Package 'ecostats'

November 11, 2020

Title Code and Data Accompanying the Eco-Stats Text

Version 0.1.4

Description Functions and data supporting the Eco-Stats text (Warton, forthcoming, Springer), and solutions to exercises. Functions include tools for using simulation envelopes in diagnostic plots, and a function for diagnostic plots of multivariate linear models. Datasets mentioned in the package are included here (where not available elsewhere) and vignette solutions to textbook exercises will be forthcoming at a later time.

License LGPL (>= 2.1)

Imports GET, mgcv, mvtnorm

Encoding UTF-8

LazyData true

RoxygenNote 7.1.1

Suggests knitr, rmarkdown, testthat

VignetteBuilder knitr

NeedsCompilation no

Author David Warton [aut, cre],
Christopher Chung [ctb]

Maintainer David Warton <david.warton@unsw.edu.au>

Repository CRAN

Date/Publication 2020-11-11 10:00:06 UTC

R topics documented:

aphids *Aphid data*

Description

A study of the effects of bird exclusion on biological control of aphids in oat and wheat fields in Germany (Grass *et al.* 2017). Many thanks to Ingo for providing the raw data. In each of two fields (one of oats, one of wheat) there were eight plots, four with plastic netting to exclude birds, and four without. Aphid abundance was counted on seven different occasions over the first 38 weeks following netting. The expectation was that aphid numbers would decrease on bird exclusion, because an important food source to tree sparrows is aphid predators, hoverflies and ladybird beetles, so presence of birds may be limit the effectiveness of a biological control of aphids.

Usage

```
data(aphids)
```

Format

A list containing two dataframes, `oat` and `wheat`, depending on the crop. Each dataframe contains:

Plot The plot ID, a factor with eight levels

Treatment A factor indicating whether birds were excluded, `excluded` or `present`.

Time The number of days since netting was applied.

counts Aphid abundance (counts)

logcount log(y+1)-transformed aphid abundance

References

Grass *et al.* (2017) Insectivorous birds disrupt biological control of cereal aphids. Ecology, **98** 1583-90.

Examples

```
data(aphids)
cols=c(rgb(1,0,0,alpha=0.5),rgb(0,0,1,alpha=0.5)) #transparent colours
with(aphids$oat, interaction.plot(Time,Plot,logcount,legend=FALSE,
                              col=cols[Treatment], lty=1, ylab="Counts [log(y+1) scale]",
                                xlab="Time (days since treatment)") )
                              legend("bottomleft",c("Excluded","Present"),col=cols,lty=1)
```

aphidsBACI *Aphid data as a BACI design*

Description

A subset from a study of the effects of bird exclusion on biological control of aphids in a German oat field (Grass *et al.* 2017). Many thanks to Ingo for providing the raw data. The full data are in aphids, here we provide just two sampling times, 3 days and 30 days after netting. OK, this is not quite a BACI design, 3 days after netting is not quite "before" but effects at this point should be negligible...

There were eight plots, four with plastic netting to exclude birds, and four without. Aphid abundance was counted on 5 shoots per plot. The expectation was that aphid numbers would decrease on bird exclusion, because an important food source to tree sparrows is aphid predators, hoverflies and ladybird beetles, so presence of birds may be limit the effectiveness of a biological control of aphids.

Usage

```
data(aphidsBACI)
```

Format

A dataframe containing:

Plot The plot ID, a factor with eight levels

Treatment A factor indicating whether birds were excluded, `excluded` or `present`.

Time The data of sampling (June 18th or July 15th, 3 and 30 days after netting, respectively)

counts Aphid abundance (counts)

logcount log(y+1)-transformed aphid abundance

References

Grass *et al.* (2017) Insectivorous birds disrupt biological control of cereal aphids. Ecology, **98** 1583-90.

Examples

```
data(aphidsBACI)
plot(logcount~interaction(Treatment,Time),data=aphidsBACI,col=c("lightgreen","pink"))
```

cpredict *Conditional Predictions for Multivariate Linear Model Fits*

Description

Predicted values using full conditional models derived from a multivariate linear model (`mlm`) object. The full conditionals model each response as a linear model with all other responses used as predictors in addition to the regressors specified in the formula of the `mlm` object.

