Package: jNSMR (via r-universe)

November 19, 2024

Title Interface to the 'Java Newhall Simulation Model' (jNSM) ``A Traditional Soil Climate Simulation Model''

Version 0.3.1

Author Soil and Plant Science Division Staff

Maintainer Andrew G. Brown <andrew.g.brown@usda.gov>

Description Provides methods to create input, read output, and run the routines from the legacy Java Newhall Simulation Model (jNSM) for soil climate. Currently this package uses a modified version of the jNSM v1.6.1 which is available for download here: <https://www.nrcs.usda.gov/wps/portal/nrcs/detail/?cid=nrcs142p2_053559> and the source code found here <https://github.com/drww/newhall/>. The system requirements of the extraction and installation tools (Windows .EXE archive) at the official download link may not be met on your system but the core Java class files are stored in a platform-independent format (a Java JAR file; e.g. newhall-1.6.1.jar) which is a core dependency in this package. Several more recent modifications to the Newhall JAR file allow for higher throughput and more efficient batching of many simulations allowing for larger-than-memory raster-based inputs and outputs.

LazyData true

Roxygen list(markdown = TRUE)

RoxygenNote 7.3.1

Encoding UTF-8

License BSD_3_clause + file LICENSE

SystemRequirements Java

Depends R (>= 4.1.0)

Imports utils, parallel, data.table, rJava, terra

Suggests knitr, testthat (>= 3.0.0), rmarkdown, prism, daymetr, geodata

Language en-US

URL https://ncss-tech.github.io/jNSMR,
https://github.com/ncss-tech/jNSMR

VignetteBuilder knitr

Config/pak/sysreqs libgdal-dev gdal-bin libgeos-dev make default-jdk libproj-dev libsqlite3-dev

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

RemoteUrl https://github.com/ncss-tech/jNSMR

RemoteRef HEAD

RemoteSha 9532cbd324e7bd84dde4c593728d69c757a0a45e

Contents

`BASICSimulationModel` *Create an instance of* BASICSimulationModel

Description

Create an instance of *BASICSimulationModel*

Usage

```
BASICSimulationModel()
```

Value

an instance of *BASICSimulationModel* class

`CSVFileParser` *Create an instance of* CSVFileParser

Description

Create an instance of *CSVFileParser*

Usage

```
CSVFileParser(pathname)
```

Arguments

`pathname`	*character* containing `pathname`

Value

an instance of *CSVFileParser* class

`CSVResultsExporter` *Create an instance of* CSVResultsExporter

Description

Create an instance of *CSVResultsExporter*

Usage

```
CSVResultsExporter(results, pathname)
```

Arguments

`results`	*NewhallResults* `jobjRef`
`pathname`	a character containing pathname

Value

an instance of *CSVResultsExporter* class

`NewhallDataset` *Create an instance of* NewhallDataset

Description

Create an instance of *NewhallDataset*

Usage

```
NewhallDataset(
  stationName,
  country,
  latDD,
  lonDD,
  elev,
  allPrecipsDbl,
  allAirTempsDbl,
  pdbegin,
  pdend,
  smcsawc,
  checkargs = TRUE
)
```

Arguments

`stationName`	*character*; station name
`country`	*character*; country
`latDD`	*double*; latitude decimal degrees
`lonDD`	*double*; longitude decimal degrees
`elev`	*double*; station elevation
`allPrecipsDbl`	*double*; length 12 precipitation, monthly (millimeters of water)
`allAirTempsDbl`	*double*; length 12 air temperature, monthly (degrees Celsius)
`pdbegin`	*integer*; beginning year
`pdend`	*integer*; ending year
`smcsawc`	*double*; soil moisture control section available water capacity (millimeters)
`checkargs`	*logical*; check argument length and data types? Default: `FALSE`

Value

an instance of *NewhallDataset*

Examples

```
input_direct <- NewhallDataset(
  stationName = "WILLIAMSPORT",
  country = "US",
  latDD = 41.24,
  lonDD = -76.92,
  elev = 158.0,
  allPrecipsDbl = c(44.2, 40.39, 113.54, 96.77, 95, 98.55,
                    66.04, 13.46, 54.86, 6.35, 17.53, 56.39),
  allAirTempsDbl = c(-2.17, 0.89, 3.72, 9.11, 16.28, 21.11,
                     22.83, 21.94, 19.78, 10.5, 5.33, -1.06),
  pdbegin = 1930,
  pdend = 1930,
  smcsawc = 200.0
)
```

`NewhallDatasetFromPath` *Create an instance of* NewhallDataset *from XML or CSV file*

Description

Create an instance of *NewhallDataset* from XML or CSV file

Usage

```
NewhallDatasetFromPath(pathname, .parser = XMLFileParser)

xml_NewhallDataset(pathname)

csv_NewhallDataset(pathname)
```

Arguments

pathname	*character* containing pathname
.parser	either XMLFileParser or CSVFileParser

