Package ‘ACDm’

July 21, 2025

Version 1.1.0

Title Tools for Autoregressive Conditional Duration Models

Depends R(>= 3.0.2)

LinkingTo Rcpp(>= 0.12.10)

Imports broom, dplyr, ggplot2, graphics, numDeriv, plyr, Rcpp, Rsolnp, zoo

Suggests optimx, rgl,

Description Provides tools for autoregressive conditional duration (ACD, Engle and Russell, 1998) models. Functions to create trade, price, or volume durations from transaction data, perform diurnal adjustments, fit various ACD models, and test them.

License GPL (>= 3)

NeedsCompilation yes

Author Markus Belfrage [aut, cre]

Maintainer Markus Belfrage <markus.belfrage@gmail.com>

Repository CRAN

Date/Publication 2025-07-16 18:20:02 UTC

Contents

`ACDm-package` *ACD Modelling*

Description

Package for Autoregressive Conditional Duration (ACD, Engle and Russell, 1998) models. Creates trade, price or volume durations from transactions (tic) data, performs diurnal adjustments, fits various ACD models and tests them.

Credit

The author would like to thank the department of statistics at Hanken School of Economics, as the bulk of this work was done there while working as a research assistant.

Author(s)

Markus Belfrage

Maintainer: Markus Belfrage <markus.belfrage@gmail.com>

References

Engle R.F, Russell J.R. (1998) *Autoregressive Conditional Duration: A New Model for Irregularly Spaced Transaction Data*, Econometrica, 66(5): 1127-1162.

acdFit *ACD (Autoregressive Conditional Duration) Model Fitting*

Description

This function estimates various ACD models with various assumed error term distributions, using Maximum Likelihood Estimation.

The currently available models (conditional mean specifications) are: Standard ACD, Log-ACD (two alternative specifications), AMACD, ABACD, BACD, BCACD, AACD, EXACD, SNIACD, LSNIACD, TACD, and TAMACD.

The currently available distributions are: Exponential (also used for QML), Weibull, Burr, generalized Gamma, generalized F, Q-Weibull, and Mixture inverse Gaussian.

Usage

```
acdFit(durations = NULL, model = "ACD", dist = "exponential",
    order = NULL, startPara = NULL,  dailyRestart = FALSE, optimFnc = "optim",
    method = "Nelder-Mead", output = TRUE, bootstrapErrors = FALSE,
    forceErrExpec = TRUE, fixedParamPos = NULL, bp = NULL,
    exogenousVariables = NULL, optimFncArgs = list(),
    control = list())
```

Arguments

`durations`	either (1) a data frame including, at least, a column named 'durations' or 'adjDur' (for adjusted durations), or (2) a vector of durations
`model`	the conditional mean model specification. Must be one of `"ACD"`, `"LACD1"`, `"LACD2"`, `"AMACD"`, `"BACD"`, `"ABACD"`, `"SNIACD"` or `"LSNIACD"`. See 'Details' for detailed model specification.
`dist`	the assumed error term distribution. Must be one of `"exponential"`, `"weibull"`, `"burr"`, `"gengamma"`, `"genf"`, `"qweibull"`, `"mixqwe"`, `"mixqww"`, or `"mixinvgauss"`. See 'Details' for detailed model specification.
`order`	a vector detailing the order of the particular ACD model. For example an ACD(p, q) specification should have `order = c(p, q)`.
`startPara`	a vector with parameter values to start the maximization algorithm from. Must be in the correct order according to the model specification (see Details).
`dailyRestart`	if `TRUE` the conditional duration will start fresh every new trading day. Can only be used if the `durations` arguments included the clock time of the durations, or if the `time` argument was provided.
`optimFnc`	Specifies which optimization function to use for the estimation. `"optim"`, `"nlminb"`, `"solnp"`, and `"optimx"` are available.
`method`	Argument passed to the optimization function if `optimFnc = "optim"` or `optimFnc = "optimx"` were chosen. Specifies the optimization algorithm. See the help files for optim, nlminb or solnp.

`output` if `FALSE` the estimation results won't be printed.

`bootstrapErrors` if `TRUE` the standard errors will be computed by using bootstrap simulations. Currently only works with the standard ACD model.

`forceErrExpec` if `TRUE` the expectation of the error terms' distribution will be forced to be 1, otherwise the distribution parameter specifying the mean will be set to 1 to ensure identification.

`fixedParamPos` a logical vector of `TRUE` and `FALSE`. Can only be used if the argument `startPara` was provided, and should be of the same length. Each element represents the respective start parameter and if `TRUE`, this parameter will be held fixed when estimating the other parameters.

`bp` a numeric vector of breakpoints, used if the `model` is any of `"SNIACD"`, `"LSNIACD"`, `"TACD"`, `"TAMACD"`. The default is a single breakpoint at 1.

`exogenousVariables` specifies the columns in the `durations` data.frame that should be used as exogenous variables when fitting the model. Must be a vector, either with the column positions or the names of the columns. It is highly recommended to standardize the exogenous variables before running the estimation. The models use the current exogenous variable and not their lagged values.

`optimFncArgs` a list of values passed to the optimaization function.

`control` a list of control values:

- **maxit** maximum number of iterations performed by the numerical maximization algorithm.
- **trace** An integer. If set to a non-zero value, the parameter values at each call of the log-likelihood function during optimization will be recorded, and the resulting search path of the maximum-likelihood estimate will be plotted. This argument is also passed on to the underlying optimizer; see the help files for optim, nlminb or solnp for details.
- **B** number of bootstrap samples

Details

The `startPara` argument is a vector of the parameter values to start from. The length of the vector naturally depends on the `model` and `distribution`. The first elements represent the model parameters, and the last elements the distribution parameters. For example for an ACD(1,1) with Weibull errors the first 3 elements are $\omega, \alpha_1, \beta_1$ for the model, and the last is γ for the Weibull distribution.

The family of ACD models are

$$x_i = \mu_i \epsilon_i,$$

where different specifications of the conditional mean duration μ_i and the error term ϵ_i give rise to different models as shown below.

When exogenous variables are used, they are added in the form of

$$\sum_{j=1}^{k} \xi_j z_j$$

to the right hand side of the equations, where z_j are the exogenous variables.

