Package 'envirem'

October 13, 2022

Type Package

Title Generation of ENVIREM Variables

Version 2.3

Date 2021-05-13

Author Pascal O. Title, Jordan B. Bemmels

Maintainer Pascal Title <pascal.title@stonybrook.edu>

Depends raster, palinsol

Imports RSAGA, sf, methods, knitr

Suggests rgdal

Description

Generation of bioclimatic rasters that are complementary to the typical 19 bioclim variables.

License GPL (>= 2)

URL http://envirem.github.io

BugReports https://github.com/ptitle/envirem/issues

NeedsCompilation yes

Encoding UTF-8

RoxygenNote 7.1.1

ByteCompile true

Repository CRAN

Date/Publication 2021-05-14 07:40:19 UTC

R topics documented:

`aridityIndexThornthwaite` *aridityIndexThornthwaite*

Description

Generates thornthwaite aridity index raster.

Usage

```
aridityIndexThornthwaite(precipStack, PETstack, precipScale = 1)
```

Arguments

`precipStack`	rasterStack of monthly precipitation.
`PETstack`	rasterStack of monthly potential evapotranspiration. Layer names are assumed to end in the month number.
`precipScale`	integer; scaling factor for the precipitation data, see envirem for additional details.

Details

`Thornthwaite aridity index = 100d / n` where d = sum of monthly differences between precipitation and PET for months where precip < PET where n = sum of monthly PET for those months

Value

RasterLayer, unitless

Author(s)

Pascal Title

References

Thornthwaite, C.W. (1948). An approach toward a rational classification of climate. *Geographical Review*, **38**, 55-94.

See Also

Requires rasters created with monthlyPET.

Examples

```
# Find example rasters
rasterFiles <- list.files(system.file('extdata', package='envirem'), full.names=TRUE)
env <- stack(rasterFiles)

# identify the appropriate layers
meantemp <- grep('mean', names(env), value=TRUE)
solar <- grep('solrad', names(env), value=TRUE)
maxtemp <- grep('tmax', names(env), value=TRUE)
mintemp <- grep('tmin', names(env), value=TRUE)

# read them in as rasterStacks
meantemp <- stack(env[[meantemp]])
solar <- stack(env[[solar]])
maxtemp <- stack(env[[maxtemp]])
mintemp <- stack(env[[mintemp]])
tempRange <- abs(maxtemp - mintemp)

# get monthly PET
pet <- monthlyPET(meantemp, solar, tempRange)

precip <- grep('prec', names(env), value=TRUE)
precip <- stack(env[[precip]])

# set up naming scheme - only precip is different from default
assignNames(precip = 'prec_##')

aridityIndexThornthwaite(precip, pet)

# set back to defaults
assignNames(reset = TRUE)
```

assignNames *Defining variable names*

Description

The naming scheme for the different input variables are defined via a custom environment, which only needs to be done once.

Usage

```
assignNames(tmin, tmax, tmean, precip, solrad, reset)
```

Arguments

tmin	naming scheme for minimum temperature
tmax	naming scheme for maximum temperature
tmean	naming scheme for mean temperature
precip	naming scheme for precipitation
solrad	naming scheme for solar radiation
reset	if TRUE, then names are set to default values

Details

The .var environment contains the naming scheme for the input variables, and this will be queried by the various functions in this R package. The user should use this function to define the names of the variables, up until the variable number, and after the variable number (prefix and suffix relative to the number). This is done by providing a template of the naming, and placing ## where the numbers would be (1:12). For example, if your minimum temperature rasters are named as worldclim_v2_LGM_ccsm4_minTemp_1_land.tif, then you should define the following: "worldclim_v2_LGM_ccsm4_min
for tmin. File extensions should not be included at all (not as a suffix).

This only needs to be done once during your R session. For any variable name, if these tags are removed, and the file extension is removed, only the variable number should remain (the month number).

When using the assignNames() function, you can specify as many or as few as needed.

Variable numbers can have zero-padding. This is handled automatically. Therefore, bio_1 or bio_01 are both fine, and nothing needs to be specified.

The default values are tmin_, tmax_, tmean_, precip_, and et_solrad_, with no suffix. You can use the function varnames() to see the current assigned values.

Examples

```
varnames()
assignNames(precip = 'precip_##_5arcmin')
assignNames(solrad = 'solar_##', tmin = 'minTemp##')
varnames()

# set back to default
assignNames(reset = TRUE)
```

`climaticMoistureIndex` *Climatic Moisture Index*

Description

Generate climatic moisture index.

Usage

```
climaticMoistureIndex(annualPrecip, PET, precipScale = 1)
```

Arguments

`annualPrecip`	rasterLayer of annual precipitation (bioclim 12)
`PET`	rasterLayer of annual potential evapotranspiration
`precipScale`	integer; scaling factor for the precipitation data, see envirem for additional details.