NewhallDatasetMetadata

Create an instance of NewhallDatasetMetadata

Description

Create an instance of *NewhallDatasetMetadata*

Usage

```
NewhallDatasetMetadata(
  stationName,
  stationId = character(length(stationName)),
  elev = numeric(length(stationName)),
  stationStateProvidence = character(length(stationName)),
  stationCountry = character(length(stationName)),
  mlraName = character(length(stationName)),
  mlraId = numeric(length(stationName)),
  contribFirstName = character(length(stationName)),
  contribLastName = character(length(stationName)),
  contribTitle = character(length(stationName)),
  contribOrg = character(length(stationName)),
  contribAddress = character(length(stationName)),
  contribCity = character(length(stationName)),
  contribStateProvidence = character(length(stationName)),
  contribPostal = character(length(stationName)),
  contribCountry = character(length(stationName)),
  contribEmail = character(length(stationName)),
  contribPhone = character(length(stationName)),
  notes = numeric(length(stationName)),
  runDate = rep(Sys.Date(), length(stationName)),
  modelVersion = rep(newhall_version(), length(stationName)),
  unitSystem = rep("metric", length(stationName)),
  soilAirOffset = rep(1.2, length(stationName)),
  amplitude = rep(0.66, length(stationName)),
  network = character(length(stationName))
)
```

Arguments

`stationName`	*character*; station name
`stationId`	*character*; station ID
`elev`	*double*; station elevation
`stationStateProvidence`	*character*; station state / providence
`stationCountry`	*character*; station country
`mlraName`	*character*; Major Land Resource Area (MLRA) name
`mlraId`	*integer*; Major Land Resource Area ID
`contribFirstName`	*character*; contributor first name
`contribLastName`	*character*; contributor last name
`contribTitle`	*character*; contributor title
`contribOrg`	*character*; contributor organization
`contribAddress`	*character*; contributor address
`contribCity`	*character*; contributor city
`contribStateProvidence`	*character*; contributor state / providence
`contribPostal`	*character*; contributor postal code
`contribCountry`	*character*; contributor country
`contribEmail`	*character*; contributor email
`contribPhone`	*character*; contributor phone
`notes`	*character* (may have length >1); notes
`runDate`	*character*; run date
`modelVersion`	*character*; model version
`unitSystem`	*character*; unit system either "cm" or "in"
`soilAirOffset`	*double*; soil-air temperature offset
`amplitude`	*double*; soil-air temperature amplitude
`network`	*character*; network

Value

an instance of *NewhallDatasetMetadata*

newhall_batch.default *Run Newhall Soil Climate Simulations*

Description

newhall_batch() provides an interface to multiple runs of the jNSM BASICSimulationModel() for the CSV batch file input format used in jNSM 1.6.0, plus the SpatRaster and RasterStack R object types.

- newhall_batch(<character>) - one or more paths to jNSM Comma-Separated Value '.csv' batch files; see details for required column names
- newhall_batch(<SpatRaster>) - a SpatRaster object, containing the required column names as layers
- newhall_batch(<RasterStack>)- a RasterStack object, containing the required column names as layers

Usage

```
## Default S3 method:
newhall_batch(
  .data = NULL,
  unitSystem = "metric",
  soilAirOffset = ifelse(unitSystem %in% c("in", "english"), 4.5, 2.5),
  amplitude = 0.66,
  hasOHorizon = FALSE,
  isSaturated = FALSE,
  verbose = TRUE,
  toString = TRUE,
  checkargs = TRUE,
  cores = NULL,
  core_thresh = NULL,
  file = NULL,
  nrows = NULL,
  overwrite = NULL
)

newhall_batch(
  .data,
  unitSystem = "metric",
  soilAirOffset = ifelse(unitSystem %in% c("in", "english"), 4.5, 2.5),
  amplitude = 0.66,
  hasOHorizon = FALSE,
  isSaturated = FALSE,
  verbose = TRUE,
  toString = TRUE,
  checkargs = TRUE,
  cores = 1,
  core_thresh = 25000L,
  file = paste0(tempfile(), ”.tif”),
  nrows = nrow(.data),
  overwrite = TRUE
)

## S3 method for class 'character'
newhall_batch(
  .data,
  unitSystem = ”metric”,
  soilAirOffset = ifelse(unitSystem %in% c(”in”, ”english”), 4.5, 2.5),
  amplitude = 0.66,
  hasOHorizon = FALSE,
  isSaturated = FALSE,
  verbose = TRUE,
  toString = TRUE,
  checkargs = TRUE,
  cores = 1,
  core_thresh = 25000L,
  file = paste0(tempfile(), ”.tif”),
  nrows = nrow(.data),
  overwrite = TRUE
)

## S3 method for class 'SpatRaster'
newhall_batch(
  .data,
  unitSystem = ”metric”,
  soilAirOffset = ifelse(unitSystem %in% c(”in”, ”english”), 4.5, 2.5),
  amplitude = 0.66,
  hasOHorizon = FALSE,
  isSaturated = FALSE,
  verbose = TRUE,
  toString = FALSE,
  checkargs = TRUE,
  cores = 1,
  core_thresh = 25000L,
  file = paste0(tempfile(), ”.tif”),
  nrows = nrow(.data)/(terra::ncell(.data)/core_thresh),
  overwrite = TRUE
)

## S3 method for class 'RasterBrick'
newhall_batch(
  .data,
  unitSystem = ”metric”,
  soilAirOffset = ifelse(unitSystem %in% c(”in”, ”english”), 4.5, 2.5),
  amplitude = 0.66,
  hasOHorizon = FALSE,
  isSaturated = FALSE,
  verbose = TRUE,
  toString = TRUE,
  checkargs = TRUE,
  cores = 1,
  core_thresh = 25000L,
  file = paste0(tempfile(), ".tif"),
 nrows = ifelse(ncol(.data) * nrow(.data) < core_thresh, nrow(.data), floor(ncol(.data)
    * nrow(.data)/(core_thresh * cores))),
  overwrite = TRUE
)

## S3 method for class 'RasterStack'
newhall_batch(
  .data,
  unitSystem = "metric",
  soilAirOffset = ifelse(unitSystem %in% c("in", "english"), 4.5, 2.5),
  amplitude = 0.66,
  hasOHorizon = FALSE,
  isSaturated = FALSE,
  verbose = TRUE,
  toString = TRUE,
  checkargs = TRUE,
  cores = 1,
  core_thresh = 25000L,
  file = paste0(tempfile(), ".tif"),
 nrows = ifelse(ncol(.data) * nrow(.data) < core_thresh, nrow(.data), floor(ncol(.data)
    * nrow(.data)/(core_thresh * cores))),
  overwrite = TRUE
)
```