Conditional mean duration μ_i specifications according to the `model` argument:

ACD(p, q) specification: (Engle and Russell, 1998)

$$\mu_i = \omega + \sum_{j=1}^{p} \alpha_j x_{i-j} + \sum_{j=1}^{q} \beta_j \mu_{i-j}$$

The element order of the `startPara` vector is $(\omega, \alpha_j ..., \beta_j ...)$.

LACD1(p, q): (Bauwens and Giot, 2000)

$$\ln \mu_i = \omega + \sum_{j=1}^{p} \alpha_j \ln \epsilon_{i-j} + \sum_{j=1}^{q} \beta_j \ln \mu_{i-j}$$

The element order of the `startPara` vector is $(\omega, \alpha_j ..., \beta_j ...)$.

LACD2(p, q): (Lunde, 1999)

$$\ln \mu_i = \omega + \sum_{j=1}^{p} \alpha_j \epsilon_{i-j} + \sum_{j=1}^{q} \beta_j \ln \mu_{i-j}$$

The element order of the `startPara` vector is $(\omega, \alpha_j ..., \beta_j ...)$.

AMACD(p, r, q) (Additive and Multiplicative ACD): (Hautsch , 2012)

$$\mu_i = \omega + \sum_{j=1}^{p} \alpha_j x_{i-j} + \sum_{j=1}^{r} \nu_j \epsilon_{i-j} + \sum_{j=1}^{q} \beta_j \mu_{i-j}$$

The element order of the `startPara` vector is $(\omega, \alpha_j ..., \nu_j ..., \beta_j ...)$.

ABACD(p, q) (Augmented Box-Cox ACD): (Hautsch, 2012)

$$\mu_i^{\delta_1} = \omega + \sum_{j=1}^{p} \alpha_j \left(|\epsilon_{i-j} - \nu| + c|\epsilon_{i-j} - \nu|\right)^{\delta_2} + \sum_{j=1}^{q} \beta_j \mu_{i-j}^{\delta_1}$$

The element order of the `startPara` vector is $(\omega, \alpha_j ..., \beta_j ..., c, \nu, \delta_1, \delta_2)$.

BACD(p, q) (Box-Cox ACD): (Hautsch, 2003)

$$\mu_i^{\delta_1} = \omega + \sum_{j=1}^{p} \alpha_j \epsilon_{i-j}^{\delta_2} + \sum_{j=1}^{q} \beta_j \mu_{i-j}^{\delta_1}$$

The element order of the `startPara` vector is $(\omega, \alpha_j ..., \beta_j ...)$.

EXACD(p, q) (Exponential ACD): (Dufour and Engle, 2000)

$$\ln \mu_i = \omega + \sum_{j=1}^{p} \left(\alpha_j \epsilon_{i-j} + \delta_j |\epsilon_{i-j} - 1| + \sum_{j=1}^{q} \beta_j \mu_{i-j} \right)$$

The element order of the `startPara` vector is $(\omega, \alpha_j ..., \beta_j ..., \delta_j ...)$.

SNIACD(p, q, M) (Spline News Impact ACD): (Hautsch, 2012, with a slight difference)

$$\mu_i = \omega + \sum_{j=1}^{p} (\alpha_{j-1} + c_0)\epsilon_{i-j} + \sum_{j=1}^{p} \sum_{k=1}^{M} (\alpha_{j-1} + c_k) 1_{(\epsilon_{i-j} \leq \bar{\epsilon_k})} + \sum_{j=1}^{q} \beta_j \mu_{i-j},$$

where $1_{()}$ is an indicator function and $\alpha_0 = 0$.
The element order of the `startPara` vector is $(\omega, c_k ..., \alpha_j ..., \beta_j ...)$ (The number of α-parameters are $p - 1$).

The distribution of the error term ϵ_i specifications according to the `dist` argument:

Exponential distribution, `dist = "exponential"`**:**

$$f(\epsilon) = \exp(-\epsilon)$$

Weibull distribution, `dist = "weibull"`**:**

$$f(\epsilon) = \theta \gamma \epsilon^{\gamma - 1} e^{-\theta \epsilon^{\gamma}},$$

where $\theta = [\Gamma(\gamma^{-1} + 1)]^{\gamma}$ if `forceErrExpec = TRUE`.

Note that this is a diffrent parameterization then the one in `stats::dweibull`. While the `shape` parameter in `stats::dweibull` is the same as γ, the `scale` parameter is equal to $\theta^{-\gamma}$.

Burr distribution, `dist = "burr"`**:**

$$f(\epsilon) = \frac{\theta \kappa \epsilon^{\kappa - 1}}{(1 + \sigma^2 \theta \epsilon^{\kappa})^{\frac{1}{\sigma^2} + 1}},$$

where,

$$\theta = \sigma^{2\left(1 + \frac{1}{\kappa}\right)} \frac{\Gamma\left(\frac{1}{\sigma^2} + 1\right)}{\Gamma\left(\frac{1}{\kappa} + 1\right) \Gamma\left(\frac{1}{\sigma^2} - \frac{1}{\kappa}\right)},$$

if `forceErrExpec = TRUE`.
The element order of the `startPara` vector is $(modelparameters, \kappa, \sigma^2)$.

Generalized Gamma distribution, `dist = "gengamma"`**:**

$$f(\epsilon) = \frac{\gamma \epsilon^{\kappa \gamma - 1}}{\lambda^{\kappa \gamma} \Gamma(\kappa)} \exp\left\{ - \left(\frac{\epsilon}{\lambda} \right)^{\gamma} \right\}$$

where $\lambda = \frac{\Gamma(\kappa)}{\Gamma(\kappa+\frac{1}{\gamma})}$ if `forceErrExpec = TRUE`. The element order of the `startPara` vector is $(modelparameters, \kappa, \gamma)$.

Generalized F distribution, `dist = "genf"`**:**

$$f(\epsilon) = \frac{\gamma\epsilon^{\kappa\gamma-1}[\eta + (\epsilon/\lambda)^{\gamma}]^{-\eta-\kappa}\eta^{\eta}}{\lambda^{\kappa\gamma}B(\kappa,\eta)},$$

where $B(\kappa, \eta) = \frac{\Gamma(\kappa)\Gamma(\eta)}{\Gamma(\kappa+\eta)}$, and if `forceErrExpec = TRUE`,

$$\lambda = \frac{\Gamma(\kappa)\Gamma(\eta)}{\eta^{1/\gamma}\Gamma(\kappa + 1/\gamma)\Gamma(\eta - 1/\gamma)}.$$

The element order of the `startPara` vector is $(modelparameters, \kappa, \eta, \gamma)$.

q-Weibull distribution, `dist = "qweibull"`**:**

$$f(\epsilon) = (2-q)\frac{a}{b^a}\epsilon^{a-1}\left[1 - (1-q)\left(\frac{\epsilon}{b}\right)^a\right]^{\frac{1}{1-q}}$$

where if `forceErrExpec = TRUE`,

$$b = \frac{(q-1)^{\frac{1+a}{a}}}{2-q}\frac{a\Gamma(\frac{1}{q-1})}{\Gamma(\frac{1}{a})\Gamma(\frac{1}{q-1} - \frac{1}{a} - 1)}.$$

The element order of the `startPara` vector is $(modelparameters, a, q)$.

