Package ‘robustST’

February 5, 2015

Type Package

Title Package to robustly fit skew-t parameters

Version 1.0.0

Date 2015-01-15

Author Joshua M. Browning <rockclimber112358@gmail.com>

Maintainer Joshua M. Browning <rockclimber112358@gmail.com>

Description This package provides functions for robust fitting of the skew-t distributon.

URL https://github.com/rockclimber112358/Robust-Skew-T

License GPL (>= 3)

Imports

Depends sn (>= 1.1-2)

LazyData yes

ZipData no

VignetteBuilder knitr

Suggests knitr, ggplot2

R topics documented:

`fast.TLE.BST` *Wrapper to fastTLE assuming bivariate skew-t data.*

Description

Wrapper to fastTLE assuming bivariate skew-t data.

Usage

```
fast.TLE.BST(data, k, initial = c(0, 0, 1, 0, 1, 0, 0, 1000), ...)
```

Arguments

`data`	The data used in the analysis.
`k`	The proportion of observations used. If k=1, the (non-robust) MLE will be computed.
`initial`	A vector of length eight containing an initial guess for xi1, xi2, omega11, omega12, omega22, alpha1, alpha2, and nu. Defaults to a zero mean, identity omega, no skew, and large nu distribution (i.e. approximately N(0, I)). Note: do not set the nu parameter to Inf, as this causes issues with the optimization.

Value

Same as fastTLE.

`fast.TLE.normal` *Wrapper to fastTLE assuming normal data.*

Description

Wrapper to fastTLE assuming normal data.

Usage

```
fast.TLE.normal(data, k, initial = c(0, 1))
```

Arguments

`data`	The data used in the analysis.
`k`	The proportion of observations used. If k=1, the (non-robust) MLE will be computed.
`initial`	A vector of length two containing an initial guess for the mean and standard deviation.

Value

Same as fastTLE.

`fast.TLE.ST` *Wrapper to fastTLE assuming univariate skew-t data.*

Description

Wrapper to fastTLE assuming univariate skew-t data.

Usage

```
fast.TLE.ST(data, k, initial = c(0, 1, 0, 1000), ...)
```

Arguments

`data`	The data used in the analysis.
`k`	The proportion of observations used. If k=1, the (non-robust) MLE will be computed.
`initial`	A vector of length four containing an initial guess for xi, omega, alpha, and nu.

Value

Same as fastTLE.

`fastTLE` *Fast TLE*

Description

An method of generating robust estimates for parameters of a distribution, different from the capped likelihood approach. Rather than capping the deviance, a few of the worst observations (i.e. largest deviance) are not used in the estimation of the MLE. Thus, this procedure is iterative; the worst observations may change as the MLE changes, and the MLE may change as the set of observations used changes. Fast TLE is an algorithm which is NOT guaranteed to find the optimum, but typically obtains a good approximation quickly.

Usage

```
fastTLE(initial, MLE, negLogLik, data, k, trace = F, ...)
```

Arguments

`initial` A vector for the initial guess for the parameters. Results will depend on this initial value, so it may be reasonable to run with several starting values.

`MLE` A function which takes two arguments: data and start. data is supplied as an argument to fastTLE. This function should return a parameter vector of the same length as initial. start may not do anything (as in the case of normal data) but must be an argument. For optimizations, start should be the best guess, and the last MLE will be supplied.

`negLogLik` A function which takes two arguments: param and data. param should be a vector of the parameters wished to optimize and data is supplied as an argument to fastTLE. This function is used to compute the negative log-likelihood of the observations and hence determines which observations are used in computing the MLE at each iteration. Thus, it should return a vector of negative log-likelihoods for each observation.

`data` The data which should be fed into the data argument of likelihood. data should be a matrix of the data. If it's not a matrix, fastTLE will attempt to coerce it to a matrix (with a warning).

`k` A value between 0 and 1 specifying the should be used to estimate the MLE. fastTLE will use the k*100 observations with the smallest likelihood. Note: This code was not designed to handle ties. This will not be a problem with continuous data, but there could be issues with discrete data.

`trace` Should output be printed as the algorithm proceeds?