Arguments

`.data` a *data.frame* or *character* vector of paths to CSV files; or a SpatRaster or RasterStack containing the same data elements and names as included in the batch `data.frame`/CSV format

`unitSystem` Default: `"metric"` OR `"mm"` OR `"cm"` use *millimeters* of rainfall (default for the BASIC model); set to `unitSystem="english"` OR `unitSystem="in"` to transform English (inches of precipitation; degrees Fahrenheit) inputs to metric (millimeters of precipitation; degrees Celsius) before running simulation

`soilAirOffset` air-soil temperature offset. Conventionally for jNSM: `2.5` for metric units (default); `4.5` for english units. Can optionally be specified as a layer in a raster input.

`amplitude` difference in amplitude between soil and air temperature sine waves. Default `0.66`. Can optionally be specified as a layer in a raster input.

`hasOHorizon` Used for cryic soil temperature regime criteria. Default: `FALSE`. Can optionally be specified as a layer in a raster input.

`isSaturated` Used for cryic soil temperature regime and aquic soil moisture regime mask. Default: `FALSE`. Can optionally be specified as a layer in a raster input.

`verbose` print message about number of simulations and elapsed time

`toString` call `toString()` method on each *NewhallResults* object and store in `output` column of result?

`checkargs` *logical*; check argument length and data types for each run? Default: `TRUE`

`cores` integer. Number of cores; used only for processing *SpatRaster* or *Raster** input. Default: `1` processes batches sequentially.

`core_thresh` integer. Approximate number of cells to target per core and batch; used to calculate default value for `nrows`. Default `25000` cells.

`file` character. Path to write incremental raster processing output for large inputs that do not fit in memory; passed to `terra::writeStart()` and used only for processing *SpatRaster* or *Raster** input; defaults to a temporary file created by `tempfile()` if needed

`nrows` integer. Number of rows to use per block; passed to `terra::readValues()` `terra::writeValues()`; used only for processing *SpatRaster* or *Raster** input; defaults to number of rows in `.data` if it is small. If the number of cells in `.data` exceeds `core_thresh`, then the number of rows is calculated based on the number of cells in `.data`, `core_thresh` and `cores`.

`overwrite` logical. Overwrite `file`? passed to `terra::writeStart()`; defaults to `TRUE` if needed

Details

Required inputs:

The main inputs to the model are monthly precipitation and air temperature for each site, the location, the soil available water storage, and the elevation.

The following columns and names are required in the input data/object:

- Latitude and Longitude in WGS84 Decimal Degrees: "latDD", "lonDD"
- Monthly Air Temperature (degrees C or F): "tJan", "tFeb", "tMar", "tApr", "tMay", "tJun", "tJul", "tAug", "tSep", "tOct", "tNov", "tDec"
- Monthly Precipitation (millimeters or inches of rain): "pJan", "pFeb", "pMar", "pApr", "pMay", "pJun", "pJul", "pAug", "pSep", "pOct", "pNov", "pDec"
- Profile Available Water Storage (millimeters; Default `200`): "awc"
- Elevation (meters or feet): "elev"

Dry v.s. Moist:

The concept of "dry" versus "moist" is expressed semi-quantitatively in the Newhall model with three different categories of moisture being recognized: "moist", "moist/dry" and "dry."