Value

a list of class `"acdFit"` with the following slots:

`durations`	the durations object used to fit the model.
`muHats`	a vector of the estimated conditional mean durations
`residuals`	the residuals from the fitted model, calculated as durations/mu
`model`	the model for the conditional mean durations
`order`	the order of the model
`distribution`	the assumed error term distribution
`distCode`	the internal code used to represent the distribution
`mPara`	a vector of the estimated conditional mean duration parameters
`dPara`	a vector of the estimated error distribution parameters
`Npar`	total number of parameters
`goodnessOfFit`	a data.frame with the log likelihood, AIC, BIC, and MSE calculated as the mean squared deviation of the durations and the estimated conditional durations.

parameterInference
: a data.frame with the estimated coefficients and their standard errors and p-values

forcedDistPara
: the value of the unfree distribution parameter. If `forceErrExpec = TRUE` were used, this parameter is a function of the other distribution parameters, to force the mean of the distribution to be one. Otherwise the parameter was fixed at 1 to ensure identification.

comments

hessian
: the numerical hessian of the log likelihood evaluated at the estimate

N
: number of observations

evals
: number of log-likelihood evaluations needed for the maximization algorithm

convergence
: if the maximization algorithm converged, this value is zero. (see the help file optim, nlminb or solnp)

estimationTime
: time required for estimation

description
: who fitted the model and when

robustCorr
: only available for QML estimation (choosing the exponential distribution) for the standard ACD(p, q) model. The robust correlation matrix of the parameter estimates.

Author(s)

Markus Belfrage

References

Bauwens, L., and P. Giot (2000) *The logarithmic ACD model: an application to the bid-ask quote process of three NYSE stocks.* Annales d'Economie et de Statistique, 60, 117-149.

Dufour, A., & Engle, R. F. (2000) *The ACD Model: Predictability of the Time Between Consecutive Trades* (Discussion Papers in Finance: 2000-05). ISMA Centre, University of Reading.

Engle R.F, Russell J.R. (1998) *Autoregressive Conditional Duration: A New Model for Irregularly Spaced Transaction Data*, Econometrica, 66(5): 1127-1162.

Grammig, J., and Maurer, K.-O. (2000) *Non-monotonic hazard functions and the autoregressive conditional duration model.* Econometrics Journal 3: 16-38.

Hautsch, N. (2003) *Assessing the Risk of Liquidity Suppliers on the Basis of Excess Demand Intensities.* Journal of Financial Econometrics (2003) 1 (2): 189-215

Hautsch, N. (2012) *Econometrics of Financial High-Frequency Data.* Berlin, Heidelberg: Springer.

Lunde, A. (1999): *A generalized gamma autoregressive conditional duration model,* Working paper, Aalborg University.

Examples

```
data(adjDurData)
fitModel <- acdFit(durations = adjDurData, model = "ACD",
           dist = "exponential", order = c(1,1), dailyRestart = TRUE)
```

acdFit-methods *Methods for class acdFit*

Description

`residuals.acdFit()` returns the residuals and `coef.acdFit()` returns the coefficients of a fitted ACD model of class 'acdFit', while `print.acdFit()` prints the essential information. `predict.acdFit()` predicts the next `N` durations by thier expected value.

Usage

```
## S3 method for class 'acdFit'
residuals(object, ...)
## S3 method for class 'acdFit'
coef(object, returnCoef = "all", ...)
## S3 method for class 'acdFit'
print(x, ...)
## S3 method for class 'acdFit'
predict(object, N = 10, ...)
```

Arguments

`object`	the fitted ACD model of class 'acdFit' (as returned by the function `acdFit`).
`x`	same as `object`, ie. an object of class 'acdFit'.
`returnCoef`	on of "all", "distribution", or "model". Specifies whether all estimated parameters should be returned or only the distribution parameters or the model (for the conditional mean duration) parameters.
`N`	the number of the predictions in `predict`.
`...`	additional arguments to `print`.

acf_acd *Autocorrelation function plots for ACD models*

Description

plots the ACF (Auto Correlation Function) for the durations, diurnally adjusted durations, and residuals.

Usage

```
acf_acd(fitModel = NULL, conf_level = 0.95, max = 50, min = 1,
        plotDur = TRUE, plotAdjDur = TRUE, plotResi = TRUE)
```

Arguments

- `fitModel` — a fitted model of class "acdFit", or a data.frame containing at least one the columns "durations", "adjDur", or "residuals". Can also be a vector of durations or residuals.
- `conf_level` — the confidence level of the confidence bands
- `max` — the largest lag to plot
- `min` — the smallest lag to plot
- `plotDur`, `plotAdjDur`, `plotResi` — logical falgs. If `FALSE`, the respective ACF wont be plotted.

Value

returns a data.frame with the values of the sample autocorrelations for each lag and variable.

Author(s)

Markus Belfrage

Examples

```
data(adjDurData)
fitModel <- acdFit(adjDurData)
acf_acd(fitModel, conf_level = 0.95, max = 50, min = 1)

data(durData)
f <- acf_acd(durData)
f
```

BurrDist *The Burr Distribution*

Description

Density, distribution function, quantile function, random generation and calculation of the expected value for the Burr distribution with parameters theta, kappa and sig2.

Usage

```
dburr(x, theta = 1, kappa = 1.2, sig2 = 0.3, forceExpectation = F)
pburr(x, theta = 1, kappa = 1.2, sig2 = .3, forceExpectation = F)
qburr(p, theta = 1, kappa = 1.2, sig2 = .3, forceExpectation = F)
rburr(n = 1, theta = 1, kappa = 1.2, sig2 = .3, forceExpectation = F)
burrExpectation(theta = 1, kappa = 1.2, sig2 = .3)
```

Arguments

`x`	vector of quantiles.
`p`	vector of probabilities.
`n`	number of observations..
`theta`, `kappa`, `sig2`	parameters, see 'Details'.
`forceExpectation`	logical; if `TRUE`, the expectation of the distribution is forced to be 1 by letting `theta` be a function of the other parameters.