Usage

```
cpredict(object, standardize = TRUE, ...)
```

Arguments

`object`	a `mlm` object, typically the result of calling `lm` with a matrix response.
`standardize`	logical defaults to `TRUE`, standardising responses so they are comparable across responses.
`...`	further arguments passed to `predict.lm`, in particular, `newdata`. However, this function was not written to accept non-default values for `se.fit`, `interval` or `terms`.

Details

Predictions using an `mlm` object but based on the full conditional model, that is, from a linear model for each response as a function of all responses as well as predictors. This can be used in plots to diagnose the multivariate normality assumption.

By default predictions are standardised to facilitate overlay plots of multiple responses, as in `plotenvelope`.

This function behaves much like `predict.lm`, but currently, standard errors and confidence intervals around predictions are not available.

Value

A matrix of predicted values from full conditional models.

Author(s)

David Warton <david.warton@unsw.edu.au>

See Also

`cresiduals`, `lm`, `plotenvelope`, `predict.lm`

Examples

```
data(iris)
# fit a mlm:
iris.mlm=lm(cbind(Sepal.Length,Sepal.Width,Petal.Length,Petal.Width)~Species,data=iris)
# predict each response conditionally on the values of all other responses:
cpredict(iris.mlm)
```

`cresiduals` *Extract Conditional Residuals from Multivariate Linear Model Fits*

Description

Residuals from full conditionals of a Multivariate Linear Model (`mlm`) object. The full conditional for each response is a linear model with all other responses used as predictors in addition to the regressors specified in the formula of the `mlm` object. This is used to diagnose the multivariate normality assumption in `plotenvelope`.

Usage

```
cresiduals(object, standardize = TRUE, ...)
```

Arguments

`object`	a `mlm` object, typically the result of calling `lm` with a matrix response.
`standardize`	logical defaults to `TRUE`, to return studentized residuals using `rstandard` so they are comparable across responses.
`...`	further arguments passed to `residuals.lm` or `rstandard`.

Details

A `residuals` function for `mlm` objects, which returns residuals from a full conditional model, that is, a linear model of each response against all responses as well as predictors, which can be used to diagnose the multivariate normality assumption. These can be standardized (`standardize=TRUE`) to facilitate overlay plots of multiple responses, as in `plotenvelope`.

Value

A matrix of residuals

Author(s)

David Warton <david.warton@unsw.edu.au>

See Also

`cpredict`, `lm`, `plotenvelope`, `residuals`, `rstandard`

Examples

```
data(iris)
# fit a mlm:
iris.mlm=lm(cbind(Sepal.Length,Sepal.Width,Petal.Length,Petal.Width)~Species,data=iris)
# construct full conditional residuals:
cresiduals(iris.mlm)
```

`plotenvelope` *Diagnostic Plots for a Fitted Object with Simulation Envelopes*

Description

Produces diagnostic plots of a fitted model `y`, and adds global envelopes constructed by simulating new residuals, to see how departures from expected trends compare to what might be expected if the fitted model were correct. Global envelopes are constructed using the `GET` package (Myllymäki et al 2017) for simultaneous control of error rates over the whole plot, by simulating new responses from the fitted model then recomputing residuals (which can be computationally intensive), or alternatively, by simulating residuals directly from the (multivariate) normal distribution (faster, but not always a smart move). Options for diagnostic plots to construct are a residual vs fits, a normal quantile plot, or scale-location plot, along the lines of `plot.lm`.

Usage

```
plotenvelope(
  y,
  which = 1:2,
  sim.method = "refit",
  n.sim = 199,
  conf.level = 0.95,
  type = "st",
  transform = NULL,
 main = c("Residuals vs Fitted Values", "Normal Quantile Plot", "Scale-Location Plot"),
  xlab = c("Fitted values", "Theoretical Quantiles", "Fitted Values"),
  ylab = c("Residuals", "Residuals", expression(sqrt("|Residuals|"))),
  col = par("col"),
  line.col = if (do.smooth) "slateblue4" else "olivedrab",
  envelope.col = adjustcolor(line.col, 0.1),
  do.smooth = FALSE,
  plot.it = TRUE,
  ...
)
```

Arguments

`y`	is *any* object that responds to `residuals`, `predict` and (if `sim.method="refit"`) `simulate` and `update`.

