Package ‘EcoHydRology’

February 19, 2015

Version 0.4.12
Title A community modeling foundation for Eco-Hydrology.
Author Fuka DR, Walter MT, Archibald JA, Steenhuis TS, and Easton ZM
Maintainer Daniel Fuka <drf28@cornell.edu>
Depends R (>= 2.10), operators, topmodel, DEoptim, XML
Description This package provides a flexible foundation for scientists, engineers, and policy makers to base teaching exercises as well as for more applied use to model complex eco-hydrological interactions.
License GPL-2
Repository CRAN
Date/Publication 2014-04-04 08:09:55
KeepSource TRUE
NeedsCompilation no

R topics documented:

EcoHydRology-package .............................................. 2
alter_files ........................................................... 3
AtmosphericEmissivity ........................................... 4
BaseflowSeparation ................................................ 5
build_gsod_forcing_data .......................................... 6
calib_swat_ex ....................................................... 7
change_params ...................................................... 8
ConvertFlowUnits .................................................. 9
declination .......................................................... 10
EnvirEnergy ........................................................ 11
EstCloudiness ...................................................... 12
EvapHeat ............................................................. 13
get_cfsr_laton ...................................................... 14
get_gsod_stn ....................................................... 15
get_usgs_gage ...................................................... 17
GroundHeat .......................................................... 18
Description

This package provides a flexible foundation for scientists, engineers, and policy makers to base teaching exercises as well as for more applied use to model complex eco-hydrological interactions.

Details

<table>
<thead>
<tr>
<th>Package:</th>
<th>EcoHydRology</th>
</tr>
</thead>
<tbody>
<tr>
<td>Type:</td>
<td>Package</td>
</tr>
<tr>
<td>Version:</td>
<td>0.4.12</td>
</tr>
<tr>
<td>Date:</td>
<td>2013-09-27</td>
</tr>
<tr>
<td>License:</td>
<td>GPL-2</td>
</tr>
<tr>
<td>LazyLoad:</td>
<td>yes</td>
</tr>
</tbody>
</table>
Included Energy Budget Functions:
AtmosphericEmissivity
EnvirEnergy
EstCloudiness
EvapHeat
GroundHeat
Longwave
NetRad
PotentialSolar
RainHeat
SatVaporDensity
SensibleHeat
Solar
declination
slopefactor
solarangle
solaraspect
transmissivity

Included Data Handeling Functions:
get_gsod_stn
build_gsod_forcing_data

Included data:
GSOD_history

Author(s)
Fuka DR, Walter MT, Archibald JA, Steenhuis, TS, and Easton ZM
Maintainer: Daniel Fuka <drf28@cornell.edu>

---

**alter_files**

_a function to alter files called from swat_objective_function_

#### Description
A function to alter files called from swat_objective_function

#### Usage

```r
alter_files(change_params)
```

#### Arguments

- `change_params`
Atmospheric Emissivity

**Author(s)**
Daniel R. Fuka

**Description**
The emissivity of the atmosphere [-]

**Usage**
AtmosphericEmissivity(airtemp, cloudiness, vp=NULL, opt="linear")

**Arguments**
- **airtemp**
  Air temperature: air temperature [°C]
- **cloudiness**
  Cloudiness: fraction of the sky covered in clouds [-]
- **vp**
  Vapor Pressure: [kPa]
- **opt**
  Option: either "linear" for a linear relationship between clear sky emissivity and temperature, or "Brutsaert" to use Brutsaert's (1975) clear sky emissivity formulation - this requires input for vapor pressure

**Value**
The emissivity of the atmosphere [-]

**Author(s)**
Fuka, D.R., Walter, M.T., Archibald, J.A.

**References**

**Examples**
```
temp=15
clouds=.5
AtmEm=AtmosphericEmissivity(temp,clouds)
print(AtmEm)
```
**Description**

This function reads a streamflow dataset and produces a baseflow dataset. It can be run using 1, 2 or 3 passes.

**Usage**

```
BaseflowSeparation(streamflow, filter_parameter = 0.925, passes = 3)
```

**Arguments**

- `streamflow`: A vector containing streamflow values
- `filter_parameter`: The value recommended by Nathan and McMahon (1990) is 0.925, however, the user might want to play with this value (0.9-0.95)
- `passes`: The number of times you want the filter to pass over your data. 1-3

**Value**

This will return a 2 column data frame with nrow = length of input streamflow data. The first column contains baseflow, while the second contains quickflow, both in the same units as the input.

**Author(s)**

Josephine Archibald

**References**


**Examples**

```r
### Look at a dataset for Owasco Lake in NY:
data(OwascoInlet)
summary(OwascoInlet)

## Get an approximation for baseflow using a 3 pass filter:
bfs<-BaseflowSeparation(OwascoInlet$Streamflow_m3s, passes=3)

## You can check out how this looks with the hydrograph function:
hydrograph(input=OwascoInlet,streamflow2=bfs[,1])
```
build_gsod_forcing_data

Parsing gzipped GSOD dataset.

Description

Parsing of the gzipped GSOD forcing data as returned from get_gsod_stn()

Usage

build_gsod_forcing_data()

Author(s)

Daniel R. Fuka

Examples

```
# After running get_gsod_stn() with addis as a temp directory then:
#
## Not run:
tmp_ppt<-build_gsod_forcing_data()
addis

## End(Not run)
## The function is currently defined as
function()
{
tmpdir=readline("Please enter a temp directory where you stored the
*.*_gz datafiles? \n")
files=dir(tmpdir,"*.gz",full.names=T)
start_year=min(substr(files,nchar(files)-9,nchar(files)-6))
end_year=max(substr(files,nchar(files)-9,nchar(files)-6))
alldates=data.frame(fdate=seq(from=as.Date(paste(start_year,"-01-01",sep="")), to=as.Date(paste(end_year,"-12-31",sep="")), by=1))

stn=matrix()
tmin=matrix()
tmax=matrix()
ppt=matrix()
fdate=matrix()
for (tmpfile in files)
{

# There is more data in this dataset we can extract later as we need it.
#
```
A function demonstrate an example series of steps to calibrate a reach in the SWAT2005 watershed model.

Description
A function demonstrate an example series of steps to calibrate a reach in the SWAT2005 watershed model. This should act only as a simple demonstration, and should be modified to individuals use.

Usage
```
calib_swat_ex(flowgage, rch = 3)
```

Arguments
- `flowgage`: A list in the format that would be returned from the `get_usgs_gage`
- `rch`: The reach in the output.rch file you wish to calibrate against.

Details
This should act only as a simple demonstration, and should be modified to individuals use.

Value
List (of length 2) containing the results of the internally called DEoptim function. See DEoptim for more information.
Author(s)

Daniel R. Fuka

Examples

```r
## Not run:
#
install.packages("SWATmodel")
#
# Load library, test SWAT numerics, define a flow gage id for USGS gage, or build similar list
# of data as presented in ?get_usgs_gage
#
library(SWATmodel)
testSWAT2005()
flowgage_id="04216500"
flowgage=get_usgs_gage(flowgage_id)
#
# Obtain basic set of historical forcing data from TAMU/Cornell/IWMI CFSR repository
#
hist_wx=get_cfsr_latlon(flowgage$declat,flowgage$declon)
#
# Build a generic 9 HRU SWAT initialization, which builds and changes into directory
# named as flowgage_id above.
#
build_swat_basic(dirname=flowgage_id,iyr="1979",nbyr=6,flowgage$area,
flowgage$elev,flowgage$declat,flowgage$declon,hist_wx=hist_wx)
#
# Simple calibration based on the outflow from RCH 3 of the simple SWAT initialization
#
calib_swat_results=calib_swat_ex(flowgage,rch=3)
#
# Graph output as hydro graph and pred/obs graphics
plot(calib_swat_results$flowdata,flowgage$flowdata)

## End(Not run)
```

<table>
<thead>
<tr>
<th>change_params</th>
<th>Example dataframe of parameters, ranges, and search strings for optimizing swat2005</th>
</tr>
</thead>
</table>

Description

Example dataframe of parameters, ranges, and search strings for optimizing swat2005

Usage

```r
data(change_params)
```
ConvertFlowUnits

Format

A data frame with 15 observations on the following 11 variables.