Details

The PDF for the Burr distribution is (as in e.g. Grammig and Maurer, 2000):

$$f(x) = \frac{\theta \kappa x^{\kappa - 1}}{(1 + \sigma^2 x^{\kappa})^{\frac{1}{\sigma^2} + 1}}$$

Value

`dburr` gives the density (PDF), `qburr` the quantile function (inverted CDF), `rburr` generates random deviates, and `burrExpectation` returns the expected value of the distribution, given the parameters.

Author(s)

Markus Belfrage

References

Grammig, J., and Maurer, K.-O. (2000) *Non-monotonic hazard functions and the autoregressive conditional duration model*. Econometrics Journal 3: 16-38.

`computeDurations` *Durations computation*

Description

Computes durations from a data.frame containing the time stamps of transactions. Trade durations, price durations and volume durations can be computed (if the appropriate data columns are given).

Usage

```
computeDurations(transactions, open = "10:00:00", close = "18:25:00",
rm0dur = TRUE, type = "trade", priceDiff = .1, cumVol = 10000)
```

Arguments

`transactions`	a data.frame with, at least, transaction time in a column named 'time' (see Details)
`open`	the opening time of the exchange. Transactions done outside the trading hours will be ignored.
`close`	the closing time of the exchange.
`rm0dur`	if `TRUE` zero-durations will be removed and transactions done on the same second will be aggregated, e.g. price will then be the volume weighted avrage price of the aggregated transactions.
`type`	the type of durations to be computed. Either "trade", "price", or "volume".
`priceDiff`	only if `type = "price"`. Price durtions are (here) defind as the duration until the price has changed by at least 'priceDiff' in absolute value.
`cumVol`	only if `type = "cumVol"`. Volume durtions are (here) defind as the duration until the cumulative traded volume since the last duration has surpassed 'cumVol'.

Details

The data.frame must include a column named 'time' with the time of each transaction, in a time format recognizable by POSIXlt or strings in format "yyyy-mm-dd hh:mm:ss". If the column 'price' or 'volume' is included its also possible to compute price- and volume durations (see arguments `priceDiff` and `cumVol`)

Value

a data.frame with columns:

`time`	the calander time of the start of each duration spell.
`price`	the volume weighted avrage price of the shares traded during the spell of the duration.
`volume`	the volume (total shares traded) during the duration spell.
`Ntrans`	number of transactions done during the spell.
`durations`	the computed durations.

Author(s)

Markus Belfrage

Examples

```
## Not run:
#only the first 3 days of data:
data(transData)
durDataShort <- computeDurations(transData[1:56700, ])
head(durDataShort)
## End(Not run)
```

DataFiles — *Time Series Data Sets*

Description

The data file `transData` is the base data used in all of the examples. It is a data.frame with rows representing a single transaction and has the columns 'time', 'price', giving the trade price, and 'volume', giving the number of shares traded for the transaction. The data set is based on real transactions but has been obfuscated by transforming the dates, price and volume, for proprietary reasons. It covers two weeks of nearly 100 000 transactions, recorded with 1 second precision.

The `durData` data.frame is simply the trade durations formed from `transData` using the function `durData <- computeDurations(transData)`

The `adjDurData` data object is in turn created by `adjDurData <- diurnalAdj(durData, aggregation = "all")` to add diurnally adjusted durations.

`defaultSplineObj` is an estimated cubic spline of the diurnal component using the sample data. It is used when simulating from `sim_ACD()` with the argument `diurnalFactor` set to `TRUE`, when no user `splineObj` is provided.

dgenf — *The generalized F distribution*

Description

Density and distribution function for the generalized F distribution. Warning: the distribution function `pgenf` and `genfHazard` are computed numerically, and may not be precise!

Usage

```
dgenf(x, kappa = 5, eta = 1.5, gamma = .8, lambda = 1, forceExpectation = F)
pgenf(q, kappa = 5, eta = 1.5, gamma = .8, lambda = 1, forceExpectation = F)
genfHazard(x, kappa = 5, eta = 1.5, gamma = .8, lambda = 1, forceExpectation = F)
```

Arguments

`x, q`
: vector of quantiles.

`kappa, eta, gamma, lambda`
: parameters, see 'Details'.

`forceExpectation`
: logical; if `TRUE`, the expectation of the distribution is forced to be 1 by letting theta be a function of the other parameters.

Details

The PDF for the generelized F distribution is:

$$f(\epsilon) = \frac{\gamma \epsilon^{\kappa\gamma - 1}[\eta + (\epsilon/\lambda)^{\gamma}]^{-\eta-\kappa}\eta^{\eta}}{\lambda^{\kappa\gamma}B(\kappa,\eta)},$$

where $B(\kappa, \eta) = \frac{\Gamma(\kappa)\Gamma(\eta)}{\Gamma(\kappa+\eta)}$ is the beta function.

`Discreetly mixed Q-Weibull and exponential`

Discreet mix of the Q-Weibull and the exponential distributions

---

Description

Density (PDF), distribution function (CDF), and hazard function for a discreetly mixed distribution of the Q-Weibull and the exponential distributions.

Usage

```
dmixqwe(x, pdist = .5, a = .8, qdist = 1.5, lambda = .8, b = 1, forceExpectation = F)
pmixqwe(q, pdist = .5, a = .8, qdist = 1.5, lambda = .8, b = 1, forceExpectation = F)
mixqweHazard(x, pdist = .5, a = .8, qdist = 1.5, lambda = .8, b = 1, forceExpectation = F)
```

Arguments

`x, q`	vector of quantiles.
`pdist, a, qdist, lambda, b`	parameters, see 'Details'.
`forceExpectation`	logical; if `TRUE`, the expectation of the distribution is forced to be 1 by letting `b` be a function of the other parameters.

Details

The PDF for the mixed distribution is:

$$f(x) = p(2-q)\frac{a}{b^a}x^{a-1}\left[1-(1-q)\left(\frac{x}{b}\right)^a\right]^{\frac{1}{1-q}} + (1-p)\frac{1}{\lambda}exp(-\frac{x}{\lambda})$$

if `forceExpectation = TRUE` the `b` parameter is a function of the other parameters to force the expectation to be 1.

