

# The extRemes Package

January 10, 2007

**Version** 1.54

**Date** March 15 2004

**Title** Extreme value toolkit.

**Author** Eric Gilleland, Rick Katz, Greg Young

**Maintainer** Eric Gilleland <ericg@ucar.edu>

**Depends** R (>= 1.7.0),tcltk,ismev

**Description** Uses Stuart Coles' (Coles, Stewart, "An introduction to statistical modeling of extreme values", Springer-Verlag, London 2001) S-plus functions as ported to the R programming language (ismev) by Alec Stephenson (<http://www.maths.lancs.ac.uk/~stephena/>). This toolkit provides a Graphical User Interface (GUI) to ismev for pedagogical purposes.

**License** GPL Version 2 or later.

**URL** <http://www.assessment.ucar.edu/toolkit/>

## R topics documented:

Denmint . . . . .	2
Denversp . . . . .	3
Flood . . . . .	4
FitCoPrec . . . . .	5
HEAT . . . . .	6
Ozone4H . . . . .	7
PORTw . . . . .	8
Peak . . . . .	9
Potomac . . . . .	11
Rsum . . . . .	12
Tphap . . . . .	13
as.extRemesDataObject . . . . .	14
bisearch . . . . .	15
boot.matrix . . . . .	16
boot.sequence . . . . .	17

clearlog . . . . .	19
damage . . . . .	20
dclust . . . . .	21
decluster.runs . . . . .	22
eiAnalyze . . . . .	24
exi.intervals . . . . .	25
extRemes internal . . . . .	26
extRemes . . . . .	28
extremalindex . . . . .	30
extremes.gui . . . . .	31
fpp . . . . .	32
ftcanmax . . . . .	34
gen.gev . . . . .	35
gev.parameterCI . . . . .	37
return.level . . . . .	38
rlplot . . . . .	40
<b>Index</b>	<b>42</b>

---

 Denmint

*Denver Minimum Temperature*


---

## Description

Daily minimum temperature (degrees centigrade) for Denver, Colorado from 1949 through 1999.

## Usage

```
data(Denmint)
```

## Format

A data frame with 18564 observations on the following 5 variables.

**Time** a numeric vector indicating the line number (time from first entry to the last).

**Year** a numeric vector giving the year.

**Mon** a numeric vector giving the month of each year.

**Day** a numeric vector giving the day of the month.

**Min** a numeric vector giving the minimum temperature in degrees Fahrenheit.

## Source

Colorado Climate Center, Colorado State University (<http://ulysses.atmos.colostate.edu>).

**Examples**

```
data(Denmint)
plot(Denmint[,3], Denmint[,5], xlab="", xaxt="n", ylab="Minimum Temperature (deg. F)")
axis(1, at=1:12, labels=c("Jan", "Feb", "Mar", "Apr", "May", "Jun", "Jul", "Aug", "Sep", "Oct", "Nov", "D
```

---

Denversp

*Denver July hourly precipitation amount.*


---

**Description**

Hourly precipitation (mm) for Denver, Colorado in the month of July from 1949 to 1990.

**Usage**

```
data(Denver)
```

**Format**

A data frame with 31247 observations on the following 4 variables.

**Year** a numeric vector giving the number of years from 1900.

**Day** a numeric vector giving the day of the month.

**Hour** a numeric vector giving the hour of the day (1 to 24).

**Prec** a numeric vector giving the precipitation amount (mm).

**Details**

These observations are part of an hourly precipitation dataset for the United States that has been critically assessed by Collander et al. (1993). The Denver hourly precipitation dataset is examined further by Katz and Parlange (1995). Summer precipitation in this region near the eastern edge of the Rocky Mountains is predominantly of local convective origin (Katz and Parlange (1005)).

**Source**

Katz, Richard W. and Parlange, Marc B., Generalizations of chain-dependent processes: Application to hourly precipitation, *Water Resources Research* Vol. 31 No. 5: 1331–1341, May 1995.

**References**

Collander, R.S., Tollerud, E.I., Li, L., and Viront-Lazar, A., Hourly precipitation data and station histories: A research assessment, in *Preprints, Eighth Symposium on Meteorological Observations and Instrumentation*, pp. 153–158, American Meteorological Society, Boston, 1993.