- filetype: a factor with levels `PJNgw`, `PJNmgt`, `PJNsol`, `basinsNbsn`
- parameter: a factor with levels `ALPHA_BF`, `Ave\ Density`, `CN2`, `Depth`, `ESCO`, `GW_DELAY`, `GWQMN`, `Ksat\`. (est\).
- alter_type: a factor with levels `new`, `percent`
- min: a numeric vector
- max: a numeric vector
- current: a numeric vector
- multi: a logical vector
- frformat: a factor with levels `A27, 10f12`
- fwformat: a factor with levels `F16.4`, `F20`

Examples

```r
data(change_params)
## maybe str(change_params); plot(change_params) ... 
```

---

ConvertFlowUnits Unit conversion for flow rates

Description

Converts volumetric flow (cfs, cms, cmd) to depth flow rate over a watershed (mm/d). Or, it converts a depth flow rate to volumetric (mm/d to cfs)

Usage

```r
ConvertFlowUnits(cfs = NULL, cmd=NULL, cms = NULL, WA, mmd = NULL, AREAunits = "mi2")
```

Arguments

- **cfs**: Input flow in cubic feet per second
- **cmd**: Input flow in cubic meters per day
- **cms**: Input flow in cubic meters per second
- **WA**: Watershed area. Can be entered in square miles (default), or square km (in this case, change the AREAunits to "km2")
- **mmd**: Input flow in mm/d
- **AREAunits**: Units of the watershed area ("mi2" or "km2")
declination

Details
Note, only one flow input should be used. (i.e., user should define cfs OR cmd OR cms OR mmd). Watershed area must always be defined.

Value
converted flow rate in either mm/d (if converting from volumetric flow) or cfs (if converting from flow depth in mm/d)

Author(s)
Josephine Archibald

Examples

```r
data(OwascoInlet)
OwascoInlet$Streamflow_mmd <- ConvertFlowUnits(cms=OwascoInlet$Streamflow_m3s, WA=271., AREAunits="km2")

# # The following commented example isn't currently working in Linux systems, but should
# # work in windows/macs
#
### # Get some streamflow (reported in cubic meters per day here):
# OI <- get_usgs_gage("04235299", "2013-03-01", "2013-05-20")
# FC <- get_usgs_gage("04234000", "2013-03-01", "2013-05-20")

# # Convert to mm/d
# OwascoInlet_mmd <- ConvertFlowUnits(cmd=OI$flowdata$flow, WA=OI$area, AREAunits="km2")
# FallCreek_mmd <- ConvertFlowUnits(cmd=FC$flowdata$flow, WA=FC$area, AREAunits="km2")

# # Compare the watershed area normalized flow depth for two watersheds near Ithaca NY :
# hydrograph(streamFlow=OwascoInlet_mmd, streamflow2=FallCreek_mmd, timeSeries=FC$flowdata$mdate,
# stream.label="flow depth (mm/d)")
# legend("topright", legend=c("Owasco Inlet", "Fall Creek"), lty=c(1,2), col=c("black", "red"))
```

description

Solar Declination

Description
Solar declination for a given day of the year.

Usage
declemation(Jday)

Arguments

| Jday | Julian date or day of the year [day] |
Author(s)

M. Todd Walter

Examples

```bash
### Should be DIRECTLY executable !! ----
### ==> Define data, use random,
### or do help(data=index) for the standard data sets.

### The function is currently defined as

```bash
function(jday){
  # solar declination [rad]
  #
  # jday: Julian date or day of the year [day]
  #
  return(0.4102*sin(pi*(jday-80)/180))
}
```

---

**EnvirEnergy**

**The Surface Net Energy Budget**

Description

Net Energy=S+La-Lt+S+E+R+G, where S is the net incident solar radiation, La is the atmospheric long wave radiation, Lt is the terrestrial long wave radiation, H is the sensible heat exchange, E is the energy flux associated with the latent heats of vaporization and condensation, G is ground heat conduction, P is heat added by precipitation.

Usage

```bash
EnvirEnergy(lat, Jday, Tx, Tn, wind, rain, relativehumidity,
  albedo=0.2, cloudiness=NULL, forest=0, slope=0, aspect=0,
  surftemp=(Tx+Tn)/2, surfemissivity=0.97)
```

Arguments

```bash
lat          latitdue [rad]
Jday         Julian day / day of the year, 1-365 [day]
Tx           Maximum daily temperature [C]
Tn           Minimum daily temperature [C]
wind         Average daily windspeed [m/s]
rain         precipitation [mm/day]
relativehumidity  Relative humidity; if negative, air vapor density will be approximated [-]
albedo       #albedo: surface albedo or reflectivity [-]
```
cloudiness          Fraction of the sky covered in clouds, if no value provided, cloudiness will be approximated [-]
forest             Forest or vegetation cover [-]
Slope              Slope of the ground [rad]
aspect              Ground aspect [rad from north]
surftemp           Surface temperature [C]
surfemissivity     Surface emissivity [-]

Value

NetEnergy         Surface net energy

Author(s)

Walter, M.T., Fuka, D.R. Maintainer: Daniel Fuka <drf28@cornell.edu>

Examples

```r
# The function is currently defined as
function(lat,Jday,Tx,Tn,wind,relativehumidity,cloudiness,albedo,forest,
slope,aspect,surftemp,surfemissivity){
  if(cloudiness<0) {cloudiness<-EstCloudiness(lat,Jday,Tx,Tn)}
  airtemp<- (Tx+Tn)/2 # average daily air temperature [C]

  return(Solar(lat,Jday,Tx,Tn,albedo,forest,slope,aspect)+
    Longwave(AtmosphericEmissivity(airtemp,cloudiness),airtemp)-
    Longwave(surfemissivity,surftemp)+SensibleHeat(surftemp,airtemp,wind)+
    EvapHeat(surftemp,airtemp,relativehumidity,Tn,wind)+
    RainHeat(airtemp,rain)+GroundHeat()
}
```

---

EstCloudiness          Estimated Cloudiness

Description

Estimates the cloudiness of the atmosphere by scaling to atmospheric transmissivity

Usage