`...` Currently not implemented.

Value

A list of two elements: the estimated MLE and a vector specifying which observations were used in the estimation.

`getLogLikelihoodBound` *Get Log Likelihood Bound*

Description

For the robust skew-t, we will occassionally need to understand which values of the negative log-likelihood are "extreme", and this depends on the current estimate of the skew-t parameters. This function computes the negative log-likelihood of the 1-alpha quantile for the skew-t distribution with provided density parameters.

Usage

```
getLogLikelihoodBound(dp, alpha = 0.01)
```

Arguments

`dp` The density parameters of the skew-t.

`alpha` The quantile of the negative log-likelihood.

Details

Caution: This is an approximation! It is very accurate for most cases, but it is not perfect. In particular, when alpha is close to 1, the approximation becomes fairly poor. Fortunately, this is usually not a scenario of much importance as we tend to be more interested in the case of alpha close to 0 (i.e. the tails) rather than 1.

Value

The value of the negative log-likelihood that corresponds to the 1-alpha quantile of the skew-t with density parameters as provided in dp.

`getStartingEstimate` *Get Starting Estimate*

Description

This function is used to provide a reasonable starting estimate for the skew-t parameters. The logic replicates what is implemented in the sn package (in particular, the st.mple function).

Usage

```
getStartingEstimate(y, x = matrix(1, nrow = NROW(y), ncol = 1), w = rep(1,
  NROW(y)))
```

Arguments

`y`	The data for which the skew-t is being fit.
`x`	The design matrix, if y is assumed to be residuals from some fit. This parameter has not been tested thoroughly, so use with caution! It defaults to a matrix of 1's with 1 column, and this is equivalent to simply fitting a skew-t to y.
`w`	The observation weights used in the skew-t fitting.

Value

An initial estimate for the density parameters.

`marginal` *Convert multivariate skew-t parameters into bivariate*

Description

This function converts parameters of a multivariate skew-t distribution into parameters for a bivariate skew-t using formulas from wind_radiosonde_QC_01.pdf (from Mandy).

Usage

```
marginal(xi, omega, alpha, nu, r = 1, alphaAdj = F)
```

Arguments

`xi`	Vector of parameters for the skew-t distribution.
`omega`	matrix of parameters for the skew-t distribution.
`alpha`	Vector of parameters for the skew-t distribution.
`nu`	Numeric value of parameter for the skew-t distribution.
`r`	This function is currently very specific to the 16x16 case. r specifies which of the 8 distributions you wish to see, and then extracts the (r, r+8) parameters from that distribution.
`alphaAdj`	If TRUE, computes the correct alpha for the marginal (per Azzalini et.al's "Distributions generated by perturbations of symmetry", page 18 and Capitano et.al's "Graphical models for skew-normal variates", page 15). If false, just subsets alpha.

Value

A list of the 4 parameters of the bivariate skew-t.

`mst.pdev.grad.robust` *Gradient of robust, penalized deviance for multivariate skew-t*

Description

This function computes the gradient of the robust-ified, penalized deviance for the multivariate skew-t (with respect to the optimization parameters).

Usage

```
mst.pdev.grad.robust(param, x, y, k = 2, ...)
```

Arguments

`param`	Optimization parameters derived from xi, Omega, alpha, nu. For conversion, see the functions optpar2dplist and dplist2optpar.
`x`	A matrix of the independent variables for the fit. Typically just a matrix of ones.
`y`	A matrix of dependent variables.
`k`	Parameter controlling the robustness of the fit. The largest possible value for the negative log-likelihood is 2*k, and the negative log-likelihood is adjusted down whenever it is larger than k.

Value

The gradient of the robust deviance with respect to dp.

`mst.pdev.robust` *Computes the robust-ified, penalized deviance for the multivariate skew-t.*

Description

Computes the robust-ified, penalized deviance for the multivariate skew-t.

Usage

```
mst.pdev.robust(param, x, y, k = 2, ...)
```

Arguments

`param`	Optimization parameters derived from xi, Omega, alpha, nu. For conversion, see the functions optpar2dplist and dplist2optpar.
`x`	A matrix of the independent variables for the fit. Typically just a matrix of ones.
`y`	A matrix of dependent variables.
`k`	A parameter controlling the robustness of the fit. The largest possible value for the negative log-likelihood is 2*k, and the negative log-likelihood is adjusted down whenever it is larger than k.

