Package: hhh4ZI (via r-universe)

December 7, 2024

Type Package

Title Zero-Inflated Endemic-Epidemic Count Time Series

Version 0.3.2

Description Extending 'surveillance::hhh4()' for zero-inflated infectious disease counts as described in Lu and Meyer (2023) <doi:10.1002/bimj.202100408>. This package is a partial fork of 'surveillance' and intended as a proof of concept for this model extension. It will be merged with other extensions in the future.

License GPL-2

LazyData true

RoxygenNote 7.2.3

Depends R (>= 3.6.0), methods, grDevices, graphics, stats, utils, surveillance

Imports sp, VGAM

Suggests numDeriv, maxLik, gridExtra, scales

Repository https://ee-lib.r-universe.dev

RemoteUrl https://github.com/Junyi-L/hhh4ZI

RemoteRef HEAD

RemoteSha 73cfa152b1e4e0284887f42d00fdad699369d1b8

Contents

Coverage 2
hhh4ZI 3
measles 6
oneStepAhead 7
pit.oneStepAhead_hhh4ZI 9
plot.hhh4ZI 9
scores 15
simulate.hhh4ZI 16
update.hhh4ZI 17
upgrade.hhh4 18

Index **19**

`Coverage` *MMR coverage levels in the 16 states of Germany*

Description

Coverage levels at school entry for the first and second dose of the combined measles-mumps-rubella (MMR) vaccine for the years 2005 to 2018, estimated from children presenting vaccination documents at school entry examinations.

Usage

```
TotalKids

VacPass

Dosis1

Dosis2
```

Format

Four data frames, each with 14 rows (years) and 16 columns (states):

- TotalKids Number of children examined.
- VacPass Number of children who presented vaccination documents.
- Dosis1 Percentage of children with vaccination documents, who received at least 1 dose of MMR vaccine.
- Dosis2 Percentage of children with vaccination documents, who received at least 2 doses of MMR vaccine.

Coverage levels were derived from vaccination documents presented at medical examinations, which are conducted by local health authorities at school entry each year. Records include information about the receipt of 1st and 2nd doses of MMR, from year 2005 to 2018. Note that information from children who did not present a vaccination document on the day of the medical examination, is not included in the estimated coverage.

Source

https://www.gbe-bund.de, Gesundheitsbericherstattung des Bundes; Queried on 23 February 2021.

hhh4ZI *Fitting zero-inflated HHH Models with Random Effects and Neighbourhood Structure*

Description

Fits a zero-inflated autoregressive negative binomial (hhh4) model to a univariate or multivariate time series of counts. The characteristic feature of hhh4 models is the additive decomposition of the conditional mean into *epidemic* and *endemic* components (Held et al, 2005). The inflated parameter is a logit-linear predictor and can have autoregressive terms.

Usage

```
hhh4ZI(
  stsObj,
 control = list(ar = list(f = ~-1, offset = 1, lag = 1), ne = list(f = ~-1, offset = 1,
   lag = 1, weights = neighbourhood(stsObj) == 1, scale = NULL, normalize = FALSE), end
   = list(f = ~1, offset = 1), zi = list(f = ~1, lag = 1, lag.unitSpecific = FALSE),
   family = c("NegBin1", "NegBinM"), subset = 2:nrow(stsObj), optimizer = list(stop =
   list(tol = 1e-05, niter = 100), regression = list(method = "nlminb"), variance =
   list(method = "Nelder-Mead")), verbose = FALSE, start = list(fixed = NULL, random =
    NULL, sd.corr = NULL),
     data = list(t = stsObj@epoch - min(stsObj@epoch)),
    keep.terms = FALSE),
  check.analyticals = FALSE
)
```

Arguments

`stsObj`	object of class "`sts`" containing the (multivariate) count data time series.
`control`	a list containing the model specification and control arguments, the parts relating to hhh4 model is the same as in `surveillance::hhh4`:

- `ar` Model for the autoregressive component given as list with the following components:
  - f = ~ -1 a formula specifying $\log(\lambda_{it})$.
  - offset = 1 optional multiplicative offset, either 1 or a matrix of the same dimension as `observed(stsObj)`.
  - lag = 1 a positive integer meaning autoregression on $y_{i,t-lag}$.
- `ne` Model for the neighbour-driven component given as list with the following components:
  - f = ~ -1 a formula specifying $\log(\phi_{it})$.
  - offset = 1 optional multiplicative offset, either 1 or a matrix of the same dimension as `observed(stsObj)`.
  - lag = 1 a non-negative integer meaning dependency on $y_{j,t-lag}$.