**Examples**

```
data(Denversp)
plot(Denversp[,1], Denversp[,4], xlab="", ylab="Hourly precipitation (mm)", xaxt="n")
axis(1, at=c(50, 60, 70, 80, 90), labels=c("1950", "1960", "1970", "1980", "1990"))
```

---

Flood

*United States total economic damage resulting from floods.*

---

### Description

United States total economic damage (in billions of U.S. dollars) caused by floods by hydrologic year from 1932-1997. See Pielke and Downton (2000) for more information.

### Usage

data(Flood)

### Format

A data frame with 66 observations on the following 5 variables.

**OBS** a numeric vector giving the line number.

**HYEAR** a numeric vector giving the hydrologic year.

**USDMG** a numeric vector giving total economic damage (in billions of U.S. dollars) caused by floods.

**DMGPC** a numeric vector giving damage per capita.

**LOSSPW** a numeric vector giving damage per unit wealth.

### Details

From Pielke and Downton (2000):

The National Weather Service (NWS) maintains a national flood damage record from 1903 to the present, and state level data from 1983 to the present. The reported losses are for "significant flood events" and include only direct economic damage that results from flooding caused by rainfall and/or snowmelt. The annual losses are based on "hydrologic years" from October through September. Flood damage per capita is computed by dividing the inflation-adjusted losses for each hydrological year by the estimated population on 1 July of that year ([www.census.gov](http://www.census.gov)). Flood damage per million dollars of national wealth uses the net stock of fixed reproducible tangible wealth in millions of current dollars (see Pielke and Downton (2000) for more details).

### Source

NWS web site: <http://www.nws.noaa.gov/oh/hic>

### References

Gilleland, Eric and Katz, Richard W. Tutorial for the 'Extremes Toolkit: Weather and Climate Applications of Extreme Value Statistics.' <http://www.assessment.ucar.edu/toolkit>, 2005.

Katz, Richard W., Parlange, Marc B. and Naveau, Philippe, Statistics of extremes in hydrology, *Advances in Water Resources*, 25:1287–1304, 2002.

Pielke, Roger A. Jr. and Downton, Mary W., Precipitation and damaging floods: trends in the United States, 1932-97, *Journal of Climate*, 13 (20):3625–3637, 2000.

### Examples

```
data(Flood)
plot(Flood[,2], Flood[,3], type="l", lwd=2, xlab="hydrologic year", ylab="Total economic da
```

---

FtCoPrec

*Daily precipitation amounts in Fort Collins, Colorado.*

---

### Description

Daily precipitation amounts (inches) from a single rain gauge in Fort Collins, Colorado. See Katz et al. (2002) Sec. 2.3.1 for more information and analyses.

### Usage

```
data(FtCoPrec)
```

### Format

A matrix with dimension 36524 by 5. Columns are: "obs", "month", "day", "year" and "Prec"; where "Prec" is the daily precipitation amount (inches).

### Source

Colorado Climate Center, Colorado State University (<http://ulysses.atmos.colostate.edu>).

### References

Gilleland, Eric and Katz, Richard W. Tutorial for the 'Extremes Toolkit: Weather and Climate Applications of Extreme Value Statistics.' <http://www.assessment.ucar.edu/toolkit>, 2005.

Katz, Richard W., Parlange, Marc B. and Naveau, Philippe. Statistics of extremes in hydrology. *Advances in Water Resources*, 25:1287–1304, 2002.

### Examples

```
data(FtCoPrec)
str(FtCoPrec)
plot(FtCoPrec[,"month"], FtCoPrec[,"Prec"], xlab="month", ylab="daily precipitation (inches)

# See Gilleland et al. (2005) for more examples using these dta with extRemes.
```

---

HEAT

*Summer maximum and minimum temperature in Phoenix, Arizona.*

---

### Description

Summer maximum and minimum temperature (degrees Fahrenheit) for July through August 1948 through 1990 at Sky Harbor airport in Phoenix, Arizona.

### Usage

```
data(HEAT)
```

### Format

A data frame with 43 observations on the following 3 variables.

**Year** a numeric vector giving the number of years since 1900.

**Tmax** a numeric vector giving the Summer maximum temperatures in degrees Fahrenheit.

**Tmin** a numeric vector giving the Summer minimum temperatures in degrees Fahrenheit.

### Details

Data is Summer maximum and minimum temperature for the months of July through August from 1948 through 1990.