Value

The "robust" skew-t deviance evaluated at observations (x,y) and with parameters dp.

`n.dev` *Normal deviance*

Description

Normal deviance

Usage

```
n.dev(dp, y)
```

Arguments

`dp`	The "density parameters". More specifically, just a vector of length two containing the mean and standard deviation parameters.
`y`	A vector of observed values.

Value

The sum of the negative log-likelihoods of a normal with parameters dp evaluated at the values y.

`n.dev.gh` *Gradient of Normal Deviance*

Description

This function computes the derivative of the normal deviance (where the deviance is computed at observations y and with parameters dp).

Usage

```
n.dev.gh(dp, y)
```

Arguments

`dp`	The density parameters, in this case a vector of the mean and sd.
`y`	A vector of observations.

Value

A vector of length 2 giving the derivative of the deviance with respect to the mean and standard deviation.

`n.dev.gh.robust` *Gradient of Normal Deviance, Adjusted for Robustness*

Description

This function is the analog of n.dev.gh, but is applied with a robust adjustment. It takes advantage of the fact that the gradient of the robust deviance for all observations y is the sum of the gradients of the deviances for each individual observation.

Usage

```
n.dev.gh.robust(dp, y, k = 2)
```

Arguments

`dp`	The density parameters, in this case a vector of the mean and sd.
`y`	A vector of observations.
`k`	The robust adjustment parameter.

Value

A vector of length 2 giving the derivative of the (robust) deviance with respect to the mean and standard deviation.

`n.dev.robust` *Robust Normal deviance*

Description

This function computes the deviance for a set of observations (y), a provided mean and standard deviation

Usage

```
n.dev.robust(dp, y, k = 2)
```

Arguments

`dp`	The "density parameters". More specifically, just a vector of length two containing the mean and standard deviation parameters.
`y`	The observed value.
`k`	The parameter controlling the robustness adjustment. See ?Psi.

Value

The sum of the negative log-likelihoods of a normal with parameters dp evaluated at the values y. However, the individual negative log-likelihood values are adjusted by Psi.

`plot_results` *Plot results*

Description

This function takes the results object as created by runSim and generates plots summarizing the simulation. These plots are saved in the users current working directory.

Usage

```
plot_results(results, prefix, xi = 0, omega = 1, alpha = 0, nu = 10000)
```

Arguments

`results`	A results object, as returned by runSim.
`prefix`	A prefix to be added to the saved plots.
`xi`	A parameter of the skew-t distribution. This should be the parameter used in the simulation.
`omega`	A parameter of the skew-t distribution. This should be the parameter used in the simulation.
`alpha`	A parameter of the skew-t distribution. This should be the parameter used in the simulation.
`nu`	A parameter of the skew-t distribution. This should be the parameter used in the simulation.

Value

No results are returned, but plots are saved in the user's current directory.

`Psi` *Robust adjustment of the negative log-likelihood*

Description

Function implementing the robustness adjustment to the negative log-likelihood.

Usage

```
Psi(originalNLL, k)
```

Arguments

`originalNLL`	The original negative log-likelihood
`k`	The parameter controlling the amount of robustification. Negative log-likelihoods larger than k are reduced, and the adjusted value is always less than 2*k.

Details

Essentially, if the negative log-likelihood value is too large, it is bounded via Psi(). The psi() function bounds the derivative.

Value

The adjusted negative log-likelihood.

`psi.grad` *Robust adjustment of the derivative of the negative log-likelihood*

Description

This function implements an adjustment to the derivative of the density function to make estimation of parameters more robust. If a negative log-likelihood is supplied, then psi returns an adjusted negative log-likelihood. Essentially, if the negative log-likelihood value is too large, it is bounded via this function.

Usage

```
psi.grad(originalNLL, k)
```

Arguments

`originalNLL`	The original negative log-likelihood
`k`	A parameter controlling the amount of robustification. Negative log-likelihoods larger than k are reduced, and the adjusted value is always less than 2*k.