  - weights = neighbourhood(stsObj) == 1 neighbourhood weights w_{ji}. The default corresponds to the original formulation by Held et al (2005), i.e., the spatio-temporal component incorporates an unweighted sum over the lagged cases of the first-order neighbours. See Paul et al (2008) and Meyer and Held (2014) for alternative specifications, e.g., `W_powerlaw`. Time-varying weights are possible by specifying an array of `dim()` `c(nUnits, nUnits, nTime)`, where `nUnits=ncol(stsObj)` and `nTime=nrow(stsObj)`.
  - scale = NULL optional matrix of the same dimensions as `weights` (or a vector of length `ncol(stsObj)`) to scale the `weights` to `scale * weights`.
  - normalize = FALSE logical indicating if the (scaled) `weights` should be normalized such that each row sums to 1.
- `end` Model for the endemic component given as list with the following components
  - f = ~ 1 a formula specifying $\log(\nu_{it})$.
  - offset = 1 optional multiplicative offset e_{it}, either 1 or a matrix of the same dimension as `observed(stsObj)`.
- `zi` Model for the zero inflation component given as list with the following components:
  - f = ~ -1 a formula specifying $logit(\gamma_{it})$.
  - lag = 1 a vector of positive integers meaning autoregression on the respective $y_{i,t-lag}$ terms. Use `NULL` or `integer(0)` to exclude AR terms.
  - lag.unitSpecific logical indicating if the autoregressive parameter in the zero inflation part is unit specific.
- `family` Distributional family – the Negative Binomial distribution. The overdispersion parameter can be assumed to be the same for all units (`"NegBin1"`), to vary freely over all units (`"NegBinM"`), or to be shared by some units (specified by a factor of length `ncol(stsObj)` such that its number of levels determines the number of overdispersion parameters). Note that `"NegBinM"` is equivalent to `factor(colnames(stsObj), levels = colnames(stsObj))`.
- `subset` Typically `2:nrow(obs)` if model contains autoregression.
- `optimizer` a list of three lists of control arguments.
  The `"stop"` list specifies two criteria for the outer optimization of regression and variance parameters: the relative `tolerance` for parameter change using the criterion `max(abs(x[i+1]-x[i])) / max(abs(x[i]))`, and the maximum number `niter` of outer iterations.
  Control arguments for the single optimizers are specified in the lists named `"regression"` and `"variance"`. `method="nlminb"` is the default optimizer for regression update, however, the `methods` from `optim` may also be specified (as well as `"nlm"` but that one is not recommended here). For the variance updates, only Nelder-Mead optimization (`method="Nelder-Mead"`) is provided. All other elements of these two lists are passed as `control` arguments to the chosen `method`, e.g., if `method="nlminb"` adding `iter.max=50` increases the maximum number of inner iterations from 20 (default) to 50.
- `verbose` non-negative integer (usually in the range `0:3`) specifying the amount of tracing information to be output during optimization.

- `start` a list of initial parameter values replacing initial values set via `fe` and `ri`. Since **surveillance** 1.8-2, named vectors are matched against the coefficient names in the model (where unmatched start values are silently ignored), and need not be complete, e.g., `start = list(fixed = c("-log(overdisp)" = 0.5))` (default: 2) for a `family = "NegBin1"` model. In contrast, an unnamed start vector must specify the full set of parameters as used by the model.
- `data` a named list of covariates that are to be included as fixed effects (see `fe`) in any of the 3 component formulae. By default, the time variable `t` is available and used for seasonal effects created by `addSeason2formula`. In general, covariates in this list can be either vectors of length `nrow(stsObj)` interpreted as time-varying but common across all units, or matrices of the same dimension as the disease counts `observed(stsObj)`.
- `keep.terms` logical indicating if the terms object used in the fit is to be kept as part of the returned object. This is usually not necessary, since the terms object is reconstructed by the `terms`-method for class `"hhh4ZI"` if necessary (based on `stsObj` and `control`, which are both part of the returned `"hhh4ZI"` object).

`check.analyticals`
: logical (or a subset of `c("numDeriv", "maxLik")`), indicating if (how) the implemented analytical score vector and Fisher information matrix should be checked against numerical derivatives at the parameter starting values, using the packages **numDeriv** and/or **maxLik**. If activated, `hhh4` will return a list containing the analytical and numerical derivatives for comparison (no ML estimation will be performed). This is mainly intended for internal use by the package developers.