`which` a vector specifying the diagnostic plots to construct:

1. residual vs fits, with a smoother
2. normal quantile plot
3. scale-location plot, with smoother

These are the first three options in `plot.lm` and a little is borrowed from that code. A global envelope is included with each plot around where we expect to see the data (or the smoother, if requested, for plots 1 and 3) when model assumptions are satisfied. If not fully captured by the global envelope, there is some evidence that the model assumptions are not satisfied.

`sim.method` How should residuals be simulated? The default `"refit"` option constructs new responses (via `simulate`), refits the model (via `update`), then recomputes residuals, often known as a *parametric bootstrap*. This is computationally intensive but gives a robust answer. This is the only suitable option if residuals are not expected to be normal when assumptions are satisfied (like when using `glm` with a non-Gaussian family). Alternatively, `"norm"` simulates from a normal distribution, matches means and variances (and covariances for multivariate responses) to the observed residuals. The `"stand.norm"` option sets means to zero and variances to one, which is appropriate when residuals should be standard normal when assumptions are satisfied (as for any object fitted using the `mvabund` package, for example). These options are faster but more approximate than `"refit"`.

`n.sim` the number of simulated sets of residuals to be generated, to which the observed residuals will be compared. The default is 199 datasets.

`conf.level` the confidence level to use in constructing the envelope.

`type` the type of global envelope to construct, see `global_envelope_test` for details. Default `"st"` uses studentized envelope tests to protect for unequal variance, which has performed well in simulations in this context.

`transform` a character vector pointing to a function that should be applied to both axes of the normal quantile plot. The most common use is to set `transform="pnorm"` for a PP-plot.

`main` the plot title (if a plot is produced). A vector of three titles, one for each plot. If only one value is given that will be used for all plots.

`xlab` x axis label (if a plot is produced). A vector of three labels, one for each plot. If only one value is given that will be used for all plots.

`ylab` y axis label (if a plot is produced). A vector of three labels, one for each plot. If only one value is given that will be used for all plots.

`col` color of points

`line.col` color of the line on diagnostic plots about which points should show no trend if model assumptions are satisfied. Defaults to "olivedrab". Because it's cool. If `do.smooth=TRUE` this is the color of the smoother (defaulting to "slateblue4").

`envelope.col` color of the global envelope around the expected trend. All data points should always stay within this envelope (and will for a proportion `conf.level` of datasets satisfying model assumptions). If a smoother has been used on the residual vs fits or scale-location plot, the envelope is constructed around the smoother, that is, the smoother should always stay within the envelope.

do.smooth	logical defaults to FALSE (no smoother). If TRUE, a smoother is drawn on residual vs fits and scale-location plots, and the global envelope is around the *smoother* not the data. No smoother is added to the normal quantile plot.
plot.it	logical. Should the result be plotted? If not, a list of analysis outputs is returned, see *Value*.
...	further arguments sent through to plot

Details

A challenge when interpreting diagnostic plots is understanding the extent to which deviations from the expected pattern could be due to random noise (sampling variation) rather than actual assumption violations. This function is intended to assess this, by simulating multiple realizations of residuals (and fitted values) in situations where assumptions are satisfied, and plotting a global simulation envelope around these at level conf.level.

This function can take any fitted model, and construct any of three diagnostic plots, as determined by which:

1. Residual vs fits plot (optionally, with a smoother)
2. Normal quantile plot
3. Scale-Location plot (optionally, with smoother)

and see if the trend is behaving as expected if the model were true. As long as the fitted model responds to residuals and predict (and when sim.method="refit", simulate) then a simulation envelope will be constructed for each plot.

Simulation envelopes are global, constructed using the GET-package, meaning that (for example) a 95% global envelope on a quantile plot should contain *all* residuals for 95% of datasets that satisfy model assumptions. So if *any* data points lie outside the quantile plot's envelope we have evidence that assumptions of the fitted model are not satisfied. The GET-package was originally constructed to place envelopes around functions, motivated by the problem of diagnostic testing of spatial processes (Myllymäki et al 2017), but it can equally well be applied here, by treating the set of residuals (ordered according to the x-axis) as point-wise evaluations of a function. For residual vs fits and scale-location plots, global envelopes are constructed for the *smoother*, not for the data, hence we are looking to see if the smoother is wholly contained within the envelope. The smoother is constructed using gam from the mgcv package.

