# Package: ecdfHT (via r-universe)

October 31, 2024

**Type** Package

**Title** Empirical CDF for Heavy Tailed Data

**Version** 0.1.1

**Date** 2016-08-31

**Author** John P Nolan <jpnolan@american.edu>

**Maintainer** John P Nolan <jpnolan@american.edu>

**Description** Computes and plots a transformed empirical CDF (ecdf) as a diagnostic for heavy tailed data, specifically data with power law decay on the tails. Routines for annotating the plot, comparing data to a model, fitting a nonparametric model, and some multivariate extensions are given.

**Depends** R (>= 3.3.0)

**Imports** rgl

**License** GPL (>= 3)

**LazyData** TRUE

**RoxygenNote** 5.0.1

**NeedsCompilation** no

**Date/Publication** 2016-09-09 18:03:25

**Repository** https://jpnolan.r-universe.dev

**RemoteUrl** https://github.com/cran/ecdfHT

**RemoteRef** HEAD

**RemoteSha** b614bcde04a03a3bb1cceef09d2bbb4ff37ee27a

## Contents

---

`ecdfHT-package` *ecdfHT: A package to plot an empirical cdf for heavy tailed data*

---

## Description

The `ecdfHT` package computes and plot a transformed empirical cdf for data. This is useful because a standard empirical cdf (ecdf) gives little information about the tails of the data when there are extreme values.

## Details

The transform is nonparametric: linear in the middle of the data and matched to a log-log transform on the tails, where the tail regions are determined by quantiles. If the data has power law behavior on the tails, the plot is linear on those tails, so this plot can be used as a graphical diagnostic to determine if a data set is heavy tailed.

In addition, there are functions to

- annotate the plot, add custom axes and grid lines
- overlay proposed models on the plot
- fit the tails using linear regression on the transformed tails
- combine the empirical cdf in the middle and the above fit on the tails to get a semi-parametric fit to the data
- compute cdf, pdf, quantiles, and simulate from the semi-parametric fit
- some multivariate plots that look at tail behavior of multiple components and some idea of the dependence.

I will try to fix the code if you provide a simple demonstration of a bug. Polite suggestions for improvements will be considered if there is time available.

## See Also

`ecdfHT` for a basic plot, `ecdfHT.draw` for annotations and additions to a basic plot, `ecdfHT.fit` to fit a semi-parametric model to the data, `pecdfHT` to compute the cdf, pdf, quantiles and simulate from a semi-parametric model, `ecdfHT.multivar` for multivariate generalizations.

ecdfHT *Plot a transformed empirical cdf*

## Description

Produces a basic plot showing a transformed empirical cdf for heavy tailed data. It uses a log-log transform on the tails, which shows power law decay as linear.

## Usage

```
ecdfHT(x, scale.q = c(0.25, 0.5, 0.75), show.axes.labels = TRUE,
  show.plot = TRUE, type = "p", ...)
```

## Arguments

| | |
|---|---|
| x | A vector of data |
| scale.q | A vector of 3 probabilites; specifies the quantiles of the data to use for the left tail, mid region, and right tail |
| show.axes.labels | Boolean value: indicates whether default labels are plotted or not. (Use function ecdfHT.axes to add custom labels) |
| show.plot | Boolean value: show plot or only do calculations |
| type | Type of plot, passed to plot. Use type='p' for points, type='l' for lines |
| ... | Optional graphical parameters, e.g. col='red' |

## Details

Most of the work is done by ecdfHT.draw and the associated helper functions.

Assuming no repeats in x, ecdf = (standard ecdf - (1/2))/n, like type=5 in the R function quantile. So instead of taking values 1/n, 2/n, 3/n, ... , k/n, ..., 1 it takes values 1/(2n), 3/(2n), ..., (2k-1)/(2n), ..., (2n-1)/(2n). This avoids 0 at lower endpoint and 1 at upper endpoint, which causes problems when we extend tails with a power law. (If there are m repeated x values, then the corresponding jump in the ecdf at that point is m/n instead of 1/n.)

The default values scale.q=c(.25,.5,.75) splits the data into quartiles; picking different quantiles splits the data into 4 different groups: the lowest group is the left tail, i.e. all values less than the quantile corresponding to scale.q[1]; the next group is between the left tail and center = quantile scale.q[2]); the third group is the center and quantile scale.q[3]; the last group is the upper tail. For two-sided data, it makes sense to use something like (p,0.5,1-p) for scale.q, where p is choosen to determine where the tail regions begin.

