Package 'rwrfhydro'

April 30, 2015

Type Package

Title R tools for the WRF Hydro Model

Version 1.0.0

Date 2015-05-01

Maintainer James McCreight <jamesmcc@ucar.edu>

Description A community-contributed tool box for managing, analyzing, and visualizing WRF Hydro (and HydroDART) input and output files in R.

License https://github.com/mccreigh/rwrfhydro/blob/master/LICENSE.md

Depends R (>= 3.1.0)

Imports plyr (>= 1.8.1), grid (>= 3.1.2), lubridate (>= 1.3.3), ncdf4 (>= 1.13), ggplot2 (>= 1.0.0), ggmap (>= 2.3), reshape2 (>= 1.4.1), doMC (>= 1.3.3), foreach (>= 1.4.2), curl (>= 0.5), dataRetrieval (>= 2.1.2), raster (>= 2.3), httr (>= 0.6.1), devtools (>= 1.7.0), jsonlite (>= 0.9.14)

LazyData true

Suggests testthat, knitr, rgdal, pander, ptw

BuildVignettes false

VignetteBuilder knitr

NeedsCompilation no

Author James McCreight [aut, cre], Aubrey Dugger [aut]

R topics documented:

`CalcFdc` *Calculate flow duration curve statistics*

Description

`CalcFdc` calculates flow exceedance probabilities and adds them as a new column to the dataframe.

Usage

```
CalcFdc(strDf, strCol = "q_cms")
```

Arguments

strDf	The streamflow time series dataframe (e.g., output from ReadFrxstPts or ReadUsgsGage). The data frame must contain a column of streamflow values.
strCol	The name of the column containing the streamflow values (DEFAULT="q_cms").

Details

CalcFdc reads a streamflow time series dataframe (modeled or observed) and outputs a modified dataframe with calculated percent exceedances.

Value

A dataframe containing the original input data with a new column added called "<strCol>.fdc" containing the flow exceedance probabilities.

See Also

Other flowDurationCurves: CalcFdcSpline; PlotFdcCompare; PlotFdc

Examples

```
## Take the time series of observed 5-minute streamflow values for Fourmile
## Creek (stored in the dataframe obsStr5min.fc in the column "q_cms") and
## return the same dataframe with a new column called "q_cms.fdc"
## containing the flow exceedance probabilities.
## Not run:
obsStr5min.fc <- CalcFdc(obsStr5min.fc, "q_cms")

## End(Not run)
```

CalcFdcPerf *Computes flow duration curve statistics for WRF-Hydro streamflow output*

Description

CalcFdcPerf calculates flow duration curve statistics for streamflow output.

Usage

```
CalcFdcPerf(strDf.mod, strDf.obs, strCol.mod = "q_cms",
  strCol.obs = "q_cms", stdate = NULL, enddate = NULL)
```

Arguments

strDf.mod	The forecast point output dataframe (required). Assumes only one forecast point per file, so if you have multiple forecast points in your output dataframe, use subset to isolate a single forecast point's data. Also assumes model output and observation both contain POSIXct fields (called "POSIXct").
strDf.obs	The observed streamflow dataframe. Assumes only one gage per file, so if you have multiple gages in your dataframe, use subset to isolate a single gage's data. Also assumes model output and observation both contain POSIXct fields (called "POSIXct").
strCol.mod	The column name for the streamflow time series for the MODEL data (default="q_cms")
strCol.obs	The column name for the streamflow time series for the OBSERVED data (default="q_cms")
stdate	Start date for statistics (DEFAULT=NULL, all records will be used). Date MUST be specified in POSIXct format with appropriate timezone (e.g., as.POSIXct("2013-05-01 00:00:00", format="%Y-%m-%d %H:%M:%S", tz="UTC"))
enddate	End date for statistics (DEFAULT=NULL, all records will be used). Date MUST be specified in POSIXct format with appropriate timezone (e.g., as.POSIXct("2013-05-01 00:00:00", format="%Y-%m-%d %H:%M:%S", tz="UTC"))

Details

CalcFdcPerf reads a model forecast point streamflow timeseries (i.e., created using ReadFrxstPts) and a streamflow observation timeseries (i.e., created using ReadUsgsGage) and calculates flow duration curve statistics at various exceedance thresholds (e.g., 10%, 20%, etc.). The tool will subset data to matching time periods (e.g., if the observed data is at 5-min increments and modelled data is at 1-hr increments, the tool will subset the observed data to select only observations on the matching hour break).

Flow Duration Curve Statistics:
(mod = model output, obs = observations)

- p.exceed: exceedance threshold (e.g., 0.2 means a flow value that is exceeded 20% of the time)
- q.mod: MODEL flow value at specified exceedance threshold (in native flow units)
- q.obs: OBSERVED flow value at specified exceedance threshold (in native flow units)
- q.err: difference between model and observed flow values [mod-obs] (in native flow units)
- q.perr: percent error in model flow [(mod-obs)/obs]

Value

A new dataframe containing the flow duration curve statistics.

Examples

```
## Take forecast point model output for Fourmile Creek (modStrh.mod1.fc)
## and a corresponding USGS gage observation file (obsStrh.fc), both at an
## hourly time step, and calculate flow duration curve statistics. The
## model forecast point data was imported using ReadFrxstPts and the gage
## observation data was imported using ReadUsgsGage.

## Not run:
```

```
CalcFdcPerf(modStr1h.allrt.fc, obsStr5min.fc)

Output:
 p.exceed    q.mod   q.obs
 0.1         3.07    5.25
 0.2         1.35    2.31
 0.3         0.82    1.06
 0.4         0.48    0.65
 0.5         0.29    0.45
 0.6         0.18    0.34
 0.7         0.14    0.25
 0.8         0.11    0.19
 0.9         0.08    0.16

## End(Not run)
```

`CalcFdcSpline` *Generate a spline-fit funtion for a flow duration curve*

Description

`CalcFdcSpline` generates a spline fit function for a flow duration curve.

Usage

```
CalcFdcSpline(strDf, strCol = "q_cms")
```

Arguments

`strDf`	The streamflow time series dataframe (e.g., output from `ReadFrxstPts` or `ReadUsgsGage`) already processed through `CalcFdc`. The data frame must contain a column of streamflow values and a corresponding column (named "<strCol>.fdc") of flow exceedance probabilities.
`strCol`	The name of the column containing the streamflow values (DEFAULT="q_cms").

Details

`CalcFdcSpline` reads the percent exceedances generated by `CalcFdc` and outputs a fitted spline function useful for plotting or estimating flow at specified exceedance thresholds (e.g., 20% exceedance probability).

Value

A spline function fitted to flow (y) vs. flow exceedance probabilities (x).

See Also

Other flowDurationCurves: `CalcFdc`; `PlotFdcCompare`; `PlotFdc`

Examples

```
## Take a time series of observed 5-minute streamflow values for Fourmile
## Creek (stored in the dataframe obsStr5min.fc in the column "q_cms") that
## has already been processed through "CalcFdc" (so also contains a column
## named "q_cms.fdc") and return a spline fit function.
## Not run:
fdc.obsStr5min.fc <- CalcFdcSpline(obsStr5min.fc, "q_cms")

## Use the spline function to estimate the flow value that is exceeded 20% of the time.

fdc.obsStr5min.fc(0.2)
> 0.72

## End(Not run)
```

`CalcModPerf` *Computes model performance statistics for WRF-Hydro flux output*

Description

`CalcModPerf` calculates model performance statistics for flux output.

Usage

```
CalcModPerf(flxDf.mod, flxDf.obs, flxCol.mod = "q_cms",
  flxCol.obs = "q_cms", stdate = NULL, enddate = NULL,
  subdivisions = 1000)
```

Arguments

`flxDf.mod`	The flux output dataframe (required). Assumes only one forecast point per file, so if you have multiple forecast points in your output dataframe, use subset to isolate a single forecast point's data. Also assumes model output and observation both contain POSIXct fields (called "POSIXct").
`flxDf.obs`	The observed flux dataframe. Assumes only one observation point per file, so if you have multiple observation points in your dataframe, use subset to isolate a single point's data. Also assumes model output and observation both contain POSIXct fields (called "POSIXct").
`flxCol.mod`	The column name for the flux time series for the MODEL data (default="q_cms")
`flxCol.obs`	The column name for the flux time series for the OBSERVED data (default="q_cms")
`stdate`	Start date for statistics (DEFAULT=NULL, all records will be used). Date MUST be specified in POSIXct format with appropriate timezone (e.g., as.POSIXct("2013-05-01 00:00:00", format="%Y-%m-%d %H:%M:%S", tz="UTC"))
`enddate`	End date for statistics (DEFAULT=NULL, all records will be used). Date MUST be specified in POSIXct format with appropriate timezone (e.g., as.POSIXct("2013-05-01 00:00:00", format="%Y-%m-%d %H:%M:%S", tz="UTC"))
`subdivisions`	Number of subdivisions used in flow duration curve integration (DEFAULT=1000); increase value if integrate function throws an error.

Details

`CalcModPerf` reads a model flux time series (i.e., created using `ReadFrxstPts`) and an observation time series (i.e., created using `ReadUsgsGage`) and calculates model performance statistics (Nash-Sutcliffe Efficiency, Rmse, etc.) at various time scales and for low and high fluxes. The tool will subset data to matching time periods (e.g., if the observed data is at 5-min increments and modelled data is at 1-hr increments, the tool will subset the observed data to select only observations on the matching hour break).

Performance Statistics:
(mod = model output, obs = observations, n = sample size)

- n: sample size
- nse: Nash-Sutcliffe Efficiency

$$nse = 1 - (sum((obs - mod)^2)/sum((obs - mean(obs))^2))$$

- nselog: log-transformed Nash-Sutcliffe Efficiency

$$nselog = 1 - (sum((log(obs) - log(mod))^2)/sum((log(obs) - mean(log(obs)))^2))$$

- cor: correlation coefficient

$$cor = cor(mod, obs)$$

- rmse: root mean squared error

$$rmse = sqrt(sum((mod - obs)^2)/n)$$

- rmsenorm: normalized root mean squared error

$$rmsenorm = rmse/(max(obs) - min(obs))$$

- bias: percent bias

$$bias = sum(mod - obs)/sum(obs) * 100$$

- mae: mean absolute error

$$mae = mean(abs(mod - obs))$$

- errcom: error in the center-of-mass of the flux, where center-of-mass is the hour/day when 50% of daily/monthly/water-year flux has occurred. Reported as number of hours for daily time scale and number of days for monthly and yearly time scales.
- errmaxt: Error in the time of maximum flux. Reported as number of hours for daily time scale and number of days for monthly and yearly time scales).
- errfdc: Error in the integrated flow duration curve between 0.05 and 0.95 exceedance thresholds (in native flow units).

Time scales/Flux types:

- ts = native model/observation time step (e.g., hourly)
- daily = daily time step
- monthly = monthly time step
- yearly = water-year time step
- max10 = high flows; restricted to the portion of the time series where the observed flux is in the highest 10% (native time step)
- min10 = low flows; restricted to the portion of the time series where the observed flux is in the lowest 10% (native time step)

Value

A new dataframe containing the model performance statistics.

See Also

Other modelEvaluation: CalcModPerfMulti; CalcNoahmpFluxes; CalcNoahmpWatBudg

Examples

```
## Take forecast point model output for Fourmile Creek (modStrh.mod1.fc) and a corresponding
## USGS gage observation file (obsStrh.fc), both at an hourly time step, and calculate
## model performance statistics. The model forecast point data was imported using ReadFrxstPts
## and the gage observation data was imported using ReadUsgsGage.


## Not run:
CalcModPerf(modStr1h.allrt.fc, obsStr5min.fc)

> Output:
          nse nselog  cor rmse rmsenorm  bias  mae errcom errmaxt errfdc
ts       0.57   0.61 0.79 1.43     9.48 -28.0 0.70     NA      NA  -0.42
daily    0.71   0.64 0.87 1.17     9.86 -28.1 0.61   0.19   -2.25  -0.37
monthly  0.80   0.70 0.93 0.89    12.73 -26.6 0.53  -0.18   -0.96     NA
yearly   0.05   0.37 0.36 0.55    41.50  -6.5 0.45  -1.50   -3.38     NA
max10   -7.50 -15.94 0.19 3.82    38.89 -24.5 0.04     NA      NA     NA
min10   -2.84  -1.83 0.10 0.05    33.36 -23.7   NA     NA      NA     NA

## End(Not run)
```

CalcModPerfMulti *Computes model performance statistics for WRF-Hydro flux output*

Description

CalcModPerfMulti calculates model performance statistics for flux output.

