Package: fit.models (via r-universe)

September 16, 2024

Version 0.64

Date 2020-08-02

Title Compare Fitted Models

Description The fit.models function and its associated methods (coefficients, print, summary, plot, etc.) were originally provided in the robust package to compare robustly and classically fitted model objects. See chapters 2, 3, and 5 in Insightful (2002) 'Robust Library User's Guide' <http://robust.r-forge.r-project.org/Robust.pdf>). The aim of the fit.models package is to separate this fitted model object comparison functionality from the robust package and to extend it to support fitting methods (e.g., classical, robust, Bayesian, regularized, etc.) more generally.

Author Kjell Konis [aut, cre]

Maintainer Kjell Konis <kjellk@gmail.com>

Imports lattice, stats

Suggests MASS

License GPL

RoxygenNote 7.1.0

NeedsCompilation no

Date/Publication 2020-08-02 14:30:02 UTC

Repository https://kjellpk.r-universe.dev

RemoteUrl https://github.com/cran/fit.models

RemoteRef HEAD

RemoteSha 2548545703702dbc11c8a2b9ceda8da77777386e

Contents

`center` *Calculate Location Estimate for a Fitted Model Object*

Description

Returns the location estimated from a location/scatter-type fitted model object.

Usage

```
center(object, ...)
```

Arguments

`object`	a fitted model object, typically. Sometimes also a `summary()` object of such a fitted model.
`...`	additional arguments for method functions.

ddPlot.covfm *Distance - Distance Plot*

Description

For a `covfm` object containing 2 models, this function plots the Mahalanobis distance from the first model on the y-axis and the Mahalanobis distance from the second model on the x-axis.

Usage

```
ddPlot.covfm(x, level = 0.95, strip = "", id.n = 3, ...)
```

Arguments

`x`	a `"covfm"` object.
`level`	a single numeric value between 0 and 1 giving the chi-squared percent point used to compute the outlyingness threshold.
`strip`	a character string printed in the "strip" at the top of the plot.
`id.n`	a single nonnegative integer specifying the number of extreme points to label in the plot.
`...`	additional arguments are passed to `xyplot`.

Value

if the models can be compared then the plotted `trellis` object is invisibly returned. Otherwise `x` is returned invisibly.

designMD *Design Matrix Mahalanobis Distance*

Description

Returns the squared Mahalanobis distance of all rows in the design (model) matrix X and the sample mean vector μ of the columns of X with respect to the sample covariance matrix Σ. This is (for vector x' a row of X) defined as

$$d^2 = (x - \mu)'\Sigma^{-1}(x - \mu)$$

where

$$\mu = colMeans(X)$$

and

$$\Sigma = cov(X).$$

Usage

```
designMD(object, ...)
```

Arguments

object	a fitted model object with a model.matrix method.
...	additional arguments are ignored.

Value

a numeric vector containing the squared Mahalanobis distances.

Examples

```
stack.lm <- lm(stack.loss ~ ., data = stackloss)

# Mahalanobis distance (not squared)
sqrt(designMD(stack.lm))
```

---

distancePlot.covfm *Side-by-Side Mahalanobis Distance Plot*

Description

Produces side-by-side plots of Mahalanobis distance computed using the location and covariance matrix estimates contained in each element of a covfm object.

Usage

```
distancePlot.covfm(x, level = 0.95, id.n = 3, ...)
```

Arguments

x	a "covfm" object.
level	a single numeric value between 0 and 1 giving the chi-squared percent point used to compute the outlyingness threshold.
id.n	a single nonnegative integer specifying the number of extreme points to label in the plot.
...	additional arguments are passed to xyplot.

Value

the trellis object is invisibly returned.

ellipsesPlot.covfm *Ellipses Plot - Visual Correlation Matrix Comparison*

Description

When there are 3 or more variables in the data, this function produces a matrix with ellipses drawn in the upper triangle. The ellipse in cell i, j of the plot is drawn to be a contour of a standard bivariate normal with correlation ρ_{ij}. One ellipse is drawn in each cell for each model in the `covfm` object. When there are 2 variables in the data, this function produces a scatter plot of the data with an overlaid 95% confidence ellipse for each model in the `covfm` object.

Usage

```
ellipsesPlot.covfm(x, ...)
```

Arguments

`x`	a `"covfm"` object.
`...`	additional arguments are ignored.

Value

x is invisibly returned.

fit.models *Fit dot Models*

Description

Fit a statistical model using different estimators (e.g., robust and least-squares) or combine fitted models into a single object. Generic methods then produce side-by-side comparisons of the parameter estimates and diagnostic plots.