### Source

U.S. National Weather Service Forecast office at the Phoenix Sky Harbor Airport.

### References

Balling, R.C., Jr., Skindlov, J.A. and Phillips, D.H., The impact of increasing summer mean temperatures on extreme maximum and minimum temperatures in Phoenix, Arizona. *Journal of Climate*, 3:1491–1494, 1990.

Gilleland, Eric and Katz, Richard W. Tutorial for the 'Extremes Toolkit: Weather and Climate Applications of Extreme Value Statistics.' <http://www.assessment.ucar.edu/toolkit>, 2005.

Tarleton, Lesley F. and Katz, Richard W., Statistical explanation for trends in extreme summer temperatures at Phoenix, A.Z., *Journal of Climate*, 8 (6):1704–1708, 1995.

### Examples

```
data(HEAT)
str(HEAT)
plot(HEAT)
```

---

Ozone4H

*Ground-level ozone order statistics.*

---

## Description

Ground-level ozone order statistics from 1997 at 513 monitoring stations in the eastern United States.

## Usage

```
data(Ozone4H)
```

## Format

A data frame with 513 observations on the following 5 variables.

**station** a numeric vector identifying the station (or line) number.

**r1** a numeric vector giving the maximum ozone reading (ppb) for 1997.

**r2** a numeric vector giving the second-highest ozone reading (ppb) for 1997.

**r3** a numeric vector giving the third-highest ozone reading (ppb) for 1997.

**r4** a numeric vector giving the fourth-highest ozone reading (ppb) for 1997.

## Details

Ground level ozone readings in parts per billion (ppb) are recorded hourly at ozone monitoring stations throughout the country during the "ozone season" (roughly April to October). These data are taken from a dataset giving daily maximum 8-hour average ozone for 5 ozone seasons (including 1997). The new U.S. Environmental Protection Agency (EPA) National Ambient Air Quality Standard (NAAQS) for ground-level ozone is based on a three-year average of fourth-highest daily 8-hour maximum ozone readings (see <http://www.epa.gov/air/criteria.html>).

For more analysis on the original data regarding the U.S. EPA NAAQS for ground-level ozone, see Fuentes (2003), Gilleland and Nychka (2005) and Gilleland et al. (2005b). For an example of using these data with **extRemes**, see Gilleland et al. (2005a). These data are in the form required by the `rlarg.fit` function of Stuart Coles available in the R package **ismev**; see Coles (2001) for more on the  $r$ -th largest order statistic model and the function `rlarg.fit`.

## Source

Data was originally provided by the U.S. EPA (<http://www.epa.gov/epahome/ozone.htm>), and can be obtained (daily maximum 8-hour average ozone for all five seasons from 1995 to 1999) from the Geophysical Statistics Project (GSP) at the National Center for Atmospheric Research (NCAR) at

<http://www.cgd.ucar.edu/stats/data.shtml>

along with the original longitude/latitude coordinates for the stations.

## References

- Coles, Stuart. An Introduction to Statistical Modeling of Extreme Values. Springer-Verlag, London, 2001.
- Fuentes, Montserrat. Statistical assessment of geographic areas of compliance with air quality. *Journal of Geophysical Research*, 108(D24), 2003.
- Gilleland, Eric and Nychka, Douglas. Statistical Models for Monitoring and Regulating Ground-level Ozone, *Environmetrics*, 16:535–546. 2005.
- a) Gilleland, Eric and Katz, Richard W. Tutorial for the 'Extremes Toolkit: Weather and Climate Applications of Extreme Value Statistics.' <http://www.assessment.ucar.edu/toolkit>, 2005.
- b) Gilleland, Eric, Nychka, Douglas, and Schneider, Uli. Spatial models for the distribution of extremes, *Applications of Computational Statistics in the Environmental Sciences: Hierarchical Bayes and MCMC Methods* Edited by J.S. Clark & A. Gelfand. Oxford University Press, 2005. (to appear)

## Examples

```
data(Ozone4H)
str(Ozone4H)
plot(Ozone4H)
# See the extRemes tutorial (Gilleland et al. (2005a)) for a much better example (uses the G
```

---

PORTw

*Daily maximum and minimum temperature*

---

## Description

Daily maximum and minimum Winter temperature (degrees centigrade) with a covariate for the North Atlantic Oscillation index from 1927 through 1995. Data is for Winter for Port Jervis, New York (PORTw) and Spring for Sept-Iles, Quebec (SEPTsp).