See Also

`qWeibullDist` for the Q-Weibull distribution and `pmixqww` for Q-Weibull mixed with the ordinary Weibull.

`Discreetly mixed Q-Weibull and ordinary Weibull`
Discreet mix of the q-Weibull and the ordinary Weibull distributions

---

Description

Density (PDF), distribution function (CDF), and hazard function for a discreetly mixed distribution of the Q-Weibull and the ordinary Weibull distributions.

Usage

```
dmixqww(x, pdist = .5, a = 1.2, qdist = 1.5, theta = .8, gamma = 1, b = 1,
  forceExpectation = F)

pmixqww(q, pdist = .5, a = 1.2, qdist = 1.5, theta = .8, gamma = 1, b = 1,
  forceExpectation = F)

mixqwwHazard(x, pdist = .5, a = 1.2, qdist = 1.5, theta = .8, gamma = 1, b = 1,
  forceExpectation = F)
```

Arguments

`x, q`	vector of quantiles.
`pdist, a, qdist, theta, gamma, b`	parameters, see 'Details'.
`forceExpectation`	logical; if `TRUE`, the expectation of the distribution is forced to be 1 by letting `b` be a function of the other parameters.

Details

The PDF for the mixed distribution is:

$$f(x) = p(2-q)\frac{a}{b^a}x^{a-1}\left[1-(1-q)\left(\frac{x}{b}\right)^a\right]^{\frac{1}{1-q}} + (1-p)\theta\gamma x^{-\theta x^{\gamma}}$$

if `forceExpectation` = `TRUE` the `b` parameter is a function of the other parameters to force the expectation to be 1.

See Also

`qWeibullDist` for the Q-Weibull distribution and `pmixqwe` for Q-Weibull mixed with the exponential distribution.

diurnalAdj *Dirunal adjustment for durations*

Description

Performs a diurnal adjustment of the durations, i.e. removes a daily seasonal component. Four different methods of diurnal adjustment are available, namely "cubicSpline", "supsmu" (Friedman's SuperSmoother), "smoothSpline" (smoothed version of the cubic spline), or "FFF" (Flexible Fourier Form).

Usage

```
diurnalAdj(dur, method = "cubicSpline", nodes = c(seq(600, 1105, 60), 1105),
aggregation = "all", span = "cv", spar = 0, Q = 4, returnSplineFnc = FALSE)
```

Arguments

`dur` a data.frame containing the columns `durations`, containing durations, and `time`, containing the time stamps.

`method` the method used. One of "cubicSpline", "supsmu", "smoothSpline", or "FFF".

`nodes` only for `method = "cubicSpline"` or `method = "smoothSpline"`. A vector of nodes to use for the spline function, in the unit minutes after midnight. The first and last element of the vector must be the start and end of the trading day. The nodes given are actually the limits of intervalls, of wich the midpoints will be set as the nodes using the means of the intervals.

`aggregation` what type of aggregation to use. Either "weekdays", "all", or "none". If for example "weekdays" is chosen, all Mondays will have the same daily seasonal component, and so on.

`span` argument passed to supsmu if `method = "supsmu"` were chosen. Affects the smoothness of the curve, see supsmu.

`spar` argument passed to smooth.spline if `method = "smooth.spline"` were chosen. Affects the smoothness of the curve, see smooth.spline.

`Q` number of trigonometric function pairs for `method = "FFF"`.

`returnSplineFnc`
if `TRUE` instead or returning the adjusted durations a list of spline objects will be returned, containing the coefficents of the spline function. Only available for `method = "cubicSpline"`.

Value

`if returnSplineFnc = FALSE (default)`
the input data.frame 'dur' with an added column of the diurnally adjusted durations called 'adjDur'.

`if returnSplineFnc = TRUE`
a list of spline objects containing the coefficents of the spline function.

Author(s)

Markus Belfrage

Examples

```
data(durData)
diurnalAdj(durData, aggregation = "none", method = "supsmu")

## Not run:

head(durData)
f <- diurnalAdj(durData, aggregation = "weekdays", method = "FFF", Q = 3)
head(f)

f <- diurnalAdj(durData, aggregation = "all", returnSplineFnc = TRUE)
f

## End(Not run)
```

`Finite mixture of inverse Gaussian Distributions`
Finite mixture of inverse Gaussian Distribution

Description

Density (PDF), distribution function (CDF), and hazard function for Finite mixture of inverse Gaussian Distributions.

Usage

```
dmixinvgauss(x, theta = .2, lambda = .1, gamma = .05, forceExpectation = F)
pmixinvgauss(q, theta = .2, lambda = .1, gamma = .05, forceExpectation = F)
mixinvgaussHazard(x, theta = .2, lambda = .1, gamma = .05, forceExpectation = F)
```

Arguments

`x, q` vector of quantiles.

`theta, lambda, gamma`
parameters, see 'Details'.

`forceExpectation`
logical; if `TRUE`, the expectation of the distribution is forced to be 1..

Details

The finite mixture of inverse Gaussian distributions was used by Gomes-Deniz and Perez-Rodrigues (2013) for ACD-models. Its PDF is:

$$f(x) = \frac{\gamma + x}{\gamma + \theta} \sqrt{\frac{\lambda}{2\pi x^3}} \exp\left[-\frac{\lambda(x-\theta)^2}{2x\theta^2} \right].$$

If `forceExpectation = TRUE` the distribution is transformed by dividing the random variable with its expectation and using the change of variable function.

References

Gomez-Deniz, E. and Perez-Rodriguez, J.V. (2016) *Mixture Inverse Gaussian for Unobserved Heterogeneity in the Autoregressive Conditional Duration Model.* Communications in Statistics - Theory and Methods, http://dx.doi.org/10.1080/03610926.2016.1200094

`GeneralizedGammaDist` *The generelized Gamma distribution*

Description

Density (PDF), distribution function (CDF), quantile function (inverted CDF), random generation and hazard function for the generelized Gamma distribution with parameters gamma, kappa and lambda.

