via prepClimateStation. If null, climate statioon data is searched in Workflow/tworkflow.
rasters	SpatRaster. Either a single raster with layers fitting for the desired time stamp. Or a bundled raster with fitting time stamps on the according layers. If empty, climodr searches for files created via prepRasterData() in Output/rfinal.
unit	character.
input	character. Either "single" and "bundle". Depends on what output one has choose. "single" means a SpatRaster that only fits to one date, "bundle" is a SpatRaster contianing layers with time stamps fitting to the climate station data. Default = "bundle".
assign_raster_dates	character. One of "floor", "ceiling" or "round". How should climate data be matched to a layer? • floor: Climate data is matched to the earliest date of the time interval (e.g. first day of a month) • ceiling: Climate data is matched to the latest date of the time interval (e.g. last day of a month) • round: Climate data is matched to the closest data available. (e.g. split between two months.)
...	arguments passed down from other functions.

Value

data.frame

See Also

prep.csv, proc.csv, spat.csv, calc.indices

Examples

```
sdfsdf
```

---

fin.csv *Final aggregation for CSV-Data*

Description

Extract the raster values of all raster layers from a scene at the station coordinates at each time stamp. The extracted data will be attached to the station data so there is a .csv-file with coordinates, sensor data (response values) and extracted raster data (predictor values). The data is ready to be used for modelling.

Usage

```
fin.csv(
  envrmt = .GlobalEnv$envrmt,
  x = NULL,
  method = "monthly",
  crs = NULL,
  save_output = TRUE,
  ...
)
```

Arguments

envrmt	variable name of your envrmt list created using climodr's envi.create function. Default = envrmt.
x	Data frame. Climate station produced by spat.csv. Or produced via prepClimateStation. If null, climate statioon data is searched in Workflow/tworkflow.
method	character. Either "daily", monthly" or "annual". Also depends on the available data.
crs	character. If null, coordinate reference system from project files will be taken. Otherwise data will be reprojected into this crs.
save_output	logical. If cleaned data should be saved permanently in the Environment put save_output = TRUE. Otherwise the output will be saved in the temporary directory. Default: FALSE.
...	arguments passed down from other functions.

Value

List

See Also

prep.csv, proc.csv, spat.csv, calc.indices

Examples

```
#create climodr environment and allow terra-functions to use 70% of RAM
envrmt <- envi.create(proj_path = tempdir(),
                      memfrac = 0.7)

# Load the climodr example data into the current climodr environment
clim.sample(envrmt = envrmt)

#prepare csv-files
prep.csv(envrmt = envrmt,
         method = "proc",
         save_output = TRUE)

#process csv-files
```

```
csv_data <- proc.csv(envrmt = envrmt,
                     method = "monthly",
                     rbind = TRUE,
                     save_output = TRUE)

# Crop all raster bands
crop.all(envrmt = envrmt,
         method = "MB_Timeseries",
         overwrite = TRUE)

# Calculate Indices from cropped raster bands
calc.indices(envrmt = envrmt,
             vi = "all",
             bands = c("blue", "green", "red",
                       "nir", "nirb",
                       "re1", "re2", "re3",
                       "swir1", "swir2"),
             overwrite = TRUE)

#extract station coordinates
csv_spat <- spat.csv(envrmt = envrmt,
                     method = "monthly",
                     des_file = "plot_description.csv",
                     save_output = TRUE)


#extract predictor values from raster files
csv_fin <- fin.csv(envrmt = envrmt,
                   method = "monthly",
                   save_output = TRUE)
head(csv_fin)
```

`plot_description` *Plot description file*

Description

Contains made up coordinates for imaginary climate stations for education purposes.

Usage

```
plot_description
```

Format

`plot_description`**:**
A data frame with 10 rows and 5 columns

plot imaginary station name and code

general imaginary category of climate station

region location name of climate station

lat Latitude Coordinate of climate station

lon Longitude Coordinate of climate station

elevation elevation of climate station

Source

Randomly created coordinates and stations extracted from a random climate map.