Value

`hhh4ZI` returns an object of class `"hhh4ZI"`, which inherits from class `"hhh4"`, and is a list containing the following components:

- coefficients named vector with estimated (regression) parameters of the model
- se estimated standard errors (for regression parameters)
- cov covariance matrix (for regression parameters)
- Sigma estimated variance-covariance matrix of random effects
- Sigma.orig estimated variance parameters on internal scale used for optimization
- call the matched call
- dim vector with number of fixed and random effects in the model
- loglikelihood (penalized) loglikelihood evaluated at the MLE
- margll (approximate) log marginal likelihood should the model contain random effects
- convergence logical. Did optimizer converge?
- mu The fitted mean values in hhh4 model part
- fitted.values fitted mean values in zero-inflated model
- gamma fitted zero inflation parameter

- control control object of the fit
- terms the terms object used in the fit if `keep.terms = TRUE` and `NULL` otherwise
- stsObj the supplied `stsObj`
- lags named integer vector of length three containing the lags used for the epidemic components `"ar"`, `"ne"` and `"zi"` respectively. The corresponding lag is `NA` if the component was not included in the model.
- nObs number of observations used for fitting the model
- nTime number of time points used for fitting the model
- nUnit number of units (e.g. areas) used for fitting the model
- runtime the `proc.time`-queried time taken to fit the model, i.e., a named numeric vector of length 5 of class `"proc_time"`

Examples

```
neW1 <- neighbourhood(measles) == 1
fit <- hhh4ZI(measles,
  control = list(
    ar = list(f = ~1),
    ne = list(f = ~1, weights = neW1, normalize = TRUE),
   end = list(f = ~1),
    zi = list(f = ~1),
    family = "NegBin1",
    verbose = TRUE,
    keep.terms = TRUE
  )
)
summary(fit)
sim_data <- simulate(fit, simplify = FALSE)
```

`measles` *Measles in the 16 states of Germany*

Description

Bi-weekly number of measles cases in the 16 states (Bundeslaender) of Germany for the years 2005 to 2018.

Usage

```
measles
```

Format

An object of class `sts` with 364 rows and 16 columns.

Details

The `population` slot contains the population fractions of each state from 2005 to 2018, obtained from the Federal Statistical Office of Germany.

Source

SurvStat@RKI 2.0 (https://survstat.rki.de), Robert Koch Institute; Queried on 26 March 2021.

Statistisches Bundesamt (https://www.destatis.de/DE/Home/_inhalt.html); Queried on 10 April 2021.

`oneStepAhead` *Predictive Model Assessment for hhh4ZI Models*

Description

Computes successive one-step-ahead predictions from a model fit. There is a method for `hhh4` models from **surveillance** (see the documentation of `oneStepAhead` there) and a derived method for ZI-extended models fitted from `hhh4ZI`. The documentation of the arguments is inherited from the former; where `"hhh4"` is mentioned, `"hhh4ZI"` works interchangeably.

Predictions can be inspected using `quantile`, `confint` and `plot` methods.

Usage

```
oneStepAhead(result, tp, ...)

## S3 method for class 'hhh4'
oneStepAhead(
  result,
  tp,
  type = c("rolling", "first", "final"),
  which.start = c("current", "final"),
  keep.estimates = FALSE,
  verbose = type != "final",
  cores = 1,
  ...
)

## S3 method for class 'hhh4ZI'
oneStepAhead(
  result,
  tp,
  type = c("rolling", "first", "final"),
  which.start = c("current", "final"),
  keep.estimates = FALSE,
  verbose = TRUE,
  cores = 1,
  ...
)

## S3 method for class 'oneStepAhead_hhh4ZI'
quantile(x, probs = c(2.5, 10, 50, 90, 97.5)/100, ...)

## S3 method for class 'oneStepAhead_hhh4ZI'
confint(object, parm, level = 0.95, ...)

## S3 method for class 'oneStepAhead_hhh4ZI'
plot(x, unit = 1, probs = 1:99/100, start = NULL, ...)
```

