Package ‘dynatop’

January 18, 2022

Title An Implementation of Dynamic TOPMODEL Hydrological Model in R

Version 0.2.1

Description An R implementation and enhancement of the Dynamic TOPMODEL semi-distributed hydrological model originally proposed by Beven and Freer (2001) \<doi:10.1002/hyp.252\>. The ‘dynatop’ package implements code for simulating models which can be created using the ‘dynatopGIS’ package.

License GPL-2

Encoding UTF-8

LazyData true

ByteCompile true

Imports R6, zoo, xts, Rcpp

LinkingTo Rcpp

Depends R (>= 4.0.0)

BugReports https://github.com/waternumbers/dynatop/issues

URL https://waternumbers.github.io/dynatop/,

https://github.com/waternumbers/dynatop

RoxygenNote 7.1.2

Suggests raster, rgdal, knitr, rmarkdown, covr, bookdown, pkgdown, testthat (>= 3.0.0)

VignetteBuilder knitr

NeedsCompilation yes

Author Paul Smith [aut, cre] (<https://orcid.org/0000-0002-0034-3412>),

Peter Metcalfe [aut]

Maintainer Paul Smith <paul@waternumbers.co.uk>

Repository CRAN

Date/Publication 2022-01-18 08:02:42 UTC
**R topics documented:**

- dynatop-package .................................................. 2
- dynatop .............................................................. 2
- evap_est ............................................................... 6
- resample_xts ......................................................... 7
- Swindale .............................................................. 8

**Index**

<table>
<thead>
<tr>
<th>dynatop-package</th>
<th>dynatop</th>
</tr>
</thead>
</table>

**Description**

This package contains the core code for the R implementation of dynamic TOPMODEL

**dynatop**

**R6 Class for Dynamic TOPMODEL**

**Description**

R6 Class for Dynamic TOPMODEL

R6 Class for Dynamic TOPMODEL

**Methods**

**Public methods:**

- `dynatop$new()`
- `dynatop$add_data()`
- `dynatop$clear_data()`
- `dynatop$initialise()`
- `dynatop$sim_hillslope()`
- `dynatop$sim_channel()`
- `dynatop$sim()`
- `dynatop$get_channel_inflow()`
- `dynatop$plot_channel_inflow()`
- `dynatop$get_gauge_flow()`
- `dynatop$plot_gauge_flow()`
- `dynatop$get_obs_data()`
- `dynatop$get_model()`
- `dynatop$get_mass_errors()`
- `dynatop$get_states()`
• `dynatop$plot_state()`
• `dynatop$clone()`

**Method `new()`:** Creates a `dynatop` class object from the a list based model description as generated by `dynatopGIS`.

*Usage:*
```
dynatop$new(model, use_states = FALSE, delta = 1e-13)
```

*Arguments:*
- `model` a dynamic TOPMODEL list object
- `use_states` logical if states should be imported
- `delta` error term in checking redistribution sums
- `drop_map` logical if the map should be dropped

*Details:* This function makes some basic consistency checks on a list representing a dynamic TOPMODEL model. The checks performed and basic ‘sanity’ checks. They do not check for the logic of the parameter values nor the consistency of states and parameters. Sums of the redistribution matrices are checked to be in the range 1 +/- delta.

*Returns:* `invisible(self)` suitable for chaining

**Method `add_data()`:** Adds observed data to a `dynatop` object

*Usage:*
```
dynatop$add_data(obs_data)
```

*Arguments:*
- `obs_data` an xts object of observed data

*Details:* This function makes some basic consistency checks on the observations to ensure they have uniform timestep and all required series are present.