Details

Usually, we're interested in df/dp, where f is the negative log-likelihood and p a parameter. df/dp should be updated with df/dp*psi(f, p) to be "robustified".

Value

Returns

`robustST` *Fit a robust Skew-t*

Description

Fits a robust version of the multivariate skew-t, done by bounding the negative log-likelihood for each observation.

Usage

```
robustST(y, x = matrix(1, nrow = NROW(y)), robust = T,
  method = c("nlminb", "constrOptim"), w = rep(1, nrow(x)), k = 10,
  start = NULL)
```

Arguments

`y`	A vector or matrix of observations to fit the skew-t to.
`x`	A matrix of ones, or matrix of independent variables for skew-t regression (use caution, as this feature has not been tested!)
`robust`	Should the robust estimator be used?
`w`	A vector of case weights, defaults to a vector of ones.
`k`	A parameter controlling the "robustness" of the fit. The maximum value for the negative log-likelihood for any observation is 2*k. Thus, as k->Inf the estimator approaches the MLE. k values around 8 or 10 seem to perform well.
`start`	The starting values for the optimization. If NULL, reasonable values are automatically chosen.
`method:`	constrOptim uses a constrained algorithm, forcing nu and omega>0. However, the implementation for multivariate skew-t fitting enforces this by default, so nlminb and constrOptim should be very similar for multivariate data. For univariate, constrOptim is recommended. For multivariate, constrOptim is also recommended as it appears to be faster.

Value

A named list containing the results of the fit. beta vector is equivalent to the mean estimate if x = matrix of 1's, and omega/alpha/nu are the parameters of the skew-t. A convergence flag is also returned, indicating if the solution is a true optimum.

robustSTChangingK *Fit a robust Skew-t, iteratively update k*

Description

Fits a robust version of the multivariate skew-t, done by bounding the negative log-likelihood for each observation. Re-compute the value of k each time new parameter estimates are available.

Usage

```
robustSTChangingK(y, x = matrix(1, nrow = NROW(y)), robust = T,
  method = c("nlminb", "constrOptim"), w = rep(1, nrow(x)), pValue = 0.01,
  start = NULL)
```

Arguments

y	A vector or matrix of observations to fit the skew-t to.
x	A matrix of ones, or matrix of independent variables for skew-t regression (use caution, as this feature has not been tested!)
robust	Should the robust estimator be used?
w	A vector of case weights, defaults to a vector of ones.
pValue	A parameter controlling the "robustness" of the fit. Given current parameter estimates, a (1-pValue) constructed, and observations in this region will not be adjusted during the optimization. However, values outside this region will have their likelihood adjusted down, and hence will have less influence on the M-estimator. As pValue->1 the estimator approaches the MLE.
start	The starting values for the optimization. If NULL, reasonable values are automatically chosen.
method:	constrOptim uses a constrained algorithm, forcing nu and omega>0. However, the implementation for multivariate skew-t fitting enforces this by default, so nlminb and constrOptim should be very similar for multivariate data. For univariate, constrOptim is recommended. For multivariate, constrOptim is also recommended as it appears to be faster.

Value

A named list containing the results of the fit. beta vector is equivalent to the mean estimate if x = matrix of 1's, and omega/alpha/nu are the parameters of the skew-t. A convergence flag is also returned, indicating if the solution is a true optimum.

`robustSTOnceK` *Fit a robust Skew-t, estimate k once*

Description

Fits a robust version of the multivariate skew-t, done by bounding the negative log-likelihood for each observation. Compute the value of k once based on the initial (non-robust) estimates of the parameters.