Details

`P/PET - 1` when `P < PET`
`1 - PET/P` when `P >= PET`

Value

rasterLayer ranging from -1 to +1.

Author(s)

Pascal Title

References

Willmott, C. & Feddema, J. (1992). A More Rational Climatic Moisture Index. *The Professional Geographer*, **44**, 84-88.

Vörösmarty, C.J., Douglas, E.M., Green, P.A. & Revenga, C. (2005). Geospatial Indicators of Emerging Water Stress: An Application to Africa. *AMBIO: A Journal of the Human Environment*, **34**, 230-236.

Examples

```
# Find example rasters
rasterFiles <- list.files(system.file('extdata', package='envirem'), full.names=TRUE)
env <- stack(rasterFiles)

# identify the appropriate layers
meantemp <- grep('mean', names(env), value=TRUE)
solar <- grep('solrad', names(env), value=TRUE)
maxtemp <- grep('tmax', names(env), value=TRUE)
mintemp <- grep('tmin', names(env), value=TRUE)

# read them in as rasterStacks
meantemp <- stack(env[[meantemp]])
solar <- stack(env[[solar]])
maxtemp <- stack(env[[maxtemp]])
mintemp <- stack(env[[mintemp]])
tempRange <- abs(maxtemp - mintemp)

# get monthly PET
pet <- monthlyPET(meantemp, solar, tempRange)

# get mean annual PET
annualPET <- sum(pet)

climaticMoistureIndex(env[['bio_12']], annualPET)
```

`continentality` *Continentality*

Description

Generate Continentality index.

Usage

```
continentality(tmax, tmin, tempScale = 1)
```

Arguments

`tmax`	rasterLayer of average temperature of the warmest month
`tmin`	rasterLayer of average temperature of the coldest month
`tempScale`	integer; scaling factor for the temperature data, see envirem for additional details.

Details

`continentality index = tmax - tmin`

Value

rasterLayer in units of degrees C.

Author(s)

Pascal Title

References

Rivas-Martínez, S. & Rivas-Sáenz, S. "Synoptical Worldwide Bioclimatic Classification System". Available online at http://www.globalbioclimatics.org/ [accessed 15 February 2016]

Sayre, R., Comer, P., Warner, H. & Cress, J. (2009) *A new map of standardized terrestrial ecosystems of the conterminous United States: US Geological Survey Professional Paper 1768.* Reston, VA.

See Also

thermicityIndex

Examples

```
# Find example rasters
rasterFiles <- list.files(system.file('extdata', package='envirem'), full.names=TRUE)
env <- stack(rasterFiles)

# identify appropriate layers
tmean <- grep('tmean', names(env))
tmin <- grep('tmin', names(env))
tmax <- grep('tmax', names(env))

tmean <- env[[tmean]]
tmin <- env[[tmin]]
tmax <- env[[tmax]]

# calculate temperature extremes
temp <- otherTempExtremes(tmean, tmin, tmax)

meantempWarmest <- temp[['meanTempWarmest']]
meantempColdest <- temp[['meanTempColdest']]

continentality(meantempWarmest, meantempColdest, tempScale = 10)
```

dataTypeCheck *Data Type Check*

Description

Determines the best data type to implement when writing the raster to file

Usage

```
dataTypeCheck(r)
```

Arguments

r	raster object

Details

Function to determine the most memory efficient data type given whether or not the raster contains integer or non-integer values, and the range of those values, based on the definitions described in dataType.

Author(s)

Pascal Title

Examples

```
rasterFiles <- list.files(system.file('extdata', package='envirem'), full.names=TRUE)
r <- raster(rasterFiles[1])
dataTypeCheck(r)
```

embergerQ *Emberger's pluviometric quotient*

Description

Calculate Emberger's pluviometric quotient.

Usage

```
embergerQ(P, M, m, tempScale = 1, precipScale = 1)
```

Arguments

P	rasterLayer, total annual precipitation
M	rasterLayer, mean max temperature of the warmest month
m	rasterLayer, mean min temperature of the coldest month
tempScale	integer; scaling factor for the temperature data, see envirem for additional details.
precipScale	integer; scaling factor for the precipitation data, see envirem for additional details.

Details

`Q = 2000 P / [(M + m + 546.4) * (M - m)]`

Value

rasterLayer in mm / degrees C

Author(s)

Pascal Title

References

Daget, P. (1977) Le bioclimat méditerranéen: analyse des formes climatiques par le système d'Emberger. *Vegetatio*, **34**, 87–103.

