Package ‘glogis’

March 16, 2018

Version 1.0-1
Date 2018-03-16
Title Fitting and Testing Generalized Logistic Distributions
Description Tools for the generalized logistic distribution (Type I, also known as skew-logistic distribution), encompassing basic distribution functions (p, q, d, r, score), maximum likelihood estimation, and structural change methods.
LazyLoad yes
Depends R (>= 2.10.0), zoo
Suggests strucchange, fxregime, lattice
Imports graphics, stats, sandwich
License GPL-2 | GPL-3
NeedsCompilation no
Author Achim Zeileis [aut, cre] (<https://orcid.org/0000-0003-0918-3766>), Thomas Windberger [aut]
Maintainer Achim Zeileis <Achim.Zeileis@R-project.org>
Repository CRAN
Date/Publication 2018-03-16 08:47:31 UTC

R topics documented:

breakpoints.glogisfit ........................................... 2
glogis .............................................................. 4
glogisfit ............................................................. 6
HICP ................................................................. 9

Index 11
breakpoints.glogisfit  Segmented Fitting of the Generalized Logistic Distribution

Description

Fitting univariate generalized logisitc distributions (Type I: skew-logistic with location, scale, and shape parameters) to segments of time series data.

Usage

```r
## S3 method for class 'glogisfit'
breakpoints(obj, h = 0.15, breaks = NULL, ic = "LWZ",
            hpc = "none", ...)

## S3 method for class 'breakpoints.glogisfit'
refit(object, ...)

## S3 method for class 'breakpoints.glogisfit'
coef(object, log = TRUE, ...)

## S3 method for class 'breakpoints.glogisfit'
fitted(object, type = c("mean", "variance", "skewness"), ...)

## S3 method for class 'breakpoints.glogisfit'
confint(object, parm = NULL, level = 0.95, breaks = NULL,
         meas = NULL, ...)
```

Arguments

- `obj` an object of class `glogisfit`.
- `h` numeric. Minimal segment size either given as fraction relative to the sample size or as an integer giving the minimal number of observations in each segment.
- `breaks` integer specifying the maximal number of breaks to be calculated. By default the maximal number allowed by `h` is used.
- `ic` character specifying the default information criterion that should be employed for selecting the number of breakpoints. Default is "LWZ" (Liu-Wu-Zidek criterion, a modified BIC). Instead the classic "BIC" can be used.
- `hpc` a character specifying the high performance computing support. Default is "none", can be set to "foreach".
- `object` an object of class `breakpoints.glogisfit` as returned by the `breakpoints` method.
- `log` logical option in `coef` method indicating whether scale and shape parameters should be reported in logs (default) or the original levels.
- `type` character specifying which moments of the segmented fitted distribution should be extracted.
breakpoints.glogisfit

parm integer. Either parm or breaks may be set, see below.
level numeric. The confidence level to be used.
meat. function. A function for extracting the meat of a sandwich estimator from a fitted object. By default, the inverse of bread is used, i.e., a correctly specified model is assumed.
arguments passed to methods.

Details

To test whether sequences (typically time series) of observations follow the same generalized logistic distribution, the stability of the parameters can be tested. If there is evidence for parameter instability, breakpoints can be estimated to find segments with stable parameters.

The methods from the strucchange and fxregime packages are leveraged. For testing, the generalized M-fluctuation tests from strucchange can directly be employed using gefp. For breakpoint estimation, the methods documented here provide a user interface to some internal functionality from the fxregime packages. They employ the (unexported) workhorse function gbreakpoints which is modeled after breakpoints from the strucchange package but employing user-defined estimation methods.

Optional support for high performance computing is available in the breakpoints method, currently using foreach for the dynamic programming algorithm. If hpc = "foreach" is to be used, a parallel backend should be registered before. See foreach for more information.

Value

breakpoints.glogisfit returns an object of class "breakpoints.glogisfit" that inherits from "gbreakpointsfull".

References


See Also

glogisfit, fxregimes, breakpoints

Examples

## artificial data with one structural change
set.seed(1081)
x <- c(rglogis(50, -1, scale = 0.5, shape = 3), rglogis(50, 1, scale = 0.5, shape = 1))
x <- zoo(x, yearmon(seq(2000, by = 1/12, length = 100)))

## full sample estimation
gf <- glogisfit(x)
The Generalized Logistic Distribution (Type I: Skew-Logistic)

Description

Density, distribution function, quantile function and random generation for the logistic distribution with parameters location and scale.