Arguments

`result`	fitted hhh4 model (class "hhh4").
`tp`	numeric vector of length 2 specifying the time range in which to compute one-step-ahead predictions (for the time points `tp[1]+1`, ..., `tp[2]+1`). If a single time index is specified, it is interpreted as `tp[1]`, and `tp[2]` is set to the penultimate time point of `result$control$subset`.
`...`	unused (argument of the generic).
`type`	The default "rolling" procedure sequentially refits the model up to each time point in `tp` and computes the one-step-ahead predictions for the respective next time point. The alternative `type`s are no true one-step-ahead predictions but much faster: "first" will refit the model for the first time point `tp[1]` only and use this specific fit to calculate all subsequent predictions, whereas "final" will just use `result` to calculate these. The latter case thus gives nothing else than a subset of `result$fitted.values` if the `tp`'s are part of the fitted subset `result$control$subset`.
`which.start`	Which initial parameter values should be used when successively refitting the model to subsets of the data (up to time point `tp[1]`, up to `tp[1]+1`, ...) if `type="rolling"`? Default ("current") is to use the parameter estimates from the previous time point, and "final" means to always use the estimates from `result` as initial values. Alternatively, `which.start` can be a list of `start` values as expected by hhh4, which then replace the corresponding estimates from `result` as initial values. This argument is ignored for "non-rolling" `type`s.
`keep.estimates`	logical indicating if parameter estimates and log-likelihoods from the successive fits should be returned.
`verbose`	non-negative integer (usually in the range `0:3`) specifying the amount of tracing information to output. During `hhh4` model updates, the following verbosity is used: `0` if `cores` > 1, otherwise `verbose-1` if there is more than one time point to predict, otherwise `verbose`.
`cores`	the number of cores to use when computing the predictions for the set of time points `tp` in parallel (with mclapply). Note that parallelization is not possible in the default setting `type="rolling"` and `which.start="current"` (use `which.start="final"` for this to work).
`x`	an object of class "oneStepAhead" or "hhh4".
`probs`	numeric vector of probabilities with values in [0,1].

`object`	an object of class `"oneStepAhead"`.
`parm`	unused (argument of the generic).
`level`	required confidence level of the prediction interval.
`unit`	single integer or character selecting a unit for which to produce the plot.
`start`	x-coordinate of the first prediction. If `start=NULL` (default), this is derived from `x`.

`pit.oneStepAhead_hhh4ZI` *Non-Randomized Version of the PIT Histogram (for Count Data)*

Description

Non-Randomized Version of the PIT Histogram (for Count Data)

Usage

```
## S3 method for class 'oneStepAhead_hhh4ZI'
pit(x, units = NULL, ...)

## S3 method for class 'hhh4ZI'
pit(x, subset = x$control$subset, units = seq_len(x$nUnit), ...)
```

Arguments

`x`	an object of class `"oneStepAhead_hhh4ZI"` or `"hhh4ZI"`, respectively.
`units`	integer or character vector indexing the units for which to compute the PIT histogram. By default, all units are considered.
`...`	further arguments passed to `pit.default`.
`subset`	subset of time points for which to produce the PIT histogram. Defaults to the subset used for fitting the model.

`plot.hhh4ZI` *Plots for Fitted* `hhh4ZI` *Models*

Description

Forks of `plot.hhh4` et al. to support zero-inflated models fitted with `hhh4ZI`.