## Usage

```
data(PORTw)
```

## Format

A data frame with 68 observations on the following 16 variables.

**Year** a numeric vector giving the year.

**MTMAX** a numeric vector giving the mean maximum temperatures (degrees centigrade) over a one month period.

**MTMIN** a numeric vector giving the mean minimum temperatures (degrees centigrade). over a one month period.

**STDTMAX** a numeric vector giving the standard deviations of maximum temperatures (degrees centigrade) over a one month period.

**STDMIN** a numeric vector giving the standard deviations of minimum temperatures (degrees centigrade) over a one month period.

**TMX1** a numeric vector giving the maximum temperature (degrees centigrade) over a one month period.

**TMN0** a numeric vector giving the minimum temperature (degrees centigrade) over a one month period.

**MDTR** a numeric vector giving the mean diurnal temperature range (degrees centigrade).

**AOindex** a numeric vector giving the Atlantic Oscillation index (see Thompson and Wallace (1998)).

### Details

See Wettstein and Mearns (2002) for a much more detailed explanation of the above variables.

### Source

National Oceanic and Atmospheric Administration/National Climate Data Center (NOAA/NCDC).

### References

Gilleland, Eric, and Katz, Richard W., Tutorial for the 'Extremes Toolkit: Weather and Climate Applications of Extreme Value Statistics.' <http://www.assessment.ucar.edu/toolkit>, 2005.

Thompson, D.W.J. and Wallace, J.M. The Arctic Oscillation signature in the wintertime geopotential height and temperature fields, *Geophys. Res. Lett.*, 25: 1297–1300, 1998.

Wettstein, Justin J. and Mearns, Linda O., The influence of the North Atlantic-Arctic Oscillation on mean, variance and extremes of temperature in the northeastern United States and Canada. *Journal of Climate*, 15:3586–3600, 2002.

### Examples

```
data(PORTw)
str(PORTw)
par(mfrow=c(2,1))
plot(PORTw["TMX1"], type="l", lwd=2, xlab="", xaxt="n", ylab="Maximum Temperature (C)")
plot(PORTw["TMN0"], type="l", lwd=2, xlab="", xaxt="n", ylab="Minimum Temperature (C)")
par(mfrow=c(1,1))
# See Gilleland et al. (2005) for more examples with these data using extRemes.
```

---

Peak

*Salt River peak stream flow.*

---

### Description

Peak stream flow data from 1924 through 1999 for the Salt River near Roosevelt, Arizona.

**Usage**

```
data(Peak)
```

**Format**

A data frame with 75 observations on the following 2 variables.

**Year** a numeric vector giving the year.

**Flow** a numeric vector giving the peak stream flow (cfs).

**Winter** a numeric vector giving the Winter seasonal mean Darwin pressure (mb–1000).

**Spring** a numeric vector giving the Spring seasonal mean Darwin pressure (mb–1000).

**Summer** a numeric vector giving the Summer seasonal mean Darwin pressure (mb–1000).

**Fall** a numeric vector giving the Fall seasonal mean Darwin pressure (mb–1000) (see Katz et al. (2002) Sec. 5.2.2).

**Details**

Peak stream flow in cfs (1 cfs=0.028317  $m^3/s$ ) data for water years (October through September) from 1924 through 1999 for the Salt River near Roosevelt, Arizona. Data for 1986 are missing. Also includes seasonal mean Darwin pressures (mb–1000).

Several analyses have been performed on streamflow at this location (see, e.g., Anderson and Meerschaert (1998), Dettinger and Diaz (2000); and, for extreme stream flow, Katz et al. (2002) Sec. 5.2.2).

**Source**

U.S. Geological Survey (<http://water.usgs.gov/nwis/peak>) for Salt River peak flows. NOAA Climate Prediction Center (<http://www.cpc.ncep.noaa.gov/data/indices>) for seasonal mean Darwin pressures.

**References**

Anderson, P.L. and Meerschaert, M.M. Modeling river flows with heavy tails. *Water Resour Res*, (34):2271–2280, 1998.