*Returns:* `invisible(self)` suitable for chaining

**Method `clear_data()`:** Clears all forcing and simulation data except current states

*Usage:*
```
dynatop$clear_data()
```

*Returns:* `invisible(self)` suitable for chaining

**Method `initialise()`:** Initialises a `dynatop` object in the simpliest way possible.

*Usage:*
```
dynatop$initialise(tol = 2 * .Machine$double.eps, max_it = 1000)
```

*Arguments:*
- `tol` tolerance for the solution for the saturated zone
- `max_it` maximum number of iterations to use in the solution of the saturated zone

*Returns:* `invisible(self)` suitable for chaining

**Method `sim_hillslope()`:** Simulate the hillslope output of a `dynatop` object

*Usage:*
```
dynatop$sim_hillslope()
```
dynatop$sim_hillslope(
  keep_states = NULL,
  sub_step = NULL,
  tol = 2 * .Machine$double.eps,
  max_it = 1000
)

Arguments:

keep_states a vector of POSIXct objects (e.g. from xts) giving the time stamp at which the states should be kept
sub_step simulation timestep in seconds, default value of NULL results in data time step
tol tolerance for the solution for the saturated zone
max_it maximum number of iterations to use in the solution of the saturated zone

Details: Both saving the states at every timestep and keeping the mass balance can generate very large data sets!!

Method sim_channel(): Simulate the channel output of a dynatop object

Usage:

dynatop$sim_channel()

Returns: invisible(self) for chaining

Method sim(): Simulate the hillslope and channel components of a dynatop object

Usage:

dynatop$sim(
  keep_states = NULL,
  sub_step = NULL,
  tol = 2 * .Machine$double.eps,
  max_it = 1000
)

Arguments:

keep_states a vector of POSIXct objects (e.g. from xts) giving the time stamp at which the states should be kept
sub_step simulation timestep in seconds, default value of NULL results in data time step
tol tolerance for the solution for the saturated zone
max_it maximum number of iterations to use in the solution of the saturated zone
mass_check Flag indicating is a record of mass balance errors shuld be kept

Details: Calls the sim_hillslope and sim_channel in sequence. Both saving the states at every timestep and keeping the mass balance can generate very large data sets!!

Returns: invisible(self) for chaining

Method get_channel_inflow(): Return channel inflow as an xts series or list of xts series

Usage:

dynatop$get_channel_inflow(total = FALSE, separate = FALSE)

Arguments:
Method plot_channel_inflow(): Plot the channel inflow

Usage:
dynatop$plot_channel_inflow(total = FALSE, separate = FALSE)

Arguments:
total logical if total inflow is to be plotted
separate logical if the surface and saturated zone inflows should be plotted separately

Method get_gauge_flow(): Return flow at the gauges as an xts series

Usage:
dynatop$get_gauge_flow(gauge = colnames(private$time_series$gauge_flow))

Arguments:
gauge names of gauges to return (default is all gauges)

Method plot_gauge_flow(): Get the flow at gauges

Usage:
dynatop$plot_gauge_flow(gauge = colnames(private$time_series$gauge_flow))

Arguments:
gauge names of gauges to return (default is all gauges)

Method get_obs_data(): Get the observed data

Usage:
dynatop$get_obs_data()

Method get_model(): Return the model

Usage:
dynatop$get_model()

Method get_mass_errors(): Return the model

Usage:
dynatop$get_mass_errors()

Method get_states(): Return states

Usage:
dynatop$get_states(record = FALSE)

Arguments:
record logical TRUE if the record should be returned. Otherwise the current states returned

Method plot_state(): Plot a current state of the system

Usage:
dynatop$plot_state(state, add_channel = TRUE)
Arguments:
state the name of the state to be plotted
add_channel Logical indicating if the channel should be added to the plot

Method clone(): The objects of this class are cloneable with this method.

Usage:
dynatop$clone(deep = FALSE)

Arguments:
deep Whether to make a deep clone.