Usage

```
## S3 method for class 'hhh4ZI'
plot(x, type = c("fitted", "maxEV", "season", "maps", "ri", "neweights"), ...)

plotHHH4ZI_fitted(
  x,
  units = 1,
  names = NULL,
  col = c("grey85", "blue", "orange"),
  pch = 19,
  pt.cex = 0.6,
  pt.col = 1,
  par.settings = list(),
  legend = TRUE,
  legend.args = list(),
  legend.observed = FALSE,
  decompose = NULL,
  total = FALSE,
  meanHHH = NULL,
  ...
)

plotHHH4ZI_fitted1(
  x,
  unit = 1,
  main = NULL,
  col = c("grey85", "blue", "orange"),
  pch = 19,
  pt.cex = 0.6,
  pt.col = 1,
  border = col,
  start = x$stsObj@start,
  end = NULL,
  xaxis = NULL,
  xlim = NULL,
  ylim = NULL,
  xlab = "",
  ylab = "No. infected",
  hide0s = FALSE,
  decompose = NULL,
  total = FALSE,
  meanHHH = NULL
)

plotHHH4ZI_maps(
  x,
  which = c("mean", "endemic", "epi.own", "epi.neighbours", "zi"),
  prop = FALSE,
  main = which,
  zmax = NULL,
  col.regions = NULL,
  labels = FALSE,
  sp.layout = NULL,
  ...,
  map = x$stsObj@map,
  meanHHH = NULL
)

plotHHH4ZI_maxEV(
  ...,
  matplot.args = list(),
  refline.args = list(),
  legend.args = list()
)

plotHHH4ZI_ri(
  x,
  component,
  exp = FALSE,
  at = list(n = 10),
  col.regions = cm.colors(100),
  colorkey = TRUE,
  labels = FALSE,
  sp.layout = NULL,
  gpar.missing = list(col = "darkgrey", lty = 2, lwd = 2),
  ...
)

plotHHH4ZI_season(
  ...,
  components = NULL,
  intercept = FALSE,
  xlim = NULL,
  ylim = NULL,
  xlab = NULL,
  ylab = "",
  main = NULL,
  par.settings = list(),
  matplot.args = list(),
  legend = NULL,
  legend.args = list(),
  refline.args = list(),
  unit = 1,
  period = NULL
)
```

```
plotHHH4ZI_neweights(x, plotter = boxplot, ..., exclude = 0, maxlag = Inf)
```

Arguments

`x`	a fitted `hhh4ZI` object.
`type`	type of plot: either `"fitted"` component means of selected `units` along time along with the observed counts, or `"season"`ality plots of the model components and the epidemic dominant eigenvalue (which may also be plotted along overall time by `type="maxEV"`, especially if the model contains time-varying neighbourhood weights or unit-specific epidemic effects), or `"maps"` of the fitted mean components averaged over time, or a map of estimated region-specific random intercepts (`"ri"`) of a specific model `component`. The latter two require `x$stsObj` to contain a map.
`...`	For `plotHHH4_season` and `plotHHH4_maxEV`, one or more `hhh4`-fits, or a single list of these. Otherwise further arguments passed on to other functions. For the `plot`-method these go to the specific plot `type` function. `plotHHH4_fitted` passes them to `plotHHH4_fitted1`, which is called sequentially for every unit in `units`. `plotHHH4_maps` and `plotHHH4_ri` pass additional arguments to `spplot`, and `plotHHH4_neweights` to the `plotter`.
`units`, `unit`	integer or character vector specifying a single `unit` or possibly multiple `units` to plot. It indexes `colnames(x$stsObj)`. In `plotHHH4_fitted`, `units=NULL` plots all units. In the seasonality plot, selection of a unit is only relevant if the model contains unit-specific intercepts or seasonality terms.
`names`, `main`	main title(s) for the selected `unit(s)` / `components`. If `NULL` (default), `plotHHH4_fitted1` will use the appropriate element of `colnames(x$stsObj)`, whereas `plotHHH4_season` uses default titles.
`col`, `border`	length 3 vectors specifying the fill and border colors for the endemic, autoregressive, and spatio-temporal component polygons (in this order).
`pch`, `pt.cex`, `pt.col`	style specifications for the dots drawn to represent the observed counts. `pch=NA` can be used to disable these dots.
`par.settings`	list of graphical parameters for `par`. Sensible defaults for `mfrow`, `mar` and `las` will be applied unless overridden or `!is.list(par.settings)`.
`legend`	Integer vector specifying in which of the `length(units)` frames the legend should be drawn. If a logical vector is supplied, `which(legend)` determines the frame selection, i.e., the default is to drawn the legend in the first (upper left) frame only, and `legend=FALSE` results in no legend being drawn.
`legend.args`	list of arguments for `legend`, e.g., to modify the default positioning `list(x="topright", inset=0.02)`.
`legend.observed`	logical indicating if the legend should contain a line for the dots corresponding to observed counts.
`decompose`	if `TRUE` or (a permutation of) `colnames(x$stsObj)`, the fitted mean will be decomposed into the contributions from each single unit and the endemic part instead of the default endemic + AR + neighbours decomposition.