Examples

```
# Find example rasters
rasterFiles <- list.files(system.file('extdata', package='envirem'), full.names=TRUE)
env <- stack(rasterFiles)

embergerQ(env[['bio_12']], env[['bio_5']], env[['bio_6']], tempScale = 10)
```

---

envirem *envirem*

Description

Generation of bioclimatic rasters that are complementary to the typical 19 bioclim variables.

Details

Package: envirem
Type: Package
Version: 2.2
Date: 2020-06-03
License: GPL-2 | GPL-3

NOTE: Temperature rasters are now assumed by default to be in degrees C and precipitation in mm. rasters in degrees C * 10. Worldclim v2 uses degrees C. CHELSA has several options, depending on whether rasters are downloaded as floating point or integer. Therefore, there is an argument `tempScale` to specify the units of temperature, and `precipScale` to define precipitation units:

For example:
If using worldclim v1 data where temperature is in degrees C * 10, specify `tempScale = 10`.
If using worldclim v2 where temperature is in degrees C, specify `tempScale = 1`.
For CHELSA, read the documentation and carefully examine the rasters.

If a function does not have the `tempScale` argument, then the function is not sensitive to the units of the input temperature rasters.

Of course, it is also perfectly acceptable to leave `tempScale = 1` and `precipScale = 1` and modify the input rasters yourself.

The main function for generating ENVIREM rasters is `generateRasters`. A complete tutorial of this R package can be found at `http://envirem.github.io`.

Author(s)

Pascal O. Title, Jordan B. Bemmels

References

`http://envirem.github.io`

Title, P.O., Bemmels, J.B. 2018. ENVIREM: An expanded set of bioclimatic and topographic variables increases flexibility and improves performance of ecological niche modeling. Ecography 41:291–307.

`ETsolradRasters` *Extraterrestrial Solar Radiation*

Description

Generate monthly extraterrestrial solar radiation rasters.

Usage

```
ETsolradRasters(rasterTemplate, year, outputDir = NULL, ...)
```

Arguments

rasterTemplate	any rasterLayer that can be used to extract extent, resolution, projection, etc.
year	The year solar radiation should be calculated for. See details.
outputDir	destination directory for rasters, can be NULL
...	additional arguments passed to writeRaster

Details

Given the latitude values of the cells found in the raster template and the year, monthly extraterrestrial solar radiation can be calculated, using the palinsol R package. year = 0 corresponds to 1950. Although the year can take on any value, it should match the time period of the other rasters that will be used for generating ENVIREM variables. Suggestions would be year = 40 for the present, year = -6000 for the mid Holocene, and year = -21500 for the LGM.

If you are having problems with this function and the rasterTemplate is not in long/lat, try with an unprojected long/lat raster.

Value

If outputDir = NULL, a RasterStack is returned. Otherwise, rasters are written to disk in the designated directory, and nothing is returned. Naming of the layers uses the tag specified via assignNames.

Author(s)

Pascal Title

References

J. Laskar et al., A long-term numerical solution for the insolation quantities of the Earth, *Astron. Astroph.*, **428**, 261-285 2004.

Examples

```
# Find example rasters
rasterFiles <- list.files(system.file('extdata', package='envirem'), full.names=TRUE)
env <- stack(rasterFiles)

# set aside a template raster
template <- env[[1]]

# generate solar radiation for the present
solrad <- ETsolradRasters(template, year = 40, outputDir = NULL)
```

generateRasters *Execute Layer Creation*

Description

Main function to generate specified ENVIREM layers. If requested, this function will split input rasters into tiles, generate desired variables, and reassemble the results. For the distinction between this function and layerCreation, see Details.

Usage

```
generateRasters(
  var,
  maindir,
  prefix = "",
  outputDir = "./",
  rasterExt = ".tif",
  nTiles = 1,
  tempScale = 1,
  precipScale = 1,
  overwriteResults = TRUE,
  outputFormat = "GTiff",
  tempDir = "~/temp",
  gdalinfoPath = NULL,
  gdal_translatePath = NULL,
  useCompression = TRUE
)
```

Arguments

var	a vector of variable names to generate, see Details.
maindir	path to directory of input rasters
prefix	prefix to append to output filename
outputDir	output directory.
rasterExt	the file extension of the input rasters
nTiles	the number of tiles to split the rasters when tiling is requested, must be a perfect square
tempScale	integer; scaling factor for the temperature data, see envirem for additional details.
precipScale	integer; scaling factor for the precipitation data, see envirem for additional details.
overwriteResults	logical, should existing rasters be overwritten
outputFormat	output format for rasters, see writeRaster for options

tempDir	temporary directory for raster tiles that will be created and then removed
gdalinfoPath	path to gdalinfo binary, leave as NULL if it is in the default search path.
gdal_translatePath	path to gdal_translate binary, leave as NULL if it is in the default search path.
useCompression	logical; should compression options be used to achieve smaller file sizes. Only pertains to format GTiff.