For one-sided data, it makes sense to use scale.q=c(0,0,p). In this case, the first two groups are empty and the effect is to divide the data into two groups: a moderate/lower range and a right tail. See the example below with nonnegative data.

The transformations h(x) acts on these different regions. It is linear on the middle two regions and logarithmic on the tails. The transformation g(p) acts on the corresponding values of the ecdf

described above. The basic plot shows (h(x[i]),g(ecdf[i])): the first component is a monotonic transform of the x values, the second component is a monotonic transform of the ecdf. See the accompanying vignette for exact definitions: go to the package index and click on User guides, package vignettes and other documentation.

## Value

An object of class 'ecdfHT.transform' which gives the information necessary to draw the plot and later add other curves and labels. This list is returned invisibly and contains the following fields:

**scale.q** vector of length 3, copied from the input argument

**scale.x** vector of length 3, the quantiles from the data corresponding to scale.q

**xsort** vector of the sorted, unique data values

**ecdf** nonstandard empirical cdf, see details

**xx** transformed x values: xx[i]=h(xsort[i])

**yy** transformed ecdf values: yy[i]=g(ecdf[i])

## See Also

ecdfHT.draw for annotations and additions to a basic plot

## Examples

```
x <- rcauchy( 1000 )
ecdfHT( x )
title("basic ecdfHT plot")

xabs <- abs(x)
ecdfHT( xabs, scale.q=c(0,0,.75) )
title("one sided data")
```

---

# ecdfHT.draw *Graph and annotate an ecdfHT plot*

---

## Description

Does the computations and plotting for ecdfHT and can be used to add to an existing plot.

## Usage

```
ecdfHT.draw(transform.info, x, p, show.plot = TRUE, new.plot = FALSE,
  show.ci = FALSE, xlab = "x", ylab = "", ...)

ecdfHT.axes(transform.info, x.labels = c(), y.labels = c(),
  show.vert.gridlines = FALSE, show.horiz.gridlines = FALSE, ...)
```

```
ecdfHT.h(x, t)

ecdfHT.g(p, q)
```

## Arguments

| | |
|---|---|
| transform.info | A list with information about the transformation, computed in ecdfHT |
| x | The data, a vector of double precision numbers. Assumbed to be sorted and have distinct values. |
| p | Probabilities, a vector of doubles. Typically p[i]=(i=0.5)/length(x), unless there are repeats in x. |
| show.plot | Boolean value: indicates whether to plot or not. |
| new.plot | Boolean value: indicates whether to produce a new plot or add to an existing plot. |
| show.ci | Boolean value: indicates whether or not confidence intervals are shown. |
| xlab | String to label the horizontal axis. |
| ylab | String to label the vertical axis. |
| ... | Optional parameters for the plot, e.g. col='red'. |
| x.labels | Vector of numbers specifying the location of the labels on the horizontal axis |
| y.labels | Vector of numbers specifying the location of the labels on the vertical axis |
| show.vert.gridlines | Boolean value indicating whether or not vertical grid lines should be drawn. |
| show.horiz.gridlines | Boolean value indicating whether or not horizontal grid lines should be drawn. |
| t | A vector of length 3 that specifies the x values that determine the left tail, middle, and right tail |
| q | A vector of length 3 that specifies the quantile values that determine the left tail, middle, and right tail. |

## Details

ecdfHT.draw computes transform and plots. ecdfHT.axes draws axes on the plot; it can be used to manually select tick marks, etc. ecdfHT.h computes the function h(x) for the transformation of the horizontal axis. ecdfHT.g computes the function g(p) for the transformation of the vertical axis.

Always call ecdfHT first to produce the basic plot, then use ecdfHT.draw to add other curves to the plot as in the examples below

## Value

A list of values used in the plot, see return value of ecdfHT.

ecdfHT.h returns the vector y=h(x;t), ecdfHT.g returns the vector y=g(p;q)