`prep.csv` *Preparing CSV-Data*

Description

Crops input data to the extent size and removes NA-Values

Usage

```
prep.csv(envrmt = .GlobalEnv$envrmt, method = "proc", save_output = TRUE, ...)
```

Arguments

`envrmt`	variable name of your envrmt list created using climodr's `envi.create` function. Default = envrmt.
`method`	character. "proc" for ready-to-use data in separate .csv-files. "tube" for raw-data from the Tube Data Base. Default "proc"-Method.
`save_output`	logical. If cleaned data should be saved permanently in the Environment put save_output = TRUE. Otherwise the output will be saved in the temporary directory. Default: FALSE.
`...`	arguments passed down from other functions.

Value

List

See Also

`proc.csv`, `spat.csv`, `fin.csv`

Examples

```
#create climodr environment and allow terra-functions to use 70% of RAM
envrmt <- envi.create(proj_path = tempdir(),
                      memfrac = 0.7)

# Load the climodr example data into the current climodr environment
clim.sample(envrmt = envrmt)

#prepare csv-files
prep.csv(envrmt = envrmt,
         method = "proc",
         save_output = TRUE)

#check the created csv files
csv_files <- grep("_no_NAs.csv$",
                  list.files(envrmt$path_tworkflow),
                  value=TRUE)
csv_files
```

`prepClimateStations` *Preparing Climate Station Data*

Description

Crops input data to the extent size and removes NA-Values

Usage

```
prepClimateStations(
  envrmt = .GlobalEnv$envrmt,
  x,
  pattern = NULL,
  metadata = NULL,
  time_column,
  time_format,
  station_ids,
  sensors,
  aggregation_interval = NULL,
  start = NULL,
  end = NULL,
  crs = NULL,
  sep = ",",
  dec = ".",
  save_output = TRUE
)
```

Arguments

`envrmt`	variable name of your envrmt list created using climodr's `envi.create` function. Default = envrmt.
`x`	dataframe or climodr environment path. A data frame containing climate station data or one of the Input paths of the envrmt, where your climate station data is stored. Valid inputs for a climodr environment path are `"envrmt$path_tabular"` or `"envrmt$path_vector"`. The 'envrmt'-variable itself has to be the same as it is called in your global environment.
`pattern`	character. Some string indicating that file is a climate station. E.g. "Climate_Station_*.csv" All climate station files in input folder must contain this pattern in filename.
`metadata`	vector. Only needed if climate station data comes in .csv format. Must be one single Metadata-file in .csv format, stored in "Input/dep".The purpose of this argument is to extract the coordinates from the metadata-file and assign them to the corresponding climate station data. The Metadata should describe the additional information of the climate stations. Usually, and in this case, this means there is one climate station per row, and the columns necessary for climodr to use this file are the station names (MUST be the same names as in the climate station data) and seperated in two columns the x coordinate and the y coordinate. THe file should contain column names, but no row name and no additional rows to skip. A vector for this argument always consisting out of 5 entries: • Name of your metadata-file in your Input/dep folder • the column name for the climate station names • the column name consisting of your X coordinates from the climate stations • the column name of your y coordinates from the climate Stations • the coordinate reference system for these coordinates
`time_column`	character. The name of the time column in your climate station data.
`time_format`	character. How is your time column formated? (e.g. YYYY-MM-DD)
`station_ids`	character. The name of the station id column of your climate station data.
`sensors`	vector. Containing all column names of each sensor in your climate station data.
`aggregation_interval`	character. To what format should your data be aggregated? Same as units in `lubridate::floor_date()`. Valid intervals are `second`, `minute`, `hour`, `day`, `week`, `month`, `bimonth`, `quarter`, `season`, `halfyear` and `year`.
`start`	character. When do you want your climate station data to start? Has to be exactly the same format as your input climate station data.
`end`	character. When do you want your climate station data to end? Has to be exactly the same format as your input cliamte station data.
`crs`	character. Destined coordinate reference system. If crs = NULL, climod searches for "res_area.tif"-file in Input/dep, which is climodrs prefered way to operate. In here, the file is supposed to have your desired raster resolution and coordinate reference systeme, that climodr simply copies from that file.

sep	character. Which seperator is used for the columns in the climate station data? (Default = ",")
dec	character. Which symbol is used to mark decimal digits? (Default = ".")
...	arguments passed down from other functions.