```r
EstCloudiness(Tx=(-999), Tn=(-999), trans=NULL, transMin = 0.15, transMax = 0.75, opt = "linear")
```
Arguments

- Tx: maximum daily temperature [C]
- Tn: minimum daily temperature [C]
- trans: transmissivity of the atmosphere (value between 0-1)
- transMin: Transmissivities equal and below this value will return a cloudiness value of 1 (for linear approximation)
- transMax: Transmissivities equal and above this value will return a cloudiness value of 0 (for linear approximation)
- opt: Currently there are two options: "linear" (Campbell 1985) and "Black" (1956)

Author(s)

M. Todd Walter, Josephine Archibald

References


Black, J.N. (1956) The Distribution of Solar Radiation over the Earth’s Surface. Theoretical and Applied Climatology 7:2 165-189

Examples

EstCloudiness(trans=0.2)
EstCloudiness(20,12,opt="Black")

EvapHeat

Evaporative heat exchange between a wet surface and the surrounding air [kJ m\(^{-2}\) d\(^{-1}\)]. This function is only intended for wet surfaces, i.e., it assumes the vapor density at the surface is the saturation vapor density

Usage

EvapHeat(surftemp, airtemp, relativehumidity=NULL, Tn=NULL, wind=2)
get_cfsr_latlon

Arguments

- surftemp: surface temperature [C]
- airtemp: average daily air temperature [C]
- relativehumidity: relative humidity, 0-1 [-]
- Tn: minimum daily air temperature, assumed to be dew point temperature if relativehumidity unknown [C]
- wind: average daily windspeed [m/s]

Note

This function will use the relative humidity to estimate air vapor density if the value passed is between 0-1. Otherwise, it will assume the minimum daily air temperature is approximately the dew point temp.

Author(s)

M. Todd Walter

References


Examples

EvapHeat(surftemp=15, airtemp=5, relativehumidity=0.7)

get_cfsr_latlon  This is a function to grab daily summaries of the CFSR from the drfuka.org service

description

This is a function to grab daily summaries of the CFSR from the drfuka.org service

Usage

get_cfsr_latlon(declat, declon, emailaddr, timeoff = 0, interprow = 2)

Arguments

- declat
- declon
- emailaddr
- timeoff
- interprow
get_gsod_stn

Author(s)
Daniel R Fuka

Examples

## Not run:

```r
hist_wx = get_cfsr_latlon(45, -72, "dan@dan.com", timeoff=0, interppow=2)
plot(hist_wx$TMX)
```

## End(Not run)

## The function is currently defined as

```r
function (declat, declon, emailaddr, timeoff = 0, interppow = 2)
{
  library(XML)
  options(timeout = 120)
  declat, "&lon=", declon, "&timeoff=", timeoff, "&interppow=",
  interppow, ".&submit=Submit", sep = "")
  hist_wx = readHTMLTable(url, which = 1, header = T, colClasses = c("character",
  "numeric", "numeric", "numeric", "numeric", "numeric",
  "numeric", "numeric"))
  hist_wx$DATE = as.Date(hist_wx$DATE, format = "%Y-%m-%d")
  return(hist_wx)
}
```

---

get_gsod_stn Obtain Global Summary of Day data

Description

This function will grab available GSOD data for stations and years for which the data is available.

Usage

```r
get_gsod_stn()
```

Author(s)

Daniel R. Fuka

Examples

```r
# Example of downloading data for ADDIS ABABA
## Not run: get_gsod_stn()
# Please enter a temp directory ....
addis
# What is the station name you want to look for?
```
get_gsod_stn

ADDIS  A
#  USAF WBAN  STATION_NAME .
#13582 634500 99999 ADDIS ABABA-BOLE .
#Good Enough?
y
#Given the start and end years, ....
n
#Date range available is 1957 to 2011 , what start year would you like?
1990
#Date range available is 1957 to 2011 , what end year would you like?
1992
#Good enough?
y

## End(Not run)

## The function is currently defined as
function(){
  tmpdir=readline("Please enter a temp directory to store the datafiles in, remember this directory when parsing with build_gsod_forcing_data()? \n")
dir.create(tmpdir)
stnlist=GSOD_history
ANSWER="n"
while(ANSWER=="n"){
stnreq=readline("What is the station name you want to look for? \n")
test=grep(stnreq,stnlist$STATION.NAME)
print(stnlist[test,])
ANSWER <- readline("Good enough? ")
}
ANSWER <- readline("Given the start and end years, would you like to update the stnlist file from NDCD? (may take a few minutes) ")
## a better version would check the answer less cursorily, and
## perhaps re-prompt
if (substr(ANSWER, 1, 1) == "y")
stnlist=read.csv(url("ftp://ftp.ncdc.noaa.gov/pub/data/gsod/ish-history.csv"))
#
# Add one more request to make sure the number of years is ok as some
# stations have really long histories.
#
working_tmp <- stnlist$STATION.NAME
has_stn<- working_tmp %in% grep(stnreq, working_tmp, value=TRUE)
dl_wbans=subset(stnlist, has_stn , USAF:END)
start_year=min(trunc(subset(stnlist, has_stn, BEGIN)/10000))
end_year=max(trunc(subset(stnlist, has_stn , END)/10000))
start_year_rec<- readline(paste("Date range available is",start_year,"to",end_year,"", what start year would you like?"))
end_year_rec<- readline(paste("Date range available is",start_year,"to",end_year,"", what end year would you like?"))
for (wbans in row.names(dl_wbans)){
  start_year=max(start_year_rec,substr(dl_wbans[wbans,"BEGIN"],1,4))
  end_year=min(end_year_rec,substr(dl_wbans[wbans,"END"],1,4))
get_usgs_gage

A function to grab daily stream flow measurements from the USGS waterdata.usgs.gov server.

Description
A function to grab daily stream flow measurements from the USGS waterdata.usgs.gov server.

Usage
get_usgs_gage(flowgage_id, begin_date="1979-01-01", end_date="2013-01-01")

Arguments
flowgage_id  flowgage_id Gage is a TEXT/String rather than numeric or the query will fail on gages with leading 0s
begin_date  begin_date begin_date is a TEXT/String for the start date for the data you want. Default is the beginning of the cfsr dataset.
end_date  end_date end_date is a TEXT/String for the end date for the data you want. Default is in the future sometime

Value
area  The area above the gage [km2]
declat  Decimal Lat
declon  Decimal Lon
flowdata  Dataframe with the historical flow data - flow is in cubic meters per day
**GroundHeat**

**Author(s)**
Daniel R Fuka

**Examples**

```r
flowgage_id="04216500"
flowgage=get_usgs_gage(flowgage_id)

## The function is currently defined as
```

<table>
<thead>
<tr>
<th>GroundHeat</th>
<th>Heat conducted to the bottom of a snowpack</th>
</tr>
</thead>
</table>

**Description**

The heat conducted to the bottom of a snowpack, assumed constant [kJ m\(^{-2}\) d\(^{-1}\)]

**Usage**

```r
GroundHeat()
```

**Author(s)**
M. Todd Walter

**Examples**

```r
## Should be DIRECTLY executable !! ----
## ==> Define data, use random,
## or do help(data=index) for the standard data sets.