Details

The function layerCreation will generate envirem rasters from input R objects (rasterStacks) and will return the result as an R object. In contrast, the function generateRasters reads in input rasters from a specified directory, splits input rasters into tiles if necessary, internally calls layerCreation and writes the result to file.

Possible variables to generate include:

annualPET
aridityIndexThornthwaite
climaticMoistureIndex
continentality
embergerQ
growingDegDays0
growingDegDays5
maxTempColdest
minTempWarmest
meanTempColdest
meanTempWarmest
monthCountByTemp10
PETColdestQuarter
PETDriestQuarter
PETseasonality
PETWarmestQuarter
PETWettestQuarter
thermicityIndex

If var = 'all', then all of the variables will be generated.

Rasters in mainDir should be named appropriately (see verifyFileStructure) and with identical resolution, origin and extent.

Output rasters are written with the most appropriate dataType, as inferred with dataTypeCheck. This will reduce the file size of these rasters.

If the goal is to use these rasters with the standalone Maxent program, we recommend outputFormat = 'EHdr'.

IMPORTANT: Temporary files can quickly fill up your hard drive when working with large rasters. There are two temporary directories to consider for this function: The tempDir directory defined as an argument in this function is used for storing intermediate files when splitting rasters into tiles (and is ignored if nTiles = 1). The raster package will use another directory for storing temporary rasters. This can be can be viewed with rasterOptions(), and can be set with

`rasterOptions(tmpdir = 'path-to-dir')`. Be sure that this is pointing to a directory with plenty of available space. Both temporary directories are automatically cleared.

Value

The requested set of rasterLayers will be written to `outputDir`.

Author(s)

Pascal Title

See Also

Naming of rasters in inputDir will be checked with `verifyFileStructure`.

`growingDegDays` *Growing degree days*

Description

Growing degree days above some base temperature.

Usage

```
growingDegDays(meantempstack, baseTemp, tempScale = 1)
```

Arguments

`meantempstack`	rasterStack of mean monthly temperature in deg C
`baseTemp`	base temperature in degrees C.
`tempScale`	integer; scaling factor for the temperature data, see envirem for additional details.

Details

growing degree days = sum of all monthly temps greater than baseTemp, multiplied by total number of days

Value

rasterLayer in degrees C * days.

Author(s)

Pascal Title

References

Prentice, I.C., Cramer, W., Harrison, S.P., Leemans, R., Monserud, R.A. & Solomon, A.M. (1992). A Global Biome Model Based on Plant Physiology and Dominance, Soil Properties and Climate. *Journal of Biogeography*, **19**, 117–134.

Examples

```
# Find example rasters
rasterFiles <- list.files(system.file('extdata', package='envirem'), full.names=TRUE)
env <- stack(rasterFiles)

meantemp <- env[[grep('tmean', names(env), value=TRUE)]]
growingDegDays(meantemp, 10, tempScale = 10)
```

---

layerCreation *Creates all layers*

Description

Generates all rasterLayers for one particular input dataset. For the distinction between this function and `generateRasters`, see `Details`.

Usage

```
layerCreation(
  masterstack,
  solradstack = NULL,
  var,
  tempScale = 1,
  precipScale = 1
)
```

Arguments

`masterstack`	rasterStack containing all monthly precipitation, min temperature, max temperature, and optionally mean temperature rasters.
`solradstack`	rasterStack of monthly solar radiation, can be `NULL` if not needed.
`var`	vector of names of variables to generate, see Details.
`tempScale`	integer; scaling factor for the temperature data, see envirem for additional details.
`precipScale`	integer; scaling factor for the precipitation data, see envirem for additional details.

Details

The function verifyFileStructure should be used to verify that the appropriate rasters are present in masterstack.

This function is called internally by generateRasters.

The function layerCreation will generate envirem rasters from input R objects (rasterStacks) and will return the result as an R object. In contrast, the function generateRasters reads in input rasters from a specified directory, splits input rasters into tiles if necessary, internally calls layerCreation and writes the result to file.

Possible variables to generate include:

annualPET
aridityIndexThornthwaite
climaticMoistureIndex
continentality
embergerQ
growingDegDays0
growingDegDays5
maxTempColdest
minTempWarmest
meanTempColdest
meanTempWarmest
monthCountByTemp10
PETColdestQuarter
PETDriestQuarter
PETseasonality
PETWarmestQuarter
PETWettestQuarter
thermicityIndex

If var = 'all', then all of the variables will be generated.

Value

rasterStack

Author(s)

Pascal Title

See Also

This function is called internally by generateRasters.