## Examples

```
set.seed(1)
x <- rcauchy( 1000 )
t.info <- ecdfHT( x, show.axes=FALSE )
ecdfHT.axes( t.info, x.labels=c(-50,-5,0,5,50), y.labels=c(.001,.01,.1,.5,.9,.99,.999),
  show.vert.gridlines=TRUE, show.horiz.gridline=TRUE, lty=2 )
q1 <- qcauchy(t.info$ecdf) # Cauchy quantiles
ecdfHT.draw( t.info, q1, t.info$ecdf, col='red',show.ci=TRUE)
q2 <- qnorm(t.info$ecdf,sd=sd(x))  # Gaussian quantiles
ecdfHT.draw( t.info, q2, t.info$ecdf, col='green',show.ci=TRUE)
title(paste("simulated Cauchy data, n=",length(x),"\nred=Cauchy cdf, green=normal cdf"))

x <- seq(-5,5,1)
t <- c(-3,0,3)
ecdfHT.h(x,t)
p <- seq(0.05,.95,.1)
q <- c(.1,.5,.9)
ecdfHT.g(p,q)
```

---

`ecdfHT.fit` *Fit heavy tailed data with a semi-parameteric model*

---

## Description

Compute an interpolation of the transformed cdf in the middle with parametric power law decay on the tails.

## Usage

```
ecdfHT.fit(p, transform.info, x.min = NA, x.max = NA, add.to.plot = TRUE,
  weights = "var", ...)

ecdfHT.fit.tails(p, transform.info, weights, add.to.plot = TRUE, ...)
```

## Arguments

| | |
|---|---|
| `p` | Vector of 2 probabilities that identify the quantile where data is cut to fit power decay on lower/upper tail. Set tail.p[1]=0 to exclude lower tail fit; tail.p[2]=1 to exclude upper tail fit. |
| `transform.info` | List containing transformation information, returned from `ecdfHT` |
| `x.min` | Number describing cut-off of lower tail |
| `x.max` | Number describing cut-off of upper tail |
| `add.to.plot` | Boolean indicating whether or not the interpolation is plotted |
| `weights` | 'none' to do unweighted regression or 'var' to use weighted regression on tail with weights proportional to variance of quantile |
| `...` | Optional parameters passed to plot routines, e.g. col='red' |

## Value

An object of class 'ecdfHT.fit' specifying the interpolation. The fields in the value are:

**scale.q** vector of length 3, copied from the input argument

**scale.x** vector of length 3, the quantiles from the data corresponding to scale.q

**xsort** vector of the sorted, unique data values

**ecdf** nonstandard empirical cdf, see details

**xx** transformed x values: xx[i]=h(xsort[i])

**yy** transformed p values: yy[i]=g(ecdf[i])

**cdf.spline** monotonic spline function used to compute the cdf

**inf.cdf.spline** monotonic spline function used to compute the inverse of the cdf

**tail.p** vector of length 2; probabilities saying where the lower and upper tails begin. Note these are generally not the exact values of input variable p, rather they are the closest values to those found in ecdf

**tail.x** vector of length 2; x values where the lower and upper tails begin

**tail.c** vector of length 2; tail constants for lower and upper powerlaw fit

**tails.slope** vector of length 2; slope of tails on transformed plot

**tail.alpha** vector of length 2; exponents for lower and upper power law fit

**tail.m** integer vector of length 2; indices in xsort where tails begin

**weights** copy of input variable weights

## Examples

```
x <- rcauchy( 1000 )
a <- ecdfHT( x )
fit <- ecdfHT.fit( c(.1,.9), a, col='red' )
str(fit)
```

---

# ecdfHT.multivar *Multivariate extensions of transformed empirical cdf plot*

---

## Description

Transform multivariate data and plot using the ideas from the univariate plot.

## Usage

```
ecdfHT.multivar(x, scale.q = matrix(c(0.25, 0.5, 0.75), nrow = 3, ncol =
  ncol(x)), q0 = 0.5, radii.upper.tail.p = 0.9, p.norm = 2,
  show.axes.labels = FALSE, zscale = c(500, 1), ...)

ecdfHT.multivar.transform(x, scale.q, q0, p.norm)

ecdfHT.2d(multivar.obj, zscale = c(500, 1), ...)

ecdfHT.2d.axes(zscale)

lp.norm(x, p.norm)
```

## Arguments

| | |
|---|---|
| x | Matrix of data of size (n x d) |
| scale.q | matrix of sixe (3 x d), probabilities used to determine the scaling and centering for each component |
| q0 | quantile of radii transformation |
| radii.upper.tail.p | probability used as cutuoff to tail fit; set to 1 to suppress upper tail fit |
| p.norm | Power used in computing L^p norm |
| show.axes.labels | Boolean value, determines if axes are labeled or not |
| zscale | Vector of length 2, value of aspect ratio for the z axis when d=2 and the two 3d plots are drawn |
| ... | Optional graphical parameters, e.g. col='red' |
| multivar.obj | An object of class 'ecdfHT.multivar', see details. |