Note that the global envelopes only tell you if there is evidence of violation of model assumptions – they do not tell you whether the violations are large enough to worry about. For example, in linear models, violations of normality are usually much less important than violations of linearity or equal variance. And in all cases, modest violations tend to only have modest effects on the validity of inferences, so you need to think about how big observed violations are rather than just thinking about whether or not they are outside their simulation envelope.

The method used to simulate data for the global envelopes is controlled by sim.method. The default method (sim.method="refit") uses a *parametric bootstrap* approach: simulate new responses from the fitted model, refit the model and recompute residuals and fitted values. This directly assesses whether trends in observed trends depart from what would be expected if the fitted model were correct, without any further assumptions. For complex models or large datasets this would however be super-slow. A fast, more approximate alternative (sim.method="norm") is to simulate

new (multivariate) normal residuals repeatedly and use these to assess whether trends in the observed data depart from what would be expected for independent (multivariate) normal residuals. If residuals are expected to be standard normal, a more refined check is to simulate from the standard normal using (`sim.method="stand.norm"`). This might for example be useful when diagnosing a model fitted using the `mvabund` package, since this calculates Dunn-Smyth residuals, which are approximately standard normal when assumptions are satisfied. If `y` is a `glm` with non-Gaussian family then residuals will not be normal and `"refit"` is the only appropriate option, hence other choices will be overridden with a warning reporting that this has been done.

Note that for Multivariate Linear Models (`mlm`), `cresiduals` and `cpredict` are used to construct residuals and fitted values (respectively) from the *full conditional models* (that is, models constructed by regressing each response against all other responses together with predictors). This is done because full conditionals are diagnostic of joint distributions, so *any* violation of multivariate normality is expressed as a violation of linear model assumptions on full conditionals. Results for all responses are overlaid on a single plot, future versions of this function may have an overlay option to separately plot each response.

The simulated data and subsequent analysis are also used to obtain a *P*-value for the test that model assumptions are correct, for each plot. This tests if sample residuals or their smoothers are unusually far from the values expected of them if model assumptions were satisfied. For details see `global_envelope_test`.

Value

Up to three diagnostic plots with simulation envelopes are returned, and additionally a list of three objects used in plotting, for plots 1-3 respectively. Each is a list with five components:

`x` X-values used for the envelope. In plots 1 and 3 with `do.smooth=TRUE`, this is 500 equally spaced points covering the range of fitted values. For plot 2, this is sorted normal quantiles corresponding to observed data.

`y` In plots 1 and 3, this is the values of the smoother corresponding to `x`. For plot 2, this is the sorted residuals.

`lo` The lower bound on the global envelope, for each value of `x`.

`hi` The upper bound on the global envelope, for each value of `x`.

`p.value` A *P*-value for the test that the observed smoother or data is not unusually far from what was expected, computed in `global_envelope_test`.

Author(s)

David Warton <david.warton@unsw.edu.au>

References

Myllymäki, M., Mrkvička, T., Grabarnik, P., Seijo, H. and Hahn, U. (2017), Global envelope tests for spatial processes. J. R. Stat. Soc. B, 79: 381-404. doi:10.1111/rssb.12172

See Also

`cpredict`, `cresiduals`, `qqenvelope`

Examples

```
# fit a Poisson regression to random data:
y = rpois(50,lambda=1)
x = 1:50
rpois_glm = glm(y~x,family=poisson())
plotenvelope(rpois_glm,n.sim=99)

# Fit a multivariate linear model to the iris dataset:
data(iris)
Y = with(iris, cbind(Sepal.Length,Sepal.Width,Petal.Length,Petal.Width))
iris_mlm=lm(Y~Species,data=iris)
# check normality assumption:
plotenvelope(iris_mlm,n.sim=99,which=2)

# A few more plots, with envelopes around smoothers: (this will take several seconds to run)
plotenvelope(iris_mlm, which=1:3, do.smooth=TRUE)
# Note violation on the scale/location plot.
```

---

qqenvelope *Normal Quantile-Quantile Plots with Global Simulation Envelopes*

Description

Produces a normal QQ plot of data, or of residuals from a fitted model `y`, with a user-specified line to compare to "theoretical" quantiles, and global envelopes constructed by simulating new residuals. Global envelopes are constructed using the `GET` package for simultaneous control of error rates over the whole curve.