Of interest to the classification of climate regimes of a soil are not only when/where the soil is dry but how that moisture or lack thereof corresponds with prevailing temperature conditions.

Standard model output fields and their definitions in the latest available JAR file include::

- `"annualRainfall"` - sum of monthly precipitation values over the year

- "waterHoldingCapacity" - total water storage of soil profile in units of length (mm). Default: 200 millimeters (8 inches) of water storage. This is approximately the average water storage when calculated using SSURGO available water capacities and depths for the soils in CONUS.
- "annualWaterBalance" - sum of difference of precipitation and estimated mean potential evapotranspiration (Thornthwaite, 1948) by month
- "annualPotentialEvapotranspiration" - sum of mean monthly potential evapotranspiration (Thornthwaite, 1948)
- "summerWaterBalance" - sum of (summer months only) difference of precipitation and estimated mean potential evapotranspiration by month
- "dryDaysAfterSummerSolstice" - number of days "dry" after June 21; used in definition of Xeric moisture regime
- "moistDaysAfterWinterSolstice" - number of days "moist" after December 21; used in definition of Xeric moisture regime
- "numCumulativeDaysDry" - cumulative number of "dry" days per year
- "numCumulativeDaysMoistDry" - cumulative number of days "intermediate between moist and dry" per year
- "numCumulativeDaysMoist" - cumulative number of days "moist" per year
- "numCumulativeDaysDryOver5C" - cumulative number of days "dry" per year when the *soil temperature is over 5 degrees C*
- "numCumulativeDaysMoistDryOver5C" - cumulative number of days "intermediate between moist and dry" per year when the *soil temperature is over 5 degrees C*
- "numCumulativeDaysMoistOver5C" - cumulative number of days "moist" per year when the *soil temperature is over 5 degrees C*
- "numConsecutiveDaysMoistInSomeParts" - maximum number of consecutive days per year where some parts of the profile are "moist"
- "numConsecutiveDaysMoistInSomePartsOver8C" - maximum number of consecutive days per year where some parts of the profile are "moist" *and* the *soil temperature is over 8 degrees C*
- "temperatureRegime" - estimated Soil Temperature Regime; one of "Pergelic", "Cryic", "Frigid", "Mesic", "Thermic", "Hyperthermic", "Isofrigid", "Isomesic", "Isothermic", or "Isohyperthermic"
- "moistureRegime" - estimated Soil Moisture Regime; one of "Aridic", "Ustic", "Xeric", "Udic", "Perudic", or "Undefined"
- "regimeSubdivision1" - estimated "Moisture Regime Subdivision #1"; one of "Typic", "Weak", "Wet", "Dry", "Extreme", "Xeric", "Udic", "Aridic", or " " (See van Wambecke et al., 1981)
- "regimeSubdivision2" - estimated "Moisture Regime subdivision #2"; one of "Aridic", "Tempustic", "Tropustic", "Tempudic", "Xeric", "Udic", "Tropudic", "Undefined", or " " (See van Wambecke et al., 1981)

"Years" are based on uniform 12 months with 30 days each for a total of 360 days (no leap years).

The following elements have a many:1 relationship with model runs and are not (yet) included in the standard output, but can be accessed using an rJava object reference to a NewhallResults class.

- "meanPotentialEvapotranspiration" - estimated mean monthly potential evapotranspiration (Thornthwaite, 1948)

- `"temperatureCalendar"` - compressed (360 day) grid "calendar" showing days above 5 and 8 degrees C
- `"moistureCalendar"` - compressed (360 day) grid "calendar" showing "moist", "moist/dry" and "dry" days.

Value

When input is a *data.frame* or *character* vector of paths to CSV files, result is a a *data.frame* with key model outputs (see details) containing list columns with Java Objects for *NewhallDataset*, *NewhallResults*. If `toString=TRUE` the column `output` is a *character* containing the `toString()` output from *NewhallResults*

For `SpatRaster` input returns a `SpatRaster` containing numeric and categorical model outputs. `RasterBrick` inputs are first converted to `SpatRaster`, and a `SpatRaster` is returned

References

van Wambeke, A. and Newhall, F. and United States Soil Management Support Services (1981) Calculated Soil Moisture and Temperature Regimes of South America: A Compilation of Soil Climatic Regimes calculated by using a mathematical model developed by F. Newhall (Soil Conservation Service, USDA, 1972). SMSS : Technical Monograph : Soil management support services. New York State College of Agriculture and Life Sciences, Cornell University, Department of Agronomy. Available online: `https://books.google.com/books?id=jwtIAAAAYAAJ`

Thornthwaite, C. W. (1948). An Approach toward a Rational Classification of Climate. Geographical Review, 38(1), 55–94. https://doi.org/10.2307/210739