## The function is currently defined as
function()

# the heat conducted to the bottom of a snowpack, assumed constant [kJ m\(^{-2}\) d\(^{-1}\]

return(173)
)
```
Description
A station list to be used with the GSOD data files, showing the names and locations for each station. Note: Global summary of day contains a subset of the stations listed in this station history.

Usage
```r
data(GSOD_history)
```

Format
A data frame with 30558 observations on the following 12 variables.

- `usaf` a numeric vector
- `wban` a numeric vector
- `stationName` a factor with levels
- `ctry` a factor with levels
- `fips` a factor with levels
- `state` a factor with levels
- `call` a factor with levels
- `lat` a numeric vector
- `lon` a numeric vector
- `elev` a numeric vector
- `begin` a numeric vector
- `end` a numeric vector

Source
```url
```

Examples
```r
data(GSOD_history)
```
## hydrograph

**Plot a hydrograph**

<table>
<thead>
<tr>
<th><strong>Description</strong></th>
</tr>
</thead>
<tbody>
<tr>
<td>Creates a hydrograph with one or two streamflow data sets, and can include a precipitation hyetograph.</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th><strong>Usage</strong></th>
</tr>
</thead>
<tbody>
<tr>
<td><code>hydrograph(input = matrix(ncol = 2, nrow = 2), streamflow = input[, 2], timeSeries = input[, 1], streamflow2 = NULL, precip = NULL, begin = 1, endindex = length(streamflow), P.units = &quot;&quot;, S.units = &quot;&quot;, S1.col = &quot;black&quot;, S2.col = &quot;red&quot;, stream.label = &quot;Streamflow&quot;, streamflow3 = NULL, streamflow4 = NULL, precip2 = NULL)</code></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th><strong>Arguments</strong></th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>input</strong> This is a data frame with at least two columns of data First column: Must be a time/date series Second column: If including precip, precip. Otherwise, streamflow - AND do not include any other columns Third column: (Only include if precip is in the second column) First streamflow dataset Forth column: (optional) Second streamflow dataset</td>
</tr>
<tr>
<td><strong>streamflow</strong> vector of streamflow values - not necessary if using &quot;input&quot;</td>
</tr>
<tr>
<td><strong>timeSeries</strong> vector of time or date values - not necessary if using &quot;input&quot;</td>
</tr>
<tr>
<td><strong>streamflow2</strong> optional vector of streamflow values - not necessary if using &quot;input&quot;</td>
</tr>
<tr>
<td><strong>precip</strong> vector of precipitation values - not necessary if using &quot;input&quot;</td>
</tr>
<tr>
<td><strong>begin</strong> If you don’t want to graph the whole data set, include the index where you would like to start</td>
</tr>
<tr>
<td><strong>endindex</strong> If you don’t want to graph the whole dataset, include the index where you would like to end</td>
</tr>
<tr>
<td><strong>P.units</strong> Precipitation units (character)</td>
</tr>
<tr>
<td><strong>S.units</strong> Streamflow units (character). Users who have volumetric flow will have the superscript correctly formatted if they enter &quot;m³s&quot; or &quot;ft³s&quot;</td>
</tr>
<tr>
<td><strong>S1.col</strong> color of the line representing streamflow</td>
</tr>
<tr>
<td><strong>S2.col</strong> color of the line representing second streamflow data</td>
</tr>
<tr>
<td><strong>stream.label</strong> character string - Label for vertical axis</td>
</tr>
<tr>
<td><strong>streamflow3</strong> optional vector of additional streamflow values</td>
</tr>
<tr>
<td><strong>streamflow4</strong> optional vector of additional streamflow values</td>
</tr>
<tr>
<td><strong>precip2</strong> optional vector of a second precip gage data if you are interested in comparing precip inputs</td>
</tr>
</tbody>
</table>
**Warning**

The date series should be continuous and evenly spaced. If not, the dates will not line up accurately on the x-axis.

**Note**

This function can now take NA values. If you chose to use the input argument but are not including precipitation, input must only have two columns - you will need to add the second streamflow dataset using the "streamflow2" argument.

**Author(s)**

Josephine Archibald

**Examples**

```r
data(OwascoInlet)
head(OwascoInlet)
hydrograph(OwascoInlet)
```

---

<table>
<thead>
<tr>
<th>Longwave</th>
<th><em>Daily Longwave Radiation</em></th>
</tr>
</thead>
</table>

**Description**

Daily longwave radiation based on the Stefan-Boltzmann equation [kJ m\(^{-2}\) d\(^{-1}\)]

**Usage**

```r
Longwave(emissivity, temp)
```

**Arguments**

<table>
<thead>
<tr>
<th>emissivity</th>
<th>emissivity [-]</th>
</tr>
</thead>
<tbody>
<tr>
<td>temp</td>
<td>temperature of the emitting body [C]</td>
</tr>
</tbody>
</table>

**Author(s)**

M. Todd Walter
Lumped VSA Model

Lumped Variable Source Area (VSA) Watershed Model

Description

This model calculates streamflow and approximate saturated area percentage contributing to overland flow. It was developed in saturation-excess dominated watersheds, and is based on the Thornthwaite-Mather water budget and SCS Curve Number approach for overland runoff.

Usage

```
Lumped_VSA_model(dateSeries, P, Tmax, Tmin, Depth = NULL, SATper = NULL, AWCper = NULL, percentImpervious = 0, no_wet_class = 10, Tp = 5, latitudeDegrees = 42.38, StartCond = "avg", PETin = NULL, AWC = Depth * AWCper, SAT = Depth * SATper, SW1 = NULL, BF1 = 1, PETcap = 5, rec_coef = 0.1, Se_min = 78, Cl = 3.1, Ia_coef = 0.05, PreviousOutput = NULL, runoff_breakdown = RunoffBreakdown(Tp, HrPrcDelay = (Tp/2 - 4)))
```

Arguments

dateSeries Daily date series in the format "2013-05-21"
P Rain + Snowmelt (mm)
Tmax Maximum daily T (C)
Tmin Minimum daily T (C)
Depth Average watershed soil depth (mm) Not needed if SAT and AWC depth entered directly
SATper Porosity of the soil (volumetric fraction, 0-1) Not needed if SAT (porosity depth) entered directly
AWCper Available water capacity, Field capacity - wilting point (volumetric fraction, 0-1) Not needed if AWC entered directly
percentImpervious Percent of the watershed that is impervious (percentage, 0-100)
no_wet_class Number of wetness classes to distribute runoff over. Default is 10.
Tp Time to peak of hydrograph (hours)
latitudeDegrees latitude (degrees)
albedo Average surface albedo, defaults to average 0.23
StartCond Watershed conditions before first day of run (options are "wet", "dry", "avg")
PETin # User has the option to enter PET values (mm/day), otherwise this will be estimated from Priestley-Taylor equation, estimating radiation from temperature
AWC # AWC depth (mm)
SAT Porosity depth (mm)
SW1 Soil water on the first day (depth, mm)
BF1 Baseflow on the first day (mm/day)
PETcap Cutoff for maximal PET allowed per day (mm)
rec_coef Baseflow recession coefficient
Se_min Minimal daily CN S value. (mm)
C1 Coefficient relating daily Curve Number S to soil water
Ia_coef Initial abstraction coefficient for CN-equation. (range ~ 0.05 - 0.2)
PreviousOutput If the model is run repeatedly, previous output can be provided so that the model only needs to calculate from that point forward.
runoff_breakdown The proportion of runoff that reaches the outlet on a given day after the storm event. Can be calculated from Tp

Details
The model expects continuous input on a daily time-step, since the soil-water is calculated over time, and affects the amount of runoff that will be generated after a storm. Also, note that precipitation values are actually Rain + Snowmelt (mm). Users can use the snowmelt function to determine this if needed.

Value
Returns a data frame with modeled streamflow, baseflow, ET, and maximum wetness class generating runoff for all dates. Soil water and other modeled intermediate results are also returned. All flow values (modeled_flow, baseflow, OverlandFlow, ShallowInterflow, totQ, quickflow_combined, impervRunoff, excess) are in depth of flow per day (mm/d)

Warning
This function cannot handle NA values in input, and can only be run for a daily time-step. If Tx < Tn for any day, this will produce an error. Currently, the crop coefficients used to estimate PET are specific for deciduous northeastern USA.

Note
This function is under development and might change substantially in further versions.

Author(s)
Josephine Archibald

References
NetRad

Shaw, SB, MT Walter. 2009. Improving runoff risk estimates: Formulating runoff as a bivariate process using the SCS curve number method. Water Resources Research. 45

Thornthwaite CW, JR Mather. 1957 Instructions for computing potential evapotranspiration and water balance. Publ Climatol 3: 183-311


Weiler K. Unpublished. Determination of the Linear Bedrock Coefficient From Historical Flow Data

See Also

PET_fromTemp

Examples

data(OwascoInlet)
# First get rain and snow-melt input:
# rsm <- SnowMelt(Date=OwascoInlet$Date, precip_mm=OwascoInlet$P_mm, Tmax_C=OwascoInlet$Tmax_C, Tmin_C=OwascoInlet$Tmin_C, lat_deg=42.66)
# Calculate streamflow based on watershed characteristics:
# Results <- Lumped_VSA_model(dateSeries = OwascoInlet$Date, P = rsm$Rain_mm+rsm$SnowMelt_mm, Tmax=OwascoInlet$Tmax_C, Tmin = OwascoInlet$Tmin_C, latitudeDegrees=42.66, Tp = 5.8, Depth = 2010, SATper = 0.27, AWPer = 0.13, StartCond = "wet")
# View results:
hydrograph(streamflow=ConvertFlowUnits(cms=OwascoInlet$Streamflow_m3s, WA=106, AREAunits="mi2"), timeSeries=OwascoInlet$Date, streamflow2=Results$modeled_flow, precip=rsm$Rain_mm+rsm$SnowMelt_mm)

NetRad

Daily Net Radiation

Description

Daily net radiation [kJ m-2 d-1]

Usage

NetRad(lat, Jday, Tx, Tn, albedo = 0.18, forest = 0, slope = 0, aspect = 0, airtemp = (Tn+Tx)/2, cloudiness = "Estimate", surfemissivity = 0.97, surftemp = (Tn+Tx)/2, units = "kJm2d", AEpars=as.list(vp=NULL, opt="linear"))

Arguments

lat latitude. Default is radians, but will automatically convert from degrees if value is larger than pi/2 or less than -pi/2

Jday Day of the year [-]

Tx Max temperature [C]
OwascoInlet

Tn  Min temperature [C]
albedo  Albedo (0-1) [-]
forest  Forest canopy or shade cover (0-1) [-]
slope  Slope [radians]
aspect  Ground aspect [radians from north]
airtemp  Average air temp [C]
cloudiness  Cloudiness percentage (0-1) [-] Default value will estimate cloudiness based on daily temperature difference.
surfemissivity  Surface emissivity [-]
surftemp  Surface temperature [C]
units  Units of the result. Defaults to "kJm2d" which returns result in kJ/m2/d, the other option is "Wm2", which will make output in W/m2
AEparams  Atmospheric Emissivity options. Defaults to linear approximation. To use Brutsaert option, include vapor pressure values (kPa)

Author(s)

M. Todd Walter, Josephine Archibald

Examples

NetRad(42.4*pi/180, Jday=55, Tx=2, Tn=(-3))

---

OwascoInlet  Owasco Inlet data

Description

Streamflow and temperature data for Owasco Inlet from USGS and NRCC

Usage

data(OwascoInlet)

Format

A data frame with 888 observations on the following 6 variables.

date  a Date
P_mm  a numeric vector
Streamflow_m3s  a numeric vector
baseflow_m3s  a numeric vector
Tmax_C  a numeric vector
Tmin_C  a numeric vector
**Source**


**References**

Data taken from USGS (www.usgs.gov) and NRCC (http://www.nrcc.cornell.edu)

**Examples**