`total`	logical indicating if the fitted components should be summed over all units to be compared with the total observed counts at each time point. If `total=TRUE`, the `units`/`unit` argument is ignored.
`meanHHH`	(internal) use different component means than those estimated and available from `x`.
`start`, `end`	time range to plot specified by vectors of length two in the form `c(year,number)`, see `"sts"`.
`xaxis`	if this is a list (of arguments for `addFormattedXAxis`), the time axis is nicely labelled similar to `stsplot_time`. Note that in this case or if `xaxis = NA`, the basic time indexes `1:nrow(x$stsObj)` will be used as x coordinates, which is different from the long-standing default (`xaxis = NULL`) with a real time scale.
`xlim`	numeric vector of length 2 specifying the x-axis range. The default (`NULL`) is to plot the complete time range (`type="fitted"`) or period (`type="season"`), respectively.
`ylim`	y-axis range. For `type="fitted"`, this defaults to `c(0,max(observed(x$stsObj)[,unit]))`. For `type="season"`, `ylim` must be a list of length `length(components)` specifying the range for every component plot, or a named list to customize only a subset of these. If only one `ylim` is specified, it will be recycled for all `components` plots.
`xlab`, `ylab`	axis labels. For `plotHHH4_season`, `ylab` specifies the y-axis labels for all `components` in a list (similar to `ylim`). If `NULL` or incomplete, default mathematical expressions are used. If a single name is supplied such as the default `ylab=""` (to omit y-axis labels), it is used for all `components`.
`hide0s`	logical indicating if dots for zero observed counts should be omitted. Especially useful if there are too many.
`which`	a character vector specifying the components of the mean for which to produce maps. By default, the overall mean and all three components are shown.
`prop`	a logical indicating whether the component maps should display proportions of the total mean instead of absolute numbers.
`zmax`	a numeric vector of length `length(which)` (recycled as necessary) specifying upper limits for the color keys of the maps, using a lower limit of 0. A missing element (`NA`) means to use a map-specific color key only covering the range of the values in that map (can be useful for `prop = TRUE`). The default `zmax = NULL` means to use the same scale for the component maps and a separate scale for the map showing the overall mean.
`col.regions`	a vector of colors used to encode the fitted component means (see `levelplot`). For `plotHHH4_maps`, the length of this color vector also determines the number of levels, using 10 heat colors by default.
`labels`	determines if and how regions are labeled, see `layout.labels`.
`sp.layout`	optional list of additional layout items, see `spplot`.
`map`	an object inheriting from `"SpatialPolygons"` with `row.names` covering `colnames(x)`.
`matplot.args`	list of line style specifications passed to `matplot`, e.g., `lty`, `lwd`, `col`.

`refline.args`	list of line style specifications (e.g., `lty` or `col`) passed to `abline` when drawing the reference line (`h=1`) in plots of seasonal effects (if `intercept=FALSE`) and of the dominant eigenvalue. The reference line is omitted if `refline.args` is not a list.
`component`	component for which to plot the estimated region-specific random intercepts. Must partially match one of `colnames(ranef(x, tomatrix=TRUE))`.
`exp`	logical indicating whether to `exp`-transform the color-key axis labels to show the multiplicative effect of the region-specific random intercept on the respective component. Axis labels are then computed using `log_breaks` from package **scales** (if that is available) or `axisTicks` (as a fallback) respecting the `colorkey$tick.number` setting (default: 7). The default is `FALSE`.
`at`	a numeric vector of breaks for the color levels (see `levelplot`), or a list specifying the number of breaks `n` (default: 10) and their `range` (default: range of the random effects, extended to be symmetric around 0). In the latter case, breaks are equally spaced (on the original, non-`exp` scale of the random intercepts). If `exp=TRUE`, custom breaks (or `range`) need to be given on the exp-scale.
`colorkey`	a Boolean indicating whether to draw the color key. Alternatively, a list specifying how to draw it, see `levelplot`.
`gpar.missing`	list of graphical parameters for `sp.polygons`, applied to regions with missing random intercepts, i.e., not included in the model. Such extra regions won't be plotted if `!is.list(gpar.missing)`.
`components`	character vector of component names, i.e., a subset of `c("ar", "ne", "end")`, for which to plot the estimated seasonality. If `NULL` (the default), only components which appear in any of the models in `...` are plotted. A seasonality plot of the epidemic dominant eigenvalue is also available by including `"maxEV"` in `components`, but it only supports models without epidemic covariates/offsets.
`intercept`	logical indicating whether to include the global intercept. For `plotHHH4_season`, the default (`FALSE`) means to plot seasonality as a multiplicative effect on the respective component. Multiplication by the intercept only makes sense if there are no further (non-centered) covariates/offsets in the component.
`period`	a numeric value giving the (longest) period of the harmonic terms in the model. This usually coincides with the `freq` of the data (the default), but needs to be adjusted if the model contains harmonics with a longer periodicity.
`plotter`	the (name of a) function used to produce the plot of weights (a numeric vector) as a function of neighbourhood order (a factor variable). It is called as `plotter(Weight ~ Distance, ...)` and defaults to `boxplot`. A useful alternative is, e.g., `stripplot` from package **lattice**.
`exclude`	vector of neighbourhood orders to be excluded from plotting (passed to `factor`). By default, the neighbourhood weight for order 0 is not shown, which is usually zero anyway.
`maxlag`	maximum order of neighbourhood to be assumed when computing the `nbOrder` matrix. This additional step is necessary iff `neighbourhood(x$stsObj)` only specifies a binary adjacency matrix.