Newhall, F., Berdanier, C. (1996) Calculation of soil moisture regimes from the climatic record. National Soil Survey Center, Natural Resources Conservation Service, U.S. Dept. of Agriculture. Available online: `https://www.nrcs.usda.gov/Internet/FSE_DOCUMENTS/nrcs142p2_052248.pdf`

See Also

- `BASICSimulationModel()`: create an instance of the Java Newhall Simulation Model
- `newhall_simulation()`: run a single Newhall model instance, return `NewhallResults` object

Examples

```
library(terra)

x <- terra::rast(system.file("extdata", "prism_issr800_sample.tif", package="jNSMR"))

## optional: make larger extent (requires full cache)
# x <- c(newhall_prism_extent(ext(x) * 4), newhall_issr800_extent(ext(x) * 4))

x$elev <- 0 # elevation is not currently used by the model directly

# reduce resolution (for fast example)
x2 <- aggregate(x, 10, na.rm = TRUE)
```

```
# calculate winter, summer and annual average temperatures
d <- as.data.frame(x2)
x2$mwst <- rowMeans(d[, c("tDec","tJan","tFeb")])
x2$msst <- rowMeans(d[, c("tJun","tJul","tAug")])
x2$mast <- rowMeans(d[, paste0("t", month.abb)])
x2$dif <- x2$msst - x2$mwst
plot(x2$dif)

## 1/10th resolution
system.time({ y <- newhall_batch(x2) })

## ~1/3 resolution
# system.time({  y <- newhall_batch(aggregate(x, 3)) })

## full resolution
# system.time({  y <- newhall_batch(x) })

par(mfrow=c(2, 1))

terra::plot(y$annualWaterBalance, main = "Annual Water Balance (P-PET)")
terra::plot(y$waterHoldingCapacity, main = "Water Holding Capacity")

terra::plot(y$temperatureRegime, main = "Temperature Regime")

terra::plot(y$moistureRegime, main = "Moisture Regime")

terra::plot(y$numCumulativeDaysDryOver5C, cex.main=0.75,
            main = "# Cumulative Days Dry over 5 degrees C")
terra::plot(y$numConsecutiveDaysMoistInSomePartsOver8C, cex.main=0.75,
            main = "# Consecutive Days Moist\nin some parts over 8 degrees C")

par(mfrow=c(1,1))
```

newhall_cmip6_cache *Load CMIP6 Downscaled Future Climate Projections*

Description

newhall_cmip6_cache(): Uses the geodata package to download and cache data at the specified resolution.

newhall_cmip6_rast(): Create a *SpatRaster* object. This object contains temperature and precipitation data for the specified data set, at the specified resolution, using the standard jNSM column naming scheme.
newhall_cmip6_subset(): Used to create a subset of the CMIP6 data corresponding to the extent of an input spatial object x.

Usage

```
newhall_cmip6_cache(
  model,
  ssp,
  time,
  resolution = "10m",
  version = "2.1",
  overwrite = FALSE,
  CMIP6_PATH = file.path(newhall_data_dir("cache"), "CMIP6")
)

newhall_cmip6_rast(
  model,
  ssp,
  time,
  resolution = "10m",
  version = "2.1",
  CMIP6_PATH = file.path(newhall_data_dir("cache"), "CMIP6"),
 tiffile = list.files(file.path(CMIP6_PATH, paste0("wc", version, "_", resolution)),
    pattern = "\\.tif$", recursive = TRUE)
)

newhall_cmip6_subset(
  x,
  model,
  ssp,
  time,
  resolution = "10m",
  template = "EPSG:4326",
  CMIP6_PATH = file.path(newhall_data_dir("cache"), "CMIP6")
)
```

Arguments

`model` *character*. Climate model abbrevation. One of "ACCESS-CM2", "ACCESS-ESM1-5", "AWI-CM-1-1-MR", "BCC-CSM2-MR", "CanESM5", "CanESM5-CanOE", "CMCC-ESM2", "CNRM-CM6-1", "CNRM-CM6-1-HR", "CNRM-ESM2-1", "EC-Earth3-Veg", "EC-Earth3-Veg-LR", "FIO-ESM-2-0", "GFDL-ESM4", "GISS-E2-1-G", "GISS-E2-1-H", "HadGEM3-GC31-LL", "INM-CM4-8", "INM-CM5-0", "IPSL-CM6A-LR", "MIROC-ES2L", "MIROC6", "MPI-ESM1-2-HR", "MPI-ESM1-2-LR", "MRI-ESM2-0", "UKESM1-0-LL"

`ssp` *character*. A valid Shared Socio-economic Pathway code: "126", "245", "370" or "585"

`time` *character*. A valid time period. One of "2021-2040", "2041-2060", or "2061-2080"

`resolution` character. Either `"10m"`, `"5m"`, `"2.5m"` or `"30s"`. In minutes (or seconds) of degrees (`"EPSG:4326"`).