Usage

```
CalcModPerfMulti(flxDf.mod, flxDf.obs, flxCol.mod = "q_cms",
  flxCol.obs = "q_cms", stdate = NULL, enddate = NULL)
```

Arguments

flxDf.mod	The flux output dataframe (required). Assumes only one forecast point per file, so if you have multiple forecast points in your output dataframe, use subset to isolate a single forecast point's data. Also assumes model output and observation both contain POSIXct fields (called "POSIXct").
flxDf.obs	The observed flux dataframe. Assumes only one observation point per file, so if you have multiple observation points in your dataframe, use subset to isolate a single point's data. Also assumes model output and observation both contain POSIXct fields (called "POSIXct").
flxCol.mod	The column name for the flux time series for the MODEL data (default="q_cms")

flxCol.obs	The column name for the flux time series for the OBSERVED data (default="q_cms")
stdate	Start date for statistics (DEFAULT=NULL, all records will be used). Date MUST be specified in POSIXct format with appropriate timezone (e.g., as.POSIXct("2013-05-01 00:00:00", format="%Y-%m-%d %H:%M:%S", tz="UTC"))
enddate	End date for statistics (DEFAULT=NULL, all records will be used). Date MUST be specified in POSIXct format with appropriate timezone (e.g., as.POSIXct("2013-05-01 00:00:00", format="%Y-%m-%d %H:%M:%S", tz="UTC"))

Details

CalcModPerfMulti reads a model flux time series (i.e., created using ReadFrxstPts) and an observation time series (i.e., created using ReadUsgsGage) and calculates model performance statistics (Nash-Sutcliffe Efficiency, Rmse, etc.) at various time scales and for low and high fluxes. The tool will subset data to matching time periods (e.g., if the observed data is at 5-min increments and modelled data is at 1-hr increments, the tool will subset the observed data to select only observations on the matching hour break).

CalcModPerfMulti calculates the same statistics as CalcModPerf) but returns results as a single-row vs. multi-row dataframe. This is intended to streamline compiling statistics for multiple data runs or multiple sites.

Performance Statistics:
(mod = model output, obs = observations, n = sample size)

- n: sample size
- nse: Nash-Sutcliffe Efficiency

$$nse = 1 - (sum((obs - mod)^2)/sum((obs - mean(obs))^2))$$

- nselog: log-transformed Nash-Sutcliffe Efficiency

$$nselog = 1 - (sum((log(obs) - log(mod))^2)/sum((log(obs) - mean(log(obs)))^2))$$

- cor: correlation coefficient

$$cor = cor(mod, obs)$$

- rmse: root mean squared error

$$rmse = sqrt(sum((mod - obs)^2)/n)$$

- rmsenorm: normalized root mean squared error

$$rmsenorm = rmse/(max(obs) - min(obs))$$

- bias: percent bias

$$bias = sum(mod - obs)/sum(obs) * 100$$

- mae: mean absolute error

$$mae = mean(abs(mod - obs))$$

- errcom: error in the center-of-mass of the flux, where center-of-mass is the hour/day when 50% of daily/monthly/water-year flux has occurred. Reported as number of hours for daily time scale and number of days for monthly and yearly time scales.
- errmaxt: Error in the time of maximum flux. Reported as number of hours for daily time scale and number of days for monthly and yearly time scales).

- errfdc: Error in the integrated flow duration curve between 0.05 and 0.95 exceedance thresholds (in native flow units).

Time scales/Flux types:

- t = native model/observation time step (e.g., hourly)
- dy = daily time step
- mo = monthly time step
- yr = water-year time step
- max10 = high flows; restricted to the portion of the time series where the observed flux is in the highest 10% (native time step)
- min10 = low flows; restricted to the portion of the time series where the observed flux is in the lowest 10% (native time step)

Value

A new dataframe containing the model performance statistics.

See Also

Other modelEvaluation: CalcModPerf; CalcNoahmpFluxes; CalcNoahmpWatBudg

Examples

```
## Take forecast point model output for Fourmile Creek (modStrh.mod1.fc) and a corresponding
## USGS gage observation file (obsStrh.fc), both at an hourly time step, and calculate
## model performance statistics. The model forecast point data was imported using ReadFrxstPts
## and the gage observation data was imported using ReadUsgsGage.


## Not run:
CalcModPerfMulti(modStr1h.allrt.fc, obsStr5min.fc)

> Output:
t_n t_nse t_nselog t_cor t_rmse t_rmsenorm t_bias t_mae t_errfdc dy_n dy_nse dy_nselog
744  0.63     0.66   0.8   0.29      13.29    0.7  0.19     0.04   32   0.69      0.74
dy_cor dy_rmse dy_rmsenorm dy_bias dy_mae dy_errcom dy_errmaxt dy_errfdc mo_n
  0.83    0.26       13.23     0.6   0.18      0.03      -0.56      0.03    2
mo_nse mo_nselog mo_cor mo_rmse mo_rmsenorm mo_bias mo_mae mo_errcom mo_errmaxt
     1      0.99      1    0.01        2.46    -0.8   0.01       0.5        1.5
yr_n yr_bias yr_mae yr_errcom yr_errmaxt max10_n max10_nse max10_nselog max10_cor
   1     0.7   0.01         0          3      75     -5.05         -6.9     -0.84
max10_rmse max10_rmsenorm max10_bias max10_mae min10_n min10_nse min10_nselog
      0.72          80.01      -27.9       0.6     138     -2.64        -2.43
min10_cor min10_rmse min10_rmsenorm min10_bias min10_mae
     0.59       0.11          53.89       -7.3      0.09

## End(Not run)
```

`CalcNoahmpFluxes` *Calculate water fluxes from NoahMP output*

Description

`CalcNoahmpFluxes` calculates water balance fluxes from accumulated water terms.

Usage

```
CalcNoahmpFluxes(ldasoutDf)
```

Arguments

`ldasoutDf`	The LDASOUT dataframe

Details

Read a dataframe derived from NoahMP LDASOUT output (i.e., using `GetMultiNcdf`) and calculate water budget component fluxes from accumulated water variables. NOTE: Currently only works for runs using NoahMP as the LSM.

Value

The input dataframe with new water flux columns added.

See Also

Other modelEvaluation: `CalcModPerfMulti`; `CalcModPerf`; `CalcNoahmpWatBudg`

Examples

```
## Take a NoahMP LDASOUT dataframe for a model run of Fourmile Creek
## and generate a dataframe with water fluxes added.
## Not run:
modLDASOUT1d.nort.fc <- CalcNoahmpFluxes(modLDASOUT1d.nort.fc)

## End(Not run)
```

`CalcNoahmpWatBudg` *Calculate water balance from WRF-Hydro (w/NoahMP) output*

Description

`CalcNoahmpWatBudg` calculates water budget components from WRF-Hydro (w/NoahMP) model output.

Usage

```
CalcNoahmpWatBudg(ldasoutDf, rtoutDf = NULL, gwoutDf = NULL,
  sfcrt = FALSE, soildeps = c(100, 300, 600, 1000), basarea = NULL)
```

Arguments

ldasoutDf	The LDASOUT dataframe (required)
rtoutDf	The RTOUT dataframe, if overland or subsurface routing was turned on (default=NULL)
gwoutDf	The GW_OUT dataframe, if groundwater model was turned on (default=NULL)
sfcrt	A flag whether surface overland flow routing was active. All other routing options are determined based on the input dataframes, as needed (e.g., if gwoutDf is provided, it is assumed that the groundwater model was active). (default=FALSE)
soildeps	A list of soil layer depths in mm (top to bottom, default=c(100, 300, 600, 1000))
basarea	The basin area in square km (necessary only if gwoutDf is provided)

Details

CalcNoahmpWatBudg reads dataframes derived from WRF-Hydro output (i.e., using GetMultiNcdf) and calculates water budget partitioning (e.g., surface runoff, evaporation, groundwater). Assumes WRF-Hydro output dataframes have already been masked to the desired basin. See GetMultiNcdf documentation for examples of how to do this. NOTE: Currently only works for model runs using NoahMP as the LSM.

REQUIRED variables (these terms must be in your input dataframes):

- LDASOUT: ACCPRCP, ACCECAN, ACCETRAN, ACCEDIR, SFCRNOFF, UGDRNOFF, SOIL_M (all layers), SNEQV, CANICE, CANLIQ
- RTOUT (optional, use if overland or subsurface routing were activated): QSTRMVOLRT, SFCHEADSUBRT, QBDRY
- GWOUT (optional, use if groundwater bucket model was activated): q_cms, POSIXct

OUTPUT water budget terms (may vary depending on model configuration):

- LSM_PRCP: Total precipitation (mm)
- LSM_ECAN: Total canopy evaporation (mm)
- LSM_ETRAN: Total transpiration (mm)
- LSM_EDIR: Total surface evaporation (mm)
- LSM_DELSWE: Change in snowpack snow water equivalent (mm)
- LSM_DELCANWAT: Change in canopy water storage (liquid + ice) (mm)
- LSM_SFCRNOFF: Surface runoff from LSM *(for an LSM-only run)* (mm)
- LSM_UGDRNOFF: Subsurface runoff from LSM *(for an LSM-only run)* (mm)
- LSM_DELSOILM: Change in total soil moisture storage (mm)
- HYD_QSTRMVOL: Total runoff into channel from land *(routing model only)* (mm)
- HYD_DELSFCHEAD: Change in surface storage *(routing model only)* (mm)
- HYD_QBDRY: Total flow outside of domain *(routing model only)* (mm)
- HYD_GWOUT: Total groundwater outflow *(routing model only)* (mm)
- HYD_DELGWSTOR: Change in groundwater storage *(routing model only)* (mm)
- WB_SFCRNOFF: Total surface runoff used in the water budget calculation (*either* LSM_SFCRNOFF or HYD_QSTRMVOL) (mm)
- WB_GWOUT: Total groundwater outflow used in the water budget calculation (*either* LSM_UGDRNOFF or HYD_GWOUT) (mm)

- ERROR: Remainder in water budget (mm)
- RUN_FRAC: Runoff fraction, runoff/precipitation
- EVAP_FRAC: Evaporative fraction, evapotranspiration/precipitation
- STOR_FRAC: Change in storage fraction, storagechange/precipitation

Value

A new dataframe containing the water budget components in mm.

See Also

Other modelEvaluation: CalcModPerfMulti; CalcModPerf; CalcNoahmpFluxes

Examples

```
## Take a NoahMP LDASOUT dataframe for a model run of Fourmile Creek with no routing
## options turned on and return a water budget summary.

## Not run:
wb.nort.fc <- CalcNoahmpWatBudg(modLDASOUT1d.nort.fc)
wb.nort.fc
##
## Take NoahMP LDASOUT, HYDRO model RTOUT, and GW_outflow dataframes for a model
## run of Fourmile Creek with subsurface, overland, and groundwater routing options
## turned on and return a water budget summary. The default soil depths were used
## and the basin is 63.1 km2. NOTE: We MUST specify with the sfcrt flag that overland
## flow routing was turned on. Otherwise the LSM surface runoff term is used.

wb.allrt.fc <- CalcNoahmpWatBudg(modLDASOUT1d.allrt.fc,
                                 modRTOUT1h.allrt.fc,
                                 modGWOUT1h.allrt.fc,
                                 sfcrt=TRUE, basarea=63.1)
wb.allrt.fc

## End(Not run)
```

CalcSnodasCoords *Calculate the SNODAS coordinates.*

Description

CalcSnodasCoords Calculate the SNODAS coordinates.

Usage

```
CalcSnodasCoords()
```

Value

A list of lon and lat

See Also

Other SNODAS: GetSnodasDepthSweDate; PutSnodasCoordsNcdf; PutSnodasNcdf; ReadSnodasDepthSweDate

Examples

```
snodasCoords <- CalcSnodasCoords()
```

`CalcStatsRS` *Calculates summary statistics from a remote sensing time series*

Description

`CalcStatsRS` takes a raster stack/brick of RS images and generates a time series dataframe of summary statistics across the domain for each time step.

Usage

```
CalcStatsRS(inStack)
```

Arguments

`inStack`	The name of the raster stack or brick to calculate statistics on.

Details

`CalcStatsRS` takes a raster stack or brick of remote sensing images over a time period (as created by `ConvertRS2Stack` or `SmoothStack`) and generates a dataframe object that summarizes all cells in the RS image at each time step. The statistics calculated are mean, min, max, and standard deviation. This tool is useful for evaluating how a smoothing function is impacting the time series of images.

Value

A dataframe of statistics by time period (date).

See Also

Other MODIS: `ConvertRS2Stack`; `ConvertStack2NC`; `GetMODIS`; `InsertRS`; `SmoothStack`

Examples

```
## Calculate domain statistics and plot the mean.

## Not run:
stats.lai.b <- CalcStatsRS(lai.b)
with(stats.lai.b, plot(POSIXct, mean, typ='l'))

## End(Not run)
```

ConvertRS2Stack *Convert a set of MODIS images to a raster stack and, optionally, a NetCDF file.*

Description

ConvertRS2Stack takes a set of pre-processed MODIS TIF files and creates a raster stack and, optionally, a NetCDF file.