Examples
## the vignettes contains further details of the method calls.
data("Swindale") # example data
cch_mdl <- dynatop$new(Swindale$model) # create with model
cch_mdl$add_data(Swindale$obs) # add observations
cch_mdl$initialise() # initialise model
cch_mdl$sim() # simulate model

---

**evap_est**

Create sinusoidal time series of potential evapotranspiration input

Description

Generate series of potential evapotranspiration

Usage

evap_est(ts, eMin = 0, eMax = 0)

Arguments

ts as vector of POSIXct data/times
eMin Minimum daily PE total (m or mm)
eMax Maximum daily PE total (m or mm)

Details

Dynamic TOPMODEL requires a time series of potential evapotranspiration in order to calculate and remove actual evapotranspiration from the root zone during a run. Many sophisticated physical models have been developed for estimating PE and AE, including the Priestly-Taylor (Priestley and Taylor, 1972) and Penman-Monteith (Montieth, 1965) methods. These, however, require detailed meteorological data such as radiation input and relative humidities that are, in general, difficult to obtain. Calder (1983) demonstrated that a simple approximation using a sinusoidal variation in potential evapotranspiration to be a good approximation to more complex schemes.
If the insolation is also taken to vary sinusoidally through the daylight hours then, ignoring diurnal meteorological variations, the potential evapotranspiration during daylight hours for each year day number can be calculated (for the catchment’s latitude). Integration over the daylight hours allows the daily maximum to be calculated and thus a sub-daily series generated.

Value

Time series (xts) of potential evapotranspiration totals for the timesteps given in same units as eMin and eMax

References


Examples

```r
## Generating daily PET data for 1970
## the values of eMin and eMax may not by not be realistic
st <- as.POSIXct("1970-01-01 00:00:00",tz="GMT")
fn <- as.POSIXct("1971-01-01 00:00:00",tz="GMT")
daily_ts <- seq(st,fn,by=24*60*60)
dpet <- evap_est(daily_ts,0,1)

## create hourly data for the same period
st <- as.POSIXct("1970-01-01 01:00:00",tz="GMT")
fn <- as.POSIXct("1971-01-01 00:00:00",tz="GMT")
hour_ts <- seq(st,fn,by=1*60*60)
hpet <- evap_est(hour_ts,0,1)

## the totals should be the same...
stopifnot(all.equal(sum(hpet), sum(dpet)))
```

resample_xts

Functions to resample an xts time series

Description

Takes an xts time series object and resamples then to a new time step.

Usage

deval_xts(x, dt, is.rate = FALSE)
Arguments

- obs: A times series (xts) object with a POSIXct index.
- dt: New time interval in seconds.
- is.rate: If TRUE then these are rates i.e m/hr. Otherwise they are absolute values accumulated within the preceding time interval. Values are scaled before returning so resampling is conservative.

Details

Time series of observation data are often of different temporal resolutions, however the input to most hydrological models, as is the case with the Dynamic TOPMODEL, requires those data at the same interval. This provides a method to resample a collection of such data to a single interval.

Because of the methods used the results:
- are not accurate when the input data does not have a constant timestep. The code issued a warning and proceeds assuming the data are equally spaced with the modal timestep.
- do not guarantee the requested time step but returns a series with the timestep computed from an integer rounding the ratio of the current and requested time step.

Value

An xts object with the new timestep

Examples

```r
# Resample Swindale Rainfall to hourly intervals
require(dynatop)
data("Swindale")
obs <- Swindale$obs
cobs <- resample_xts(obs, dt=60*60) # hourly data
dobs <- resample_xts(cobs,dt=15*60) # back to 15 minute data
cdobs <- resample_xts(dobs,dt=60*60) # back to hourly data - checks time stamp conversion
obs <- obs[zoo::index(obs)<=max(zoo::index(cobs)),]
# check totals
stopifnot( all.equal(sum(obs),sum(cobs)) )
stopifnot( all.equal(sum(obs),sum(dobs)) )
stopifnot( all.equal(cobs,cdobs) )
```

Description

This data set contains a processed model and observation data for Swindale.

Usage

data(Swindale)
Swindale

Format

An object of class list of length 2.

See Also

dynatop

Examples

require(dynatop)
data(Swindale)

# Show it
# plot(obs)
Index

* datasets
  Swindale, 8

dynatop, 2, 9
dynatop-package, 2
evp_est, 6
resample_xts, 7
Swindale, 8