Dettinger, M.D. and Diaz, H.F. Global characteristics of stream flow seasonality and variability. *J. Hydrometeorol* (1):289–310, 2000.

Katz, Richard W., Parlange, Marc B. and Naveau, Philippe. Statistics of extremes in hydrology. *Advances in Water Resources*, (25):1287–1304, 2002.

**Examples**

```
data(Peak)
str(Peak)
# Fig. 9 of Katz et al. (2002) Sec. 5.2.2.
plot(Peak[, "Year"], Peak[, "Flow"]/1000, type="l", yaxt="n",
      xlab="Water year (Oct-Sept)", ylab="Annual peak flow (thousand cfs)")
axis(2, at=c(0, 40, 80, 120), labels=c("0", "40", "80", "120"))
```

---

Potomac

*Potomac River peak stream flow data.*

---

### Description

Potomac River peak stream flow (cfs) data for water years (Oct-Sep) 1895 through 2000 at Point Rocks, Maryland.

### Usage

```
data(Potomac)
```

### Format

A data frame with 106 observations on the following 2 variables.

**Year** a numeric vector giving the water year (Oct-Sep).

**Flow** a numeric vector the peak stream flow (cfs; 1 cfs = 0.028317 cubic meters per second).

### Details

Potomac River peak stream flow (cfs) data for water years (Oct-Sep) 1895 through 2000 at Point Rocks, Maryland.

These data (up to 1986) have been analyzed by Smith (1987) and this entire dataset by Katz et al. (2002) Sec. 2.3.2.

### Source

U.S. Geological Survey (<http://water.usgs.gov/nwis/peak>).

### References

Katz, Richard W., Parlange, Marc B. and Naveau, Philippe. Statistics of extremes in hydrology. *Advances in Water Resources*, (25):1287–1304, 2002.

Smith, J.A. Regional flood frequency analysis using extreme order statistics of the annual peak record. *Water Resour Res* (23):1657–1666, 1987.

### Examples

```
data(Potomac)
str(Potomac)
# Fig. 3 of Katz et al. (2002) Sec. 2.3.2.
plot(Potomac[, "Year"], Potomac[, "Flow"]/1000, yaxt="n", ylim=c(0,500), type="l", lwd=1.5,
      xlab="Water Year (Oct-Sept)", ylab="Annual peak flow (thousand cfs)")
axis(2, at=seq(0, 500, 100), labels=as.character(seq(0, 500, 100)))
```

---

Rsum

*Hurricane frequency dataset.*

---

### Description

This dataset gives the number of hurricanes per year (from 1925 to 1995) as well as the ENSO state and total monetary damage.

### Usage

```
data(Rsum)
```

### Format

A data frame with 71 observations on the following 4 variables.

**Year** a numeric vector giving the year.

**EN** a numeric vector giving the ENSO state (-1 for La Niña, 1 for El Niño and 0 otherwise).

**Ct** a numeric vector giving the number of hurricanes for the corresponding year.

**TDam** a numeric vector giving the total monetary damage (millions of U.S. dollars).

### Details

More information on these data can be found in Pielke and Landsea (1998) or Katz (2002).

### Source

[http://sciencepolicy.colorado.edu/pielke/hp\\_roger/hurr\\_norm/data.html](http://sciencepolicy.colorado.edu/pielke/hp_roger/hurr_norm/data.html)

### References

Gilleland, Eric and Katz, Richard W. Tutorial for the 'Extremes Toolkit: Weather and Climate Applications of Extreme Value Statistics.' <http://www.assessment.ucar.edu/toolkit>, 2005.

Katz, Richard W., Stochastic modeling of hurricane damage. *Journal of Applied Meteorology*, 41:754–762, 2002.

Pielke, Roger A. and Landsea, CW., Normalized hurricane damages in the United States: 1925-95. *Weather and Forecasting*, 13 (3):621–631, 1998.

### Examples

```
data(Rsum)
str(Rsum)
plot(Rsum)
```

```
# Reproduce Fig. 1 of Katz (2002).
plot(  Rsum["Year"], Rsum["TDam"]/1000, type="h", xlab="Year",
```

```

        ylab="Total damage (billion U.S. dollars)",
        ylim=c(0,80), lwd=2)

# Reproduce Fig. 2 of Katz (2002).
plot(Rsum[, "Year"], Rsum[, "Ct"], type="h", xlab="Year", ylab="Number of Hurricanes", ylim=c(0,

# See Gilleland et al. (2005) for more examples using these data with extRemes.