Usage

```
fit.models(model.list, ...)
```

Arguments

`model.list`	a list or a character vector containing names of modeling functions. Only required when `fit.models` is being used to fit models (rather than combine already fitted models into a `fit.models` object).
`...`	see details.

Details

There are two distinct ways the `fit.models` function can be used.

The first is to fit the same model using different estimators. In this case, `model.list` should be a character vector or a list where each element is the name of a modeling function and the remaining arguments (in `...`) are the common arguments to the functions in `model.list`. For example, the following command fits robust and least squares linear models to Brownlee's Stack Loss Plant Data.

```
fit.models(c("rlm", "lm"), stack.loss ~ ., data = stackloss)
```

The resulting `fit.models` object is a list with the output of

```
rlm(stack.loss ~ ., data = stackloss)
```

in the first element and

```
lm(stack.loss ~ ., data = stackloss)
```

in the second. The class attribute of the returned list is set (in this case) to `"lmfm"` which is the `fit.models` class (fmclass) for comparing linear-model-like fits.

The second use of fit.models is to combine fitted model objects. In this case, `fit.models` combines its arguments into a fit.models object (a list where element i is occupied by argument i and sets the class attribute to the appropriate `fit.models` class.

Value

The returned object is a list containing the fitted models. The class of the retuned object depends on the classes of the model objects it contains.

See Also

`fmclass.add.class` for adding a class to an existing fit.models class and `fmclass.register` to create a new fit.models class.

Examples

```
# First, use fit.models to fit robust and least squares linear
# regression models to Brownlee's Stack Loss Plant Data.

# Step 1: rlm (robust linear model) is in the MASS package.
library(MASS)

# Step 2: tell fit.models rlm can be compared to lm
fmclass.add.class("lmfm", "rlm")

fm1 <- fit.models(c("rlm", "lm"), stack.loss ~ ., data = stackloss)

summary(fm1) #rlm does not provide p-values or Multiple R-squared


# Second, use fit.models to combine fitted models into a
```

```
# fit.models object.

lm.complete <- lm(stack.loss ~ ., data = stackloss)
lm.clean <- lm(stack.loss ~ ., data = stackloss, subset = 5:20)

fm2 <- fit.models(lm.clean, lm.complete)

summary(fm2)
plot(fm2)


# Name the models in the fit.models object.

fm3 <- fit.models(c(Robust = "rlm", "Least Squares" = "lm"),
                  stack.loss ~ ., data = stackloss)

fm4 <- fit.models(Clean = lm.clean, Complete = lm.complete)
```

`fmclass.register` *Register Comparable Functions*

Description

The `fit.models` package maintains a list of comparable models. These functions provide an api to modify this list.

Usage

```
fmclass.register(fmclass, classes, validation.function = NULL)

fmclass.add.class(fmclass, class, warn = TRUE)
```

Arguments

`fmclass`	a character string naming the fit.models class to be added.
`classes`	a character vector naming one or more classes that can be compared by the methods defined for the fit.models class in `fmclass`.
`validation.function`	a function returning `TRUE` when the models are comparable.
`class`	a character string specifying a class compatible with the methods of `fmclass`.
`warn`	a logical value. If TRUE, a warning is printed if `class` is already registered.

Details

See the package vignette.

Value

a null value is invisibly returned.

`overlaidKernelDensityPlot` *Overlaid Kernel Density Estimate Plot*

Description

Produces an overlaid kernel density plot.

Usage

```
overlaidKernelDensityPlot(x, fun, ...)
```

Arguments

`x`	a `fit.models` object.
`fun`	a function to extract the desired quantity from `x`.
`...`	additional arguments are passed to `densityplot`.

Value

the `trellis` object is invisibly returned.

`overlaidQQPlot` *Overlaid Normal QQ Plot*

Description

Produces an overlaid normal QQ plot.

Usage

```
overlaidQQPlot(x, fun, ...)
```

Arguments

`x`	a `fit.models` object.
`fun`	a function to extract the desired quantity from `x`.
`...`	additional arguments are passed to `qqmath`.

Value

the `trellis` object is invisibly returned.

overlaidSimpleRegressionPlot *Scatter Plot with Overlaid Fits*

Description

Produces a scatter plot of the data with overlaid fits.

Usage

```
overlaidSimpleRegressionPlot(x, lwd.reg, col.reg, ...)
```

Arguments

x	a fit.models object.
lwd.reg	a vector with length equal to the number of models in x specifying the line widths used in the plot.
col.reg	a vector with length equal to the number of models in x specifying the line colors used in the plot.
...	additional arguments are passed to xyplot.