Value

data frame with aggregated climate station data

See Also

replacement function for the old prep.csv, proc.csv, spat.csv functions.

Examples

```
# create climodr environment and allow terra-functions to use 70% of RAM
envrmt <- envi.create(proj_path = tempdir(),
                      memfrac = 0.7)

# load example data
clim.sample(envrmt = envrmt)

# prepare climate station data
climdata <- prepClimateStations(envrmt = envrmt,
                                x = envrmt$path_tabular,
                                pattern = "Station",
                                metadata = c("plot_description.csv",
                                             "plot",
                                             "lon",
                                             "lat",
                                             "+proj=longlat +datum=WGS84"),
                                time_column = "datetime",
                                time_format = "%Y-%m-%dT%H",
                                station_ids = "plotID",
                                sensors = "Ta_200",
                                aggregation_interval = "month")
head(climdata)
```

---

prepRasterData *Prepare Raster Data*

Description

Crops input data to the extent size and reprojects them into project Coordinate reference system.

Usage

```
prepRasterData(
  envrmt = .GlobalEnv$envrmt,
  datepos,
  dateformat,
  crs = NULL,
  ext = NULL,
  mask = NULL,
  add_spectral_indices = FALSE,
  bands = NULL,
  output = "Single",
  overwrite = FALSE
)
```

Arguments

envrmt	variable name of your envrmt list created using climodr's envi.create function. Default = envrmt.
datepos	numeric vector. At which position in the filename does the date of your raster data start and where does it end?
dateformat	character. The format of your date. See `base::strptime()`
crs	Coordinate reference system Used to crop all images in folder_path.
ext	SpatRaster, SpatVector or SpatExtent. Extent all data is cropped into. Default: Smallest Extent in folder_path.
mask	SpatVector or filename. Assign a polygon to mask your raster files. Either add file from global environment or the filename of your mask in Input/vector. Potentially reduces calculation times, as fewer pixels are contained in rasters after masking.
add_spectral_indices	logical. Should spectral indices be calculated from input raster bands?
bands	vector. If spectral indices should be calculated, this vector should contain the names of your required spectral bands. The order in the vector is fixed like this: c("blue", "green", "red", "nir", "swir"). Just switch out names in the vector with your equivalent band names.
output	charakter. Either "single" or "bundle". "single" saves your dated raster data separately per date in Output/rfinal. "bundle" marks the corresponding layers with a time stamp and safes the dated rasters in a single file. Works more efficiently than "single", as long as there are more than one dates.
...	arguments passed down from other functions.

Value

SpatRaster-Stack. Also saved to /workflow/rworkflow

See Also

`fin.csv`, `calc.indices`

Examples

```
#create climodr environment and allow terra-functions to use 70% of RAM
```

`proc.csv` *Processing CSV-Data*

Description

Calculate averaged sensor values aggregated to a given time interval.

Usage

```
proc.csv(
  envrmt = .GlobalEnv$envrmt,
  method = "monthly",
  rbind = TRUE,
  save_output = TRUE,
  ...
)
```

Arguments

`envrmt`	variable name of your envrmt list created using climodr's `envi.create` function. Default = envrmt.
`method`	character. Either "daily", monthly" or "annual". Also depends on the available data.
`rbind`	logical. Create a single file with all climate stations. If FALSE, every station will be saved in a seperate file.
`save_output`	logical. If data should be saved permanently in the Environment put save_output = TRUE. Otherwise the output will be saved in the temporary directory. Default: TRUE.
`...`	arguments passed down from other functions.