Usage

```
ConvertRS2Stack(inPath, matchStr, begin = NULL, end = NULL, noData = NULL,
  noDataQual = "exact", valScale = 1, valAdd = 0, outFile = NULL,
  varName = NULL, varUnit = NULL, varLong = NULL, varNA = -1e+36)
```

Arguments

inPath	The path to the directory that holds the already processed MODIS TIF files.
matchStr	The regular expression for filename matching (e.g., "*Lai_1km.tif").
begin	Date string for the start date to include. The date string should follow MODIS package convention (e.g., "2011.06.01"). (DEFAULT=NULL, all files are processed).
end	Date string for the end date to include. The date string should follow MODIS package convention (e.g., "2011.06.01"). (DEFAULT=NULL, all files are processed).
noData	Value for specifying "no data" per the MODIS product. Should be combined with a qualifier. For example, for the MOD15A2 (LAI) product, valid value range is 0-100. So the noData value should be set to 100 and the noDataQual to "max". (DEFAULT=NULL, no additional "no data" screening is applied)
noDataQual	Qualifier for specifying "no data" per the MODIS product. Should be combined with a nodata value. For example, for the MOD15A2 LAI product, valid value range is 0-100. So the noData value should be set to 100 and the noDataQual to "max". Options are: "exact" (value == noData are converted to NA), "min" (value < noData are converted to NA), "max" (value > noData are converted to NA). (DEFAULT="exact", only exact matches are converted to NA)
valScale	The scale factor to apply to the image values per the MODIS product. The adjustment is applied as VALUE * valScale + valAdd. For example, for the MOD15A2 LAI product, the scale factor is 0.1. (DEFAULT = 1)
valAdd	The addition value to apply to the image values per the MODIS product. The adjustment is applied as VALUE * valScale + valAdd. (DEFAULT = 0)
outFile	OPTIONAL name for an output NetCDF file. A NetCDF file will only be created if this file name is provided. Images will be exported after the "no data" and scale adjustments are made. If you want to do smoothing or other time series processing, do not export a NetCDF file here but do processing on the raster stack and then use ConvertStack2NC to export the processed brick to a NetCDF file.
varName	Name for the NetCDF export variable. Only required if outFile is provided.
varUnit	Units for the NetCDF export variable. Only required if outFile is provided.

varLong	Long name for the NetCDF export variable. Only required if outFile is provided.
varNA	Value to set for "NA" or "no data". Default is -1.e+36. Only required if outFile is provided.

Details

ConvertRS2Stack scans the specified directory and imports pre-processed MODIS TIF files matching the specified expression and combines them into an R raster stack object and, optionally, an output NetCDF file. Files in the input directory should be already processed through the GetMODIS tool or follow the same file naming convention used by the MODIS runGdal tool. See the R MODIS package for more on those specifications.

Value

A raster stack.

See Also

Other MODIS: CalcStatsRS; ConvertStack2NC; GetMODIS; InsertRS; SmoothStack

Examples

```
## Import the already processed LAI TIF images into a raster stack. Use
## the full time series of images.

## Not run:
lai.b <- ConvertRS2Stack("/d6/adugger/WRF_Hydro/RS/MODIS_ARC/PROCESSED/BCNED_LAI",
                         "*Lai_1km.tif", noData=100, noDataQual="max",
                         valScale=0.1, valAdd=0)

## Export a subset of the already processed LAI TIF images into an output netcdf file

lai.b <- ConvertRS2Stack("/d6/adugger/WRF_Hydro/RS/MODIS_ARC/PROCESSED/BCNED_LAI",
                         "*Lai_1km.tif", begin=c("2011.06.01", end="2011.06.30",
                         noData=100, noDataQual="max", valScale=0.1, valAdd=0,
                         outFile="BCNED_LAI.nc", varName="LAI",
                         varUnit="(m^2)/(m^2)", varLong="Leaf area index")

## End(Not run)
```

ConvertStack2NC *Convert a raster stack to a NetCDF file.*

Description

ConvertStack2NC takes a raster stack of RS images and outputs a NetCDF file with a time dimension and specified variable name & metadata.

Usage

```
ConvertStack2NC(inStack, outFile = NULL, varName = NULL, varUnit = NULL,
  varLong = NULL, varNA = -1e+36)
```

Arguments

inStack	The name of the raster stack to export.
outFile	Name for an output NetCDF file.
varName	Name for the NetCDF export variable.
varUnit	Units for the NetCDF export variable.
varLong	Long name for the NetCDF export variable.
varNA	Value to set for "NA" or "no data". Default is -1.e+36.

Details

ConvertStack2NC converts a raster stack to an output NetCDF file. The raster stack should be already processed through the ConvertRS2Stack tool or follow the same layer (date) naming convention.

See Also

Other MODIS: CalcStatsRS; ConvertRS2Stack; GetMODIS; InsertRS; SmoothStack

Examples

```
## Export the raster stack of LAI images created through ConvertRS2Stack
## to a NetCDF file. Use the full time series of images.

## Not run:
ConvertStack2NC(lai.b, outFile="BCNED_LAI.nc", varName="LAI",
                varUnit="(m^2)/(m^2)", varLong="Leaf area index")

## End(Not run)
```

ExportGeogrid *Creates a georeferenced TIF from a geogrid variable*

Description

ExportGeogrid takes a NetCDF geogrid and converts the specified variable into a georeferenced TIF file.

Usage

```
ExportGeogrid(inFile, inVar, outFile)
```

Arguments

inFile	The geogrid NetCDF file.
inVar	The name of the variable to be converted.
outFile	The geoTiff filename to create.

Details

ExportGeogrid takes a standard geogrid in NetCDF format and converts the specified variable to a georeferenced geoTiff for use in standard GIS tools.

Examples

```
## Export the HGT_M field from the geogrid file geo_em.d01_1km.nc
## to a geoTiff called geogrid_hgt.tif.

## Not run:
ExportGeogrid("geo_em.d01_1km.nc", "HGT_M", "geogrid_hgt.tif")

## End(Not run)
```

`FindUsgsStns` *Discover USGS stations using huc8 code or lat/lon/radius.*

Description

`FindUsgsStns` wraps `dataRetrieval::whatNWISsites` with common options for finding USGS gauges. See the dataRetrieval package for more deails on what else could be passed or how this could be modified. One improvement here would be to order results by proximity to supplied lat/lon and filter to nClosest.

Usage

```
FindUsgsStns(stnLon = NULL, stnLat = NULL, within = NULL, huc8 = NULL,
  siteType = "ST", hasDataTypeCd = "iv")
```

Arguments

`stnLon`	optional
`stnLat`	optional
`within`	optional, goes with stnLon and stnLat and specifies a search radius in decimal degrees.
`huc8`	optional EIGHT digit HUC code.
`siteType`	the type of USGS site to look for.
`hasDataTypeCd`	the kind of data of interest (iv=instantaneous, dv=daily, etc.)

See Also

Other streamObs: `GetSiteHuc`; `GetUsgsHucData`; `GetUsgsIvProduct`; `ImproveHucMeta`; `LoadMetaDB`; `PlotPrettyUsgs`; `PrettyUsgs`; `QueryHaveSite`; `QuerySiteData`; `QuerySiteInfo`; `QuerySiteName`; `QuerySiteProd`; `SaveHucData`; `[.prettyUsgs`; `subset.prettyUsgs`

Examples

```
stnDf <- FindUsgsStns(huc8='10190005')
stnDf <- FindUsgsStns(huc8=c('10190005','03160203'))
stnDf <- FindUsgsStns(stnLon=254.67374999999998408,stnLat=40.018666670000001773,within=.001)
stnDf <- FindUsgsStns(stnLon=c(254.67374999999998408,-87.747224700000004),
                      stnLat=c(40.018666670000001773, 31.864042489999999),within=.001)
```

`gages2Attr` *Gages-II attributes*

Description

These are the attributes taken from the USGS Gages-II data set.
http://water.usgs.gov/GIS/metadata/usgswrd/XML/gagesII_Sept2011.xml
The original gages-II atributes are in `gages2Attr` additional attributes may be placed in `gages2AttrPlus`.

Usage

```
gages2Attr
```

Format

data.frame: 9322 obs. of 16 variables:

STAID Station ID from USGS NWIS

STANAME Station Name from USGS NWIS

CLASS Classification (Ref or Non-ref)

AGGECOREGI Aggregated ecoregion

DRAIN_SQKM Drainage area, sq km

HUC02 Hydrologic Unit Code, 2-digit

HUC08 ONLY in gages2AttrPlus: Hydrologic Unit Code, 8-digit

LAT_GAGE Latitude, decimal degrees

LNG_GAGE Longitude, decimal degrees

STATE State at gage location

HCDN_2009 If gage is part of HCDN-2009

ACTIVE09 If gage active in water year 2009

FLYRS1900 Number of complete years of flow data between 1900 and 2009

FLYRS1950 Number of complete years of flow data between 1950 and 2009

FLYRS1990 Number of complete years of flow data between 1990 and 2009

Citation

Originator: James Falcone
Publication_Date: 20111012
Title: GAGES-II: Geospatial Attributes of Gages for Evaluating Streamflow
Geospatial_Data_Presentation_Form: vector digital data Publication_Information:
Publication_Place: Reston, Virginia
Publisher: U.S. Geological Survey
Online_Linkage: http://water.usgs.gov/lookup/getspatial?gagesII_Sept2011

See Also

Other gages2: `gages2AttrPlus`

`gages2AttrPlus` *Gages-II attributes*

Description

These are the attributes taken from the USGS Gages-II data set.
http://water.usgs.gov/GIS/metadata/usgswrd/XML/gagesII_Sept2011.xml
The original gages-II atributes are in `gages2Attr` additional attributes may be placed in `gages2AttrPlus`.

Usage

```
gages2AttrPlus
```

Format

data.frame: 9322 obs. of 16 variables:

STAID Station ID from USGS NWIS

STANAME Station Name from USGS NWIS

CLASS Classification (Ref or Non-ref)

AGGECOREGI Aggregated ecoregion

DRAIN_SQKM Drainage area, sq km

HUC02 Hydrologic Unit Code, 2-digit

HUC08 ONLY in gages2AttrPlus: Hydrologic Unit Code, 8-digit

LAT_GAGE Latitude, decimal degrees

LNG_GAGE Longitude, decimal degrees

STATE State at gage location

HCDN_2009 If gage is part of HCDN-2009

ACTIVE09 If gage active in water year 2009

FLYRS1900 Number of complete years of flow data between 1900 and 2009

FLYRS1950 Number of complete years of flow data between 1950 and 2009

FLYRS1990 Number of complete years of flow data between 1990 and 2009

Citation

Originator: James Falcone
Publication_Date: 20111012
Title: GAGES-II: Geospatial Attributes of Gages for Evaluating Streamflow
Geospatial_Data_Presentation_Form: vector digital data Publication_Information:
Publication_Place: Reston, Virginia
Publisher: U.S. Geological Survey
Online_Linkage: http://water.usgs.gov/lookup/getspatial?gagesII_Sept2011

See Also

Other gages2: `gages2Attr`

GetMODIS *Get MODIS data and process to match geogrid*

Description

GetMODIS downloads, mosaics, and resamples MODIS data to match input geogrid.

Usage

```
GetMODIS(geogrdPath, prodName, outDir, begin, end, resampTyp = "near")
```

Arguments

geogrdPath	The pathname to the geogrid file (i.e., geo_em.d01.nc).
prodName	The MODIS product name to download/process. Run the MODIS package getProducts() for a complete list of supported products.
outDir	Directory name to store processed TIF files. This is the equivalent to the MODIS package's "job" name. This is a directory name only and NOT a full path. The directory will be created in the preset MODIS package outDirPath.
begin	Date string for the start date to download/process MODIS tiles. The date string should follow MODIS package convention (e.g., "2011.06.01").
end	Date string for the end date to download/process MODIS tiles. The date string should follow MODIS package convention (e.g., "2011.06.01").
resampTyp	Resampling type. GDAL possible options (depends on version): 'near', 'bilinear', 'cubic', 'cubicspline', 'lanczos', 'mode', 'average' (DEFAULT='near'). Note that MODIS nodata values MAY look like real values to GDAL, so use caution with any methods other than 'near' and 'mode'. Fix in future version.

Details

GetMODIS reads a geogrid file and parameters on MODIS product and data range and downloads MODIS tiles where they do not exist (using the R MODIS package), mosaics where necessary, clips, reprojects, and resamples (using nearest neighbor) to match the geogrid. Results in a set of TIF files per the MODIS package specifications.

Please see documentation on the R MODIS package for details on required installs and workspace setup. This tool builds off of the MODIS runGdal tool, so follows MODIS file directory structure (see MODISoptions). This tool requires a local GDAL installation in addition to the required R packages.