Value

the trellis object is invisibly returned.

plot.covfm *Plot Method*

Description

Generic plot method for "covfm" objects.

Usage

```
## S3 method for class 'covfm'
plot(x, which.plots = 1:4, ...)
```

Arguments

x	a covfm object.
which.plots	either "ask" (character string) or an integer vector specifying which plots to draw. The plot options are (1) Mahalanobis Distance, (2) Ellipses Matrix, (3) Screeplot (Eigenvalues of Covariance Estimate), and (4) Distance - Distance Plot.
...	additional arguments are passed to the plot subfunctions.

Value

x is returned invisibly.

`plot.glmfm` *Comparison Diagnostic Plots for Generalized Linear Models*

Description

Produces a set of comparison diagnostic plots. The plot options are

1. Deviance Residuals vs. Predicted Values,
2. Response vs. Fitted Values,
3. Normal QQ Plot of Pearson Residuals,
4. Normal QQ Plot of Deviance Residuals,
5. Pearson Residuals vs. Mahalanobis Distance,
6. Sqrt Deviance Residuals vs. Predicted Values.

Usage

```
## S3 method for class 'glmfm'
plot(x, which.plots = 1:6, ...)
```

Arguments

`x`	a `glmfm` object.
`which.plots`	either `"ask"` (character string) or an integer vector specifying which plots to draw. In the later case, the plot numbers are given above.
`...`	other parameters to be passed through to plotting functions.

Value

x is invisibly returned.

Side Effects

The selected plots are drawn on a graphics device.

See Also

`sideBySideQQPlot` for 4 and 5 and `sideBySideScatterPlot` for the others.

Examples

```
# From ?glm:
# A Gamma example, from McCullagh & Nelder (1989, pp. 300-2)

clotting <- data.frame(
    u = c(5,10,15,20,30,40,60,80,100),
    lot1 = c(118,58,42,35,27,25,21,19,18),
    lot2 = c(69,35,26,21,18,16,13,12,12))

lot1 <- glm(lot1 ~ log(u), data = clotting, family = Gamma)
lot2 <- glm(lot2 ~ log(u), data = clotting, family = Gamma)

fm <- fit.models(lot1, lot2)
plot(fm)
```

`plot.lmfm` *Comparison Diagnostic Plots for Linear Regression Models*

Description

Produces a set of comparison diagnostic plots. The plot options are

1. Normal QQ Plot of Residuals,
2. Kernel Density Estimate of Residuals,
3. Residuals vs. Mahalanobis Distance,
4. Residuals vs. Fitted Values,
5. Sqrt Residuals vs. Fitted Values,
6. Response vs. Fitted Values,
7. Residuals vs. Index (Time),
8. Overlaid Normal QQ Plot of Residuals,
9. Overlaid Kernel Density Estimate of Residuals,
10. Scatter Plot with Overlaid Fits (for simple linear regression models).

Usage

```
## S3 method for class 'lmfm'
plot(x, which.plots = 1:10, ...)
```

Arguments

`x`	an `lmfm` object.
`which.plots`	either `"ask"` (character string) or an integer vector specifying which plots to draw. In the later case, the plot numbers are given above.
`...`	additional parameters are ignored.

Value

x is invisibly returned.

Side Effects

The selected plots are drawn on a graphics device.

See Also

See sideBySideQQPlot for 2, sideBySideKernelDensityPlot for 3, sideBySideIndexPlot for 8, overlaidQQPlot for 9, overlaidKernelDensityPlot for 10, overlaidSimpleRegressionPlot for 11, and sideBySideScatterPlot for the others.

Examples

```
data(stackloss)
stack.lm <- lm(stack.loss ~ ., data = stackloss)
stack.clean <- lm(stack.loss ~ ., data = stackloss, subset = 5:20)
fm <- fit.models(stack.clean, stack.lm)
plot(fm)
```

---

screePlot.covfm *Comparison Screeplot*

Description

Overlaid screeplot for the models in a "covfm" object.

Usage

```
screePlot.covfm(x, npcs, strip = "", ...)
```

Arguments

x	a "covfm" object.
npcs	a postive integer value specifying the number of components to be plotted.
strip	a character string printed in the "strip" at the top of the plot.
...	additional arguments are passed to xyplot.

Value

the trellis object is invisibly returned.

sideBySideIndexPlot *Comparison Index (Time) Plot*

Description

Produces side-by-side index (time) plots.

Usage

```
sideBySideIndexPlot(x, fun, level = 0.95, id.n = 3, ...)
```

Arguments

x	a fit.models object.
fun	a function to extract the desired quantity from x.
level	a numeric value between 0 and 1 specifying the confidence level used to draw the threshold in the plot.
id.n	a non-negative integer value specifying the number of extreme points to identify.
...	any additional arguments are passed to xyplot.