Examples

```
# Find example rasters
rasterFiles <- list.files(system.file('extdata', package='envirem'), full.names=TRUE)
```

```
# create stack of temperature and precipitation rasters
# and stack of solar radiation rasters
solradFiles <- grep('solrad', rasterFiles, value=TRUE)
worldclim <- stack(setdiff(rasterFiles, solradFiles))
solar <- stack(solradFiles)

# set up naming scheme - only precip is different from default
assignNames(precip = 'prec_##')

# generate all possible envirem variables
layerCreation(worldclim, solar, var='all', tempScale = 10)

# set back to defaults
assignNames(reset = TRUE)
```

`monthCountByTemp` *Month count by temperature*

Description

Number of months with mean temperature greater than some base temp.

Usage

```
monthCountByTemp(tempStack, minTemp = 10, tempScale = 1)
```

Arguments

`tempStack`	rasterStack of monthly mean temperature in degrees C
`minTemp`	reference temperature in degrees C
`tempScale`	integer; scaling factor for the temperature data, see envirem for additional details.

Value

rasterLayer with values representing counts of months.

Author(s)

Pascal Title

References

Metzger, M.J., Bunce, R.G.H., Jongman, R.H.G., Sayre, R., Trabucco, A. & Zomer, R. (2013). A high-resolution bioclimate map of the world: a unifying framework for global biodiversity research and monitoring. *Global Ecology and Biogeography*, **22**, 630-638.

Examples

```
# Find example rasters
rasterFiles <- list.files(system.file('extdata', package='envirem'), full.names=TRUE)
env <- stack(rasterFiles)

# identify the appropriate layers
meantemp <- grep('mean', names(env), value=TRUE)
meantemp <- env[[meantemp]]
monthCountByTemp(meantemp, 10, tempScale = 10)
```

`monthlyPET` *monthly PET*

Description

Monthly potential evapotranspiration

Usage

```
monthlyPET(Tmean, RA, TD, tempScale = 1)
```

Arguments

`Tmean`	rasterStack of monthly mean temperature
`RA`	rasterStack of monthly extraterrestrial solar radiation
`TD`	rasterStack of monthly temperature range
`tempScale`	integer; scaling factor for the temperature data, see envirem for additional details.

Details

`PET = 0.0023 * RA * (Tmean + 17.8) * TD ^ 0.5`

Value

rasterStack of monthly PET in mm / month

Author(s)

Pascal Title

References

Hargreaves, G. L., Hargreaves, G. H., & Riley, J. P. (1985). Irrigation water requirements for Senegal River basin. *Journal of Irrigation and Drainage Engineering*, **111**, 265-275.

Zomer, R.J., Trabucco, A., Bossio, D.A. & Verchot, L.V. (2008). Climate change mitigation: A spatial analysis of global land suitability for clean development mechanism afforestation and reforestation. *Agriculture, Ecosystems and Environment*, **126**, 67-80.

Zomer, R.J., Trabucco, A., Van Straaten, O. & Bossio, D.A. (2006) *Carbon, Land and Water: A Global Analysis of the Hydrologic Dimensions of Climate Change Mitigation through Afforestation/Reforestation. International Water Management Institute Research Report 101*. Colombo, Sri Lanka.

Examples

```
# Find example rasters
rasterFiles <- list.files(system.file('extdata', package='envirem'), full.names=TRUE)
env <- stack(rasterFiles)

# identify the appropriate layers
meantemp <- grep('mean', names(env), value=TRUE)
solar <- grep('solrad', names(env), value=TRUE)
maxtemp <- grep('tmax', names(env), value=TRUE)
mintemp <- grep('tmin', names(env), value=TRUE)

# read them in as rasterStacks
meantemp <- stack(env[[meantemp]])
solar <- stack(env[[solar]])
maxtemp <- stack(env[[maxtemp]])
mintemp <- stack(env[[mintemp]])
tempRange <- abs(maxtemp - mintemp)

monthlyPET(meantemp, solar, tempRange, tempScale = 10)
```

---

otherTempExtremes *Temperature Extremes*

Description

Generates max temp of the coldest month, min temp of the warmest month, mean temp of the coldest month, mean temp of the warmest month.

Usage

```
otherTempExtremes(meantempStack, mintempStack, maxtempStack)
```

Arguments

meantempStack	rasterStack of monthly mean temperature
mintempStack	rasterStack of monthly min temperature
maxtempStack	rasterStack of monthly max temperature

Value

rasterStack of maxTempColdest, minTempWarmest, meanTempColdest, meanTempWarmest, in same units as input rasters.