NOTE: This tool currently only works for geogrid files in Lambert Conformal Conic projection.

Value

Empty

See Also

Other MODIS: CalcStatsRS; ConvertRS2Stack; ConvertStack2NC; InsertRS; SmoothStack

Examples

```
## First, specify the target directories for the MODIS package.
## Not run:
MODISoptions(localArcPath="/d1/WRF_Hydro/RS/MODIS_ARC",
             outDirPath="/d1/WRF_Hydro/RS/MODIS_ARC/PROCESSED")

## Then, use GetMODIS to download the MODIS MOD15A2 (FPAR/LAI) product for
## all tiles that overlap our Fourmile Creek domain (as specified by the
## geo_em.d01.nc file), mosaic if multiple tiles, clip to the domain extent,
## and resample to the domain grid cells. The raw HDF files will be stored in
## /d1/WRF_Hydro/RS/MODIS_ARC/ and the final processed TIF files will be
## stored in /d1/WRF_Hydro/RS/MODIS_ARC/PROCESSED/Fourmile_LAI/.

GetMODIS(geogrdPath="/d1/WRF_Hydro/Fourmile_fire/DOMAIN/geo_em.d01.nc",
         prodName="MOD15A2", outDir="Fourmile_LAI",
         begin="2011.01.01", end="2011.01.31")

## End(Not run)
```

`GetMultiNcdf` *Get WRF Hydro output/restart (scalar) timeseries spread over multiple files.*

Description

`GetMultiNcdf` is designed to get *all* your output/restart data which are spread over multiple files. Three collated lists specify 1) file groups, 2) variables for each file group, and 3) indices or statistics for each variable in each file group. The names of the lists must match. See examples for details. While the routine can read and summarize raster data at each time via specificied statistics, it only returns scalar timeseries. (It may be possible to extend to return both scalar and raster data if there's demand.)

Usage

```
GetMultiNcdf(filesList, variableList, indexList, env = parent.frame(),
  parallel = FALSE)
```

Arguments

`filesList`	The list of file groups. Names must match those in the other lists.
`variableList`	The list of variables for each file group. Names must match filesList.
`indexList`	The list of indices or statistics to be applied to each variable.
`env`	The environment where the stat function lives
`parallel`	Logical, this is the .parallel argument of plyr functions. Parallelization is at the file level (not file group).Typcially we achieve parallelization using the DoMC package. See examples.

Value

A dataframe (in an awesome format).

Examples

```
# This example only shows data for 3 dates, because of limitation of package data.
# Find the package data directory on your machine
## Not run:
tcPath <- '~/wrfHydroTestCases/'
fcPath <- paste0(tcPath,'Fourmile_Creek/')
dataPath <- paste0(fcPath,'/RUN.RTTESTS/OUTPUT_CHRT_DAILY/')
fileList - These are the groups of files.
lsmFiles <- list.files(path=dataPath, pattern='LDASOUT_DOMAIN', full.names=TRUE)
hydroFiles <- list.files(path=dataPath, pattern='HYDRO_RST', full.names=TRUE)
fileList <- list( lsm=lsmFiles, hydro=hydroFiles)

# varList - Define which variables are desired for each file group.
lsmVars   <- list(TRAD='TRAD', SWE='SNEQV')
## smc1-4 will correspond to the vertical layers.
hydroVars <- list(streamflow='qlink1', smc1='sh2ox', smc2='sh2ox',
                  smc3='sh2ox', smc4='sh2ox')
# Note that the outer names collate with fileList.
variableList <- list(lsm=lsmVars, hydro=hydroVars)

# indexList - Define what indices/stats are desired for each variable.
# Note that only scalars can be returned for each entry. Spatial fields can
# be summarized via statistics.
# Show how to define your own useful stats to use.
# For basin average and max we need the basin mask (this is a non-standard
# field in the fine grid file).
basinMask <- ncdump(paste0(fcPath,'DOMAIN/hydro_OrodellBasin_100m.nc'),
                    'basn_msk_geogrid')
nc_close(fineGridNc)
basAvg <- function(var) sum(basinMask*var)/sum(basinMask)
basMax <- function(var) max(ceiling(basinMask)*var)
basinKm2 <- sum(basinMask)  ## just asking about the total area of the basin.

# Note that the list names at this level collate with the variable names
# in VarList. You are responsible for entering the correct indices. Note
# that these are reverse order from what is shown in "ncdump -h".
lsmInds   <- list(TRAD=list(start=c(1,1,1), end=c(21,7,1), stat='basAvg'),
                  SNEQV=list(start=c(1,1,1), end=c(21,7,1), stat='basMax'))
hydroInds <- list(qlink1=1,
                  smc1=list(start=c(1,1,1), end=c(21,7,1), stat='basAvg'),
                  smc2=list(start=c(1,1,2), end=c(21,7,2), stat='basAvg'),
                  smc3=list(start=c(1,1,3), end=c(21,7,3), stat='basAvg'),
                  smc4=list(start=c(1,1,4), end=c(21,7,4), stat='basAvg') )
indexList <- list( lsm=lsmInds, hydro=hydroInds)

library(doMC)   ## Showing parallelization, which is at the file level within
registerDoMC(3) ## each file grous; pointless to be longer than your timeseries.
fileData <- GetMultiNcdf(file=fileList,var=variableList, ind=indexList,
                         parallel=TRUE)

# plot
# the lsm and hyro output for this spinup were at different times...
library(ggplot2)
library(scales)
ggplot( fileData, aes(x=POSIXct, y=value, color=fileGroup)) +
  geom_line() + geom_point() +
```

```
  facet_wrap(~variableGroup, scales='free_y', ncol=1) +
  scale_x_datetime(breaks = date_breaks("5 days")) + theme_bw()

## End(Not run)
```

GetPkgMeta *Get a package's metadata fields and associated entries.*

Description

GetPkgMeta Get metadata fields and associated entries from a package's documentation (e.g. "keyword" or "concepts".)

Usage

```
GetPkgMeta(meta = c("concept", "keyword"), package = "rwrfhydro",
  keyword = NULL, concept = NULL, quiet = FALSE, listMetaOnly = FALSE,
  byFunction = "")
```

Arguments

meta	Character the metadata field.
package	Character The package to query for metadata.
keyword	Character A specific keyword to look for.
concept	Character A specific concept to look for.
quiet	Logical Do not print summary to screen.
listMetaOnly	Logical Just return the meta categories (without functons)?
byFunction	Character Vector of functions for which concepts and keywords are desired.

Value

List of metadata fields in alphabetical order with corresponding entries.

Examples

```
GetPkgMeta()
GetPkgMeta('keyword', package='ggplot2')
GetPkgMeta(concept = 'foo', key='hplot' )
print( GetPkgMeta(concept = c('dataMgmt','foo'), key=c('foo','hplot') , quiet=TRUE))
str( GetPkgMeta(concept = c('dataMgmt','foo'), key=c('foo','hplot') , quiet=TRUE))
str( GetPkgMeta(concept = c('dataMgmt','DART'), key=c('internal','hplot') , quiet=TRUE))
GetPkgMeta(byFunction=c('MkDischargeVariance','SaveHucData'))
```

GetSiteHuc *Given a USGS site code, return its HUC8.*

Description

GetSiteHuc returns a HUC8 given a USGS site name.

Usage

```
GetSiteHuc(site)
```

Arguments

site	Character USGS site number.

Value

character HUC8

See Also

Other streamObs: FindUsgsStns; GetUsgsHucData; GetUsgsIvProduct; ImproveHucMeta; LoadMetaDB; PlotPrettyUsgs; PrettyUsgs; QueryHaveSite; QuerySiteData; QuerySiteInfo; QuerySiteName; QuerySiteProd; SaveHucData; [.prettyUsgs; subset.prettyUsgs

Examples

```
GetSiteHuc(FindUsgsStns(stnLon=254.67374999999998408,
                        stnLat=40.018666670000001773,
                        within=.001)$site_no)
huc <- GetSiteHuc(gages2AttrPlus$STAID[1:3])
names(huc)<-gages2AttrPlus$STAID[1:3]
```

GetSnodasDepthSweDate *Get and unpack the SNODAS snow depth and SWE tarball for given dates.*

Description

GetSnodasDepthSweDate Get and unpack the SNODAS snow depth and SWE tarball for a given date.

Usage

```
GetSnodasDepthSweDate(datePOSIXct, outputDir = ".", overwrite = FALSE,
  quiet = TRUE, parallel = (foreach::getDoParWorkers() > 1))
```

Arguments

datePOSIXct	The date in POSIXct format for which data is desired. Only whole days matter.
outputDir	The directory where the data are to be archived. This directory is checked to see if the data exist when overwrite=FALSE
overwrite	When false: If the depth and SWE files exist on disk, dont grab the tarball. When false: If the tarball exists on disk but depth and SWE files dont, just unpack the tarball. When true: Pull new tarball and overwrite any existing files with the same date.
quiet	Passed to curl, to show it's progress (typcially too fast to matter).
parallel	Logical Defaults to (foreach::getDoParWorkers>1), so if you've set up parallelization it is automatically used.

Value

Logical was the file "got"?

See Also

Other SNODAS: CalcSnodasCoords; PutSnodasCoordsNcdf; PutSnodasNcdf; ReadSnodasDepthSweDate

Examples

```
snodasGot <- GetSnodasDepthSweDate(as.POSIXct('2015-02-28'),
                                   outputDir = path.expand('~'))
```

GetUsgsHucData *Get all the USGS streamgage information within a HUC8.*

Description

GetUsgsHucData gets all the USGS streamgage information within a HUC8. If an output path is supplied, an existing database is examined for existing records and the data retrieved from the USGS only extends the available and is saved in the data base. Could eventually use startDate and endDate to tweak this for calls which dont care about archival.

Usage

```
GetUsgsHucData(huc8, outPath = NULL,
  metaDBFileName = "usgsDataRetrieval.metaDatabase.RData", update = FALSE)
```

Arguments

huc8	Character The eight-digit HUC code.
outPath	Character The path to the database directory
metaDBFileName	Character The name of the database metadata file.
update	Logical, If huc exists on disk, update it byt getting new data from NWIS?

See Also

Other streamObs: FindUsgsStns; GetSiteHuc; GetUsgsIvProduct; ImproveHucMeta; LoadMetaDB; PlotPrettyUsgs; PrettyUsgs; QueryHaveSite; QuerySiteData; QuerySiteInfo; QuerySiteName; QuerySiteProd; SaveHucData; [.prettyUsgs; subset.prettyUsgs

Examples

```
## Not run:
bldrHucData    <- GetUsgsHucData(huc8='10190005')
satilpaHucData <- GetUsgsHucData(huc8='03160203')
vansHucData    <- GetUsgsHucData(huc8='03020104')

## End(Not run)
```

InsertRS *Inserts pre-processed images into appropriate forcing NetCDF files by date.*

Description

InsertRS takes a raster stack or brick of RS images and exports individual images to matching (by date) forcing NetCDF files.

Usage

```
InsertRS(inFile, forcPath, forcName = "LDASIN_DOMAIN1", varName = NULL,
  varUnit = NULL, varLong = NULL, varNA = -1e+36, overwrite = TRUE)
```

Arguments

inFile	The name of the raster stack/brick or NetCDF file (full pathname) to export.
forcPath	Path to the forcing data you want to modify. Forcing data files MUST match the size/resolution of the images in the inFile.
forcName	The suffix for the forcing data files to modify (DEFAULT="LDASIN_DOMAIN1")
varName	Name for the NetCDF variable to export. The varibale will be copied as-is, so make sure it matches the variable name needed in the forcing data.
varUnit	Units for the NetCDF export variable. Only required if the inFile is a raster stack/brick. If the inFile is a NetCDF file, the units will carry over.
varLong	Long name for the NetCDF export variable. Only required if the inFile is a raster stack/brick. If the inFile is a NetCDF file, the longname will carry over.
varNA	Value to set for "NA" or "no data". Default is -1.e+36.
overwrite	Boolean to allow the tool to overwrite existing variables if found in the forcing data. (DEFAULT=TRUE)

Details

InsertRS takes a raster stack or brick (as created by ConvertRS2Stack or SmoothStack) or a NetCDF file (as created by ConvertStack2NC) of RS images and exports each layer (time step) to the appropriate time step forcing file. Only looks for the date (not time) and inserts at the 00:00 hour on that date. The input stack, brick, or file should be already processed through the ConvertRS2Stack, SmoothStack, or ConvertStack2NC tools or follow the same layer (date) naming convention.