Usage

```
dgengamma(x, gamma = 0.3, kappa = 1.2, lambda = 0.3, forceExpectation = F)
pgengamma(x, gamma = .3, kappa = 3, lambda = .3, forceExpectation = F)
qgengamma(p, gamma = .3, kappa = 3, lambda = .3, forceExpectation = F)
rgengamma(n = 1, gamma = .3, kappa = 3, lambda = .3, forceExpectation = F)
gengammaHazard(x, gamma = .3, kappa = 3, lambda = .3, forceExpectation = F)
```

Arguments

`x`	vector of quantiles.
`p`	vector of probabilities.
`n`	number of observations..
`gamma, kappa, lambda`	parameters, see 'Details'.
`forceExpectation`	logical; if `TRUE`, the expectation of the distribution is forced to be 1 by letting `theta` be a function of the other parameters.

Details

The PDF for the generelized Gamma distribution is:

$$f(x) = \frac{\gamma x^{\kappa\gamma-1}}{\lambda^{\kappa\gamma}\Gamma(\kappa)} \exp\left\{-\left(\frac{x}{\lambda}\right)^{\gamma}\right\}$$

Value

`dgengamma` gives the density (PDF), `pgengamma` gives the distribution function (CDF), `qgengamma` gives the quantile function (inverted CDF), `rgenGamma` generates random deviates, and `genGammaHazard` gives the hazard function.

Author(s)

Markus Belfrage

`plotDescTrans` *Transactions plots*

Description

Plots (1) the price over time, (2) volume traded over time for a given interval, and (3) number of transactions over time for a given interval.

Usage

```
plotDescTrans(trans, windowunit = "hours", window = 1)
```

Arguments

`trans`	a data.frame with the column 'time', 'price', and 'volume'. Currently only works if all of those are available.
`windowunit`	the unit of the time interval. One of "secs", "mins", "hours", or "days".
`window`	a positive integer giving the length of the interval.

Examples

```
## Not run:
data(transData)
plotDescTrans(transData, windowunit = "hours", window = 1)
## End(Not run)
```

`plotHazard` *Hazard function plot*

Description

Estimates and plots the hazard function from an estimatated ACD model.

Usage

```
plotHazard(fitModel, breaks = 20, implied = TRUE, xstop)
```

Arguments

`fitModel`	an estimated model of class acdFit. Can also be a numerical vector.
`breaks`	the number of quantiles used to estimate the hazard.
`implied`	a logical flag. If TRUE then the implied hazard function using the distribution parameter estimates will be plotted together with the nonparametric estimate of the error term hazard function.
`xstop`	where to stop plotting the implied hazard.

Details

This estimator of the hazard function is based on the one used by Engle and Russell (1998). It is modified sligthly to decrease its bias and inconsistency. However, the estimator is still not fully consistent when using a fixed number of breaks (quantiles).

Author(s)

Markus Belfrage

References

Engle, R.F and Russell, J.R. (1998) Autoregressive Conditional Duration: A New Model for Irregularly Spaced Transaction Data. *Econometrica*, 66(5): 1127-1162.

Examples

```
## Not run:

data(adjDurData)
fitModelWei <- acdFit(adjDurData, dist = "wei")
plotHazard(fitModelWei)

## End(Not run)
```

`plotHistAcd` *Mean duration plot*

Description

Plots the mean duration over time at chosen interval length

Usage

```
plotHistAcd(durations, windowunit = "mins", window = 1)
```

Arguments

`durations`	a data.frame containing the durations and their time stamps.
`windowunit`	the unit of the time interval. One of "secs", "mins", "hours", or "days".
`window`	a positive integer giving the length of the interval.

Author(s)

Markus Belfrage

Examples

```
data(durData)
plotHistAcd(durData, windowunit = "days", window = 1)

## Not run:

plotHistAcd(durData, windowunit = "mins", window = 30)

## End(Not run)
```

`plotLL` *Plots the response surface of the log likelihood of a fitted model.*

Description

Plots the log likelihood for a fitted model against either one or two of the parameters at a time. This can help to find issues with for example poor identification of a model.

Usage

```
plotLL(fitModel, parameter1 = 1, parameter2 = NULL,
       param1sequence = NULL, param2sequence = NULL, startpoint = NULL,
       length.out = NULL, returnOutput = FALSE)
```

Arguments

`fitModel`	a fitted model of class acdFit.
`parameter1`	the first parameter for the log likelihood to be plotted against. Either the index of the parameter as an integer, or the name of the parameter.
`parameter2`	the second parameter for the log likelihood to be plotted against. Either the index of the parameter as an integer, or the name of the parameter. If left empty, a plot with only the `parameter1` will be drawn.
`param1sequence, param2sequence`	the sequence of points from which the log likelihood is computed. If left empty, the log likelihood will be computed at 21 points spanning between MLE-3*SD and MLE+3*SD in the one dimensional case, and the 11x11 points for the same range in the two dimensional case.
`startpoint`	a vector of size equal to the number of parameters in the model. If this is supplied, the log likelihood will be evaluated at this point instead of the point of the MLE (for the parameters not in `parameter1` and `parameter2`).
`length.out`	(optional) overrides the default number of points (for each dimension) where the log likelihood is computed.
`returnOutput`	a logical flag. If set to `TRUE`, the values of the response surface will be returned. See 'value' below.

Value

Only if `returnOutput = TRUE`

1. For the one dimensional case: a data.frame with the columns 'logLikelihood', and 'param1sequence' for all the values of the `parameter1` witch the log likelihood was evaluated at

2. For the two dimensional case: a list with the following items:

`para1`	a vector with the sequence of the `parameter1` values.
`para2`	a vector with the sequence of the `parameter2` values.
`z`	a matrix with the log likelihood values. The element at the ith row and jth column is evaluated at the ith `para1` value and jth `para2` value.

Author(s)

Markus Belfrage

Examples

```
## Not run:


data(adjDurData)
#Indicates identification issues with the generelized gamma distibution:
#(Try a diffrent 'startPara' in acdFit() to get slightly a better fit)
fit <- acdFit(durations = adjDurData[1:3000, ], dist = "gengamma")
seq1 <- seq(500, 1000, 50)
```

```
seq2 <- seq(.02, 0.045, 0.001)
plotLL(fitModel = fit, parameter1 = "kappa", parameter2 = "gamma",
       param1sequence = seq1, param2sequence = seq2)

## End(Not run)
```

`plotRollMeanAcd` *Plots rolling means of durations*

Description

Plots rolling means of durations

Usage

```
plotRollMeanAcd(durations, window = 500)
```

Arguments

`durations`	a data.frame containing the column 'time' and 'durations'.
`window`	the length of the rolling window.