Author(s)

Pascal Title

Examples

```
# Find example rasters
rasterFiles <- list.files(system.file('extdata', package='envirem'), full.names=TRUE)
env <- stack(rasterFiles)

# identify appropriate layers
tmean <- grep('tmean', names(env))
tmin <- grep('tmin', names(env))
tmax <- grep('tmax', names(env))

tmean <- env[[tmean]]
tmin <- env[[tmin]]
tmax <- env[[tmax]]

# calculate temperature extremes
otherTempExtremes(tmean, tmin, tmax)
```

---

pacificCentric *Center raster on the Pacific*

Description

Takes a raster that is centered on 0 longitude (default) and recenters it on the Pacific

Usage

```
pacificCentric(r, crop = TRUE)
```

Arguments

r	rasterLayer or rasterStack in unprojected geographic coordinates
crop	logical, should raster then be cropped to longitude [100, 300]

Details

Cropping to [100, 300] is equivalent to [100, -60]

Value

rasterLayer or rasterStack

Author(s)

Pascal Title

Examples

```
# Find example rasters
rasterFiles <- list.files(system.file('extdata', package='envirem'), full.names=TRUE)
tmin1 <- raster(grep('tmin_1\\.', rasterFiles, value=TRUE))

pacificCentric(tmin1, crop = TRUE)
```

---

petExtremes *PET Extremes*

Description

Calculates mean PET of the coldest, warmest, wettest and driest quarters.

Usage

```
petExtremes(PETstack, precipStack, meantempStack)
```

Arguments

PETstack	rasterStack of monthly PET, layer names assumed to end in month numbers
precipStack	rasterStack of monthly precipitation
meantempStack	rasterStack of monthly mean temperature

Details

Generates mean monthly PET for the warmest, coldest, wettest and driest 3 consecutive months.

Value

rasterStack of PETColdestQuarter, PETWarmestQuarter, PETWettestQuarter, PETDriestQuarter in mm / month.

Author(s)

Pascal Title

References

Metzger, M.J., Bunce, R.G.H., Jongman, R.H.G., Sayre, R., Trabucco, A. & Zomer, R. (2013). A high-resolution bioclimate map of the world: a unifying framework for global biodiversity research and monitoring. *Global Ecology and Biogeography*, **22**, 630-638.

See Also

monthlyPET

Examples

```
# Find example rasters
rasterFiles <- list.files(system.file('extdata', package='envirem'), full.names=TRUE)
env <- stack(rasterFiles)

# identify the appropriate layers
meantemp <- grep('mean', names(env), value=TRUE)
solar <- grep('solrad', names(env), value=TRUE)
maxtemp <- grep('tmax', names(env), value=TRUE)
mintemp <- grep('tmin', names(env), value=TRUE)
precip <- grep('prec', names(env), value=TRUE)

# read them in as rasterStacks
meantemp <- stack(env[[meantemp]])
solar <- stack(env[[solar]])
maxtemp <- stack(env[[maxtemp]])
mintemp <- stack(env[[mintemp]])
tempRange <- abs(maxtemp - mintemp)
precip <- stack(env[[precip]])

# set up naming scheme - only precip is different from default
assignNames(precip = 'prec_##')

# get monthly PET
pet <- monthlyPET(meantemp, solar, tempRange)

petExtremes(pet, precip, meantemp)

# set back to defaults
assignNames(reset = TRUE)
```

PETseasonality *PET seasonality*

Description

Seasonality of potential evapotranspiration

Usage

```
PETseasonality(PETstack)
```

Arguments

PETstack	rasterStack of monthly PET rasters

Details

PET seasonality = 100 * standard deviation of monthly PET.

Value

rasterLayer in mm / month

Author(s)

Pascal Title

References

Metzger, M.J., Bunce, R.G.H., Jongman, R.H.G., Sayre, R., Trabucco, A. & Zomer, R. (2013). A high-resolution bioclimate map of the world: a unifying framework for global biodiversity research and monitoring. *Global Ecology and Biogeography*, **22**, 630-638.

See Also

monthlyPET

Examples

```
# Find example rasters
rasterFiles <- list.files(system.file('extdata', package='envirem'), full.names=TRUE)
env <- stack(rasterFiles)

# identify the appropriate layers
meantemp <- grep('mean', names(env), value=TRUE)
solar <- grep('solrad', names(env), value=TRUE)
maxtemp <- grep('tmax', names(env), value=TRUE)
mintemp <- grep('tmin', names(env), value=TRUE)

# read them in as rasterStacks
meantemp <- stack(env[[meantemp]])
solar <- stack(env[[solar]])
maxtemp <- stack(env[[maxtemp]])
mintemp <- stack(env[[mintemp]])
tempRange <- abs(maxtemp - mintemp)

# get monthly PET
pet <- monthlyPET(meantemp, solar, tempRange)
```

```
PETseasonality(pet)
```

`split_raster` *Split raster into tiles*

Description

Splits a rasterLayer into tiles

Usage

```
split_raster(
  file,
  s = 2,
  outputDir,
  gdalinfoPath = NULL,
  gdal_translatePath = NULL
)
```

Arguments

`file`	full path to a raster
`s`	division applied to each side of the raster (s=2 -> 4 tiles)
`outputDir`	path and directory name for output
`gdalinfoPath`	path to gdalinfo binary. Set to `NULL` if default search path is sufficient.
`gdal_translatePath`	path to gdal_translate binary. Set to `NULL` if default search path is sufficient.