Value

List

See Also

`prep.csv`, `spat.csv`, `fin.csv`

Examples

```
#create climodr environment and allow terra-functions to use 70% of RAM
envrmt <- envi.create(proj_path = tempdir(),
                      memfrac = 0.7)

# Load the climodr example data into the current climodr environment
clim.sample(envrmt = envrmt)

#prepare csv-files
prep.csv(envrmt = envrmt,
         method = "proc",
         save_output = TRUE)

#process csv-files
csv_data <- proc.csv(envrmt = envrmt,
                     method = "monthly",
                     rbind = TRUE,
                     save_output = TRUE)
head(csv_data)
```

---

readClimateData *Read climate station data*

Description

Readable climate station data for now: .csv, .txt, .gpkg, .shp

Usage

```
readClimateData(
  station_list,
  folder,
  station_ids,
  metadata = NULL,
  sep = ",",
  dec = "."
)
```

Arguments

station_list	list. A list containing all paths of each climate station that should be read.
folder	path. Folder where all the climate station data is stored in. Specifically made for climodr-environment. Either "$path_tabular" or "$path_vector".
station_ids	character. The name of the station id column of your climate station data.
metadata	vector. 5 Entries. Name of your metadata-file in your Input/dep folder, the column name for the climate stations, the name of your X and Y columns and the coordinate reference system. (e.g. c("metadata.csv", "plotID", "lon", "lat", "+proj=longlat +datum=WGS84")). NULL if data is vector_data.

sep	character. Which seperator is used for the columns in the climate station data? (Default = ",")
dec	character. Which symbol is used to mark decimal digits? (Default = ".")

Value

data frame or list. Depends if climate stations are vector files or tabular files. Tabular files return a single data frame, vector files return a list with the same data frame and a metadata vector, containting information about the metadata lost in conversion.

See Also

prepClimateStations

Examples

```
# create climodr environment and allow terra-functions to use 70% of RAM
envrmt <- envi.create(proj_path = tempdir(),
                      memfrac = 0.7)

# load example data
climsample(envrmt = envrmt)

# read climate station data
climate_stations <- list.files(envrmt$path_tabular, pattern = "Station")
csd <- readClimateData(station_list = climate_stations,
                       folder = envrmt$path_tabular,
                       station_ids = "plotID",
                       metadata = c("plot_description.csv",
                                    "plot",
                                    "x",
                                    "y",
                                    "+proj=longlat +datum=WGS84"))
head(csd)
```

---

res_area *Resolution and Area*

Description

This raster contains the area of interest as well as the desired model resolution (100 m * 100 m) and the project extent.

Usage

```
res_area
```

Format

`res_area`:
A binary Raster of pixels with value 1 in extent that belong to example area

class SpatRaster
dimensions 57, 71, 1 (nrow, ncol, nlyr)
resolution 100, 100 (x, y)
extent 805732, 812832, 5890310, 5896010 (xmin, xmax, ymin, ymax)
coord. ref. WGS 84 / UTM zone 32N (EPSG:32632)
name res_area
min/max 0/1

Source

Randomly created binary Spat Raster file with the project resolution of 100 m per pixel. Created in (QGIS)https://www.qgis.org/download/thank-you/.

---

sch_201707 *Spatial Raster File for Vignette*

Description

A spatial Raster file from a random area choose for the Vignette or as dummy data.

Usage

```
sch_201707
```

Format

`sch_201707`:
A Spat Raster with 8 spectral bands

class SpatRaster
dimensions 86, 151, 10 (nrow, ncol, nlyr)
resolution 100, 100 (x, y)
extent 801522.5, 816622.5, 5888973, 5897573 (xmin, xmax, ymin, ymax)
coord. ref. WGS 84 / UTM zone 32N (EPSG:32632)
names blue, green, red, nir, nirb, re1, re2, re3, swir1, swir2
min/max 33.90298/5479.6602

Source

Randomly created Spat Raster file from (Sentinel-2 Data)https://browser.dataspace.copernicus.eu/?zoom=10&lat=52.96601&
THEME&visualizationUrl=U2FsdGVkX1%2BIsGtPIR6VONjAIVeaSIsxAeEPdOA40qQAQbcMBNQU8hm72ssEgHpSw4
07-01T00%3A00%3A00.000Z&toTime=2017-07-31T23%3A59%3A59.999Z&layerId=1_TRUE_COLOR.

sch_dgm *Digital Ground Model for Vignette*

Description

A Digital Ground Model file from a random area choose for the Vignette or as dummy data.