Value

the trellis object is invisibly returned.

sideBySideKernelDensityPlot *Comparison Kernel Density Estimate Plot*

Description

Produces side-by-side kernel density estimate plots.

Usage

```
sideBySideKernelDensityPlot(x, fun, ...)
```

Arguments

x	a fit.models object.
fun	a function to extract the desired quantity from x.
...	additional arguments are passed to xyplot.

Value

the trellis object is invisibly returned.

`sideBySideQQPlot` *Compaison QQ Plot*

Description

Produces side-by-side QQ plots. An optional simulated confidence envelope can be included in each plot.

Usage

```
sideBySideQQPlot(
  x,
  fun,
  envelope = TRUE,
  half.normal = FALSE,
  n.samples = 250,
  level = 0.95,
  id.n = 3,
  qqline = TRUE,
  ...
)
```

Arguments

`x`	a `fit.models` object.
`fun`	a function to extract the desired quantity from `x`.
`envelope`	a logical value. If `TRUE` a `level` confidence envelope is simulated for each QQ plot.
`half.normal`	a logical value. If `TRUE` the plot is drawn using the absolute values.
`n.samples`	a positive integer value giving the number of samples to compute in the simulation of the confidence envelope.
`level`	a numeric value between 0 and 1 specifying the confidence level for the envelope.
`id.n`	a non-negative integer value specifying the number of extreme points to identify.
`qqline`	a logical value. If `TRUE`, a QQ line is included in the plot.
`...`	additional arguments are passed to `xyplot`.

Value

the `trellis` object is invisibly returned.

sideBySideScatterPlot *Comparison Scatter Plot*

Description

Produces a side-by-side scatter plot.

Usage

```
sideBySideScatterPlot(object, x.fun, y.fun, ...)
```

Arguments

object	a fit.models object.
x.fun	a function to extract the x-axis quantity.
y.fun	a function to extract the y-axis quantity.
...	additional arguments.

Value

the trellis object is invisibly returned.

summary.covfm *Summary Method*

Description

Generic summary method for "covfm" objects.

Usage

```
## S3 method for class 'covfm'
summary(object, corr = FALSE, ...)
```

Arguments

object	a "covfm" object.
corr	a logical value passed as an attribute to the print method. When TRUE, correlations are compared in the textual output.
...	additional arguments for the summary method.

summary.glmfm *Comparison Summaries for Generalized Linear Models*

Description

Compute a summary of each model in a glmfm object.

Usage

```
## S3 method for class 'glmfm'
summary(object, correlation = FALSE, ...)
```

Arguments

object	a glmfm object.
correlation	a logical value. If TRUE, correlation matrices of the coefficient estimates are included in each summary.
...	additional arguments required by the generic summary function.

Value

a list with class summary.glmfm whose elements are summaries of each model in object.

Examples

```
# From ?glm:
# A Gamma example, from McCullagh & Nelder (1989, pp. 300-2)

clotting <- data.frame(
    u = c(5,10,15,20,30,40,60,80,100),
    lot1 = c(118,58,42,35,27,25,21,19,18),
    lot2 = c(69,35,26,21,18,16,13,12,12))

lot1 <- glm(lot1 ~ log(u), data = clotting, family = Gamma)
lot2 <- glm(lot2 ~ log(u), data = clotting, family = Gamma)

fm <- fit.models(lot1, lot2)
summary(fm)
```

summary.lmfm *Comparison Summaries for Linear Regression Models*

Description

Compute a summary of each model in an lmfm object.

Usage

```
## S3 method for class 'lmfm'
summary(object, correlation = FALSE, ...)
```

Arguments

object	an lmfm object.
correlation	a logical value. If TRUE, the correlation matrices for the coefficients are included in the summaries.
...	additional arguments required by the generic summary function.

Value

a list with class summary.lmfm whose elements are summaries of each model in object.

Examples

```
data(stackloss)
m1 <- lm(stack.loss ~ ., data = stackloss)
m2 <- lm(stack.loss ~ . - Acid.Conc., data = stackloss)

fm <- fit.models(m1, m2)
print(fm.sum <- summary(fm))
```

vcor *Calculate Correlation Matrix for a Fitted Model Object*

Description

Retrieve a correlation matrix estimate from a fitted model object. The default method uses cov2cor to scale the covariance matrix returned by vcov.

Usage

```
vcor(object, ...)
```

Arguments

`object`	a fitted model object, typically. Sometimes also a `summary()` object of such a fitted model.
...	additional arguments for method functions.

Index