Package ‘hydroState’

November 27, 2024

Title Hidden Markov Modelling of hydrological state change

Version 0.2.0.0

Maintainer Tim Peterson <tim.peterson@monash.edu>

Depends R (>= 3.4.2)

Description HydroState identifies regime changes in streamflow runoff not explained by variations in precipitation. The package allows a flexible set of Hidden Markov Models of annual, seasonal or monthly streamflow runoff to be built that includes precipitation as a predictor of runoff. Suites of models can be built for a single site, ranging from one to three states and each with differing combinations of error models and autocorrelation terms, allowing the most parsimonious model (by AIC) to easily be identified. The entire package is written in R S4 object oriented code and accesible with specific functions for users. See Peterson TJ, Saft M, Peel MC & John A (2021), Watersheds may not recover from drought, Science, DOI: 10.1126/science.abd5085.

Imports methods, DEoptim, sn, truncnorm, diagram, padr, zoo

BugReports https://github.com/peterson-tim-j/HydroState/issues

URL https://github.com/peterson-tim-j/HydroState

License GPL-3

Encoding UTF-8

LazyData true

RoxygenNote 7.3.1

ByteCompile true

Collate 'abstracts.R'
'parameters.R'
'Qhat.boxcox.R'
'Qhat.burbidge.R'
'Qhat.log.R'
'Qhat.none.R'
'QhatModel.homo.normal.linear.R'
'QhatModel.homo.normal.linear.AR1.R'
'QhatModel.homo.gamma.linear.R'
'QhatModel.homo.gamma.linear.AR1.R'
'QhatModel.homo.normal.linear.AR2.R'
'QhatModel.homo.gamma.linear.AR2.R'
'QhatModel.homo.normal.linear.AR3.R'
'QhatModel.homo.gamma.linear.AR3.R'

'QhatModel.homo.skewedNormal.linear.R'
'QhatModel.homo.skewedNormal.linear.AR1.R'
'QhatModel.homo.skewedNormal.linear.AR2.R'
'QhatModel.homo.skewedNormal.linear.AR3.R'
'QhatModel.subAnnual.homo.gamma.linear.R'
'QhatModel.subAnnual.homo.gamma.linear.AR1.R'
'QhatModel.subAnnual.homo.gamma.linear.AR2.R'
'QhatModel.subAnnual.homo.gamma.linear.AR3.R'
'markov.annualHomogeneous.R'
'hydroState.R'
'hydroState.allModels.R'
'hydroState.subAnnual.allModels.R'
'markov.annualHomogeneous.flickering.R'
'wrapper.R'

Suggests knitr,
rmarkdown,
testthat (>= 3.0.0)

Config/testthat/edition 3

VignetteBuilder knitr

R topics documented:

`hydroState-package` *Overview of methods and procedures*

Description

`hydroState` provides methods to construct and evaluate hidden Markov models (HMM) of annual, seasonal, or monthly streamflow runoff with precipitation as a predictor. The state of the relationship between these observations is evaluated overtime. The package contains a default `hydroState` model to evaluate probable shifts in the intercept of the rainfall-runoff relationship or shifts in other terms can be evaluated. The default model can also be expanded with various terms to better define the relationship as discussed below, but the general workflow is as follows. Once the default or constructed model is built (`buildModel`), the model is fitted (`fitModel`) to determine the most likly rainfall-runoff state at each time-step. To assess the adequacy of the fit, the residuals are plotted (`plot.residuals`), and an adequate fit requires the residuals to be normaly distributed and not have any correlation or trends. The resulting runoff states from the fitted model can then be evaluated overtime (`plot.states`) and even exported (`get.states`) with the state values, confidence intervals, and conditional probabilities of each state. Input data requires a dataframe with flow and precipitation at annual, seasonal, or monthly timesteps, and gaps with missing data are permitted.

A `hydroState` model operates in S4, object oriented programming, and constructing a model requires the selection of three objects. The first object is the `data.transform`, or 'Qhat' object, from `select.transform` that offers various transformaitons of the observations in order to reduce skew. These transformaitons include: 'boxcox', 'log', 'burbidge', or 'none'. The second object is the `state.model`, or 'QhatModel' object, from `select.stateModel` that offers a variety of items to better define the rainfall-runoff relationship including: the state number (1, 2, or 3), residual distribution ('normal', 'gamma', or 'truc.normal'), and degree of auto-correlation ('AR1', 'AR2', or 'AR3'). Furthermore `select.stateModel` provides an option to select the parameters in the model expected to shift (`state.shift.parameters`) and apply seasonal variation within model parameters (`seasonal.paraneters`) if monthly data is provided. The third object defines the form of the `Markov`, or 'markov' object, from `select.Markov` which currently only provides a homogeneous Markov model. Further details on the selection of options within these objects for constructing a `hydroState` model are explained within each topic.