scores — *Proper Scoring Rules for* hhh4ZI *Models*

Description

The following scores are implemented: logarithmic score (logs), ranked probability score (rps), Dawid-Sebastiani score (dss), squared error score (ses). These are extended versions of the corresponding functions in **surveillance** to handle zero-inflated negative binomial predictions, which use an additional zero inflation parameter gamma.

Usage

```
## S3 method for class 'oneStepAhead_hhh4ZI'
scores(
  x,
  which = c("logs", "rps", "dss", "ses"),
  units = NULL,
  sign = FALSE,
  individual = FALSE,
  ...
)

## S3 method for class 'hhh4ZI'
scores(
  x,
  which = c("logs", "rps", "dss", "ses"),
  subset = x$control$subset,
  units = seq_len(x$nUnit),
  sign = FALSE,
  ...
)
```

Arguments

`x`	an object of class `"oneStepAhead_hhh4ZI"` or `"hhh4ZI"`, respectively.
`which`	character vector determining which scores to compute. The package **surveillance** implements the following proper scoring rules: logarithmic score (`"logs"`), ranked probability score (`"rps"`), Dawid-Sebastiani score (`"dss"`), and squared error score (`"ses"`). The normalized SES (`"nses"`) is also available but it is improper and hence not computed by default. It is possible to name own scoring rules in `which`. These must be functions of `(x, mu, size)`, vectorized in all arguments (time x unit matrices) except that `size` is `NULL` in case of a Poisson model. See the available scoring rules for guidance, e.g., `dss`.
`units`	integer or character vector indexing the units for which to compute the scores. By default, all units are considered.

sign	logical indicating if the function should also return `sign(x-mu)`, i.e., the sign of the difference between the observed counts and corresponding predictions. This does not really make sense when averaging over multiple `units` with `individual=FALSE`.
individual	logical indicating if the individual scores of the `units` should be returned. By default (`FALSE`), the individual scores are averaged over all `units`.
...	unused (argument of the generic).
subset	subset of time points for which to compute the scores.

simulate.hhh4ZI — *Simulate* `hhh4ZI` *Count Time Series*

Description

Fork of `simulate.hhh4` to support zero-inflated models fitted with `hhh4ZI`.

Usage

```
## S3 method for class 'hhh4ZI'
simulate(
  object,
  nsim = 1,
  seed = NULL,
  y.start = NULL,
  subset = 1:nrow(object$stsObj),
  coefs = coef(object),
  components = c("ar", "ne", "end"),
  simplify = nsim > 1,
  ...
)
```

Arguments

object	of class `"hhh4ZI"`.
nsim	number of time series to simulate. Defaults to `1`.
seed	an object specifying how the random number generator should be initialized for simulation (via `set.seed`). The initial state will also be stored as an attribute `"seed"` of the result. The original state of the `.Random.seed` will be restored at the end of the simulation. By default (`NULL`), neither initialization nor recovery will be done. This behaviour is copied from the `simulate.lm` method.
y.start	vector or matrix (with `ncol(object$stsObj)` columns) with starting counts for the epidemic components. If `NULL`, the observed means in the respective units of the data in `object` during `subset` are used.
subset	time period in which to simulate data. Defaults to (and cannot exceed) the whole period defined by the underlying `"sts"` object.

`coefs`	coefficients used for simulation from the model in `object`. Default is to use the fitted parameters. Note that the `coefs`-vector must be in the same order and scaling as `coef(object)`, which especially means `reparamPsi = TRUE` (as per default when using the `coef`-method to extract the parameters). The overdispersion parameter in `coefs` is the inverse of the dispersion parameter `size` in `rnbinom`.
`components`	character vector indicating which components of the fitted model `object` should be active during simulation. For instance, a simulation with `components="end"` is solely based on the fitted endemic mean.
`simplify`	logical indicating if only the simulated counts (`TRUE`) or the full `"sts"` object (`FALSE`) should be returned for every replicate. By default a full `"sts"` object is returned iff `nsim=1`.
`...`	unused (argument of the generic).