## Details

ecdfHT.multivar gives a quick graphical look at a d dimensional data set. It produces two plots: the first is a superposition of the univariate ecdfHT plots for each component; the second plot is an array of plots, showing one plot for each component.

ecdfHT.bivar produces two plots of a bivariate data set. The first one has three subplots: a scatter plot of the data, a transformed scatterplot of the data, and a univariate ecdfHT plot of the radii of the data. For the second and third subplot, ???? then g(y[,1]) is plotted against g(y[,2]) to get the second plot. For the third plot, compute radius r[i]= l_p norm of shifted and scaled data. These radii are plotted in a univariate, one-sided ecdfHT plot.

The second plot produced is a 3-dimensonal plot. It takes the first two subplots just described and adds a third dimension by looking at an ecdf for the radii. Thus the height of a point is low if the point is near the center, and increases as points move away. The first subplot shows points (x[i,1],x[i,2],ecdf of r[i]). The second plot transforms all three components: it shows (h1(x[i,1]),hs(x[i,2]),g(ecdf of r[i]), where h1(.) and h2(.) are scaled versions of h(.) from the univariate ecdfHT plot, and g(.) is as in the univariate plot. See the vignette for more detail.

## Value

ecdfHT.multivar draws several plots, returns a list (invisibly) with fields:

**x** input (n x d) matrix of data

**x.prime** (n x d) matrix of centered and shifted version of x

**y** (n x d) matrix of transformed x

**p.norm** what p-norm to use; p.norm=2 is Euclidean norm

**scale.q** copy of input argument

**radii** vector of length n, p-norm of the rows of x.prime

**q0** copy of input value

**r0** q0-th quantile of the radii

**univariate.ecdfHT** list of length d, with j-th entry the object returned by ecdfHT for the j-th column of x

**radii.ecdfHT** list returned from ecdfHT( radii, ... )

**radii.tail.fit** object returned from ecdfHT.fit for the radii )

**rgl.id** rgl id of 3d plot(s); can be used to access, change, print 3d plots

**radii.prob** if d=2, this vector gives the empirical cdf of the radii

**radii.prob2** if d=2, this vector gives the transformed empirical cdf of the radii

ecdfHT.multivar.transform computes the transformed vectors y, radii, and lp.norm computes the lp-norm of the rows of x

## Examples

```
# independent components
set.seed(2)
x <- matrix( rcauchy(4000), ncol=4 )
ecdfHT.multivar( x )

# radially symmetric
r <- rcauchy(1000)
theta <- runif(1000,min=0,max=2*pi)
x <- cbind( r*cos(theta), r*sin(theta) )
ecdfHT.multivar( x )
```

---

# pecdfHT *Compute cdf, pdf, quantiles, and simulate from a fitted distribution*

---

## Description

Use the semi-parametric fit calculated by ecdfHT.fit to evaluate the cdf F(x), pdf f(x), quantiles and simulate.

## Usage

```
pecdfHT(x, ecdfHT.fit)

decdfHT(x, ecdfHT.fit)

qecdfHT(p, ecdfHT.fit)

recdfHT(n, ecdfHT.fit)
```

## Arguments

| | |
|---|---|
| x | A vector of numbers |
| ecdfHT.fit | An object returned by ecdfHT.fit describing the interpolation. |
| p | Vector of probabilites |
| n | Number of values to simulate |

## Details

pecdfHT computes the cdf of the semi-parametric fit to the data. decdfHT computes the pdf of the semi-parametric fit to the data. This is likely very irregular and not of much value except on the tails, where the pdf calculation is computed analytically. qecdfHT computes quantiles. recdfHT simulates from a semi-parameteric distribution.

## Value

pecdfHT computes the cdf, decdfHT computes the pdf, qecdfHT computes the quantiles (inverse of the cdf), recdfHT simulates from the distribution.

## Examples

```
x <- rcauchy(1000)
a <- ecdfHT( x, show.plot=FALSE )
fit <- ecdfHT.fit( c(.1,.9), a, add.to.plot=FALSE )
pecdfHT( -3:3, fit )
decdfHT( -3:3, fit )
qecdfHT( seq(.1,.9,.1), fit )
recdfHT( 10, fit )
```

# Index