```r
data(OwascoInlet)
## maybe str(OwascoInlet) ; plot(OwascoInlet) ...
```

---

**PET_fromTemp**

*Priestley-Taylor Potential Evapotranspiration from temperature*

**Description**

Calculates potential Evapotranspiration (in meters) based on the Priestley-Taylor equation (1972). We use an estimation of net radiation based on temperature data.

**Usage**

```r
PET_fromTemp(Jday, Tmax_C, Tmin_C, lat_radians,
    AvgT = (Tmax_C + Tmin_C)/2, albedo = 0.18, TerrestEmiss = 0.97,
    aspect = 0, slope = 0, forest = 0, PTconstant=1.26,
    AEparams=list(vp=NULL, opt="linear"))
```

**Arguments**

- **Jday**
  Day of the year
- **Tmax_C**
  Maximum daily temperature (degrees C)
- **Tmin_C**
  Minimum daily temperature (degrees C)
- **lat_radians**
  Latitude (radians = decimal degrees*Pi/180)
- **AvgT**
  Average daily temperature (degrees C) (if not known, will be taken as the averages of the daily extremes)
- **albedo**
  (-) average surface albedo. Can be expressed as a single value, or as a vector with the same length as Jday, Tmax_C and Tmin_C
- **TerrestEmiss**
  (-) Surface Emissivity - defaults to 0.97
- **aspect**
  (radians) Surface aspect
- **slope**
  (radians) average slope
- **forest**
  (-) Forest or shade cover (0-1). This modifies the amount of solar radiation reaching the location of interest. It should always set to zero for landscape-wide processes regardless of the amount of forest present. Only change this if calculating PET under a canopy.
- **PTconstant**
  (-) Priestley-Taylor Constant, often 1.26
- **AEparams**
  Atmospheric Emissivity options. Defaults to linear approximation. To use Brutsaert option, include vapor pressure values (kPa)
Value

PET (potential evapotranspiration) in m

Note

We are assuming that the Ground heat flux on a daily time-step is zero

Author(s)

Josephine Archibald, M. Todd Walter

References


Examples

data(OwascoInlet)
head(OwascoInlet)
attach(OwascoInlet)

PETapprox <- PET_fromTemp(Jday=(1+as.POSIXlt(date)$yday), Tmax_C=Tmax_C,
Tmin_C=Tmin_C, lat_radians=42.45*pi/180)
plot(PETapprox*1000~date, type="l")
detach(OwascoInlet)

PotentialSolar

Potential Solar Radiation

Description

Potential solar radiation at the edge of the atmosphere [kJ m-2 d-1]

#lat: latitude [rad] #Jday: Julian date or day of the year [day]

Usage

PotentialSolar(lat, Jday)

Arguments

lat latitude in radians

Jday Day of the year
Description

Potential Solar Radiation at a particular time of day. Defaults to W/m², can also report in kJ/m²/d if units set to kJ/m²/d

Usage

```
PotSolarInst(jday, hour = 12, lat = 42.44 * pi/180, sunrise = NULL, sunset = NULL, SolarNoon = mean(c(sunrise, sunset)), units = "W/m²", latUnits = "unknown")
```

Arguments

- **jday**  
  Day of the year [-]
- **hour**  
  Time of the day in hours [0-24 hr]
- **lat**  
  Latitude. Default is radians, but will automatically convert from degrees if value is larger than 1.62 or less than -1.62
- **sunrise**  
  Time of sunrise used to calculate solar noon [0-24 hr]
- **sunset**  
  Time of sunset used to calculate solar noon [0-24 hr]
- **SolarNoon**  
  Time of solar noon. Can be calculated from sunrise and sunset times. [hr]
- **units**  
  Units of the result. Defaults to W/m²
- **latUnits**  
  Latitude units can be explicitly stated here, options are 'radians', 'degrees' or default is 'unknown', which will assume radians unless the absolute value of lat is greater than pi/2

Author(s)

Josephine Archibald

References


Examples