There is an additional option to construct all possible types of models using the `buildModelAll`, and compare them using the same `fitModel` function. The most likely model is selected based on the AIC where the best model will have the lowest AIC. To begin, it is recommended to evaluate the default model at first with one state and again with two states. This documentation is organized with four sections that define the general workflow for using `hydroState`: Build - Fit - Review - Evaluate.

I. Build

`buildModel`	build hydroState model
`select.transform`	select transformation of observations
`select.stateModel`	select type of state model
`select.Markov`	select form of Markov
`buildModelAll`	build all possible models

II. Fit

`fitModel`	fit built hydroState model(s)

III. Review

`plot.residuals`	plot residuals
`get.residuals`	get residuals

IV. Evaluate

setInitialYear	set initial year for assigning state names
plot.states	plot states
get.states	get states

Authors

Except where indicated otherwise, the methods and functions in this package were written by Tim Peterson and Thomas Westfall.

Acknowledgments

...

---

buildModel *Builds hydroState model*

Description

`buildModel` builds a hydrostate model with either a default model or the model can be specified with options from below. Every model depends on a linear base model where streamflow is a function of precipitation. The default model is an analysis of this base linear model with state shifts expected in the intercept, a_0, and standard deviation, std, of the rainfall-runoff relationship. There are additional adjustments to this model with auto-correlations terms, seasonal parameters for a sub-annual analysis, and even evaluation of other `state.shift.parameters`. The number of states and assumed error distribution can also be selected. After the model is built, the hydroState model is ready to be fitted with `fitModel`

Usage

```
buildModel(
  input.data = data.frame(year = c(), flow = c(), precip = c()),
  data.transform = "boxcox",
  parameters = list("a0", "a1", "std"),
  seasonal.parameters = list(),
  state.shift.parameters = list("a0", "std"),
  error.distribution = "truc.normal",
  flickering = FALSE,
  transition.graph = matrix(TRUE, 2, 2)
)
```

Arguments

`input.data`	dataframe of annual, seasonal, or monthly runoff and precipitation observations. Gaps with missing data in either streamflow or precipitation are permitted. Monthly data is required when using `seasonal.parameters` that assumes selected model parameters are better defined with a sinusoidal function.

`data.transform` character string with the method of transformation. The default is 'boxcox'. Other options: 'log', 'burbidge', 'none'

`parameters` character list of parameters to construct model. Required and default: `a0`, `a1`, `std`. Auto-correlation terms optional: `AR1`, `AR2`, or `AR3`.

`seasonal.parameters` character list of one or all parameters (`a0`, `a1`, `std`) defined as a sinusoidal function to represent seasonal variation. Requires monthly or seasonal data. Default is empty list.

`state.shift.parameters` character list of one or all parameters (`a0`, `a1`, `std`, `AR1`, `AR2`, `AR3`) able to shift as dependent on state. Default is `a0` and `std`.

`error.distribution` character sting of the distribution in the HMM error. Default is 'truc.normal'. Others include: 'normal' or 'gamma'

`flickering` logical T/F. T = allows more sensitive markov flickering between states over time, F = less sensitive and is default.

`transition.graph` matrix given the number of states. Default is a 2-state matrix (2 by 2): matrix(TRUE,2,2)

Details

`buildModel`

There are a selection of items to consider when defining the rainfall-runoff relationship and investigating state shifts in this relationship. hydroState provides various options for modelling the rainfall-runoff relationship.