`version`	character. Version number. Default: `"2.1"`. See `geodata::worldclim_global()` for details.
`overwrite`	Force download of new cache files? Default: `FALSE`.
`CMIP6_PATH`	Default: `file.path(newhall_data_dir("cache"), "CMIP6")`
`tiffile`	Optional: custom vector of paths to files to use to build raster. Defaults to all .TIF files in the specified cache directory and resolution.
`x`	A *SpatVector*, *SpatRaster*, *SpatExtent*, or any other object type suitable to use with `terra::crop()`.
`template`	Template *SpatRaster* or target CRS specification for re-projection. Default: `"EPSG:4326`

Value

character. Vector of file paths (to CMIP6 .TIF files).

References

- Eyring, V., Bony, S., Meehl, G. A., Senior, C. A., Stevens, B., Stouffer, R. J., and Taylor, K. E.: Overview of the Coupled Model Intercomparison Project Phase 6 (CMIP6) experimental design and organization, Geosci. Model Dev., 9, 1937-1958, doi:10.5194/gmd-9-1937-2016, 2016.
- Detailed and up-to-date description of the CMIP6 experiments protocol: https://search.es-doc.org/?project=cmip6&

See Also

`newhall_data_dir()`

---

newhall_CSVResultsExporter

Export Newhall Results, Data and Metadata to CSV file with CSVResultsExporter

---

Description

Export Newhall Results, Data and Metadata to CSV file with *CSVResultsExporter*

Usage

```
newhall_CSVResultsExporter(results, pathname)
```

Arguments

`results`	*NewhallResults* `jobjRef`
`pathname`	output CSV file path; default: `NULL`

Value

a CSV file written to specified path

`newhall_data_dir` *Newhall Data Directory*

Description

Returns a platform-specific user-level directory where data, configuration and cache files may be stored.

Usage

```
newhall_data_dir(which = c("data", "config", "cache"))
```

Arguments

`which`	One of: `"data"`, `"config"`, or `"cache"`

Value

character. Directory path.

`newhall_daymet_subset` *Load DAYMET Monthly Data at 1 kilometer Resolution*

Description

`newhall_daymet_subset()`: Used to create a subset of the DAYMET data corresponding to the extent of an input spatial object `x`.

Usage

```
newhall_daymet_subset(
  x,
  start_year = 1991,
  end_year = 2020,
  force = FALSE,
  DAYMET_PATH = tempdir()
)
```

Arguments

`x` A *SpatVector*, *SpatRaster*, *SpatExtent*, or any other object type suitable to use with `terra::ext()`

`start_year` integer. First year in range to download

`end_year` integer. Last year in range to download.

`force` logical. Force download when files exist in `DAYMET_PATH`? Default: `FALSE`

`DAYMET_PATH` Default: `file.path(newhall_data_dir("cache"), "DAYMET")`

Value

A *SpatRaster* object

References

Thornton, M.M., R. Shrestha, Y. Wei, P.E. Thornton, S-C. Kao, and B.E. Wilson. 2022. Daymet: Monthly Climate Summaries on a 1-km Grid for North America, Version 4 R1. ORNL DAAC, Oak Ridge, Tennessee, USA. doi:10.3334/ORNLDAAC/2131

See Also

`newhall_data_dir()`

`newhall_GUI` *Open the Java Newhall Graphical User Interface*

Description

This function must be called interactively.

Usage

```
newhall_GUI(command_only = FALSE)
```

Arguments

`command_only` If `TRUE` return the command that would be executed to run GUI

Details

See documentation for `system()` return result for limitations on line length, error conditions, etc.

Value

If `intern=TRUE` (default), the output of the command, one line per character string. `0` if successful. If the command could not be run for any reason, the value is 127 and a warning is issued.

newhall_issr800_cache *Load SoilWeb "ISSR-800" at 800 meter Resolution*

Description

Currently the only ISSR-800 data are only available for the contiguous (lower 48) United States. The only property cached for use in the Newhall model is the "available water holding capacity" (sum of storage for the whole profile, in millimeters). For consistency with PRISM grid the values are reprojected from `"EPSG:5070"` to `"EPSG:4269"` (see `newhall_nad83_template()`) `newhall_issr800_subset()`: Used to create a subset of the ISSR-800 soil available water storage data corresponding to the extent of an input spatial object `x`.

`newhall_issr800_rast()`: Create a *SpatRaster* object. This object contains Available Water Capacity (Storage) for at 800 meter resolution using the standard jNSM column naming scheme.

Usage

```
newhall_issr800_cache(
  ISSR800_PATH = file.path(newhall_data_dir("cache"), "SoilWeb", "800m"),
  template = newhall_nad83_template(),
  overwrite = FALSE
)

newhall_issr800_subset(
  x,
  template = newhall_nad83_template(),
  ISSR800_PATH = file.path(newhall_data_dir("cache"), "SoilWeb", "800m")
)

newhall_issr800_rast(
  ISSR800_PATH = file.path(newhall_data_dir("cache"), "SoilWeb", "800m"),
  tiffile = list.files(ISSR800_PATH, "\\.tif$", full.names = TRUE)
)
```

Arguments

`ISSR800_PATH`	Default: `file.path(newhall_data_dir("cache"), "SoilWeb", "800m")`
`template`	Template *SpatRaster* or target CRS specification for reprojection. Default: `newhall_nad83_template()`
`overwrite`	Force download of new cache files? Default: `FALSE`.
`x`	A *SpatVector*, *SpatRaster*, *SpatExtent*, or any other object type suitable to use with `terra::crop()`;
`tiffile`	Optional: custom vector of paths to files to use to build raster. Defaults to all .TIF files in the specified cache directory and resolution.