Details

GDAL must be installed for this function to work. To determine if the default search paths are sufficient, you can type in R `Sys.which('gdalinfo')` and `Sys.which('gdal_translate')`. If a path is returned, then you can leave those arguments as `NULL`.

Value

Rasters are written to the output directory.

Author(s)

Pascal Title

References

GDAL. 2015. GDAL - Geospatial Data Abstraction Library: Version 1.11.3, Open Source Geospatial Foundation, https://gdal.org

Examples

```
## Not run:
# Find example rasters
rasterFiles <- list.files(system.file('extdata', package='envirem'), full.names=TRUE)
tmin1file <- grep('tmin_1\\.', rasterFiles, value=TRUE)

We will split this raster into 4 tiles, that will be written to disk.
split_raster(tmin1file, s = 2, outputDir = '~/temp/', gdalinfoPath = NULL,
gdal_translatePath = NULL)

## End(Not run)
```

`thermicityIndex` *Compensated Thermicity index*

Description

Compensated Thermicity index

Usage

```
thermicityIndex(
  annualTemp,
  minTemp,
  maxTemp,
  continentality,
  returnCompensated = TRUE,
  tempScale = 1
)
```

Arguments

`annualTemp`	rasterLayer, mean annual temperature
`minTemp`	rasterLayer, min temp of the coldest month
`maxTemp`	rasterLayer, max temp of the coldest month
`continentality`	rasterLayer, continentality index
`returnCompensated`	logical: if `FALSE`, regular thermicity index is returned.
`tempScale`	integer; scaling factor for the temperature data, see envirem for additional details.

Details

thermicity index = tempRange + minTemp + maxTemp

The compensated thermicity index incorporates corrections designed to make this metric more appropriately comparable across the globe.

Value

rasterLayer in degrees C

Author(s)

Pascal Title

References

Rivas-Martínez, S. & Rivas-Sáenz, S. "Synoptical Worldwide Bioclimatic Classification System". Available online at http://www.globalbioclimatics.org/ [accessed 15 February 2016]

Sayre, R., Comer, P., Warner, H. & Cress, J. (2009) *A new map of standardized terrestrial ecosystems of the conterminous United States: US Geological Survey Professional Paper 1768.* Reston, VA.

See Also

continentality

Examples

```
# Find example rasters
rasterFiles <- list.files(system.file('extdata', package='envirem'), full.names=TRUE)
env <- stack(rasterFiles)

# identify appropriate layers
tmean <- grep('tmean', names(env))
tmin <- grep('tmin', names(env))
tmax <- grep('tmax', names(env))

tmean <- env[[tmean]]
tmin <- env[[tmin]]
tmax <- env[[tmax]]

# calculate temperature extremes
temp <- otherTempExtremes(tmean, tmin, tmax)

ci <- continentality(temp[['meanTempWarmest']], temp[['meanTempColdest']], tempScale = 10)

# compensated thermicity index
thermicityIndex(env[['bio_1']], env[['bio_6']], temp[['maxTempColdest']], ci, tempScale = 10)
```

`topoWetnessIndex` *Topographic Wetness Index*

Description

SAGA-GIS topographic wetness index

Usage

```
topoWetnessIndex(dem, sagaEnv)
```

Arguments

`dem`	elevation rasterLayer, with defined proj4string.
`sagaEnv`	list object returned from `RSAGA::rsaga.env`, which supplies appropriate SAGA paths, and parallelization information.

Details

If this function returns an error, there may be a conflict with the version of SAGA-GIS installed on your machine, and the version of SAGA-GIS that the RSAGA package is designed to work with.

From a DEM, this function will write an appropriate raster to disk, run an RSAGA function to calculate the topographic wetness index, and will then read it back in and return it.

This function requires that SAGA-GIS be installed on your system. SAGA-GIS can be found at http://www.saga-gis.org.

See the documentation for `RSAGA::rsaga.env` for specifying appropriate paths and parallelization details.

Value

rasterLayer, unitless

Author(s)

Pascal Title

References

Boehner, J., Koethe, R. Conrad, O., Gross, J., Ringeler, A. & Selige, T. (2002) Soil regionalization by means of terrain analysis and process parameterization. *Soil Classification 2001 European Soil Bureau, Research Report No. 7* (eds Micheli, E., Nachtergaele, F. & Montanarella, L.), pp. 213-222. Luxembourg.