Usage

dglogis(x, location = 0, scale = 1, shape = 1, log = FALSE)
pglogis(q, location = 0, scale = 1, shape = 1, lower.tail = TRUE, log.p = FALSE)
qglogis(p, location = 0, scale = 1, shape = 1, lower.tail = TRUE, log.p = FALSE)
rglogis(n, location = 0, scale = 1, shape = 1)
sglogis(x, location = 0, scale = 1, shape = 1)

Arguments

x, q vector of quantiles.
p vector of probabilities.
n number of observations. If length(n) > 1, the length is taken to be the number required.
location, scale, shape

location, scale, and shape parameters (see below).

log, log.p  logical; if TRUE, probabilities p are given as log(p).

lower.tail  logical; if TRUE (default), probabilities are \( P[X \leq x] \), otherwise, \( P[X > x] \).

Details

If location, scale, or shape are omitted, they assume the default values of 0, 1, and 1, respectively.

The generalized logistic distribution with location = \( \mu \), scale = \( \sigma \), and shape = \( \gamma \) has distribution function

\[
F(x) = \frac{1}{1 + e^{-(x-\mu)/\sigma^\gamma}}.
\]

The mean is given by location + \((\text{digamma}(\text{shape}) - \text{digamma}(1)) \times \text{scale}\), the variance by \((\text{psigamma}(\text{shape}, \text{deriv} = 1) + \text{psigamma}(1, \text{deriv} = 1)) \times \text{scale}^2\) and the skewness by \((\text{psigamma}(\text{shape}, \text{deriv} = 2) - \text{psigamma}(1, \text{deriv} = 2)) / (\text{psigamma}(\text{shape}, \text{deriv} = 1) + \text{psigamma}(1, \text{deriv} = 1))\).

dglogis are calculated by leveraging the [dpq]logis and adding the shape parameter. rglogis uses inversion.

Value

dglogis gives the probability density function, pglogis gives the cumulative distribution function, qglogis gives the quantile function, and rglogis generates random deviates. sglogis gives the score function (gradient of the log-density with respect to the parameter vector).