See Also

Other MODIS: CalcStatsRS; ConvertRS2Stack; ConvertStack2NC; GetMODIS; SmoothStack

Examples

```
## Export the raster stack of LAI images created through ConvertRS2Stack to
## the forcing data.

## Not run:
InsertRS(lai.b, forcPath="FORCING", forcName="LDASIN_DOMAIN3",
         varName="LAI", varUnit="(m^2)/(m^2)", varLong="Leaf area index")

## Export the NetCDF of LAI images created through ConvertStack2NC to the
## forcing data.

InsertRS("BCNED_LAI.nc", forcPath="FORCING", forcName="LDASIN_DOMAIN3",
         varName="LAI")

## End(Not run)
```

MkNcdf *List based creation of netcdf files.*

Description

MkNcdf List based creation of netCDF files. Works for most cases. Creates netcdf version 4 files by default. Appends to unlimited dimensions in existing files.

Usage

```
MkNcdf(varList, filename, globalAttList = NULL, overwrite = FALSE,
  force_v4 = TRUE)
```

Arguments

varList	List, the variable list. See structure in examples.
filename	Character, the name of the netcdf file to be created.
globalAttList	List, the global attribute list. See structure in examples.
overwrite	Logical, overwrite (i.e. clobber in netcdf parlance) existing file? Otherwise, the file will be appended if existing and correct (unlimited) dims are extended.
force_v4	Logical, make netcdf version 4 files.

Value

Returns the filename invisibly, if successful.

See Also

Other ncdf: ncdump

Examples

```
# Example 1 - Basic write.
varList = list()
varList[[1]] <- list( name='precipMult',
                      longname='Precipitation Multiplier',
                      units='-',
                      precision = 'double',
                      missing = -9999,
                      dimensionList = list(scalar=list(name='scalar',values=1,
                                           units='-', unlimited=FALSE,
                                           create_dimvar=FALSE)),
                      data = 1:1 )
varList[[2]] <- list( name='precipMult22',
                      longname='Precipitation Multiplier22',
                      units='-',
                      precision = 'double',
                      missing = -9999,
                      dimensionList =
                          list(  # n.b. the dimension order: charlen,z,y,x,t
                                y=list(name='y',values=2.5+((0:6)*5),
                                       units='lat', unlimited=FALSE,
                                       create_dimvar=TRUE),
                                x=list(name='x',values=c(10,20),
                                       units='lon', unlimited=FALSE,
                                       create_dimvar=TRUE)
                               ),
                     data = matrix( 1:7, nrow=7, ncol=2 ) )
globalAttList <- list()
globalAttList[[1]] <- list(name='Restart_Time',value='2012-07-05_00:00:00', precision="text")
globalAttList[[2]] <- list(name='Some reall atts',value='#$%^!!', precision="text" )

outFile1 <- path.expand('~/test1.nc')
dum <- MkNcdf( varList, globalAttList=globalAttList, filename=outFile1 )
ncdump(outFile1)
unlink(outFile1)

#Example 2 - append to an existing file's variable.
varList1 = list()
varList1[[1]] <- list( name='precipMult22',
                      longname='Precipitation Multiplier22',
                      units='-',
                      precision = 'double',
                      missing = -9999,
                      dimensionList =
                          list(  # n.b. the dimension order: charlen,z,y,x,t
                                y=list(name='y',values=2.5+((0:6)*5),
                                       units='lat', unlimited=TRUE,
                                       create_dimvar=TRUE),
                                x=list(name='x',values=c(10,20),
                                       units='lon', unlimited=FALSE,
                                       create_dimvar=TRUE)
                               ),
                     data = matrix( 1:7, nrow=7, ncol=2 ) )

varList2 = list()
varList2[[1]] <- list( name='precipMult22',
```

```
                        longname='Precipitation Multiplier22',
                        units='-',
                        precision = 'double',
                        missing = -9999,
                        dimensionList =
                            list(  # n.b. the dimension order: charlen,z,y,x,t
                                  y=list(name='y',values=99,
                                         units='lat', unlimited=TRUE,
                                         create_dimvar=TRUE),
                                  x=list(name='x',values=c(10,20),
                                         units='lon', unlimited=FALSE,
                                         create_dimvar=TRUE)
                                 ),
                       data =  c(1:2)+10 )

globalAttList <- list()
globalAttList[[1]] <- list(name='Some reall atts',value='#$%^!!', precision="text" )
outFile2 <- path.expand('~/test2.nc')
MkNcdf( varList1, globalAttList=globalAttList,
        filename=outFile2)
ncdump(outFile2)
MkNcdf( varList2, globalAttList=globalAttList,
        filename=outFile2 )
ncdump(outFile2)
unlink(outFile2)
```

NamedList *Create and or name a list with its entries.*

Description

NamedList creates a list with names equal to its entries.

Usage

```
NamedList(theNames)
```

Arguments

theNames	Vector to be coerced to character.

Value

List with names equal to entries.

Examples

```
NamedList(1:5)
```

ncdump — *Emulate ncdump -h on OSX where ncdump might not be availabe.*

Description

I just hacked print.ncdf4 just to make it look more like unix output.

Usage

```
ncdump(file, variable, quiet = FALSE)
```

Arguments

file	Character, the file to inspect.
variable	Character, a variable to return.
quiet	Logical, suppress the 'meta' dump?

Value

If variable is not set, the meta object `ncdf4::nc_open(file)` is returned. If `variable` is set, its values are returned.

See Also

Other ncdf: MkNcdf

Examples

```
# See \code{\link{MkNcdf}}.
```

PlotFdc — *Plot a flow duration curve for a single streamflow time series*

Description

PlotFdc plots a flow duration curve from a dataframe processed through CalcFdc and optional spline function processed through CalcFdcSpline.

Usage

```
PlotFdc(strDf, strCol = "q_cms", spline = TRUE, fdcProb = NULL)
```

Arguments

strDf	The streamflow time series dataframe (e.g., output from ReadFrxstPts or ReadUsgsGage) already processed through CalcFdc. The data frame must contain a column of streamflow values and a corresponding column (named "<strCol>.fdc") of flow exceedance probabilities.
strCol	The name of the column containing the streamflow values (DEFAULT="q_cms").
spline	(TRUE or FALSE) Option to add spline-fit curves to the plot (DEFAULT=TRUE).
fdcProb	Optional exceedance threshold to calculate/plot (e.g., 0.2 for 20% exceedance).

Details

PlotFdc reads the dataframe generated by CalcFdc and outputs a plot of flow vs. percent exceedances and, optionally the fitted spline curve.

Value

A plot of flow (y) vs. flow exceedance probabilities (x).

See Also

Other flowDurationCurves: CalcFdcSpline; CalcFdc; PlotFdcCompare

Examples

```
## Take a time series of observed 5-minute streamflow values for Fourmile
## Creek (stored in the dataframe obsStr5min.fc in the column "q_cms") that
## has already been processed through "CalcFdc" (so also contains a column
## named "q_cms.fdc") and plot with the 20% exceedance threshold called out.
## Not run:
PlotFdc(obsStr5min.fc, fdcProb=0.2)

## End(Not run)
```

PlotFdcCompare — *Plots a flow duration curve for up to three streamflow time series (e.g., observed & two model outputs)*

Description

PlotFdcCompare plots up to three flow duration curves for comparison.

Usage

```
PlotFdcCompare(strDf.obs, strCol.obs = "q_cms", strDf.mod1,
  strCol.mod1 = "q_cms", strDf.mod2 = NULL, strCol.mod2 = "q_cms",
  spline = TRUE, logy = TRUE, labelObs = "Observed",
  labelMod1 = "Model 1", labelMod2 = "Model 2")
```

Arguments

strDf.obs	The OBSERVED streamflow time series dataframe (e.g., output from ReadUsgsGage). The dataframe must contain a column of streamflow values and a POSIXct column.
strCol.obs	The name of the column containing the streamflow values for the OBSERVED dataframe (DEFAULT="q_cms").
strDf.mod1	The FIRST MODEL streamflow time series dataframe (e.g., output from ReadFrxstPts). The dataframe must contain a column of streamflow values and a POSIXct column.
strCol.mod1	The name of the column containing the FIRST MODEL streamflow values (DEFAULT="q_cms").

strDf.mod2	The SECOND MODEL streamflow time series dataframe (e.g., output from ReadFrxstPts). The dataframe must contain a column of streamflow values and a POSIXct column.
strCol.mod2	The name of the column containing the SECOND MODEL streamflow values (DEFAULT="q_cms").
spline	(TRUE or FALSE) Option to add spline-fit curves to the plot (DEFAULT=TRUE).
logy	(TRUE or FALSE) Optional flag to set the y-axis to log-scale (DEFAULT=TRUE).
labelObs	Optional label for the observed streamflow (DEFAULT="Observed")
labelMod1	Optional label for the FIRST MODEL (DEFAULT="Model 1")
labelMod2	Optional label for the SECOND MODEL (DEFAULT="Model 2")

Details

PlotFdcCompare reads up to three dataframes (e.g., generated by ReadFrxstPts or ReadUsgsGage) and outputs a plot of flow vs. percent exceedances and, optionally, the fitted spline curves. Intended to plot one observed flow duration curve and one or two modelled flow duration curves for comparison. The tool will subset data to matching time periods (e.g., if the observed data is at 5-min increments and modelled data is at 1-hr increments, the tool will subset the observed data to select only observations on the matching hour break). Therefore, the tool will RECALCULATE the flow exceedance probabilities on the fly.

Value

A plot of flow (y) vs. flow exceedance probabilities (x).

See Also

Other flowDurationCurves: CalcFdcSpline; CalcFdc; PlotFdc

Examples

```
## Take a time series of observed 5-minute streamflow values for Fourmile
## Creek (obsStr5min.fc) and two hourly model runs (modStrh.allrt.fc,
## modStrh.chrt.fc), all with streamflow columns named "q_cms", and plot
## the flow duration curves for all three.


## Not run:
PlotFdcCompare(obsStr5min.fc, "q_cms", modStrh.allrt.fc, "q_cms",
               strDf.mod2=modStrh.chrt.fc, strCol.mod2="q_cms",
               labelObs="Observed Fourmile Creek",
               labelMod1="All Routing", labelMod2="Channel Routing Only")

## End(Not run)
```

PlotFluxCompare *Plot time series comparing modeled and observed fluxes*

Description

PlotFluxCompare plots a time series of an observed flux (e.g., streamflow, ET) and up to 2 modelled fluxes.

Usage

```
PlotFluxCompare(strDf.obs, strCol.obs = "q_cms", strDf.mod1,
  strCol.mod1 = "q_cms", strDf.mod2 = NULL, strCol.mod2 = "q_cms",
  stdate = NULL, enddate = NULL, logy = FALSE, labelObs = "Observed",
  labelMod1 = "Model 1", labelMod2 = "Model 2",
  title = "Observed and Modelled Fluxes")
```

Arguments

strDf.obs	The OBSERVED flux time series dataframe (e.g., output from ReadUsgsGage). The dataframe must contain a column of flux values and a POSIXct column.
strCol.obs	The name of the column containing the flux values for the OBSERVED dataframe (DEFAULT="q_cms").
strDf.mod1	The FIRST MODEL flux time series dataframe (e.g., output from ReadFrxstPts). The dataframe must contain a column of flux values and a POSIXct column.
strCol.mod1	The name of the column containing the FIRST MODEL flux values (DEFAULT="q_cms").
strDf.mod2	The SECOND MODEL flux time series dataframe (e.g., output from ReadFrxstPts). The dataframe must contain a column of flux values and a POSIXct column.
strCol.mod2	The name of the column containing the SECOND MODEL flux values (DEFAULT="q_cms").
stdate	Start date for plot/statistics (DEFAULT=NULL, all records will be used). Date MUST be specified in POSIXct format with appropriate timezone (e.g., as.POSIXct("2013-05-01 00:00:00", format="%Y-%m-%d %H:%M:%S", tz="UTC"))
enddate	End date for plot/statistics (DEFAULT=NULL, all records will be used). Date MUST be specified in POSIXct format with appropriate timezone (e.g., as.POSIXct("2013-05-01 00:00:00", format="%Y-%m-%d %H:%M:%S", tz="UTC"))
logy	(TRUE or FALSE) Optional flag to set the y-axis to log-scale (DEFAULT=FALSE).
labelObs	Optional label for the observed streamflow (DEFAULT="Observed")
labelMod1	Optional label for the FIRST MODEL (DEFAULT="Model 1")
labelMod2	Optional label for the SECOND MODEL (DEFAULT="Model 2")
title	Optional for the plot (DEFAULT="Observed and Modelled Fluxes")

Details

PlotFluxCompare reads modelled and observed dataframes (e.g., as generated from ReadFrxstPts and ReadUsgsGage) and plot the time series and summary statistics. The tool will subset data to matching time periods (e.g., if the observed data is at 5-min increments and modelled data is at 1-hr increments, the tool will subset the observed data to select only observations on the matching hour break).

Value

A plot of the hydrographs.