```
PotSolarInst(jday=150, hour = 15, lat = 42.44, SolarNoon = 12.5)
PotSolarInst(jday=c(1,50,100,150), hour = c(9,10,12,17), lat = -pi/4, SolarNoon = 12.5)
```
**PTpet**

*Priestley-Taylor potential evapotranspiration*

**Description**

Returns potential evapotranspiration in m/day calculated with the Priestley-Taylor equation

**Usage**

\[
\text{PTpet}(\text{Rn}, \text{T}_C, \text{PTconstant} = 1.26)
\]

**Arguments**

- \( \text{Rn} \): Net daily radiation [kJ/m²/d]
- \( \text{T}_C \): Average daily temperature [C]
- \( \text{PTconstant} \): Priestley-Taylor constant, usually 1.26 [-]

**Author(s)**

Josephine Archibald

**References**


---

**RainHeat**

*Heat from Rain*

**Description**

Temperature added to the land from heat exchange with rain (usually in the context of snowmelt) [kJ m⁻² d⁻¹]

**Usage**

\[
\text{RainHeat}(\text{airtemp, rain})
\]

**Arguments**

- \( \text{airtemp} \): average daily air temperature [C]
- \( \text{rain} \): depth of rainfall [m]

**Examples**

\[
\text{RainHeat}(20, 0.01)
\]
RunoffBreakdown  

*Daily overland runoff breakdown*

**Description**

Determines the percentage of runoff that occurs on each day following a storm event, based on time to peak of a watershed.

**Usage**

\[
\text{RunoffBreakdown}(\text{Tp}\_\text{hr}, a = 4.5, \text{HrPrcDelay} = 4, \text{numDaysReturn} = 5)\
\]

**Arguments**

- **Tp\_hr**: Time to peak (hr).
- **a**: The ratio of the time to recession to the time to peak. Default is 4.5.
- **HrPrcDelay**: Lag time (hr).
- **numDaysReturn**: Number of days to return.

**Value**

returns a vector corresponding to the amount of runoff per day

**Author(s)**

J Archibald

**Examples**

\[
\text{RunoffBreakdown}(5)\
\]

SatVaporDensity  

*Saturated Vapor Density*

**Description**

Saturated vapor density [kg/m^3] at a given temperature

**Usage**

\[
\text{SatVaporDensity}(\text{T}\_\text{C})\
\]

**Arguments**

- **T\_C**: temperature [C]
**SatVaporPressure**

**Author(s)**
Josephine Archibald, M. Todd Walter

**References**

**Examples**
SatVaporDensity(seq(-5,40, by=5))

---

**SatVaporPressure**  
*Saturated Vapor Pressure*

**Description**
Calculates the saturated vapor pressure (kPa) at a particular temperature (degrees Celcius)

**Usage**
SatVaporPressure(T_C)

**Arguments**
T_C

**Author(s)**
Josephine Archibald

**References**
Tetens (1930) : Tetens, V.O (1930) Uber einige meteorologische. Begriffe, Zeitschrift fur Geophysik. 6, 297-309
### SatVapPressSlope

**Slope of the relationship between Saturation Vapor Pressure and Temperature**

**Description**

calculates the slope of the Saturation Vapor Pressure vs T (kPa/K)

**Usage**

SatVapPressSlope(temp_C)

**Arguments**

temp_C  
Air temperature in degrees C

**Author(s)**

Josephine Archibald

**References**


**Examples**

```r
## The function is currently defined as
function (temp_C)
{
  (2508.3/(temp_C + 237.3)^2) * exp(17.3 * temp_C/(temp_C + 237.3))
}
```

### SensibleHeat

**Sensible Heat Exchange**

**Description**

Sensible heat exchange between a surface and the surrounding air [kJ m⁻² d⁻¹]

**Usage**

SensibleHeat(surftemp, airtemp, wind)
setup_swatcal

Arguments

surftemp  surface temperature [C]
airtemp   average daily air temperature [C]
wind      average daily windspeed [m/s]

Author(s)

M. Todd Walter

Examples

```bash
# Should be DIRECTLY executable !! ----
#-- => Define data, use random,
#-- or do help(data=index) for the standard data sets.

# The function is currently defined as
function(surftemp,airtemp,wind){
    # sensible heat exchange between a surface and the surrounding air [kJ m^{-2} d^{-1}]

    #surftemp: surface temperature [C]
    #airtemp: average daily air temperature [C]
    #wind: average daily windspeed [m/s]

    latentht<-2500 #latent heat of vaporization [kJ kg^{-1}]
    heatcapacity<-1.25 #approx. heat capacity of air [kJ m^{-3} C^{-1}]
    windfunction<-5.3*(1+wind)

    return(86400*heatcapacity*(surftemp-airtemp)*windfunction/latentht)
}
```

Description

A function to setup a swat calibration, building the template *.unixorig template files, and cleaning up oddities we find in the default input files. (eg file.cio has cst.cst file which appears to be a bug)

Usage

```bash
setup_swatcal(change_params)
```

Arguments

change_params
Author(s)

Daniel Fuka drf28@cornell.edu

Examples

```r
#----- Should be DIRECTLY executable !! -----
#-- ==> Define data, use random,
#--or do help(data=index) for the standard data sets.

# The function is currently defined as
function(change_params){
  library/operators)
  for(ft in unique(change_params$filetype))(
    print(ft)
    files=list.files(.paste(ft,$",sep=""))
    for (file in files) {
      junk%<%file
      junk=gsub("\r",",junk)
      file_swatcal=paste(file,".unixorig",sep="");
      cat(junk,file=file_swatcal,sep="\n")
    }
  }
}
```

---

slopeatfactor Slope Factor

Description

Adjusts solar radiation for land slope and aspect relative to the sun, 0=level ground

Usage

`slopeatfactor(lat, Jday, slope, aspect)`

Arguments

- `lat`: latitude [rad]
- `Jday`: Julian date or day of the year [day]
- `slope`: slope of the ground [rad]
- `aspect`: ground aspect [rad from north]

Author(s)

M. Todd Walter, J. Archibald

References

Monteith and Unsworth (1990) Principles of Environmental Physics, 2nd ed. Chapter 5
**SnowMelt**

**Snow Melt Calculator**

**Description**

Calculates snow pack accumulation and melt

**Usage**

```r
calculate_snow_melt <- function(date, precip_mm, Tmax_C, Tmin_C, lat_deg, slope = 0, aspect = 0, tempHt = 1, windHt = 2, groundAlbedo = 0.25, SurfEmissiv = 0.95, windSp = 2, forest = 0, startingSnowDepth_m = 0, startingSnowDensity_kg_m3=450)
```

**Arguments**

<table>
<thead>
<tr>
<th>Variable</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>date</td>
<td>Vector of dates (class Date or character) in this format: Y-m-d</td>
</tr>
<tr>
<td>precip_mm</td>
<td>Precipitation in mm</td>
</tr>
<tr>
<td>Tmax_C</td>
<td>Daily maximum temperature (degrees C)</td>
</tr>
<tr>
<td>Tmin_C</td>
<td>Daily minimum temperature (degrees C)</td>
</tr>
<tr>
<td>lat_deg</td>
<td>Degrees latitude</td>
</tr>
<tr>
<td>slope</td>
<td>Overall slope of area of interest</td>
</tr>
<tr>
<td>aspect</td>
<td>Aspect of the area of interest</td>
</tr>
<tr>
<td>tempHt</td>
<td>height of temperature measurements (m)</td>
</tr>
<tr>
<td>windHt</td>
<td>height of wind measurements (m)</td>
</tr>
<tr>
<td>groundAlbedo</td>
<td>Ground Albedo, 0-1 ( - )</td>
</tr>
<tr>
<td>SurfEmissiv</td>
<td>Surface Emissivity, 0-1 ( - )</td>
</tr>
<tr>
<td>windSp</td>
<td>Wind speed - either a vector of measured values or a single value of average wind speed for the site (m/s)</td>
</tr>
<tr>
<td>forest</td>
<td>Forest cover (shade) - use this only when determining snowmelt under a canopy, 0-1 (-)</td>
</tr>
<tr>
<td>startingSnowDepth_m</td>
<td>The depth of the snow pack initially (m)</td>
</tr>
<tr>
<td>startingSnowDensity_kg_m3</td>
<td>The density of snow on the ground on the first day (kg/m3)</td>
</tr>
</tbody>
</table>

**Value**

This will return a 10 column data frame with nrow = length of input weather data. Column names are: Date, MaxT_C, MinT_C, Precip_mm, Rain_mm, SnowfallWatEq_mm, SnowMelt_mm, NewSsnow_m, SnowDepth_m, SnowWaterEq_mm
Warning

This function cannot handle NA values in input, and can only be run for continuous daily data. For data-sets missing values, run discontinuous segments separately.

Author(s)

Josephine Archibald, M. Todd Walter

References


Examples