```

---

Tphap

*Daily maximum and minimum temperature in Phoenix, Arizona.*


---

### Description

Daily maximum and minimum temperature (degrees Fahrenheit) for July through August 1948 through 1990 at Sky Harbor airport in Phoenix, Arizona.

### Usage

```
data(Tphap)
```

### Format

A data frame with 43 observations on the following 3 variables.

**Year** a numeric vector giving the number of years since 1900.

**Month** a numeric vector giving the month.

**Day** a numeric vector giving the day of the month.

**MaxT** a numeric vector giving the daily maximum temperatures in degrees Fahrenheit.

**MinT** a numeric vector giving the daily minimum temperatures in degrees Fahrenheit.

### Details

Data is daily maximum and minimum temperature for the summer months of July through August from 1948 through 1990.

### Source

U.S. National Weather Service Forecast office at the Phoenix Sky Harbor Airport.

### References

Balling, R.C., Jr., Skindlov, J.A. and Phillips, D.H., The impact of increasing summer mean temperatures on extreme maximum and minimum temperatures in Phoenix, Arizona. *Journal of Climate*, 3:1491–1494, 1990.

Gilleland, Eric and Katz, Richard W. Tutorial for the 'Extremes Toolkit: Weather and Climate Applications of Extreme Value Statistics.' <http://www.assessment.ucar.edu/toolkit>, 2005.

Tarleton, Lesley F. and Katz, Richard W., Statistical explanation for trends in extreme summer temperatures at Phoenix, A.Z., *Journal of Climate*, 8 (6):1704–1708, 1995.

**Examples**

```

data(Tphap)
str(Tphap)

par(mfrow=c(2,1))
hist(Tphap[, "MaxT"], main="", xlab="Max Temp", xlim=c(60,120), freq=FALSE, breaks="FD", col="red")
hist(Tphap[, "MinT"], main="", xlab="Min Temp", xlim=c(60,120), freq=FALSE, breaks="FD", col="blue")
par(mfrow=c(1,1))

# See Gilleland et al. (2005) for more examples using these data with extRemes.

```

---

```
as.extRemesDataObject
```

*Convert a data frame, matrix or vector to an object of class "extRemesDataObject"*

---

**Description**

Converts a data frame, matrix or vector to an object of class "extRemesDataObject" so that extRemes GUI dialogs can handle the data.

**Usage**

```
as.extRemesDataObject(x)
```

**Arguments**

`x` A data frame, matrix or vector.

**Details**

If an object is a vector of length  $n$ , an ' $n \times 2$ ' matrix is created where the first column is simply the line number. If a vector (e.g., `rnorm(10)`) is passed as the argument '`x`', then the second column will be labelled as '`x`'. If '`x`' is a matrix or data frame, then the column names of '`x`' will be used. Otherwise, column names will be '`x1,x2,...`'

**Value**

A list object is returned with class "extRemesDataObject" and only one component.

`data` The data passed via the '`x`' argument.

**Author(s)**

Eric Gilleland

**See Also**

[extremes.gui](#)

**Examples**

```
z <- rnorm( 100)
y <- as.extRemesDataObject(z)
# The object 'y' can now be used with the extRemes GUI dialogs.
# To open the main dialog window, use 'extremes.gui()'.
```

---

bisearch	<i>Simple bisection search.</i>
----------	---------------------------------

---

**Description**

Bisection search algorithm to find where a function crosses a particular value.

**Usage**

```
bisearch(x1, x2, f, tol = 1e-07, niter = 25, upcross.level = 0)
```

**Arguments**

x1	Low-end starting value.
x2	High-end starting value.
f	A function for which to find the crossing value.
tol	The tolerance that determines whether to break (early) from the algorithm.
niter	Number of iterations to try before quitting.
upcross.level	The level where it is desired to find where f crosses.

**Details**

This function is shamelessly stolen from the **fields** package (cf. `bisection.search`), but is slightly simpler (i.e., less useful) in that it doesn't allow for extra arguments to `f`. It is used in the `gev.parameterCI` and `gpd.parameterCI` functions to try to find analytically where the profile likelihood crosses the chi-square based critical value for obtaining confidence bounds on parameter estimates.