Usage

```
robustSTOnceK(y, x = matrix(1, nrow = NROW(y)), robust = T,
  method = c("nlminb", "constrOptim"), w = rep(1, nrow(x)), pValue = 0.01,
  start = NULL)
```

Arguments

`y`	A vector or matrix of observations to fit the skew-t to.
`x`	A matrix of ones, or matrix of independent variables for skew-t regression (use caution, as this feature has not been tested!)
`robust`	Should the robust estimator be used?
`w`	A vector of case weights, defaults to a vector of ones.
`pValue`	A parameter controlling the "robustness" of the fit. Given current parameter estimates, a (1-pValue) constructed, and observations in this region will not be adjusted during the optimization. However, values outside this region will have their likelihood adjusted down, and hence will have less influence on the M-estimator. As pValue->1 the estimator approaches the MLE.
`start`	The starting values for the optimization. If NULL, reasonable values are automatically chosen.
`method:`	constrOptim uses a constrained algorithm, forcing nu and omega>0. However, the implementation for multivariate skew-t fitting enforces this by default, so nlminb and constrOptim should be very similar for multivariate data. For univariate, constrOptim is recommended. For multivariate, constrOptim is also recommended as it appears to be faster.

Value

A named list containing the results of the fit. beta vector is equivalent to the mean estimate if x = matrix of 1's, and omega/alpha/nu are the parameters of the skew-t. A convergence flag is also returned, indicating if the solution is a true optimum.

runSim *Run simulation*

Description

This function runs a simulation to measure the performance of the robust and non-robust fitting methods on data where the distribution is known.

Usage

```
runSim(n = 100, outPct = 0, outSigma = 0, k = 6:10, fast.k = c(0.99,
  0.98, 0.95, 0.9), xi0 = runif(1, min = -10, max = 10), Omega0 = diag(x =
  10^runif(1, -1, 1), nrow = 1), alpha0 = runif(1, -4, 4), nu0 = 10^runif(1,
  log(4)/log(10), 4), pressure = NULL, type = NULL, restrict = FALSE)
```

Arguments

`n`	Sample size to simulate.
`outPct`	Percent of outliers. Actual number is round(n*outPct)
`k`	A numeric vector of constants to use in "capping" the likelihood function. Each constant is tried in turn, and results are returned for all values.
`fast.k`	A vector of k values for the fastTLE algorithm. Each value should represent the proportion of data to be used in estimating the MLE.
`xi0`	Simulated center parameter of the skew-t to simulate.
`alpha0`	Simulated skewness parameter of the skew-t to simulate.
`nu0`	Simulated heaviness of tails parameter of the skew-t to simulate.
`pressure`	Use Denver station data at this pressure level, and simulate the skew-t using Ying's code
`type`	The type of skewness to use. Must be "MVN", "obs", or "EX".
`restrict`	Should outliers be generated in the tail of the skew-t only? Only applies if pressure and type are not null.
`outSigma:`	Outliers are created by adding on an error of N(0,outSigma^2)
`omega0`	Simulated scale parameter of the skew-t to simulate.

Value

A data.frame containing the results of the simulations.

st.pdev.gh.robust — *Gradient of the robust-ified, penalized deviance for the univariate skew-t.*

Description

Gradient of the robust-ified, penalized deviance for the univariate skew-t.

Usage

```
st.pdev.gh.robust(dp, x, y, k = 2)
```

Arguments

dp	"Density parameters"? Vector of xi, omega, alpha, nu.
x	A matrix of the independent variables for the fit. Typically just a matrix of ones.
y	A vector of dependent variables
k	Parameter controlling the robustness of the fit. The largest possible value for the negative log-likelihood is 2*k, and the negative log-likelihood is adjusted down whenever it is larger than k.

Value

The gradient of the robust deviance with respect to dp.

st.pdev.robust — *Computes the robust-ified, penalized deviance for the univariate skew-t.*

Description

Computes the robust-ified, penalized deviance for the univariate skew-t.

Usage

```
st.pdev.robust(dp, x, y, k = 2, ...)
```

Arguments

dp	"Density parameters"? Vector of xi, omega, alpha, nu.
x	matrix of the independent variables for the fit. Typically just a matrix of ones, but can theoretically be a design matrix if the end goal is fitting a regression with skew-t errors.
y	A vector of dependent variables.
k	Parameter controlling the robustness of the fit. The largest possible value for the negative log-likelihood is 2*k, and the negative log-likelihood is adjusted down whenever it is larger than k.

Value

The "robust" skew-t deviance evaluated at observations (x,y) and with parameters dp.

Index