Examples

```
## Take a time series of observed 5-minute streamflow values for Fourmile
## Creek (obsStr5min.fc) and two model runs (mod1Str1h.fc, mod2Str1h.fc),
## all with streamflow columns named "q_cms", and plot the the hydrographs
## for all three over the May-June snowmelt period.

## Not run:
PlotFluxCompare(obsStr5min.fc, "q_cms", modStrh.chrt.fc, "q_cms",
                strDf.mod2=modStrh.allrt.fc, strCol.mod2="q_cms",
                labelObs="Observed Fourmile Creek at Orodell",
                labelMod1="Channel Routing Only", labelMod2="All Routing",
                title="Streamflow: Fourmile Creek",
                stdate=as.POSIXct("2013-05-01 00:00:00",
                                  format="%Y-%m-%d %H:%M:%S", tz="UTC"),
                enddate=as.POSIXct("2013-06-30 00:00:00",
                                   format="%Y-%m-%d %H:%M:%S", tz="UTC"))

## End(Not run)
```

`PlotPrettyUsgs` *Plot USGS site data which has been prettied with PrettyUsgs.*

Description

`PlotPrettyUsgs` plots USGS site data which has been prettied with `PrettyUsgs`.

Usage

```
PlotPrettyUsgs(prettyUsgs, plot = TRUE, errInnerQntl = 0.995)
```

Arguments

`prettyUsgs`	dataframe returned from PrettyUsgs
`plot`	Logical to plot before returning or not.
`errInnerQntl`	The inner quantile of the error estimate to be plotted with error bars. For example, the "68-95-99.7 rule" where specifying these as the errInnerQntl would display 1, 2, and 3 standard deviations, respectively.

Value

A function(closure) with arguments controlling the look of its graphical output. It's actual return value is a list of ggplot2 object which can be custom manipulated.

See Also

Other streamObs: `FindUsgsStns`; `GetSiteHuc`; `GetUsgsHucData`; `GetUsgsIvProduct`; `ImproveHucMeta`; `LoadMetaDB`; `PrettyUsgs`; `QueryHaveSite`; `QuerySiteData`; `QuerySiteInfo`; `QuerySiteName`; `QuerySiteProd`; `SaveHucData`; `[.prettyUsgs`; `subset.prettyUsgs`

Examples

```
# See vignette "Collect USGS stream observations and build a local database" for examples.
```

PlotWatBudg *Plot water balance from WRF-Hydro (w/NoahMP) output*

Description

PlotWatBudg plot water budget components from WRF-Hydro (w/NoahMP) model output.

Usage

```
PlotWatBudg(wbDf, plottyp = "pie")
```

Arguments

wbDf	The water budget dataframe (required)
plottyp	The plot type (pie or bar) (default=pie)

Details

Read water budget dataframe (as generated from CalcNoahmpWatBudg) and plot water budget components as a piechart or barchart. NOTE: Currently only works for runs using NoahMP as the LSM.

Value

A plot of the water budget components in mm.

Examples

```
## Plot the water budget components from a water budget dataframe generated
## using CalcNoahmpWatBudg. Plot as a piechart.

## Not run:
PlotWatBudg(wb.allrt.fc)

## Plot the same as a barchart.

PlotWatBudg(wb.allrt.fc, "bar")

## End(Not run)
```

PrettyUsgs *PrettyUsgs constructs the S3 class prettyUsgs.*

Description

PrettyUsgs beautifies data frames of USGS data. This is the S3 constructor for the PrettyUsgs object.

Usage

```
PrettyUsgs(data, tz = "UTC", metric = TRUE, na.rm = TRUE)
```

Arguments

data	Dataframe from QuerySiteData
tz	Character The timezone for the POSIXct dataTime variable to be returned.
metric	Logical. Units are either metric or not (not both).
na.rm	Logical Remove all missing observations?

Value

dataframe similar to input with improved names and/or metric variables.

See Also

Other streamObs: FindUsgsStns; GetSiteHuc; GetUsgsHucData; GetUsgsIvProduct; ImproveHucMeta; LoadMetaDB; PlotPrettyUsgs; QueryHaveSite; QuerySiteData; QuerySiteInfo; QuerySiteName; QuerySiteProd; SaveHucData; [.prettyUsgs; subset.prettyUsgs

Examples

```
## Not run:
p='~/wrfHydroTestCases/usgsDb/'
dataOrodell <- PrettyUsgs(QuerySiteData(QuerySiteName("FOURMILE CREEK AT ORODELL, CO", p),
                                        '00060', p), metric=TRUE)
## multisite and multiproduct case
dataMultiHuc <- PrettyUsgs(QuerySiteData(c('06730500','02470072'),c('00065','00060'),p))

## End(Not run)
```

PutSnodasCoordsNcdf *Put the SNODAS coordinates into a netcdf file.*

Description

PutSnodasCoordsNcdf Put the output of CalcSnodasCoords into a netcdf file.

Usage

```
PutSnodasCoordsNcdf()
```

Value

Success if the filename (which is SNODAS_Coordinates.nc), otherwise NULL.

See Also

Other SNODAS: CalcSnodasCoords; GetSnodasDepthSweDate; PutSnodasNcdf; ReadSnodasDepthSweDate

Examples

```
outFile <- PutSnodasCoordsNcdf()
ncdump(outFile)
unlink(outFile)
```

PutSnodasNcdf *Write output of ReadSnodasDepthSweDate to netcdf.*

Description

PutSnodasNcdf Put the output of ReadSnodasDepthSweDate into a netcdf file.

Usage

```
PutSnodasNcdf(snodasList, outputDir = ".")
```

Arguments

snodasList	The output of ReadSnodasDepthSweDate.
outputDir	Character. The directory path where the output files are to be written.

Value

Success if the filename which is (SNODAS_YYYYMMDD.nc), otherwise NULL.

See Also

Other SNODAS: CalcSnodasCoords; GetSnodasDepthSweDate; PutSnodasCoordsNcdf; ReadSnodasDepthSweDate

Examples

```
## Not run:
snodasGot <- GetSnodasDepthSweDate(as.POSIXct('2015-02-28'))
if(snodasGot) snodasList <- ReadSnodasDepthSweDate(as.POSIXct('2015-02-28'))
PutSnodasNcdf(snodasList)

## End(Not run)
```

QueryHaveSite *General purpose query/get for instantaneous USGS streamflow data.*

Description

QueryHaveSite

Usage

```
QueryHaveSite(site, path = ".",
  metaDBFileName = "usgsDataRetrieval.metaDatabase.RData", get = FALSE,
  retData = FALSE)
```

Arguments

site	Character USGS site number.
path	Character The path to the database.
metaDBFileName	Character The name of the metadata file.
get	Logical Get the data from NWIS if it is not local? Data are saved to local database.
retData	Logical OR Character If true return all products, otherwise return the specified product. If get from NWIS, all products are retrieved and saved locally but only specified products are returned.

Value

See retData argument.

See Also

Other streamObs: FindUsgsStns; GetSiteHuc; GetUsgsHucData; GetUsgsIvProduct; ImproveHucMeta; LoadMetaDB; PlotPrettyUsgs; PrettyUsgs; QuerySiteData; QuerySiteInfo; QuerySiteName; QuerySiteProd; SaveHucData; [.prettyUsgs; subset.prettyUsgs

Examples

```
## Not run:
haveOro <- QueryHaveSite('06727500', path='~/wrfHydroTestCases/usgsDb', retData=TRUE)

## End(Not run)
```

QuerySiteData *Returns the data for given sites from local database.*

Description

QuerySiteData gets the specified data from the local database in long format.

Usage

```
QuerySiteData(site, product, path = ".",
  metaDBFileName = "usgsDataRetrieval.metaDatabase.RData")
```

Arguments

site	Character USGS site number or vector of numbers.
product	Character USGS product code number.
path	Character path to the database.
metaDBFileName	Character The name of the metadata file.

Value

dataframe of data with pertinent attributes.

See Also

Other streamObs: FindUsgsStns; GetSiteHuc; GetUsgsHucData; GetUsgsIvProduct; ImproveHucMeta; LoadMetaDB; PlotPrettyUsgs; PrettyUsgs; QueryHaveSite; QuerySiteInfo; QuerySiteName; QuerySiteProd; SaveHucData; [.prettyUsgs; subset.prettyUsgs

Examples

```
## Not run:
p='~/wrfHydroTestCases/usgsDb/'
dataOrodell <- QuerySiteData(QuerySiteName("FOURMILE CREEK AT ORODELL, CO", p),
                             '00060', p)
siteInfo<-QuerySiteInfo(c('station_nm','site_no','stateCd'), path=p)
dataCO <- QuerySiteData(subset(siteInfo, stateCd=='08' & product=='00060')$site_no,
                        '00060', p)
dataMultiHuc <- QuerySiteData(c('06730500','02084557'),'00060',p)
Case with ill-defined variables
dataMultiHuc <- QuerySiteData('06730500','00065',p)
## This is the multisite multi product case.
dataMultiHuc <- QuerySiteData(c('06730500','02084557'),c('00065','00060'),p)

## End(Not run)
```

QuerySiteInfo *Returns the desired information from the database metadata file.*

Description

QuerySiteInfo gets the specified info from the local database.

Usage

```
QuerySiteInfo(info = NULL, path = ".",
  metaDBFileName = "usgsDataRetrieval.metaDatabase.RData")
```

Arguments

info	Character vector, information fields in HUC$prod$meta$SiteInfo$info.
path	Character The path to the database directory.
metaDBFileName	Character The name of the metadata file.

Value

dataframe of requested info with all available HUC and product codes.

See Also

Other streamObs: FindUsgsStns; GetSiteHuc; GetUsgsHucData; GetUsgsIvProduct; ImproveHucMeta; LoadMetaDB; PlotPrettyUsgs; PrettyUsgs; QueryHaveSite; QuerySiteData; QuerySiteName; QuerySiteProd; SaveHucData; [.prettyUsgs; subset.prettyUsgs

Examples

```
## Not run:
QuerySiteInfo(c('station_nm','site_no'), path='~/streamflow/OBS/')

## End(Not run)
```

QuerySiteName *Find the name (site id) for a given site (name) in the local database.*

Description

QuerySiteName returns the name (site id) for a given site ID (name) in the local database.

Usage

```
QuerySiteName(site, path = ".",
  retSiteId = tryCatch(is.na(as.numeric(site)), warning = function(w) TRUE,
  error = function(e) TRUE),
  metaDBFileName = "usgsDataRetrieval.metaDatabase.RData")
```

Arguments

site	Character USGS site number or name.
path	Character The path to the database.
retSiteId	Logical return ID (name if FALSE) This is only exposed in case there are issues, should work by default.
metaDBFileName	Character The name of the metadata file.

Value

Character Site name or number.

See Also

Other streamObs: FindUsgsStns; GetSiteHuc; GetUsgsHucData; GetUsgsIvProduct; ImproveHucMeta; LoadMetaDB; PlotPrettyUsgs; PrettyUsgs; QueryHaveSite; QuerySiteData; QuerySiteInfo; QuerySiteProd; SaveHucData; [.prettyUsgs; subset.prettyUsgs

Examples

```
## Not run:
QuerySiteName('06730500', '~/streamflow/OBS/')
QuerySiteName('BOULDER CREEK AT MOUTH NEAR LONGMONT, CO', '~/streamflow/OBS/')

## End(Not run)
```

QuerySiteProd *Find the products available for a given site in the local database.*

Description

QuerySiteProd finds the products available for a given site in the local database.

Usage

```
QuerySiteProd(site, path = ".",
  metaDBFileName = "usgsDataRetrieval.metaDatabase.RData")
```

Arguments

site	Character USGS site number.
path	Character The path to the database.
metaDBFileName	Character The name of the metadata file.

See Also

Other streamObs: FindUsgsStns; GetSiteHuc; GetUsgsHucData; GetUsgsIvProduct; ImproveHucMeta; LoadMetaDB; PlotPrettyUsgs; PrettyUsgs; QueryHaveSite; QuerySiteData; QuerySiteInfo; QuerySiteName; SaveHucData; [.prettyUsgs; subset.prettyUsgs

Examples

```
## Not run:
QuerySiteProd('06730500', '~/streamflow/OBS/')

## End(Not run)
```

ReadAmerifluxCSV *Read standard-format CSV data downloaded from Ameriflux*

Description

ReadAmerifluxCSV reads Ameriflux data table (Level 2 standardized CSV file) and creates a dataframe.

Usage

```
ReadAmerifluxCSV(pathFluxData, timeZone)
```

Arguments

pathFluxData	The full pathname to the flux time series CSV file as downloaded from an Ameriflux data server.
timeZone	The time zone for the flux data. Time zone name must be R-friendly for your current OS. See: http://stat.ethz.ch/R-manual/R-devel/library/base/html/timezones.html

Details

ReadAmerifluxCSV reads an Ameriflux Level 2 standardized CSV file and outputs a dataframe with consistent date and data columns for use with other rwrfhydro tools.