Examples

```
data(durData)
plotRollMeanAcd(durData, window = 500)
```

`plotScatterAcd` *Scatter plot for ACD models*

Description

Function to help scatter plot different variables of a fitted ACD model and superimposes a smoothed conditional mean using ggplot2. Can be used to investigate the possible need for non-linear models and issues with the diurnal adjustment.

Usage

```
plotScatterAcd(fitModel, x = "muHats", y = "residuals", xlag = 0, ylag = 0,
                              colour = NULL, xlim = NULL, ylim = NULL, alpha = 1/10,
                                smoothMethod = "auto")
```

Arguments

fitModel	a fitted model of class "acdFit"
x	the variable used on the x-axis. One of "muHats", "residuals", "durations", "adjDur", "dayTime", "time", or "index".
y	the variable used on the y-axis. One of "muHats", "residuals", "durations", "adjDur", "dayTime", "time", or "index".
xlag	number of lags used for the variable shown on the x-axis.
ylag	number of lags used for the variable shown on the y-axis.
colour	a possible third variable to be represented with a colour scale. One of "muHats", "residuals", "durations", "adjDur", "dayTime", or "time".
xlim	a vector of the limits of the x-axis to possibly zoom in on a certain region.
ylim	a vector of the limits of the y-axis to possibly zoom in on a certain region.
alpha	`alpha` parameter passed to ggplot2. For large data sets many data points will overlap. The alpha parameter can make the points transparent, making it easier to distinguish the density of different region. Takes the value between 1 (opaque) and 0 (completely transparent).
smoothMethod	value passed as `smooth` argument to ggplot2. See stat_smooth.

Author(s)

Markus Belfrage

Examples

```
## Not run:

data(adjDurData)
# The mean residuals are too small for small values of the estimated conditional
# mean, suggesting a need for a different conditional mean model specification:
fitModel <- acdFit(adjDurData)
plotScatterAcd(fitModel, x = "muHats", y = "residuals")

## End(Not run)
```

---

qqplotAcd *Quantile-Quantile plot of the residuals*

Description

Plots a QQ-plot of the residuals and the theoretical quantiles implied by the model estimates.

Usage

```
qqplotAcd(fitModel, xlim = NULL, ylim = NULL)
```

Arguments

`fitModel`	a fitted ACD model, i.e. an object of class "acdFit"
`xlim`	an optional vector of limits for the x-axis
`ylim`	an optional vector of limits for the y-axis

Examples

```
data(adjDurData)
fitModelExp <- acdFit(adjDurData, dist = "exp")
qqplotAcd(fitModelExp)
```

`qWeibullDist` *The q-Weibull distribution*

Description

Density (PDF), distribution function (CDF), quantile function (inverted CDF), random generation, exepcted value, and hazard function for the q-Weibull distribution.

Usage

```
dqweibull(x, a = .8, qdist = 1.2, b = 1, forceExpectation = F)
pqweibull(q, a = .8, qdist = 1.2, b = 1, forceExpectation = F)
qqweibull(p, a = .8, qdist = 1.2, b = 1, forceExpectation = F)
rqweibull(n = 1, a = .8, qdist = 1.2, b = 1, forceExpectation = F)
qweibullExpectation(a = .8, qdist = 1.2, b = 1)
qweibullHazard(x, a = .8, qdist = 1.2, b = 1, forceExpectation = F)
```

Arguments

`x, q`	vector of quantiles.
`p`	vector of probabilities.
`n`	number of observations.
`a, qdist, b`	parameters, see 'Details'.
`forceExpectation`	logical; if `TRUE`, the expectation of the distribution is forced to be 1 by letting `b` be a function of the other parameters.

Details

The PDF for the q-Weibull distribution is:

$$f(\epsilon) = (2-q)\frac{a}{b^a}\epsilon^{a-1}\left[1-(1-q)\left(\frac{\epsilon}{b}\right)^a\right]^{\frac{1}{1-q}}$$

The distribution was used for ACD models by Vuorenmaa (2009).

References

Vuorenmaa, T. (2009) *A q-Weibull Autoregressive Conditional Duration Model with an Application to NYSE and HSE data.* Available at SSRN: http://ssrn.com/abstract=1952550.

`resiDensityAcd` *Residual Density Histogram*

Description

Plots a density histogram of the residuals and superimposes the density implied by the model estimates.

Usage

```
resiDensityAcd(fitModel, xlim = NULL, binwidth = .1, density = FALSE)
```

Arguments

`fitModel`	a fitted ACD model, i.e. an object of class "acdFit"
`xlim`	an optional vector of limits for the x-axis
`binwidth`	the width of the bins of the density histogram.
`density`	if `TRUE` a kernel density estimate will be added

Author(s)

Markus Belfrage

Examples

```
## Not run:
data(adjDurData)
fitModelBurr <- acdFit(adjDurData, dist = "burr")
resiDensityAcd(fitModelBurr)
## End(Not run)
```

sim_ACD *ACD simulation*

Description

Simulates a sample from a specified ACD `model` and error term distribution `dist`. The error terms can also be sampled from residuals.

Usage

```
sim_ACD(N = 1000, model = "ACD", dist = "exponential", param = NULL, order = NULL,
   Nburn = 50, startX = c(1), startMu = c(1), errors = NULL, sampleErrors = TRUE,
    roundToSec = FALSE, rm0 = FALSE, bp = NULL)
```

Arguments

`N`	sample size
`model`	the class of conditional mean duration specification. One of `"ACD"`, `"LACD1"`, `"LACD2"`, `"AMACD"`,`"ABACD"`, `"SNIACD"` or `"LSNIACD"`. Can also be an object of class `acdFit`, as returned by `acdFit()`.
`dist`	the distribution of the error terms (only if `errors` are left out). Must be one of `"exponential"`, `"weibull"`, `"burr"`, `"gengamma"` or `"genf"`.
`param`	a vector of the parameters of the DGP (data generating process).
`order`	a vector describing the order of the conditional mean duration specification, e.g. `order = c(1,1)` for an ACD(1,1) model.
`Nburn`	the number of burned observations. Used to lower the effect of the start values of the simulated series.
`startX`	a vector of values to start the simulation from.
`startMu`	a vector of conditional mean values to start the simulation from.
`errors`	a vector of error terms. If provided and `sampleErrors = TRUE` the errors will be sampled from this vector (with replacement). If instead `sampleErrors = FALSE` the error terms will be matched by the `errors` vector non stochastic (must then be of the same length as `N + Nburn`)
`sampleErrors`	logical flag, see `errors` above. Default is `TRUE`.
`roundToSec`	if `TRUE` the simulated sample will be discretized with 1 second(unit) precision.
`rm0`	if `TRUE` zero durations will be removed. Will the result in a smaller sample than `N`.
`bp`	a vector of breakpoints used for `model` = "SNIACD", "LSNIACD", "TACD", "TAMACD".