- Data gaps with `input.data`: When there is missing `input.data`, special care was taken to reduce the influence of the missing time periods while making the most of the given data without infilling. For time-periods where the dependent variable, streamflow, is missing, the transition probability for these missing periods is essentially ignored by setting the conditional probability of the missing time-steps equal to one. This results in the time-step after the missing period having the same probability of being in the given state as before the missing period. For time-periods where the independent variable is missing, precipitation, the missing periods are essentially ignored when fitting the models as the likelihood is calculated from only continuous periods with values for the independent variable. Since the log-likelihood is calculated for each continuous period, the sum of the log-likelihoods provides a total log-likelihood to fit models. Again, there is no infilling of missing data.
- Transform Observations with `data.transform`: Transforms observations to remove heteroscedasticity. Often there is skew within hydrologic data. When defining relationships between observations, this skew results in an unequal variance in the residuals, heteroscedasticity. Transforming observations is often required with observations of streamflow and precipitation. There are several options to transform observations. Since the degree of transformation is not typically known, 'boxcox' is the default. Other options include: 'log', 'burbidge', and of course, 'none' when no transformation is performed.
- Model Structure with `parameters` and `seasonal.parameters`: The structure of the model depends on the `parameters`. hydroState simulates runoff, Q, as being in one of finite states, i, at every time-step, t, depending on the distribution of states at prior time steps. This results in a runoff distribution for each state that can vary overtime $(\widehat{{}_tQ_i})$. The model defines the relationship that is susceptible to state shifts with precipitation, P_t, as a predictor. This takes the form as a simple linear model $\widehat{{}_tQ_i} = f(P_t)$:

$\widehat{{}_tQ_i} = P_t a_1 + a_0$
where a_0 and a_1 are constant parameters. These parameters and the model error, std, are required parameters for every model. It is possible the relationship contains serial correlation and would be better defined with an auto-regressive term:
$\widehat{{}_tQ_i} = P_t a_1 + a_0 + AR1\widehat{{}_{t-1}Q}$
where $AR1$ is the lag-1 auto-correlation term. Either, lag-1: $AR1$, lag-2: $AR2$, and lag-3: $AR3$ auto-correlation coefficients are an option as additional parameters to better define the rainfall-runoff relationship. For sub-annual analysis, `seasonal.parameters` provides the option to assume a sinusoidal function better defines either of the constant parameters or error (a_0, a_1, std) throughout the year, i.e:
$a_0 = a_{0.disp} + a_{0.amp} * sin(2\pi(\frac{M_t}{12} + a_{0.phase}))$
where M_t is an integer month at t. Monthly streamflow and precipitation are required as `input.data` for the sub-annual analysis.

- State Dependent Parameters with `state.shift.parameters`: These are state dependent parameters where they are subject to shift in order to better explain the state of streamflow over time. Any or all of the previously chosen parameters can be selected ($a_0, a_1, std, AR1, AR2, AR3$). The default model evaluates shifts in the rainfall-runoff relationship with a_0 std as state dependent parameters.
- Distribution of the Residuals with `error.distribution`: The distribution of the residuals (error) in the model can be chosen to reduce skew and assist with making models statistically adequate (see `plot.residuals`). Either normal: 'normal', truncated normal: 'truc.normal', or gamma: 'gamma' distributions are acceptable. The default is 'truc.normal'. Sub-annual models are restricted to only a 'gamma' distribution.
- Markov flickering with `flickering`: The form of the Markov model is homogeneous where the transition probabilities are time-invariant. The model be performed with or without flickering between states. Flickering within the Markov model provides a more sensitive analysis. The default is FALSE, no flickering.
- Number of States with `transition.graph`: The number of possible states in the rainfall-runoff relationship and transition between the states is selected with the transition.graph. The default is a 2-state model in a 2 by 2 matrix with a TRUE transition to and from each state.

Value

A built hydroState model object ready to be fitted with `fitModel`

Examples

```
# Load data
data(streamflow_annual_221201)

## Build default annual hydroState model
model = buildModel(input.data = streamflow_annual_221201)

# OR

## Build annual hydroState model with specified objects
# Build hydroState model with: 2-state, normal error distribution,
# 1-lag of auto-correlation, and state dependent parameters ('a1', 'std')
model = buildModel(input.data = streamflow_annual_221201,
                   data.transform = 'boxcox',
                   parameters = list('a0','a1','std','AR1'),
                   seasonal.parameters = list(),
```

```
                    state.shift.parameters = list('a1','std'),
                    error.distribution = 'normal',
                    flickering = FALSE,
                    transition.graph = matrix(TRUE,2,2))
```

buildModelAll *Builds all hydroState models*

Description

`buildModelAll` builds all possible combinations of hydroState models

Usage

```
buildModelAll(
  input.data = data.frame(year = c(), flow = c(), precip = c()),
  ID = ""
)
```

Arguments

`input.data`	dataframe of annual, seasonal, or monthly runoff and precipitation observations. Gaps with missing data in either streamflow or precipitation are permitted, and the handling of them is further discussed in `select.Markov`. Monthly data is required when using `seasonal.parameters` that assumes selected model parameters are better defined with a sinusoidal function.
`ID`	character vector of a stream gauge identifier