Usage

```
qqenvelope(y, n.sim = 199, conf.level = 0.95, ylab = "Sample Quantiles", ...)
```

Arguments

`y`	can be a set of values for which we wish to check (multivariate) normality, or it can be *any* object that responds to the `residuals`, `simulate` and `update` functions.
`n.sim`	the number of simulated sets of residuals to be generated, to which the observed residuals will be compared. The default is 199 datasets.
`conf.level`	the confidence level to use in constructing the envelope.
`ylab`	`y` axis label (if a plot is produced).
`...`	further arguments sent through to `plotenvelope`

Details

A challenge when interpreting a `qqplot` is understanding the extent to which deviations from expected values could be due to random noise (sampling variation) rather than actual assumption violations. This function is intended to assess this, by simulating multiple realizations of residuals in situations where assumptions are satisfied, and plotting a global (or "simultaneous") simulation envelope around these at level `conf.level`. All data points should lie if assumptions are satisfied, and will do so for a proportion `conf.level` of datasets which satisfy their assumptions.

This function can take data (univariate or multivariate) and check for (multivariate) normality, or it can take a fitted model and use qq plots to interrogate residuals and see if they are behaving as we would expect them to if the model were true.

The envelope is global, constructed using the `GET-package`. So if *any* data points lie outside the envelope we have evidence that assumptions are not satisfied. The `GET-package` was originally constructed to place envelopes around functions, motivated by the problem of diagnostic testing of spatial processes (Myllymäki et al 2017), but it can equally well be applied here, by treating sorted residuals as point-wise evaluations of a function.

For further details refer to `plotenvelope`, which is called to construct the plot.

Value

a qqplot with simulation envelope is returned, and additionally:

`x`	a vector of theoretical quantiles from the standard normal sorted from smallest to largest
`y`	a vector of observed residuals sorted from smallest to largest
`lo`	lower bounds on the global simulation envelope for residuals
`hi`	upper bounds on the global simulation envelope for residuals
`p.value`	A *P*-value for the test that model assumptions are correct, using a 'parametric bootstrap' test, based on how far sample residuals depart from the values expected of them if model assumptions were satisfied.

Author(s)

David Warton <david.warton@unsw.edu.au>

References

Myllymäki, M., Mrkvička, T., Grabarnik, P., Seijo, H. and Hahn, U. (2017), Global envelope tests for spatial processes. J. R. Stat. Soc. B, 79: 381-404. doi:10.1111/rssb.12172

See Also

`qqnorm`, `qqline`, `plotenvelope`

Examples

```
# simulate some data and fit a qq plot:
y=rnorm(20)
qqenvelope(y)

# fit a multivariate linear model to the iris dataset:
data(iris)
Y = with(iris, cbind(Sepal.Length,Sepal.Width,Petal.Length,Petal.Width))
iris.mlm=lm(Y~Species,data=iris)
# check normality assumption:
qqenvelope(iris.mlm,n.sim=99)
```

`simulate.mlm` *Simulate Responses from a Multivariate Linear Model*

Description

Simulate one or more sets of responses from a Multivariate Linear Model (`mlm`) object.

Usage

```
## S3 method for class 'mlm'
simulate(object, nsim = 1, seed = NULL, ...)
```

Arguments

`object`	a `mlm` object, typically the result of calling `lm` where the response is a matrix.
`nsim`	number of replicate datasets to simulate. If `nsim` is greater than 1, the output is arranged in a 3D array.
`seed`	an object specifying if and how the random number generator should be initialized ('seeded'). Either NULL or an integer that will be used in a call to set.seed before simulating the response vectors. If set, the value is saved as the "seed" attribute of the returned value. The default, NULL will not change the random generator state, and return .Random.seed as the "seed" attribute, see 'Value'
`...`	additional optional arguments.

Details

A `simulate` function for `mlm` objects, which simulates one or more sets of responses from a Multivariate Linear Model (`mlm`) object. If multiple sets of responses are requested, they are returned in a 3D array, with simulation number along the third dimension.

The weights argument is currently ignored – a constant variance-covariance matrix assumed for `mlm`.

Value

A matrix of simulated values for the response (or an array, for `nsim` greater than 1)

Author(s)

David Warton <david.warton@unsw.edu.au>

See Also

`lm`, `simulate`

Examples

```
# fit a mlm to iris data:
data(iris)
iris.mlm=lm(cbind(Sepal.Length,Sepal.Width,Petal.Length,Petal.Width)~Species,data=iris)
# simulate new responses:
simulate(iris.mlm)
```

Index