```r
###
data(OwascoInlet)
sm <- SnowMelt(Date=OwascoInlet$date, precip_mm=OwascoInlet$P_mm,
        Tmax_C=OwascoInlet$Tmax_C, Tmin_C=OwascoInlet$Tmin_C, lat_deg=42)
sm
```

SoilStorage

Calculating S in the Curve Number Equation from soil water content

Description

This function calculates S, used in the SCS-CN equation, from the water content of the soil.

Usage

`SoilStorage(S_avg, field_capacity, soil_water_content, porosity)`

Arguments

- `S_avg`: Average S, as used normally in the CN equation, calculated from the curve number, which is based on land-use. This is in units of depth, often mm or inches
- `field_capacity`: field capacity - the amount of water that a soil can hold after drainage. (fraction)
- `soil_water_content`: Soil water content (fraction) on a given day.
- `porosity`: Saturated water content, approximately equal to the porosity of a soil (fraction)

Note

This equation is not the same as the one used in SWAT. It was given to Dr. Todd Walter by Dr. Keith E. Saxton
**Author(s)**

Josephine Archibald

**See Also**

SoilStorageSWAT

**Examples**

```r
SoilStorage(S_avg=120, field_capacity=0.2, soil_water_content=0.1, porosity=0.3)
```

---

**Solar**

**Solar Radiation**

**Description**

Solar radiation at the ground surface [kJ m\(^{-2}\) d\(^{-1}\)]

**Usage**

```r
Solar(lat, Jday, Tx, Tn, albedo=0.2, forest=0, slope=0, aspect = 0, 
units="kJm2d", latUnits = "unknown", printWarn=TRUE)
```

**Arguments**

- **lat** latitdue [rad]
- **Jday** Julian date or day of the year [day]
- **Tx** maximum daily temperature [C]
- **Tn** minimum daily temperature [C]
- **albedo** surface albedo or reflectivity [-]
- **forest** forest or vegeation cover [-]
- **slope** slope of the ground [rad]
- **aspect** ground aspect [rad from north]
- **units** Units of the result. Defaults to kJ/m2/d, changing this to "Wm2" will make output in W/m2
- **latUnits** Latitude units can be explicitly stated here, options are 'radians', 'degrees' or default is 'unknown', which will assume radians unless the absolute value of lat is greater than pi/2
- **printWarn** Will print a warning about latitude if set to TRUE
**solarangle**  
_Solar Angle_

**Description**

Angle of solar inclination from horizontal at solar noon [rad]

**Usage**

`solarangle(lat, Jday)`

**Arguments**

- `lat`  
  Latitude [rad]
- `Jday`  
  Julian date or day of the year [day]

**Examples**

```plaintext
### Should be DIRECTLY executable !! ----
### =@ Define data, use random,
### or do help(data=index) for the standard data sets.

### The function is currently defined as
function(lat,Jday){
  # angle of solar inclination from horizontal at solar noon [rad]
  #lat: latitude [rad]
  #Jday: Julian date or day of the year [day]
  # solar declination [rad]
  dec<-declination(Jday)
  return(asin(sin(lat)*sin(dec)+cos(lat)*cos(dec)*cos(0)))
}
```

**Description**

This function runs the SWAT2005 executable in the current directory.

**Usage**

`SWAT2005(hist_wx,elev,rch)`
**Arguments**

- hist_wx  
  Describe hist_wx
- elev  
  Describe elev
- rch  
  Describe rch

**Author(s)**

Daniel R. Fuka

---

**swat_general**

All files required for a general SWAT run.

---

**Description**

All files required for a general SWAT run. This initialization has 3 subbasin init, with 3 HRUs per subbasin.

**Usage**

data(swat_general)

**Format**

The format is: List of 74 files required for a general SWAT run. This initialization has 3 subbasin init, with 3 HRUs per subbasin.
DEEPST : Initial depth of water in the deep aquifer [mm] " 31 | GW_DELAY : Groundwater delay [days]" ... $ 00010001.hru file Subbasin: 2 HRU: 1 Luse: SHRB Soil: Kl15-2ab-5212 Slope: 0-5 Monday, March 21, 2011 6:57:24 PM MapWindow - SWAT intei"l__truncated__ "0.347771714415945 | HRU_FR : Fraction of subbasin area contained in HRU" " 90 | SLSUBBSN : Average slope length [m] " 0.0260677934475378 | HRU_SLP : Average slope stepness [m/m]" ... $ 000010001.mgt file Subbasin: 2 HRU: 1 Luse: SHRB Soil: Kl15-2ab-5212 Slope: 0-5 Monday, March 21, 2011 6:57:24 PM MapWindow - SWAT intei"l__truncated__ "0 | NMGT: Management code" "Initial Plant Growth Parameters" " 0 | IGRO: Land cover status: 0-none growing; 1-growing" ... $ 000010001.sol file Subbasin: 2 HRU: 1 Luse: SHRB Soil: Kl15-2ab-5212 Slope: 0-5 Monday, March 21, 2011 6:57:24 PM MapWindow - SWAT intei"l__truncated__ "Soil Name: Kl15-2ab-5212" "Soil Hydrologic Group: C" "Maximum rooting depth(m): 580.00" ... $ 000010002.chm file Subbasin: 2 HRU: 2 Luse: SHRB Soil: Kl30-2bc-4832 Slope: 0-5 Monday, March 21, 2011 6:57:24 PM MapWindow - SWAT intei"l__truncated__ "Soil Nutrient Data" " Soil Layer : 1 2 3 4 5 6 7 8 "l__truncated__ " Soil NO3 [mg/kg] : 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 "l__truncated__ ... $ 000010002.mgt file Subbasin: 2 HRU: 2 Luse: SHRB Soil: Kl30-2bc-4832 Slope: 0-5 Monday, March 21, 2011 6:57:24 PM MapWindow - SWAT intei"l__truncated__ "0 | NMGT: Management code" "Initial Plant Growth Parameters" " 0 | IGRO: Land cover status: 0-none growing; 1-growing" ... $ 000010002.sol file Subbasin: 2 HRU: 2 Luse: SHRB Soil: Kl30-2bc-4832 Slope: 0-5 Monday, March 21, 2011 6:57:24 PM MapWindow - SWAT intei"l__truncated__ "Soil Name: Kl30-2bc-4832" "Soil Hydrologic Group: C" "Maximum rooting depth(m): 580.00" ... $ 000010003.chm file Subbasin: 2 HRU: 3 Luse: GRAS Soil: Vp38-3a-5345 Slope: 0-5 Monday, March 21, 2011 6:57:24 PM MapWindow - SWAT intei"l__truncated__ "Soil Nutrient Data" " Soil Layer : 1 2 3 4 5 6 7 8 "l__truncated__ " Soil NO3 [mg/kg] : 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 "l__truncated__ ... $ 000010003.mgt file Subbasin: 2 HRU: 3 Luse: GRAS Soil: Vp38-3a-5345 Slope: 0-5 Monday, March 21, 2011 6:57:24 PM MapWindow - SWAT intei"l__truncated__ "0 | NMGT: Management code" "Initial Plant Growth Parameters" " 0 | IGRO: Land cover status: 0-none growing; 1-growing" ... $ 000010003.sol file Subbasin: 2 HRU: 3 Luse: GRAS Soil: Vp38-3a-5345 Slope: 0-5 Monday, March 21, 2011 6:57:24 PM MapWindow - SWAT inter"l__truncated__ "0.5 | SHALLST : Initial depth of water in the shallow aquifer [mm]" " 1000 | DEEPST : Initial depth of water in the deep aquifer [mm]" " 31 | GW_DELAY : Groundwater delay [days]" ... $ 000010003.hru file Subbasin: 2 HRU: 3 Luse: GRAS Soil: Vp38-3a-5345 Slope: 0-5 Monday, March 21, 2011 6:57:24 PM MapWindow - SWAT inter"l__truncated__ "Soil Name: Kl30-2bc-4832" "Soil Hydrologic Group: C" "Maximum rooting depth(m): 580.00" ... $ 000010003.chm file Subbasin: 2 HRU: 3 Luse: GRAS Soil: Vp38-3a-5345 Slope: 0-5 Monday, March 21, 2011 6:57:24 PM MapWindow - SWAT inter"l__truncated__ "Soil Nitrogen Data" " Soil Layer : 1 2 3 4 5 6 7 8 "l__truncated__ " Soil NO3 [mg/kg] : 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 "l__truncated__ ... $ 000010003.mgt file Subbasin: 2 HRU: 3 Luse: GRAS Soil: Vp38-3a-5345 Slope: 0-5 Monday, March 21, 2011 6:57:24 PM MapWindow - SWAT inter"l__truncated__ "0 | NMGT: Management code" "Initial Plant Growth Parameters" " 0 | IGRO: Land cover status: 0-none growing; 1-growing" ... $ 000010003.sol file Subbasin: 2 HRU: 3 Luse: GRAS Soil: Vp38-3a-5345 Slope: 0-5 Monday, March 21, 2011 6:57:24 PM MapWindow - SWAT inter"l__truncated__ "Soil Name: Vp38-3a-5345" "Soil Hydrologic Group: C" "Maximum rooting depth(m): 910.00" ... $ 000020000.pnd file Subbasin: 2 HRU: 3 Luse: GRAS Soil: Vp38-3a-5345 Slope: 0-5 Monday, March 21, 2011 6:57:24 PM MapWindow - SWAT interface " Pond inputs: " 0 | PND.FR : Fraction of subbasin
area that drains into ponds. The value for PND_FR should be between 0.0 and 1.0.

PND_PSA: Surface area of ponds when filled to principal spillway [ha]

PND_PSA: Surface area of ponds when filled to principal spillway [ha]

CH_W2: Main channel width [m]

CH_D: Main channel depth [m]

CH_S2: Main channel slope [m/m]

RS1: Local algal settling rate in the reach at 20 degC [m/day]

RS2: Benthic (sediment) source rate for dissolved phosphorus in the reach at 20 degC [mg dissolved P/m²/day]

LATITUDE = 20.00

LONGITUDE = 0.00

ELEV [m] = 2000.00

RAIN_YRS = 10.00

HRU_MGT: Management code

IGRO: Land cover status: 0-none growing; 1-growing

HRU_SLP: Average slope stepness [m/m]

HRU_SHP: Initial depth of water in the shallow aquifer [mm]

DEEPST: Initial depth of water in the deep aquifer [mm]
...
swat_objective_function

A simple example objective function to be modified by user

Description

A simple example objective function to be modified by user

Usage

swat_objective_function(x, calib_range, calib_params, flowgage)

Arguments

x

A vector of adjustment factors to be fed into alter_params function
calib_range

calib_range This will in the future be a range for calibration.
calib_params

See other function to make sure you have this right.
flowgage

See other function to make sure you have this right.

Author(s)

Daniel Fuka

Examples

## Not run:
Sorry, this should be custom to your project. Look and determine what you want.

## End(Not run)
swat_objective_function_rch

An example objective function for calibrating SWAT2005 model.

**Description**

An example objective function for calibrating SWAT2005 model.

**Usage**