Details

`buildModelAll`

All possible combinations of hydroState models are built for each data transformations, auto-correlation lag, and residual distribution from 1 to 3 states for investigating only state changes in the 'a0' and 'std' parameters. Note: annual time-step only

Value

A list of built hydroState models with every combination of objects ready to be fitted

Examples

```
# Load data
data(streamflow_annual_221201)

# Build all annual models
all.annual.models = buildModelAll(input.data = streamflow_annual_221201, ID = '221201')
```

`fitModel` *Fit hydroState model*

Description

`fitModel` fits hydrostate model(s) using global optimization by differential evolution DEoptim library.

Usage

```
fitModel(
  model.name,
  pop.size.perParameter = 10,
  max.generations = 500,
  doParallel = F
)
```

Arguments

`model.name`	name of the built hydroState model or name of the list containing all built models
`pop.size.perParameter`	integer that should be greater than or equal to the number of parameters in the model. The default is '10' and is sufficient for all models.
`max.generations`	integer that will stop the optimizer when set number of generations are reached. The default is '500'.
`doParallel`	TRUE/FALSE to perform fitting in parallel on all computer cores. Default is FALSE

Details

`fitModel`

After a hydroState model object is built, the model is ready to be fitted through minimizing the negative log-likelihood function. The likelihood is estimated recursively across each time-step, for each continous period of observations, and the sum of the negative log-likelihood is minimized too calibrate the model parameters. The only required input is the given name of the built hydroState model object. `fitModel` works for one built model (`buildModel`) or all (`buildModelAll`). If fitting all models be sure to install and load the parallelly library. Details on the likelihood function is as follows:

The likelihood function is estimated as:

$L_T = \delta P(x_1) + \Gamma\delta P(x_2)...\Gamma\delta P(x_T)1'$

where:

- δ is the initial state distribution, the initial probability of being in each state: $\delta = \begin{pmatrix} \delta_1 \\ 1-\delta_1 \end{pmatrix}$
- $P(x)$ is the m x m diagonal emissions matrix of the probability density for each state using a lower tail truncated Gaussian distribution or a two-parameter Gamma distribution
  - $f_{Gau}(x = \widehat{_{obs}q_t}; \mu = \widehat{_t q_i}, \sigma = \sigma_i, a = 0) = \frac{\phi(\frac{x-\mu}{\sigma})}{\sigma(1-\Phi(\frac{a-\mu}{\sigma}))}$

  - $f_{Gam}(x = \widehat{_{obs}q_t}; k = \frac{\widehat{_tq_i}^2}{\sigma_i^2}, \theta = \frac{\sigma_i^2}{\widehat{_tq_i}}) = \frac{x^{k-1}e^{\frac{x}{\theta}}}{\theta^k\Gamma(k)}$
  - where ϕ is the probability density function for the standard normal distribution, Φ is the cumulative distribution function for the standard normal distribution, k is the shape parameter, θ is the scale parameter, and $\Gamma(k)$ is the gamma function. For more details, refer to pg. 8-17 in the Supplementary Materials of "Watersheds may not recover from drought".
- Γ is the transition matrix
- T is the number of time-steps.

Value

A fitted hydroState model

Examples

```
# Load data
data(streamflow_annual_221201)

## Build default annual hydroState model
model = buildModel(input.data = streamflow_annual_221201)

## Fit built model
model = fitModel(model)

## Fit all built models
## Not run:

# Load data
data(streamflow_annual_221201)

## Build all annual models
all.annual.models = buildModelAll(input.data = streamflow_annual_221201, ID = '221201')

## Fit all
model = fitModel(all.annual.models)


## End(Not run)
```

---

get.AIC *get AIC*

Description

`get.AIC` retrieves Akaike information criteria from a fitted model.

Usage

```
get.AIC(model.name)
```

Arguments

`model.name` is the name of the fitted `hydroState` model object.

Details

`get.AIC`

The AIC is the negative log-likelihood of the model plus a penealty for model parameters. This function can be performed on a single model or a selection of models to find the lowest AIC of the set.

Value

AIC value

Examples

```
# Load fitted model
data(model.annual.fitted.221201)

## AIC of a single model
get.AIC(model.annual.fitted.221201)

## Lowest AIC of a model set
get.AIC()
```

`get.residuals` *Get residuals*

Description

The normal pseudo residuals are retrieved from the fitted model.