References


Examples

```r
## PDF and CDF
par(mfrow = c(1, 2))
x <- -100:100/10
plot(x, dglogis(x, shape = 2), type = "l", col = 4, main = "PDF", ylab = "f(x)")
lines(x, dglogis(x, shape = 1))
lines(x, dglogis(x, shape = 0.5), col = 2)
legend("topleft", c("generalized (0, 1, 2)", "standard (0, 1, 1)",
                    "generalized (0, 1, 0.5")", lty = 1, col = c(4, 1, 2), bty = "n")
plot(x, pglogis(x, shape = 2), type = "l", col = 4, main = "CDF", ylab = "F(x)")
lines(x, pglogis(x, shape = 1))
lines(x, pglogis(x, shape = 0.5), col = 2)

## artificial empirical example
set.seed(2)
x <- rglogis(1000, -1, scale = 0.5, shape = 3)
gf <- glogisfit(x)
plot(gf)
summary(gf)

Description

Fit a univariate generalized logistic distribution (Type I: skew-logistic with location, scale, and shape parameters) to a sample of observations.

Usage

glogisfit(x, ...)

## Default S3 method:
glogisfit(x, weights = NULL, start = NULL, fixed = c(NA, NA, NA),
    method = "BFGS", hessian = TRUE, ...)

## S3 method for class 'formula'
glogisfit(formula, data, subset, na.action, weights, x = TRUE, ...)

## S3 method for class 'glogisfit'
plot(x, main = ",", xlab = NULL, fill = "lightgray",
    col = "blue", lwd = 1, lty = 1, xlim = NULL, ylim = NULL,
    legend = "topright", moments = FALSE, ...)

## S3 method for class 'glogisfit'
summary(object, log = TRUE, breaks = NULL, ...)

## S3 method for class 'glogisfit'
coef(object, log = TRUE, ...)

## S3 method for class 'glogisfit'
vcov(object, log = TRUE, ...)

Arguments

x

a vector of observation (may be a ts or zoo time series).

weights

optional numeric vector of weights.

start

optional vector of starting values. The parametrization has to be in terms of location, log(scale), log(shape) where the original parameters (without logs) are as in dglogis. Default is to use c(0, 0, 0) (i.e., standard logistic). For details see below.

fixed

specification of fixed parameters (see description of start). NA signals that the corresponding parameter should be estimated. A standard logistic distribution could thus be fitted via fixed = c(NA, NA, 0).
method character string specifying optimization method, see optim for the available options. Further options can be passed to optim through ....

hessian logical. Should the Hessian be used to compute the variance/covariance matrix? If FALSE, no covariances or standard errors will be available in subsequent computations.

formula symbolic description of the model, currently only x ~ 1 is supported.

data, subset, na.action arguments controlling formula processing via model.frame.

main, xlab, fill, col, lwd, lty, xlim, ylim standard graphical parameters, see plot and par.

legend logical or character specification where to place a legend. legend = FALSE suppresses the legend. See legend for the character specification.

moments logical. If a legend is produced, it can either show the parameter estimates (moments = FALSE, default) or the implied moments of the distribution.

object a fitted glogisfit object.

log logical option in some extractor methods indicating whether scale and shape parameters should be reported in logs (default) or the original levels.

breaks interval breaks for the chi-squared goodness-of-fit test. Either a numeric vector of two or more cutpoints or a single number (greater than or equal to 2) giving the number of intervals.

... arguments passed to methods.

Details
glogisfit estimates the generalized logistic distribution (Type I: skew-logistic) as given by dglogis. Optimization is performed numerically by optim using analytical gradients. For obtaining numerically more stable results the scale and shape parameters are specified in logs. Starting values are chosen as c(θ, θ, θ), i.e., corresponding to a standard (symmetric) logistic distribution. If these fail, better starting values are obtained by running a Nelder-Mead optimization on the original problem (without logs) first.

A large list of standard extractor methods is supplied to conveniently compute with the fitted objects, including methods to the generic functions print, summary, plot (reusing hist and lines), coef, vcov, logLik, residuals, and estfun and bread (from the sandwich package).

The methods for coef, vcov, summary, and bread report computations pertaining to the scale/shape parameters in logs by default, but allow for switching back to the original levels (employing the delta method).

Visualization employs a histogramm of the original data along with lines for the estimated density. Further structural change methods for "glogisfit" objects are described in breakpoints.glogisfit.

Value
glogisfit returns an object of class "glogisfit", i.e., a list with components as follows.

coefficients estimated parameters from the model (with scale/shape in logs, if included),

vcov associated estimated covariance matrix,
loglik    log-likelihood of the fitted model,
df       number of estimated parameters,
n       number of observations,
nobs number of observations with non-zero weights,
weights the weights used (if any),
optim output from the optim call for maximizing the log-likelihood,
method the method argument passed to the optim call,
parameters the full set of model parameters (location/scale/shape), including estimated and fixed parameters, all in original levels (without logs),
moments associated mean/variance/skewness,
start the starting values for the parameters passed to the optim call,
fixed the original specification of fixed parameters,
call the original function call,
x the original data,
converged logical indicating successful convergence of optim,
terms the terms objects for the model (if the formula method was used).

References


See Also
dglogis, dlogis, breakpoints.glogisfit

Examples

```r
## simple artificial example
set.seed(2)
x <- rglogis(1000, -1, scale = 0.5, shape = 3)
gf <- glogisfit(x)
plot(gf)
summary(gf)

## query parameters and associated moments
coef(gf)
coef(gf, log = FALSE)
gf$parameters
gf$moments
```
Description

Time series data with HICP (Harmonised Index of Consumer Prices) for 21 countries (plus EU) for 1990–2010 as provided by the OECD; and corresponding seasonally adjusted inflation ratios.

Usage

data("HICP")

data("hicps")

Format

Monthly multiple "zooreg" time series with "yearmon" index from Jan 1990 (HICP) or Feb 1990 (hicps) to Dec 2010 for 21 countries (plus EU).

Details

HICP contains the raw unadjusted Harmonised Index of Consumer Prices as provided by the OECD from which unadjusted inflation rates can be easily computed (see examples).

As the different countries have rather different seasonal patterns which vary over time (especially in the 2000s), they will typically require seasonal adjustment before modeling. Hence, a seasonally adjusted version of the inflation rate series is provided as hicps, where X-12-ARIMA (version 0.3) has been employed for adjusted. An alternative seasonal adjustment can be easily computed use stl (see examples).

Source

Organisation for Economic Co-operation and Development (OECD)

http://stats.oecd.org/

References


Examples