Conrad, O., Bechtel, B., Bock, M., Dietrich, H., Fischer, E., Gerlitz, L., Wehberg, J., Wichmann, V. & Boehner, J. (2015) System for automated geoscientific analyses (SAGA) v. 2.1.4. *Geoscientific Model Development*, **8**, 1991-2007.

Examples

```
## Not run:
# Find example rasters
rasterFiles <- list.files(system.file('extdata', package='envirem'), full.names=TRUE)
elev <- raster(grep('elev', rasterFiles, value=TRUE))

# setting up appropriate RSAGA environment
sagaEnv <- RSAGA::rsaga.env(modules = '/usr/lib/x86_64-linux-gnu/saga/', cores = 2,
parallel = TRUE, version = "2.2.0")
topoWetnessIndex(elev, sagaEnv)

## End(Not run)
```

---

varnames *List Naming Scheme*

Description

Lists the naming scheme, either as default, or as modified by the user.

Usage

```
varnames()
```

Details

See assignNames.

Examples

```
varnames()
```

---

verifyFileStructure *Verify File Structure*

Description

Ensures that the necessary files are present for other functions to work properly.

Usage

```
verifyFileStructure(
  path = "./",
  returnFileNames = TRUE,
  includeSolRad = TRUE,
  rasterExt = ".tif"
)
```

Arguments

path	path to directory of rasters
returnFileNames	logical, should file paths and names be returned
includeSolRad	logical, should solar radiation files be required?
rasterExt	file extension of rasters

Details

This function searches for the following in the directory specified by path:

12 precipitation rasters

12 min temperature rasters

12 max temperature rasters

12 mean temperature rasters [optional]

12 solar radiation rasters

The naming scheme will be checked against the one defined via the custom naming environment. See link{?assignNames} for additional details.

An equivalent function for checking the names of rasters in R can be found at verifyRasterNames.

If mean temperature rasters are not detected, the raster creation functions will create mean temperature by taking the mean of the min and max.

Value

Prints messages to the console if problems are found. If returnFileNames == TRUE, then a vector of filenames is returned.

Author(s)

Pascal Title

Examples

```
rasterPath <- system.file('extdata', package='envirem')
list.files(rasterPath)

# Assign naming scheme
assignNames(precip = 'prec_##')
varnames()

# As there are no problems with these files, the list of files
# will be returned.
verifyFileStructure(rasterPath)

assignNames(reset = TRUE)
```

`verifyRasterNames` *Verify Raster Names*

Description

Given a RasterStack, this function will verify the naming scheme and check that all required rasters are present.

Usage

```
verifyRasterNames(
  masterstack = NULL,
  solradstack = NULL,
  returnRasters = FALSE
)
```

Arguments

`masterstack`	rasterStack containing all precipitation, min temperature, max temperature, and (optionally) mean temperature variables.
`solradstack`	rasterStack of monthly solar radiation
`returnRasters`	if `FALSE`, the function checks names and reports back. If `TRUE`, a RasterStack is returned with standardized names.

Details

This function checks that the following are present:

12 precipitation rasters

12 min temperature rasters

12 max temperature rasters

12 mean temperature rasters [optional]

12 solar radiation rasters

The naming scheme will be checked against the one defined via the custom naming environment. See `link{?assignNames}` for additional details.

The function can test the temp/precip rasterstack and/or the solar radiation rasterstack separately, or simultaneously.

Value

Prints messages to the console if `returnRasters = FALSE`, If `returnRasters = TRUE`, then a RasterStack is returned. This RasterStack will not include rasters that were deemed unnecessary.

Author(s)

Pascal Title

Examples

```
# Find example rasters
rasterFiles <- list.files(system.file('extdata', package='envirem'), full.names=TRUE)

# create stack of temperature and precipitation rasters
# and stack of solar radiation rasters
solradFiles <- grep('solrad', rasterFiles, value=TRUE)
worldclim <- stack(setdiff(rasterFiles, solradFiles))
solar <- stack(solradFiles)

# modify naming
names(worldclim) <- gsub('tmin_', 'minTemp', names(worldclim))
names(worldclim) <- paste0(names(worldclim), '_v1.0')
names(solar) <- gsub('et_solrad_', 'solar_', names(solar))

# but don't specify this change
varnames()

# Run check
verifyRasterNames(masterstack = worldclim, solradstack = solar, returnRasters = FALSE)

# But if we specify our naming scheme
assignNames(tmin = 'minTemp##_v1.0', tmax = 'tmax_##_v1.0', tmean = 'tmean_##_v1.0',
solrad = 'solar_##', precip = 'prec_##_v1.0')
varnames()

verifyRasterNames(masterstack = worldclim, solradstack = solar, returnRasters = FALSE)

# set back to defaults
assignNames(reset = TRUE)
```

Index