Usage

```
get.residuals(model.name)
```

Arguments

`model.name` name of the fitted hydroState model object.

Details

`get.residuals`

`get.residuals` retrieves residuals from the fitted model and exports them as a data frame.

Value

Data frame of residuals for each time-step

Examples

```
# Load fitted model
data(model.annual.fitted.221201)

## Get residuals in a dataframe
get.residuals(model.name = model.annual.fitted.221201)
```

`get.states` *get states*

Description

`get.states` retrieves results from the fitted `hydroState` model.

Usage

```
get.states(model.name)
```

Arguments

`model.name`	is the name of the fitted `hydroState` model object.

Details

`get.states`

These dataframe of results include:

- time-step: year and possibly either season or month for subannual analysis
- Viterbi State Number: state number (i.e. 1, 2, or 3) to differentiate states
- Obs. flow: streamflow observations
- Viterbi Flow: flow values of the Viterbi state including the 5% and 95% confidence intervals. These are the most likely flow state values at each time-step of the given states.
- Normal State Flow: flow values of the normal state including the 5% and 95% confidence intervals. These Normal state flow values are the values from the normal state at each time-step. When the most likely state is the Normal state for a time-step, the Viterbi flow state value equals the Normal flow state value. This Normal state can be visualized relative to the most likely Viterbi state in the "dep.variable" plot from `plot.states`.
- Conditional Prob: conditional probabilities for each state show the probability of remaining in the given state. When the conditional probability is closer to 1, there is a higher probability that hydroState model remains in that state for the next time-step.
- Emission Density: emission density for each state is the result of multiplying the conditional probabilities by the transition probabilities at each timestep.

Value

data frame of results to evaluate the rainfall-runoff states overtime

Examples

```
# Load fitted model
data(model.annual.fitted.221201)

## Set initial year to set state names
model.annual.fitted.221201 =
                setInitialYear(model.name = model.annual.fitted.221201,
                               initial.year = 1990)

## Get states
model.annual.fitted.221201.states =
                get.states(model.name = model.annual.fitted.221201)
```

`hydroState-internal` *hydroState-internal*

Description

These are undocumented internal functions not intended for the user

---

`plot.residuals` *Plot residuals*

Description

The normal pseudo residuals are plotted for review to check for outliers and validate the fit of the model. It is recommended to ensure the model fit is valid before evaluating results (i.e. `plot.states`). Furthermore, to ensure the multi-state model performs better than the one-state model, it is recommended to visually compare `plot.resdiuals` of both models.

Usage

```
## S3 method for class 'residuals'
plot(model.name, do.pdf = FALSE, ID = NULL)
```

Arguments

`model.name`	name of the fitted hydroState model object.
`do.pdf`	option to export residual plots as a pdf. Default is FALSE.
`ID`	character string of catchment identifier (i.e. gauge ID). Default is NULL. Only recommended when do.pdf = TRUE.

Details

`plot.residuals`

`plot.residuals` produces five plots to review and validate the fitted hydroState model.

- A) Time-series of normal-pseudo residuals to ensure the residuals each year are within the confidence intervals.
- B) Auto-correlation function (ACF) of normal-pseudo residuals to ensure there is no serial correlation in residuals. Lag spikes should be below confidence interval at each lag (except 0).
- C) Histogram of uniform-pseudo residuals should show uniform distribution (equal frequency for each residual value)
- D) Histogram of normal-pseudo residuals should show normal distribution centered on zero and with no skew
- E) Quantile-Quantile (Q-Q) plot where normal-pseudo residuals vs. theoretical quantities should align on the diagonal line. The last plot contains the Akaike information criterion (AIC) and Shapiro-Wilk p-value. The AIC is an estimator to determine the most parsimonious, best performing model given the number of parameters. When comparing models, the lowest AIC is the best performing model. Shapiro-Wilks test for normality in the residuals and a p-value greater than 0.05 (chosen alpha level) indicates the residuals are normally distributed; the null hypothesis that the residuals are normally distributed is not rejected.

It is recommended to export the residual plot as a PDF due to it's size. If the R plot windor is too small, two common errors can occur:

- "Error in plot.new() : figure margins too large": reset plot window with "dev.off()", enlarge plot area and re-run `plot.residuals`.
- "Error in par(op) : invalid value specified for graphical parameter "pin" if the R plot window is not reset with "dev.off", an additional `plot.residuals` attempt will result in this error.