```r
## price series
data("HICP", package = "glogis")

## corresponding raw unadjusted inflation rates (in percent)
hicp <- 100 * diff(log(HICP))

## seasonal adjustment of inflation rates (via STL)
hicps1 <- do.call("merge", lapply(1:ncol(hicp), function(i) {
  z <- na.omit(hicp[,i])
  coredata(z) <- coredata(as.ts(z) - stl(as.ts(z), s.window = 13)$time.series[, "seasonal"])
  z
}))
colnames(hicps1) <- colnames(hicp)

## load X-12-ARIMA adjusted inflation rates
data("hicps", package = "glogis")

## compare graphically for one country (Austria)
plot(hicp[, "Austria"], lwd = 2, col = "lightgray")
lines(hicps1[, "Austria"], col = "red")
lines(hicps[, "Austria"], col = "blue")
legend("topleft", c("unadjusted", "STL", "X-12-ARIMA"), lty = c(1, 1, 1),
       col = c("lightgray", "red", "blue"), bty = "n")

## compare graphically across all countries (via lattice)
if(require("lattice")) {
  trellis.par.set(theme = canonical.theme(color = FALSE))
  xyplot(merge(hicp, hicps1, hicps),
    screen = names(hicp)[rep(1:ncol(hicp), 3)],
    col = c("lightgray", "red", "blue")[[rep(1:3, each = ncol(hicp))],
    lwd = c(2, 1, 1)[rep(1:3, each = ncol(hicp))])
  }
```
Index

+ Topic **datasets**
  - HICP, 9

+ Topic **distribution**
  - glogis, 4

+ Topic **regression**
  - breakpoints.glogisfit, 2
  - glogisfit, 6
  - bread, 3, 7
  - bread.glogisfit (glogisfit), 6
  - breakdates.confint.breakpoints.glogisfit
    - (breakpoints.glogisfit), 2
  - breakpoints, 2, 3
  - breakpoints.glogisfit, 2, 7, 8
  - coef, 7
  - coef.breakpoints.glogisfit
    - (breakpoints.glogisfit), 2
  - coef.glogisfit (glogisfit), 6
  - confint.breakpoints.glogisfit
    - (breakpoints.glogisfit), 2
  - dglogis, 6–8
  - dglogis (glogis), 4
  - dlogis, 8
  - estfun, 7
  - estfun.glogisfit (glogisfit), 6
  - fitted.breakpoints.glogisfit
    - (breakpoints.glogisfit), 2
  - foreach, 3
  - fxregimes, 3
  - gefp, 3
  - glogis, 4
  - glogisfit, 2, 3, 6
  - HICP, 9
  - hicps (HICP), 9
  - hist, 7
  - hist.glogisfit (glogisfit), 6
  - index.breakpoints.glogisfit
    - (breakpoints.glogisfit), 2
  - legend, 7
  - lines, 7
  - lines.confint.breakpoints.glogisfit
    - (breakpoints.glogisfit), 2
  - lines.glogisfit (glogisfit), 6
  - logLik, 7
  - logLik.glogisfit (glogisfit), 6
  - model.frame, 7
  - optim, 7
  - par, 7
  - pglogis (glogis), 4
  - plot, 7
  - plot.glogisfit (glogisfit), 6
  - print, 7
  - print.confint.breakpoints.glogisfit
    - (breakpoints.glogisfit), 2
  - print.glogisfit (glogisfit), 6
  - print.summary.glogisfit (glogisfit), 6
  - qglogis (glogis), 4
  - refit.breakpoints.glogisfit
    - (breakpoints.glogisfit), 2
  - residuals, 7
  - residuals.glogisfit (glogisfit), 6
  - rglogis (glogis), 4
  - sglogis (glogis), 4
  - stl, 9
  - summary, 7
  - summary.glogisfit (glogisfit), 6
  - ts, 6
vcov, 7
vcov.glogisfit (glogisfit), 6
zoo, 6