Value

A dataframe containing the Ameriflux data.

See Also

Other obsDataReads: ReadAmerifluxNC; ReadCoDwrGage; ReadUsgsGage

Examples

```
## Takes a CSV file downloaded from the ORNL Amerifux website for US-NR1 (Niwot Ridge)
## and returns a dataframe.

## Not run:
obsFlux30min.usnr1 <-
  ReadAmerifluxCSV("../OBS/FLUX/AMF_USNR1_2013_L2_GF_V008.csv", "America/Denver")

## End(Not run)
```

ReadAmerifluxNC *Read standard-format NetCDF data downloaded from Ameriflux*

Description

ReadAmerifluxNC reads Ameriflux data table (Level 2 standardized NetCDF file) and creates a dataframe.

Usage

```
ReadAmerifluxNC(pathFluxData, timeZone)
```

Arguments

pathFluxData	The full pathname to the flux time series NetCDF file as downloaded from an Ameriflux data server.
timeZone	The time zone for the flux data. Time zone name must be R-friendly for your current OS. See: http://stat.ethz.ch/R-manual/R-devel/library/base/html/timezones.html

Details

ReadAmerifluxNC reads an Ameriflux Level 2 standardized NetCDF file and outputs a dataframe with consistent date and data columns for use with other rwrfhydro tools.

Value

A dataframe containing the Ameriflux data.

See Also

Other obsDataReads: ReadAmerifluxCSV; ReadCoDwrGage; ReadUsgsGage

Examples

```
## Takes a NetCDF file downloaded from the ORNL Amerifux website for US-NC2
## (North Carolina Loblolly Pine) and returns a dataframe.

## Not run:
obsFlux30min.usnc2 <-
  ReadAmerifluxNC("../OBS/FLUX/AMF_USNC2_2005_L2_WG_V003.nc", "America/New_York")

## End(Not run)
```

ReadCoDwrGage *Read standard-format text data table downloaded from CO DWR*

Description

ReadCoDwrGage reads CO DWR gage data table (streamflow and/or stage) and puts into a dataframe.

Usage

```
ReadCoDwrGage(pathGageData, returnMetric = TRUE, returnEnglish = TRUE,
  timeZone = "America/Denver")
```

Arguments

pathGageData	The full pathname to the streamflow/stage time series text file as downloaded from CO DWR. The file should include the standard CO DWR header info and the data time series with standard CO DWR columns.
returnMetric	Logical: Return variables in these units (cms and m)
returnEnglish	Logical: Return variables in english units (cfs and ft)
timeZone	(OPTIONAL) The time zone for the gage data (DEFAULT="America/Denver", which is MST/MDT). Time zone name must be R-friendly for your current OS. See: http://stat.ethz.ch/R-manual/R-devel/library/base/html/timezones.html

Details

ReadCoDwrGage reads a streamflow or stage time series data table (standard CO DWR data format) and outputs a dataframe with consistent date and data columns for use with other rwrfhydro tools.

Value

A dataframe containing the flow and/or stage data.

See Also

Other obsDataReads: ReadAmerifluxCSV; ReadAmerifluxNC; ReadUsgsGage

Examples

```
## Take a text file downloaded from the CO DWR website for 15-minute flow
## at Alamosa River and create a dataframe called "obsStr15min.alaterco".

## Not run:
obsStr15min.alaterco <-
  ReadCoDwrGage("../OBS/STRFLOW/ALATERCO_41715103512.txt")
obsStr15min.alaterco <-
  ReadCoDwrGage("../OBS/STRFLOW/ALATERCO_41715103512.txt", returnEnglish=FALSE)

## End(Not run)
```

ReadFrxstPts *Read WRF-Hydro standard-format forecast points output text file.*

Description

ReadFrxstPts reads in WRF-Hydro forecast points output text file.

Usage

```
ReadFrxstPts(pathOutfile)
```

Arguments

pathOutfile	The full pathname to the WRF-Hydro forecast points text file (frxst_pts_out.txt).

Details

ReadFrxstPts reads a standard-format WRF-Hydro forecast points output text file and creates a dataframe with consistent date and data columns for use with other rwrfhydro tools.

Value

A dataframe containing the forecast points output flow data.

See Also

Other modelDataReads: ReadGwOut; ReadLdasoutWb; ReadRtout

Examples

```
## Take a forecast point output text file for an hourly model run of Fourmile Creek
## and return a dataframe.
## Not run:
modStr1h.mod1.fc <- ReadFrxstPts("../OUTPUT/frxst_pts_out.txt")

## End(Not run)
```

ReadGwOut *Read WRF-Hydro standard-format groundwater output text file.*

Description

ReadGwOut reads in WRF-Hydro groundwater output text file.

Usage

```
ReadGwOut(pathOutfile)
```

Arguments

pathOutfile	The full pathname to the WRF-Hydro groundwater text file (GW_inflow.txt, GW_outflow.txt, or GW_zlev.txt).

Details

ReadGwOut reads a standard-format WRF-Hydro groundwater output text file (GW_inflow.txt, GW_outflow.txt, or GW_zlev.txt) and creates a dataframe with consistent date and data columns for use with other rwrfhydro tools.

Value

A dataframe containing the groundwater data.

See Also

Other modelDataReads: ReadFrxstPts; ReadLdasoutWb; ReadRtout

Examples

```
## Take a groundwater outflow text file for an hourly model run of Fourmile Creek
## and return a dataframe.
## Not run:
modGWout1h.mod1.fc <- ReadGwOut("../OUTPUT/GW_outflow.txt")

## End(Not run)
```

ReadLdasoutWb — *Read WRF-Hydro (w/NoahMP) LDASOUT data files and generate basin-wide mean water budget variables.*

Description

ReadLdasoutWb reads in WRF-Hydro (w/NoahMP) LDASOUT files and outputs a time series of basin-wide mean variables for water budget calculations.

Usage

```
ReadLdasoutWb(pathOutdir, pathDomfile, mskvar = "basn_msk", basid = 1,
  aggfact = 10, ncores = 1)
```

Arguments

pathOutdir	The full pathname to the output directory containing the LDASOUT files.
pathDomfile	The full pathname to the high-res hydro domain NetCDF file used in the model run (for grabbing the basin mask).
mskvar	The variable name in pathDomfile to use for the mask (DEFAULT="basn_msk").
basid	The basin ID to use (DEFAULT=1)
aggfact	The high-res (hydro) to low-res (LSM) aggregation factor (e.g., for a 100-m routing grid and a 1-km LSM grid, aggfact = 10)
ncores	If multi-core processing is available, the number of cores to use (DEFAULT=1). Must have doMC installed if ncores is more than 1.

Details

ReadLdasoutWb reads standard-format WRF-Hydro (w/NoahMP) LDASOUT NetCDF files and calculates basin-wide mean values for each time step suitable for running basin water budget calculations.

OUTPUT NoahMP LDASOUT water budget variables:

- ACCECAN: Mean accumulated canopy evaporation (mm)
- ACCEDIR: Mean accumulated surface evaporation (mm)
- ACCETRAN: Mean accumulated transpiration (mm)
- ACCPRCP: Mean accumulated precipitation (mm)
- CANICE: Mean canopy ice storage (mm)
- CANLIQ: Mean canopy liquid water storage (mm)
- SFCRNOFF: Mean surface runoff from LSM *(meaningful for an LSM-only run)* (mm)
- SNEQV: Mean snowpack snow water equivalent (mm)
- UGDRNOFF: Mean subsurface runoff from LSM *(meaningful for an LSM-only run)* (mm)
- SOIL_M1: Mean soil moisture storage in soil layer 1 (top) (mm)
- SOIL_M2: Mean soil moisture storage in soil layer 2 (mm)
- SOIL_M3: Mean soil moisture storage in soil layer 3 (mm)
- SOIL_M4: Mean soil moisture storage in soil layer 4 (bottom) (mm)

Value

A dataframe containing a time series of basin-wide mean water budget variables.

See Also

Other modelDataReads: ReadFrxstPts; ReadGwOut; ReadRtout

Examples

```
## Take an OUTPUT directory for a daily LSM timestep model run of Fourmile Creek and
## create a new dataframe containing the basin-wide mean values for the major water budget
## components over the time series.

## Not run:
modLdasoutWb1d.mod1.fc <-
  ReadLdasoutWb("../RUN.MOD1/OUTPUT", "../DOMAIN/Fulldom_hires_hydrofile_4mile.nc",
                ncores=16)

## End(Not run)
```

`ReadRtout` *Read WRF-Hydro RTOUT data files and generate basin-wide mean water fluxes.*

Description

`ReadRtout` reads in WRF-Hydro RTOUT files and outputs a time series of basin-wide mean water fluxes for water budget.

Usage

```
ReadRtout(pathOutdir, pathDomfile, mskvar = "basn_msk", basid = 1,
  ncores = 1)
```

Arguments

`pathOutdir`	The full pathname to the output directory containing the RTOUT files.
`pathDomfile`	The full pathname to the high-res hydro domain NetCDF file used in the model run (for grabbing the basin mask).
`mskvar`	The variable name in pathDomfile to use for the mask (DEFAULT="basn_msk").
`basid`	The basin ID to use (DEFAULT=1)
`ncores`	If multi-core processing is available, the number of cores to use (DEFAULT=1). Must have doMC installed if ncores is more than 1.

Details

`ReadRtout` reads standard-format WRF-Hydro RTOUT NetCDF files and calculates basin-wide mean values for each time step for water budget terms.

OUTPUT RTOUT water budget variables:

- QSTRMVOLRT: Mean accumulated depth of stream channel inflow (mm)
- SFCHEADSUBRT: Mean depth of ponded water (mm)
- QBDRYRT: Mean accumulated flow volume routed outside of the domain from the boundary cells (mm)

Value

A dataframe containing a time series of basin-wide mean water budget variables.

See Also

Other modelDataReads: `ReadFrxstPts`; `ReadGwOut`; `ReadLdasoutWb`

Examples

```
## Take an OUTPUT directory for an hourly routing timestep model run of Fourmile Creek (Basin ID = 1)
## and create a new dataframe containing the basin-wide mean values for the major water budget
## components over the time series.

## Not run:
modRtout1h.mod1.fc <-
  ReadRtout("../RUN.MOD1/OUTPUT", "../DOMAIN/Fulldom_hires_hydrofile_4mile.nc",
            basid=1, ncores=16)

## End(Not run)
```

ReadSnodasDepthSweDate

Read snodas snow depth and SWE into memory for a given date.

Description

ReadSnodasDepthSweDate Read (from local disk into memory) the SNODAS snow depth and SWE for a given date.

Usage

```
ReadSnodasDepthSweDate(datePOSIXct, outputDir = ".")
```

Arguments

datePOSIXct	The date in POSIXct format for which data is desired. Only whole days matter.
outputDir	The directory where the data are archived.

Value

A list with a SWE and a depth matrix.

See Also

Other SNODAS: CalcSnodasCoords; GetSnodasDepthSweDate; PutSnodasCoordsNcdf; PutSnodasNcdf

Examples

```
snodasGot <- GetSnodasDepthSweDate(as.POSIXct('2015-02-28'),
                                   outputDir = path.expand('~'))
if(snodasGot) snodasList <- ReadSnodasDepthSweDate(as.POSIXct('2015-02-28'),
                                   outputDir = path.expand('~'))
unlink(path.expand('~/us_ssmv1103*.dat.gz'))
```

ReadUsgsGage *Read standard-format text data table downloaded from USGS Water Data*

Description

ReadUsgsGage reads USGS data table (streamflow and/or stage) and puts into a dataframe.

Usage

```
ReadUsgsGage(pathGageData, returnMetric = TRUE, returnEnglish = TRUE,
  timeZone = NULL)
```

Arguments

pathGageData	The full pathname to the streamflow/stage time series text file as downloaded from USGS Water Data. The file should include the standard USGS header info and the data time series with standard USGS columns.
returnMetric	Logical: Return variables in these units (cms and m)?
returnEnglish	Logical: Return variables in english units (cfs and ft)?
timeZone	(OPTIONAL) The time zone for the USGS gage data. Only necessary if tz code is NOT provided in the input file (e.g., daily streamflow gage data files). Time zone name must be R-friendly for your current OS. See: http://stat.ethz.ch/R-manual/R-devel/library/base/html/timezones.html

Details

ReadUsgsGage reads a streamflow or stage time series data table (standard USGS Water Data format) and outputs a dataframe with consistent date and data columns for use with other rwrfhydro tools.

Value

A dataframe containing the USGS flow and/or stage data.