```r
swat_objective_function_rch(x, calib_range, calib_params, flowgage, rch, save_results = F)
```

**Arguments**

- `x` Numeric vector similar to `change_params$current` defining the scalar states of the parameters to be optimized.
- `calib_range` 2 place vector of begin and end dates for calibration.
- `calib_params` Dataframe in the format of `change_params` limited to those rows of parameters desired to calibrate. See `change_params`.
- `flowgage` list of the format of the list returned by the function `get_usgs_gage`.
- `rch` Number of the reach in the output.rch file you desire to calibrate against
- `save_results` Logical to express if you want to copy the optimal solution to the current directory, overwriting the swat input files.

**Value**

`abs\((NS - 1)\)` as needed to be able to minimize to optimal as needed for DEoptim function.

**Author(s)**

Daniel R. Fuka

testSWAT2005

A function to test the numerical correctness of the SWAT2005 exe due to bad results on some CPUs.

**Description**

A function to test the numerical correctness of the SWAT2005 exe due to bad results on some CPUs.

**Usage**

```r
testSWAT2005()
```

**Author(s)**

Daniel Fuka drf28@cornell.edu
transmissivity  

**Description**

Transmissivity Fraction of direct solar radiation passing through the atmosphere based on the Bristow-Campbell eqn

**Usage**

`transmissivity(Tx,Tn, A=0.75, C=2.4, opt="1day", JD=NULL)`

**Arguments**

- **Tx**: maximum daily temperature [C]
- **Tn**: minimum daily temperature [C]
- **A**: Maximum transmissivity in a location, varies with environmental conditions and elevation
- **C**: Empirical partitioning coefficient - set to 2.4
- **opt**: Options: "1day" uses the diurnal temperature change as Tx-Tn for any given day; "2day" depends on an average temperature change, i.e. the max temp today minus the average of the min temps of today and tomorrow; Users who do not have a continuous record of daily temperatures should use "missingdays" - note that this depends on JD input. If JD is null and "missingdays" is chosen the function will default to "1day"
- **JD**: A vector of julian days corresponding to temp measurements. Only needed if using opt="missingdays"

**Details**

Can accept either single values or vectors for Tx,Tn and JD.

**Author(s)**

M. Todd Walter, Josephine Archibald

**References**

Examples

## Assuming the temperature measurements are consecutive days, default option:
```r
transmissivity(Tx=c(10,10,10,10,10), Tn=c(1,2,6,9,2))
```

## Second option:
```r
transmissivity(Tx=c(10,10,10,10,10), Tn=c(1,2,6,9,2), opt="2day")
```

## When the days are not consecutive:
```r
transmissivity(Tx=c(10,10,10,10,10), Tn=c(1,2,6,9,2), JD=c(250,265,300,320,321), opt="missingdays")
```
Index

* Topic \textasciitilde kwd1
  GroundHeat, 18
* Topic \textasciitilde kwd2
  GroundHeat, 18
* Topic datasets
  change_params, 8
  GSOD_history, 19
  OwascoInlet, 25
* Topic package
  EcoHydRology-package, 2

alter_files, 3
AtmosphericEmissivity, 4

BaseflowSeparation, 5
build_gsod_forcing_data, 6

calib_swat_ex, 7
change_params, 8
ConvertFlowUnits, 9

deciliation, 10

EcoHydRology (EcoHydRology-package), 2
EcoHydRology-package, 2
EnvirEnergy, 11
EstCloudiness, 12
EvapHeat, 13

get_cfsr_latlon, 14
get_gsod_stn, 15
get_usgs_gage, 17
GroundHeat, 18
GSOD_history, 19

hydrograph, 20

Longwave, 21
Lumped_VSA_model, 22

NetRad, 24

OwascoInlet, 25
PET_fromTemp, 26
PotentialSolar, 27
PotSolarInst, 28
PTpet, 29
RainHeat, 29
RunoffBreakdown, 30
SatVaporDensity, 30
SatVaporPressure, 31
SatVapPresSlope, 32
SensibleHeat, 32
setup_swatcal, 33
slopefactor, 34
SnowMelt, 35
SoilStorage, 36
Solar, 37
solarangle, 38
SWAT2005, 38
swat_general, 39
swat_objective_function, 44
swat_objective_function_rch, 45

testSWAT2005, 45
transmissivity, 46