Usage

```
sch_dgm
```

Format

sch_dgm**:**
A Digital Ground Model
class SpatRaster
dimensions 86, 151, 10 (nrow, ncol, nlyr)
resolution 100, 100 (x, y)
extent 801522.5, 816622.5, 5888973, 5897573 (xmin, xmax, ymin, ymax)
coord. ref. WGS 84 / UTM zone 32N (EPSG:32632)
names elevation
min/max 48.75315/94.67307

Source

Randomly extracted Digital Ground Model.

spat.csv *Spatial aggregation for CSV-Data*

Description

Extract station coordinates from meta-data and reproject the coordinates to the project coordinate reference system.

Usage

```
spat.csv(
  envrmt = .GlobalEnv$envrmt,
  method = "monthly",
  des_file,
  crs = NULL,
  save_output = TRUE,
  ...
)
```

Arguments

envrmt	variable name of your envrmt list created using climodr's envi.create function. Default = envrmt.
method	character. Either "daily", monthly" or "annual". Also depends on the available data.
des_file	character. The filename and data type of the meta-data. (Only reads .csv)
crs	character. EPSG of the Coordinate Reference System, if no **res_area.tif** file is provided.
save_output	logical. If cleaned data should be saved permanently in the Environment put save_output = TRUE. Otherwise the output will be saved in the temporary directory. Default: TRUE
...	arguments passed down from other functions.

Value

Data Frame

See Also

prep.csv, proc.csv, fin.csv

Examples

```
#create climodr environment and allow terra-functions to use 70% of RAM
envrmt <- envi.create(proj_path = tempdir(),
                      memfrac = 0.7)

# Load the climodr example data into the current climodr environment
clim.sample(envrmt = envrmt)

#prepare csv-files
prep.csv(envrmt = envrmt,
         method = "proc",
         save_output = TRUE)

#process csv-files
csv_data <- proc.csv(envrmt = envrmt,
                     method = "monthly",
                     rbind = TRUE,
                     save_output = TRUE)

#extract station coordinates
csv_spat <- spat.csv(envrmt = envrmt,
                     method = "monthly",
                     des_file = "plot_description.csv",
                     save_output = TRUE)
head(csv_spat)
```

splitTime *Split Time*

Description

Description

Usage

```
splitTime(data, time_column, smallest_interval)
```

Arguments

`data`	Data frame. Climate station data with time column in POSIXct-format
`time_column`	Character. Column name of the column containing timestamps in POSIXct-format
`smallest_interval`	The smallest time intervall the data should be split into. Same as units in `lubridate::floor_date()`

Value

data frame with aggregated climate station data

See Also

prepClimateStations

Examples

```
# example code
```

Station_G06 *Station File (G06)*

Description

Contains made up climate data for imaginary climate stations for education purposes.

Usage

```
Station_G06
```

Format

`Station_G06`**:**

A data frame with 10 rows and 5 columns

plotID Plot ID of imaginary climate station

datetime Timestamp of record for station

Ta_200 Imaginary air temperature at 200 cm above ground

Source

Randomly created climate data at random created stations extracted from a random climate map made by climodr.

`Station_G17` *Station File (G17)*

Description

Contains made up climate data for imaginary climate stations for education purposes.

Usage

```
Station_G17
```

Format

`Station_G17`**:**

A data frame with 10 rows and 5 columns

plotID Plot ID of imaginary climate station

datetime Timestamp of record for station

Ta_200 Imaginary air temperature at 200 cm above ground

Source

Randomly created climate data at random created stations extracted from a random climate map made by climodr.

Station_G20 *Station File (G20)*

Description

Contains made up climate data for imaginary climate stations for education purposes.

Usage

```
Station_G20
```

Format

`Station_G20`:
A data frame with 10 rows and 5 columns
plotID Plot ID of imaginary climate station
datetime Timestamp of record for station
Ta_200 Imaginary air temperature at 200 cm above ground

Source

Randomly created climate data at random created stations extracted from a random climate map made by climodr.

Station_G21 *Station File (G21)*

Description

Contains made up climate data for imaginary climate stations for education purposes.

Usage

```
Station_G21
```

Format

`Station_G21`:
A data frame with 10 rows and 5 columns
plotID Plot ID of imaginary climate station
datetime Timestamp of record for station
Ta_200 Imaginary air temperature at 200 cm above ground

Source

Randomly created climate data at random created stations extracted from a random climate map made by climodr.