Details

The data are stored in centimeters on the SoilWeb server. When `newhall_issr800_cache()` saves the file the data are converted to millimeters, which is required for the Newhall model.

Value

character. Path to cached water storage GeoTIFF.

References

Walkinshaw, Mike, A.T. O'Geen, D.E. Beaudette. "Soil Properties." California Soil Resource Lab, 1 Oct. 2022, https://casoilresource.lawr.ucdavis.edu/soil-properties/.

See Also

`newhall_data_dir()`

`newhall_prism_cache` *Load PRISM Monthly "Normals" at 800 meter or 4 kilometer Resolution*

Description

`newhall_prism_cache()`: Uses the prism package to download and cache data at the specified resolution. At this time only monthly grids for 30 year Normals (1991-2020) are supported.

`newhall_prism_rast()`: Create a *SpatRaster* object. This object contains temperature and precipitation data for the specified data set, at the specified resolution, using the standard jNSM column naming scheme.
`newhall_prism_subset()`: Used to create a subset of the PRISM data corresponding to the extent of an input spatial object `x`.

`newhall_nad83_template()`: Empty `SpatRaster` corresponding to the lower 48 United States PRISM data/extent at the specified resolution.

Usage

```
newhall_prism_cache(
  resolution = "800m",
  overwrite = FALSE,
  PRISM_PATH = file.path(newhall_data_dir("cache"), "PRISM")
)

newhall_prism_rast(
  resolution = "800m",
  PRISM_PATH = file.path(newhall_data_dir("cache"), "PRISM"),
 bilfile = list.files(file.path(PRISM_PATH, resolution), "\\.bil$", recursive = TRUE)
)

newhall_prism_subset(
  x,
  resolution = "800m",
  template = newhall_nad83_template(resolution = resolution),
  PRISM_PATH = file.path(newhall_data_dir("cache"), "PRISM")
)

newhall_nad83_template(resolution = "800m")
```

Arguments

`resolution` character. Either `"800m"` (default) or `"4km"`

`overwrite` Force download of new cache files? Default: `FALSE`.

`PRISM_PATH` Default: `file.path(newhall_data_dir("cache"), "PRISM")`

`bilfile` Optional: custom vector of paths to files to use to build raster. Defaults to all .BIL files in the specified cache directory and resolution.

`x` A *SpatVector*, *SpatRaster*, *SpatExtent*, or any other object type suitable to use with `terra::crop()`.

`template` Template *SpatRaster* or target CRS specification for re-projection. Default: `newhall_nad83_template()`

Details

Currently used only for matching the ISSR800 source to PRISM.

Value

character. Vector of file paths (to PRISM .BIL files).

References

PRISM Climate Group, Oregon State University, `https://prism.oregonstate.edu`, data created 4 Feb 2014, accessed 22 Jul 2023.

See Also

`newhall_data_dir()`

`newhall_issr800_cache()` `newhall_issr800_rast()` `newhall_issr800_rast()`

Examples

```
newhall_nad83_template()

newhall_nad83_template("4km")
```

newhall_simulation *Run Newhall* BASICSimulationModel *simulation*

Description

Run Newhall *BASICSimulationModel* simulation

Usage

```
newhall_simulation(
  dataset,
  smcsawc = 200,
  soilAirOffset = 2.5,
  amplitude = 0.66,
  bsm = BASICSimulationModel(),
  toString = FALSE
)
```

Arguments

`dataset`	a *NewhallDataset* `jobjRef`
`smcsawc`	Default: `200`
`soilAirOffset`	air-soil temperature offset. Conventionally for jNSM: `2.5` for metric units (default); `4.5` for english units.
`amplitude`	Default: `0.66` amplitude difference between soil and air temperature sine waves
`bsm`	`jobjRef` for *BASICSimulationModel*; Default: `BASICSimulationModel()`
`toString`	logical; return *NewhallResults* (Default: `FALSE`), or if `TRUE` call `result.toString()` to get formatted standard output as *character*?

Value

NewhallResults `jobjRef`

newhall_version *Get Java Newhall JAR file version*

Description

Get Java Newhall JAR file version

Usage

```
newhall_version()
```

Details

This is a wrapper around accessing the public string field `NSM_VERSION` stored within the main *Newhall* class of the JAR file.

Value

character containing version number of Newhall JAR file

`newhall_worldclim_cache`

Load WorldClim Monthly Averages

Description

`newhall_worldclim_cache()`: Uses the geodata package to download and cache data at the specified resolution.