Value

Plots of residuals to evaluate model fit

Examples

```
# Load fitted model
data(model.annual.fitted.221201)

## Plot residuals
plot.residuals(model.name = model.annual.fitted.221201)
```

`plot.states` *plot States*

Description

`plot.states` produces several plots to visualize results of the states overtime. `setInitialYear` is required before `plot.states`.

Usage

```
## S3 method for class 'states'
plot(
  model.name,
  ind.variable = TRUE,
  dep.variable = TRUE,
  dep.variable.transformed = TRUE,
  cond.state.prob = TRUE,
  do.pdf = FALSE,
  ID = NULL
)
```

Arguments

`model.name`	is the name of the fitted hydroState model object.
`ind.variable`	option to plot independent variable overtime. Default is TRUE.
`dep.variable`	option to plot dependent variable and states overtime. Default is TRUE.
`dep.variable.transformed`	option to plot transformed dependent variable and states overtime. Default is TRUE.
`cond.state.prob`	option to plot the conditional state probabilities overtime for each state. Default is TRUE.
`do.pdf`	option to export plots as a pdf. Default is FALSE.
`ID`	character string of catchment identifier (i.e. gauge ID). Default is NULL. Only recommended when do.pdf = TRUE.

Details

`plot.states`

`plot.states` produces four figures of the results from the fitted hydroState model. The default produces all four:

- independent variable: precipitation
- dependent variable and states: streamflow observations, most likely state, and relative normal state estimate
- transformed dependent variable and states: transformed streamflow observations and most likely state
- conditional state probabilities for each state: probability of hydroState model remaining in given state

These are plotted on the same page, and there is an option to export plots as a pdf to the current working directory. There are also options to only plot one of the four plots.

Value

plots to evaluate rainfall-runoff states overtime along with observations and the conditional probabilities of each state.

Examples

```
# Load fitted model
data(model.annual.fitted.221201)

## Set initial year to set state names
model.annual.fitted.221201 =
                  setInitialYear(model.name = model.annual.fitted.221201,
                  initial.year = 1990)

## Plot all figures
plot.states(model.name = model.annual.fitted.221201)

## Plot only dependent variable transformed with markov states
plot.states(model.name = model.annual.fitted.221201,
             ind.variable = FALSE,
             dep.variable = FALSE,
             dep.variable.transformed = TRUE,
             cond.state.prob = FALSE)
```

---

set.seasons *Set Seasons*

Description

Aggregates monthly data to 4 seasons in a year.

Usage

```
set.seasons(
  input.data = data.frame(year = c(), month = c(), flow = c(), precip = c())
)
```

Arguments

input.data	dataframe of monthly runoff and precipitation observations. Gaps with missing data in either streamflow or precipitation are permitted, and the handling of them is further discussed in `select.Markov`. Monthly data is required when using `seasonal.parameters` that assumes selected model parameters are better defined with a sinusoidal function.

Details

`set.seasons`

Sets 4 seasons

Value

A dataframe of seasonal observations with an additional column counting the number of months in each season.

Examples

```
# Load data
data(streamflow_monthly_221201)

# aggregate monthly data to seasonal
streamflow_seasonal_221201 = set.seasons(streamflow_monthly_221201)
```

`setInitialYear` *Sets state names given initial year*

Description

sets the state names for each time-step relative to the initial year given

Usage

```
setInitialYear(model.name, initial.year)
```

Arguments

`model.name`	name of the fitted hydroState model object.
`initial.year`	integer with year (YYYY). Default is first year in input.data.

Details

`setInitialYear`

hydroState assigns names to the computed states. This requires choosing an initial year where the state value from that year will be named 'Normal'. Other state values will be given names relative to the state value in the initial year. The choice of the initial year does not affect results. It is a means to more easily interpret the difference in state values relative to each other. It is best to choose a year based on the question being asked. For example, in testing the impact of drought, a year before the beginning of the drought, 1990, was selected as an initial year when conditions were considered 'Normal' (Peterson TJ, Saft M, Peel MC & John A (2021), Watersheds may not recover from drought, Science, DOI: doi:10.1126/science.abd5085)

Value

A fitted hydroState model object with state names for each time-step ready for `plot.states`

Examples

```
# Load fitted model
data(model.annual.fitted.221201)

## Set initial year to set state names
model.annual.fitted.221201 =
                setInitialYear(model.name = model.annual.fitted.221201,
                               initial.year = 1990)
```

Index