Station_G25 *Station File (G25)*

Description

Contains made up climate data for imaginary climate stations for education purposes.

Usage

```
Station_G25
```

Format

`Station_G25`**:**
A data frame with 10 rows and 5 columns
plotID Plot ID of imaginary climate station
datetime Timestamp of record for station
Ta_200 Imaginary air temperature at 200 cm above ground

Source

Randomly created climate data at random created stations extracted from a random climate map made by climodr.

Station_G48 *Station File (G48)*

Description

Contains made up climate data for imaginary climate stations for education purposes.

Usage

```
Station_G48
```

Format

`Station_G48`**:**
A data frame with 10 rows and 5 columns
plotID Plot ID of imaginary climate station
datetime Timestamp of record for station
Ta_200 Imaginary air temperature at 200 cm above ground

Source

Randomly created climate data at random created stations extracted from a random climate map made by climodr.

Station_W10 *Station File (W10)*

Description

Contains made up climate data for imaginary climate stations for education purposes.

Usage

```
Station_W10
```

Format

`Station_W10`:
A data frame with 10 rows and 5 columns
plotID Plot ID of imaginary climate station
datetime Timestamp of record for station
Ta_200 Imaginary air temperature at 200 cm above ground

Source

Randomly created climate data at random created stations extracted from a random climate map made by climodr.

Station_W11 *Station File (W11)*

Description

Contains made up climate data for imaginary climate stations for education purposes.

Usage

```
Station_W11
```

Format

`Station_W11`:
A data frame with 10 rows and 5 columns
plotID Plot ID of imaginary climate station
datetime Timestamp of record for station
Ta_200 Imaginary air temperature at 200 cm above ground

Source

Randomly created climate data at random created stations extracted from a random climate map made by climodr.

Station_W19 *Station File (W19)*

Description

Contains made up climate data for imaginary climate stations for education purposes.

Usage

```
Station_W19
```

Format

`Station_W19`:
A data frame with 10 rows and 5 columns
plotID Plot ID of imaginary climate station
datetime Timestamp of record for station
Ta_200 Imaginary air temperature at 200 cm above ground

Source

Randomly created climate data at random created stations extracted from a random climate map made by climodr.

Station_W20 *Station File (W20)*

Description

Contains made up climate data for imaginary climate stations for education purposes.

Usage

```
Station_W20
```

Format

`Station_W20`:
A data frame with 10 rows and 5 columns
plotID Plot ID of imaginary climate station
datetime Timestamp of record for station
Ta_200 Imaginary air temperature at 200 cm above ground

Source

Randomly created climate data at random created stations extracted from a random climate map made by climodr.

Index

∗ **datasets**
ext_vignette, 20
plot_description, 23
res_area, 31
sch_201707, 32
sch_dgm, 33
Station_G06, 35
Station_G17, 36
Station_G20, 37
Station_G21, 37
Station_G25, 38
Station_G48, 38
Station_W10, 39
Station_W11, 39
Station_W19, 40
Station_W20, 40

aggregate_sensor, 3
autocorr, 4

base::strptime(), *28*

calc.indices, 6
calc.model, 7
clim.sample, 10
climplot, 11
climpred, 13
crop.all, 17

envi.create, 19, *28*
ext_vignette, 20
extractPredictors, 20

fin.csv, 21

https://browser.dataspace.copernicus.eu/?zoom=10&lat=52.96601&lng=13.86818&themeId=DEFAULT-THEME&visual
32
https://www.qgis.org/download/thank-you/,
20, *32*

lubridate::floor_date(), *26*, *35*

plot_description, 23
prep.csv, 24, *27*
prepClimateStations, *3*, 25, *31*, *35*
prepRasterData, 27
prepRasterData(), *21*
proc.csv, *27*, 29

readClimateData, 30
res_area, 31

sch_201707, 32
sch_dgm, 33
spat.csv, *27*, 33
splitTime, 35
Station_G06, 35
Station_G17, 36
Station_G20, 37
Station_G21, 37
Station_G25, 38
Station_G48, 38
Station_W10, 39
Station_W11, 39
Station_W19, 40
Station_W20, 40