`newhall_worldclim_rast()`: Create a *SpatRaster* object. This object contains temperature and precipitation data for the specified data set, at the specified resolution, using the standard jNSM column naming scheme.
`newhall_worldclim_subset()`: Used to create a subset of the WorldClim data corresponding to the extent of an input spatial object `x`.

Usage

```
newhall_worldclim_cache(
  resolution = "10m",
  version = "2.1",
  overwrite = FALSE,
  WORLDCLIM_PATH = file.path(newhall_data_dir("cache"), "WorldClim")
)

newhall_worldclim_rast(
  resolution = "10m",
  version = "2.1",
  WORLDCLIM_PATH = file.path(newhall_data_dir("cache"), "WorldClim"),
 tiffile = list.files(file.path(WORLDCLIM_PATH, paste0("wc", version, "_", resolution)),
    pattern = "\\.tif$", recursive = TRUE)
)

newhall_worldclim_subset(
  x,
  resolution = "10m",
  template = "EPSG:4326",
  WORLDCLIM_PATH = file.path(newhall_data_dir("cache"), "WorldClim")
)
```

Arguments

`resolution` character. Either `"10m"`, `"5m"`, `"2.5m"` or `"30s"`. In minutes (or seconds) of degrees (`"EPSG:4326"`).

`version` character. Version number. Default: `"2.1"`. See `geodata::worldclim_global()` for details.

`overwrite` Force download of new cache files? Default: `FALSE`.

`WORLDCLIM_PATH` Default: `file.path(newhall_data_dir("cache"), "WorldClim")`

`tiffile` Optional: custom vector of paths to files to use to build raster. Defaults to all .TIF files in the specified cache directory and resolution.

`x` A *SpatVector*, *SpatRaster*, *SpatExtent*, or any other object type suitable to use with `terra::crop()`.

`template` Template *SpatRaster* or target CRS specification for re-projection. Default: `"EPSG:4326`

Value

character. Vector of file paths (to WorldClim .TIF files).

References

Fick, S.E. and R.J. Hijmans, 2017. WorldClim 2: new 1km spatial resolution climate surfaces for global land areas. International Journal of Climatology 37 (12): 4302-4315. `https://www.worldclim.org/data/worldclim21.html`

See Also

`newhall_data_dir()`

`newhall_XMLResultsExporter` *Export Newhall Results, Data and Metadata to XML file with* XMLResultsExporter

---

Description

Export Newhall Results, Data and Metadata to XML file with *XMLResultsExporter*

Usage

```
newhall_XMLResultsExporter(dataset, results, pathname)
```

Arguments

`dataset` *NewhallDataset* `jobjRef`

`results` *NewhallResults* `jobjRef`

`pathname` output XML file path

Value

an XML file written to pathname

`newhall_XMLStringResultsExporter`
Export Newhall Results, Data and Metadata to XML string with XMLStringResultsExporter

---

Description

Export Newhall Results, Data and Metadata to XML string with *XMLStringResultsExporter*

Usage

```
newhall_XMLStringResultsExporter(dataset, results)
```

Arguments

`dataset`	*NewhallDataset* `jobjRef`
`results`	*NewhallResults* `jobjRef`

Value

character containing XML string

`writeRasterLayers` *Iterate over multiband output and write as single-layer files*

Description

This function is a simple wrapper around `terra:writeRaster()` that makes it easier to separate the individual layers of an input or output grid as separate files.

Usage

```
writeRasterLayers(x, output_dir = NULL, ...)
```

Arguments

`x`	character. Path to raster file(s) to split by layer.
`output_dir`	character. Default: `NULL` creates new directories in current working directory. Alternately specify a different path for output folders to be created.
`...`	Additional arguments to `terra::writeRaster()`.

Value

New directories are created in `output_dir` (or current working directory) based on each input file `x`.

Examples

```
library(terra)

x <- writeRaster(rast(list(a = rast(matrix(1)),
                           b = rast(matrix(2)))), "test.tif")

writeRasterLayers("test.tif", "test")

unlink(c("test.tif", "test"), recursive=TRUE)
```

`XMLFileParser` *Create an instance of* XMLFileParser

Description

Create an instance of *XMLFileParser*

Usage

```
XMLFileParser(pathname)
```

Arguments

`pathname`	*character* containing `pathname`

Value

an instance of *XMLFileParser* class

`XMLResultsExporter` *Create an instance of* XMLResultsExporter

Description

Create an instance of *XMLResultsExporter*

Usage

```
XMLResultsExporter(pathname)
```

Arguments

`pathname`	character; output path

Value

an instance of *XMLResultsExporter* class

`XMLStringResultsExporter` *Create an instance of* XMLStringResultsExporter

Description

Create an instance of *XMLStringResultsExporter*

Usage

```
XMLStringResultsExporter()
```

Value

an instance of *XMLStringResultsExporter* class

Index