Value

a numerical vector of simulated ACD durations

Author(s)

Markus Belfrage

Examples

```
x <- sim_ACD() #simulates 1000 observations from an ACD(1,1) with exp. errors as default
acdFit(x)
```

---

standardizeResi *Residual standardization*

Description

Standardizes residuals from a fitted ACD model of class 'acdFit' by a probability integral transformation (taking the CDF, using the estimated distribution parameters, of the residuals) or by returning the Cox-Snell residuals.

Usage

```
standardizeResi(fitModel, transformation = "probIntegral")
```

Arguments

`fitModel`	a fitted ACD model of class 'acdFit'.
`transformation`	type of transformation done, either "probIntegral", or "cox-snell".

Details

The probability integral transformation is done by taking the CDF of the residuals from the model estimation, using the estimated distribution parameters. Under correct specification the probability integral transformed residuals should be iid. uniform(0, 1).

The Cox-Snell residuals is the computed by taking the integrated hazard of the residuals from the model estimation, using the estimated distribution parameters. Under correct specification the probability integral transformed residuals should be iid. unit exponentially distributed.

`testRmACD` *LM test of no Remaining ACD (Meitz and Terasvirta, 2006)*

Description

Tests if there is any remaining ACD structure in the residuals (currently only works for `model = "ACD"`).

Usage

```
testRmACD(fitModel, pStar = 2, robust = TRUE)
```

Arguments

`fitModel`	a fitted ACD model, i.e. an object of class "acdFit".
`pStar`	the number of alpha parameters in the alternative hypothesis. See $p*$ under 'Details'.
`robust`	if `TRUE` the LM statistic will be calculated using the "robust" version, making its asymptotic behavior unaffected by possible misspecification of the error term distribution (Meitz and Terasvirta, 2006).

Details

For the model

$$x_i = \mu_i \phi_i \epsilon_i,$$

$$\mu_i = \omega + \sum_{j=1}^{p} \alpha_j x_{i-j} + \sum_{j=1}^{q} \beta_j \mu_{i-j},$$

$$\phi_i = 1 + \sum_{j=1}^{p*} \frac{x_{i-j}}{\mu_{i-j}},$$

the function tests the null hypothesis

$$H_0 : \phi_i = 1.$$

Value

a list of:

`chi2`	the value of the LM statistic.
`pv`	the pvalue of the test statistic.

Author(s)

Markus Belfrage

References

Meitz, M. and Terasvirta, T. (2006). *Evaluating models of autoregressive conditional duration*. Journal of Business and Economic Statistics 24: 104-124.

See Also

`testTVACD`, `testSTACD`.

Examples

```
data(adjDurData)
fitModel3000obs <- acdFit(adjDurData[1:3000,])
testRmACD(fitModel3000obs, pStar = 2, robust = TRUE)
```

---

testSTACD *LM test against Smooth Transition ACD models (Meitz and Terasvirta, 2006)*

Description

Tests if the alpha parameters and the constant should be varying with the value of the lagged durations, according to a logistic transition function (currently only works for `model = "ACD"`).

Usage

```
testSTACD(fitModel, K = 2, robust = TRUE)
```

Arguments

`fitModel`	a fitted ACD model, i.e. an object of class "acdFit".
`K`	the order of the logistic transition function used for the alternative hypothesis.
`robust`	if `TRUE` the LM statistic will be calculated using the "robust" version, making its asymptotic behavior unaffected by possible misspecification of the error term distribution (Meitz and Terasvirta, 2006).

Value

a list of:

`chi2`	the value of the LM statistic.
`pv`	the pvalue of the test statistic.

References

Meitz, M. and Terasvirta, T. (2006) Evaluating models of autoregressive conditional duration. Journal of Business and Economic Statistics 24: 104-124.

See Also

testRmACD, testTVACD.

Examples

```
data(adjDurData)
fitModel3000obs <- acdFit(adjDurData[1:3000,])
testSTACD(fitModel3000obs, K = 2, robust = TRUE)
```

---

testTVACD — *LM test against Time-Varying ACD models (Meitz and Terasvirta, 2006)*

Description

Tests if the parameters are time-varying (currently only works for `model = "ACD"`).

Usage

```
testTVACD(fitModel, K = 2, type = "total", robust = TRUE)
```

Arguments

fitModel	a fitted ACD model, i.e. an object of class "acdFit".
K	the order of the logistic transition function used for the alternative hypothesis.
type	either "total" or "intraday". If "total", the possible time varying parameters under the alternative varies over the total time of the sample, whereas for "intraday", the time variable is time of the day. See 'Details'
robust	if TRUE the LM statistic will be calculated using the "robust" version, making its asymptotic behavior unaffected by possible misspecification of the error term distribution (Meitz and Terasvirta, 2006).

Details

This function tests the fitted standard ACD model against the TVACD model of Meitz and Terasvirta (2006). The TVACD model lets the ACD parameters vary over time by a logistic transition function.

In one specification, the time variable is total time, and a test rejecting the null in favor of this alternative specification would indicate that the ACD parameters are changing over time over the total sample.

The other specification lets the parameters be intraday varying, by letting the transition variable be the time of the day. Failing this test could indicate that the diurnal adjustment was inadequate at removing any diurnal component.

Value

a list of:

`chi2`	the value of the LM statistic.
`pv`	the pvalue of the test statistic.

Author(s)

Markus Belfrage

References

Meitz, M. and Terasvirta, T. (2006). *Evaluating models of autoregressive conditional duration*. Journal of Business and Economic Statistics 24: 104-124.

See Also

`testRmACD`, `testSTACD`.

Examples

```
data(adjDurData)
fitModel5000obs <- acdFit(adjDurData[1:5000,])
testTVACD(fitModel5000obs, K = 2, type = "total", robust = TRUE)

testTVACD(fitModel5000obs, K = 2, type = "intraday", robust = TRUE)
```

Index