See Also

Other obsDataReads: ReadAmerifluxCSV; ReadAmerifluxNC; ReadCoDwrGage

Examples

```
## Take a text file downloaded from the USGS Water Data website for
## 5-minute flow at Fourmile Creekand create a dataframe called
## "obsStr5min.fc".
## Not run:
obsStr5min.fc <-
  ReadUsgsGage("../OBS/STRFLOW/5min_str_06727500_110401_140810.txt")
obsStr5min.fc <-
  ReadUsgsGage("../OBS/STRFLOW/5min_str_06727500_110401_140810.txt,
               returnEnglish=FALSE")

## End(Not run)
```

rwrfhydro *rwrfhydro: Tools for working with WRF Hydro data.*

Description

A community-contributed tool box for managing, analyzing, and visualizing WRF Hydro (and HydroDART) input and output files in R.

Getting started

Please visit the welcome documentation/README page at:
https://github.com/mccreigh/rwrfhydro

Other package help

- Top level help: `help(package='rwrfhydro')`
- Package overview vignette (long-form example) online:
http://htmlpreview.github.io/?https://github.com/mccreigh/rwrfhydro/blob/master/vignettes/overview.Rmd
- All package vignettes (long-form examples) online:
https://github.com/mccreigh/rwrfhydro/blob/master/vignettes.Rmd
- Discover rwrfhydro orgainzation into "concepts" and "keywords": `GetPkgMeta`

SaveHucData *Save the output of GetUsgsHucData to an archive.*

Description

`SaveHucData` take a list returned by `GetUsgsHucData`, create or append to an existing archive, and update metadata file. Input list is grouped by HUC codes where each HUC has data and meta lists. The output files are written to outPath with the following format HHHHHHHH.data.RData where HHHHHHHH is the HUC8 code. The name of the metadata file is configurable. The metadata file contains metadata for all files in the outPath.

Usage

```
SaveHucData(hucData, outPath,
  metaDBFileName = "usgsDataRetrieval.metaDatabase.RData",
  overwriteHucDataFile = FALSE)
```

Arguments

`hucData`	List Returned from GetUsgsHucData.
`outPath`	Character The path to the database directory
`metaDBFileName`	Character The name of the database metadata file.
`overwriteHucDataFile`	Logical Replace/overwrite the existing data file for the HUC with the current data.

See Also

Other streamObs: FindUsgsStns; GetSiteHuc; GetUsgsHucData; GetUsgsIvProduct; ImproveHucMeta; LoadMetaDB; PlotPrettyUsgs; PrettyUsgs; QueryHaveSite; QuerySiteData; QuerySiteInfo; QuerySiteName; QuerySiteProd; [.prettyUsgs; subset.prettyUsgs

Examples

```
## Not run:
files <- SaveHucData(GetUsgsHucData(huc='10190005'),
                     outPath='~/wrfHydroTestCases/usgsDb') ##boulder, CO
files <- SaveHucData(GetUsgsHucData(huc='03160203'),
                     outPath='~/wrfHydroTestCases/usgsDb') ##satilpa, AL
files <- SaveHucData(GetUsgsHucData(huc='03020104'),
                     outPath='~/wrfHydroTestCases/usgsDb')  ##vans, NC

## End(Not run)
```

SmoothStack *Run MODIS-R Whittaker smoothing over pre-processed raster stack.*

Description

SmoothStack takes a raster stack of RS images and outputs a smoothed raster brick over the same time period.

Usage

```
SmoothStack(inStack, w = NULL, t = NULL, lambda = 5000, nIter = 3,
  outputAs = "one", collapse = FALSE, outDirPath = "./",
  removeOutlier = FALSE, outlierThreshold = NULL, mergeDoyFun = "max",
  ...)
```

Arguments

inStack	The name of the raster stack to smooth.
w	FROM whittaker.raster: In case of MODIS composite the 'VI_Quality' raster-Brick, Stack or filenames. Use preStack functionality to ensure the right input.
t	FROM whittaker.raster: In case of MODIS composite the 'composite_day_of_the_year' raster-Brick, Stack or filenames. Use preStack functionality to ensure the right input.
lambda	FROM whittaker.raster: _Yearly_ lambda value passed to ?ptw:::wit2. If set as character (i.e. lambda="600"), it is not adapted to the time serie length but used as a fixed value (see details). High values = stiff/rigid spline
nIter	FROM whittaker.raster: Number of iteration for the upper envelope fitting.
outputAs	FROM whittaker.raster: Character. Organisation of output files: single each date one RasterLayer, yearly a RasterBrick for each year, one one RasterBrick for the entire time-serie.
collapse	FROM whittaker.raster: logical, if TRUE the input data is treated as _1_single_Year_ collapsing the data using the Julian date information without the year.

outDirPath	FROM whittaker.raster: Output path default is the current directory.
removeOutlier	FROM whittaker.raster: Logical. See details
outlierThreshold	FROM whittaker.raster: Numerical in the same unit as vi, used for outliers removal. See details
mergeDoyFun	FROM whittaker.raster: Especially when using argument collapse=TRUE, multiple measurements for one day can be present, here you can choose how those values are merged to one single value: "max" use the highest value, "mean" or "weighted.mean" use the mean if no weighting "w" is available and weighted.mean if it is.
...	nFROM whittaker.raster: Arguments passed to ?writeRaster (except filename is automatic), NAflag, datatype, overwrite,...

Details

SmoothStack converts a raster stack of RS images (as processed through ConvertRS2Stack) and calls the MODIS-R whittaker.raster smoothing function to generate a smoothed raster brick over the same time series as the input. All function parameters are per the whittaker.raster function except we force the timeInfo to be derived from the input raster stack so the names match and the smoothed brick can be used in other tools. The whittaker.raster tool also exports a set of smoothed TIFs, so also specify an output file directory.

Value

raster brick of smoothed images

See Also

Other MODIS: CalcStatsRS; ConvertRS2Stack; ConvertStack2NC; GetMODIS; InsertRS

Examples

```
## Take the raster stack of LAI images created through ConvertRS2Stack and
## apply a smoothing filter that also removes outliers, which we specify to
## be more than 0.5 LAI from the smoothed value.
## Not run:
lai.b.sm <- SmoothStack(lai.b,
         outDirPath="/Volumes/d1/adugger/RS/MODIS_ARC/PROCESSED/FRNTRNG_LAI_SMOOTHED",
         outputAs="one", removeOutlier=TRUE, outlierThreshold=0.5, lambda=1000,
         overwrite=TRUE)

## End(Not run)
```

VisualizeChanNtwk *Visualize WRF Hydro channel link indices and get coordinates.*

Description

VisualizeChanNtwk shows the channel network indices and provides interactive selection to retrieve link coordinates.

Usage

```
VisualizeChanNtwk(file, gaugePts = NULL, excludeInds = NULL,
  gaugeAccuracy = 12, plot = TRUE)
```

Arguments

file	A path/name to an output YYYMMDDHHmm.CHRTOUT_DOMAIN* file or a hydroDART Posterior_Diag.nc file.
gaugePts	Optional list of gauge points. Nearest stream links are found. See examples.
excludeInds	Optional index of channel network to exclude. See examples.
gaugeAccuracy	The number of digits printed for the gauge information.
plot	Logical to plot or not.

Details

Read a CHANRTOUT file and display the link indicies. Returns a function which allows interactive querying of individual links for lat and lon.

The arguments of the returned function are:

`location` Most generically, the center of the google (or other) map. More specifically, this is the argument passed to the **ggmap** for its argument of the same name. (Default=`c(lon=mean(range(linkDf$lon)),lat`

`zoom=11` The zoom level for the google (or other) map. See **ggmap** for more details.

`source='google'` The source for the underlying map. See **ggmap** package for details.

`maptype='terrain'` The map type for **ggmap**.

`padPlot=.1` The fraction of the range (in both lon and lat) of the channel network to expand the plot by.

`gaugeZoom=NULL` The name of the gauge you'd like to zoom in on. This will likely require finessing zoom and padPlot to make it look nice.

`clickSelect=FALSE` Do you want to click on the plot to query a specfic point? You only get one click per function call.

`linkShape=5` The shape code (ggplot2) for the gridded link elements.

`gaugeShape=4` The shape code (ggplot2) for the gauges.

Value

A function which allows the plot to be interactively queried once each time it is run and returns the coordinates of the selected location. Details provided in details above.

See Also

Other domain: `VisualizeDomain`

Examples

```
## See the vignette "WRF Hydro Domain and Channel Visualization", for details.
## Not run:
tcPath <- '~/wrfHydroTestCases/'
fcPath <- paste0(tcPath,'Fourmile_Creek/')
chrtFile <- paste0(fcPath,'/RUN.RTTESTS/OUTPUT_CHRT_DAILY/201308010000.CHRTOUT_DOMAIN1')
## The basic function call returns a function which you will use.
```

```
LocLinkFun<- VisualizeChanNtwk(chrtFile)
## You can look at the function arguments with
args(LocLinkFun)
## Next, click at the outlet point to see that it has (q=) 0 flow.
LocLinkFun(click=TRUE)
## Any set of valid indices can be excluded.
LocLinkFun<-VisualizeChanNtwk(chrtFile, exclude=350)
## Now clicking at the lowest point reveals that that index is 1.
LocLinkFun(click=TRUE)
## Add gauges and find their nearest links' coords
gaugePts <-
  list(orodell   =data.frame(lon=254.67374999999998408,
                             lat=40.018666670000001773),
       loganMill =data.frame(lon=254.63508330000001934,
                             lat=40.042027779999997961),
       sunshine  =data.frame(lon=254.65122220000000652,
                             lat=40.05761110000000258)  )
## You can just get the location information.
VisualizeChanNtwk(chrtFile, gaugePts=gaugePts, plot=FALSE)
## But you'll still want to exclude the lowest point.
LocLinkFun <- VisualizeChanNtwk(chrtFile, gaugePts=gaugePts, exc=350, plot=FALSE)
## Increase the accuracy of the lon/lat ouput
LocLinkFun <- VisualizeChanNtwk(chrtFile, gaugePts=gaugePts, exc=350, plot=FALSE, gaugeAccuracy=17)
## Now make the plot that was suppressed in the previous call.
LocLinkFun()
## Change the amount of padding around the domain and the shape of the gauge symbols.
LocLinkFun(pad=.3, gaugeShape=16)
## Zoom to the orodell gauge.
LocLinkFun(zoom=14, gaugeShape=16, gaugeZoom='orodell', pad=15)
## Zoom to logan mill gauge
LocLinkFun(zoom=15, gaugeShape=16, gaugeZoom='loganMill', pad=15)
LocLinkFun(zoom=15, gaugeShape=16, gaugeZoom='loganMill', pad=15, click=TRUE)

## End(Not run)
```

`VisualizeDomain` *Visualize WRF Hydro domain files.*

Description

`VisualizeDomain` creates basic plots of WRF Hydro domains.

Usage

```
VisualizeDomain(file, variable = NULL, plot = TRUE)
```

Arguments

`file`	A path/name to a geo or hydro domain file.
`variable`	An optional variable name of interest within the file.
`plot`	Logical: plot or not?

Details

Crude plots of the WRF Hydro domain files. The routine accepts a file path/name for either the geo (coarse resolution) or hydro (fine resolution) files. Spatial variables of interest are listed if none is provided. The return is a function (a closure which encapuslates the domain data) which creates a plot when called. The arguments to the function can be changed to tailor the plot (arguments are passed to ggmap and ggplot inside the function). This function (the closure) returns a ggplot object whose data can be accessed.

Value

A function which can be called to plot the data and allow adjustment of its arguments, the plotting parameters.

See Also

Other domain: VisualizeChanNtwk

Examples

```
## See the vignette "WRF Hydro Domain and Channel Visualization", for details.
## set your test case path
## Not run:
tcPath <- '~/wrfHydroTestCases/'
fcPath <- paste0(tcPath,'Fourmile_Creek/')
hydroFile<-paste0(fcPath,'/DOMAIN/hydro_OrodellBasin_100m.nc')
GgMapFunction <- VisualizeDomain(hydroFile, "CHANNELGRID")
ggMap1 <- GgMapFunction(zoom=11, pointshape=15, pointsize=7,
                        source="google", maptype="terrain")
# Add a streamflow gauge point; compare reality and the model.
orodellLonLat <- data.frame(lon=c(254.6722259521484375, 254.67374999999998408)-360,
                            lat=c(40.019321441650390625, 40.018666670000001773),
                            gauge=c('model','USGS'))
ggMap2 <- GgMapFunction(location=c(lon=orodellLonLat$lon[1], lat=orodellLonLat$lat[1]),
                        zoom=14, pointshape=15, pointsize=7,
                        source="google", maptype="terrain", plot=FALSE)
ggMap2 + geom_point(data=orodellLonLat, aes(x=lon,y=lat, shape=gauge)) +
         scale_x_continuous(limits=rev(orodellLonLat$lon+c( .01 ,-.01 ))) +
         scale_y_continuous(limits=rev(orodellLonLat$lat+c( .005,-.005)))

## End(Not run)
```

Index