`update.hhh4ZI` *Refit a* `hhh4ZI` *Model*

Description

Re-fit a `hhh4ZI` model with a modified control list, equivalently to `update.hhh4`.

Usage

```
## S3 method for class 'hhh4ZI'
update(
  object,
  ...,
  S = NULL,
  subset.upper = NULL,
  use.estimates = object$convergence,
  evaluate = TRUE
)
```

Arguments

`object`	a fitted `"hhh4ZI"` model.
`...`	elements modifying the original control list.
`S`	a named list of numeric vectors (with names `"ar"`, `"end"`, etc) to adjust the number of harmonics in the model components via `addSeason2formula`, or `NULL` (meaning no modification of seasonal terms).
`subset.upper`	refit on a subset of the data up to that time point (used by `oneStepAhead`).
`use.estimates`	logical indicating if `coef(object)` should be used as starting values for the new fit.
`evaluate`	logical indicating if the updated `hhh4ZI` call should be evaluated or returned.

upgrade.hhh4 *Refit a HHH4 model with a ZI Component*

Description

Refits a previous hhh4 fit with a ZI component using hhh4ZI.

Usage

```
## S3 method for class 'hhh4'
upgrade(object, control = list(f = ~1, lag = 1, lag.unitSpecific = FALSE), ...)
```

Arguments

`object`	a "hhh4" fit.
`control`	either a list of specifications for the `zi` component of hhh4ZI or a hhh4 control list with added `zi` specification.
`...`	further (non-`zi`) control elements.

Index

∗ **datasets**
  Coverage, 2
  measles, 6
.Random.seed, *16*

abline, *14*
addFormattedXAxis, *13*
addSeason2formula, *5*, *17*
axisTicks, *14*

boxplot, *14*

confint.oneStepAhead_hhh4ZI
  *(*oneStepAhead*)*, 7
Coverage, 2

Dosis1 *(*Coverage*)*, 2
Dosis2 *(*Coverage*)*, 2
dss, *15*

factor, *14*
fe, *5*

hhh4, *7, 8*, *12*, *18*
hhh4ZI, 3, *7*, *9*, *12*, *16–18*

layout.labels, *13*
legend, *12*
levelplot, *13, 14*
log_breaks, *14*

matplot, *13*
mclapply, *8*
measles, 6

nbOrder, *14*
nlm, *4*

oneStepAhead, *7*, 7, *17*
optim, *4*

par, *12*
pit.default, *9*
pit.hhh4ZI *(*pit.oneStepAhead_hhh4ZI*)*, 9
pit.oneStepAhead_hhh4ZI, 9
plot.hhh4, *9*
plot.hhh4ZI, 9
plot.oneStepAhead_hhh4ZI
  *(*oneStepAhead*)*, 7
plotHHH4ZI_fitted *(*plot.hhh4ZI*)*, 9
plotHHH4ZI_fitted1 *(*plot.hhh4ZI*)*, 9
plotHHH4ZI_maps *(*plot.hhh4ZI*)*, 9
plotHHH4ZI_maxEV *(*plot.hhh4ZI*)*, 9
plotHHH4ZI_neweights *(*plot.hhh4ZI*)*, 9
plotHHH4ZI_ri *(*plot.hhh4ZI*)*, 9
plotHHH4ZI_season *(*plot.hhh4ZI*)*, 9
proc.time, *6*

quantile.oneStepAhead_hhh4ZI
  *(*oneStepAhead*)*, 7

ri, *5*
rnbinom, *17*

scores, 15
set.seed, *16*
simulate, *16*
simulate.hhh4, *16*
simulate.hhh4ZI, 16
sp.polygons, *14*
SpatialPolygons, *13*
spplot, *12, 13*
stripplot, *14*
sts, *3*, *13*, *17*
stsplot_time, *13*

terms, *5*
TotalKids *(*Coverage*)*, 2

update.hhh4, *17*
update.hhh4ZI, 17
upgrade.hhh4, 18

VacPass *(*Coverage*)*, 2

W_powerlaw, *4*