**Value**

x	The estimated crossing.
fm	The difference between the function value at the found crossing and <code>upcross.level</code> .
iter	Number of iterations used to find the crossing.

**Author(s)**

Unknown

**References**

Fishbane, P.M., Gasiorowicz, S., and S.T. Thornton Physics for Scientists and Engineers, 2d ed. (Extended), Upper Saddle River, New Jersey: Prentice Hall, 1996.

**See Also**

[gev.parameterCI](#), [gpd.parameterCI](#)

**Examples**

```
# Try to solve x^2 = 0.2
# Initial guess is that it occurs between x=0 and x=1.
bisearch( 0, 1, function(x) x^2, tol=1e-20, upcross.level=0.2)
```

---

boot.matrix

*Boot Matrices.*

---

**Description**

Set up matrices for bootstrapping sequences of extreme values.

**Usage**

```
boot.matrix(x, y)
```

**Arguments**

x	Output from <code>decluster.runs</code> or <code>decluster.intervals</code> .
y	Vector of observations.

**Details**

This function merely formats the information needed by `boot.sequence` to improve efficiency.

**Value**

A list containing two (unnamed) matrices, each with columns corresponding to clusters identified in 'x.'

comp1	inter-exceedance times
comp2	Data values in 'y' corresponding to the exceedances in each cluster.

**Note**

Maintained by Eric Gilleland.

**Author(s)**

Chris Ferro

**See Also**

`boot.sequence`, `decluster.intervals`, `decluster.runs`

**Examples**

```
# Simulate 1000 uniform random variables.
x <- runif(1000)

# Perform runs declustering with run length = 1 and 90th percentile as threshold.
u <- quantile(x, 0.9)
z <- x > u
dec <- decluster.runs(z, 1)

# Make sure estimated run length is not zero before doing the rest.
if( dec[["par"]] != 0) {
  # Set up the matrices for bootstrapping.
  mat <- boot.matrix(dec, x)
  # Bootstrap with 500 iterations.
  eib <- numeric(500)
  for( i in 1:500) {
    set.seed(i)
    zb <- boot.sequence(mat[[1]],mat[[2]],u) > u
    eib[i] <- exi.intervals(zb)
  } # end of for 'i' loop.
  # Obtain bootstrapped 95th percentile confidence intervals.
  conf.int <- quantile( eib, c((1-0.95)/2, (1+0.95)/2))
} # end of if run length estimate not zero stmt.
```

---

<code>boot.sequence</code>	<i>Bootstrap a sequence.</i>
----------------------------	------------------------------

---

**Description**

Bootstrap a sequence of exceedances.

**Usage**

```
boot.sequence(tmat, ymat, u)
```

**Arguments**

<code>tmat</code>	Output from <code>boot.matrix</code> ('iet' component).
<code>ymat</code>	Output from <code>boot.matrix</code> ('ce' component).
<code>u</code>	Threshold above which the sequences of exceedances are to be bootstrapped.

**Details**

The bootstrapped sequence contains the same number of clusters as identified in the original data. Clusters of exceedances and inter-exceedance times within clusters are resampled as single entities; inter-exceedance times between clusters are resampled independently. The values of non-exceedances are set equal to the threshold 'u'.

**Value**

Vector of exceedances above 'u'.

**Warning**

The bootstrapped sequence can be longer than the original sequence.

**Note**

Maintained by Eric Gilleland.

**Author(s)**

Chris Ferro

**References**

Ferro CAT and Segers J. (2003). Inference for clusters of extreme values. *Journal of the Royal Statistical Society B* 65:545–556.

**See Also**

[boot.matrix](#)

**Examples**

```
# Simulate 1000 uniform random variables.
x <- runif(1000)

# Perform runs declustering with run length = 1 and 90th percentile as threshold.
u <- quantile(x, 0.9)
z <- x > u
dec <- decluster.runs(z, 1)

# Make sure the estimated run length is not zero before doing the rest.
if( dec[["par"]] != 0) {
  # Set up the matrices for bootstrapping.
  mat <- boot.matrix(dec, x)
  # Bootstrap with 500 iterations.
  eib <- numeric(500)
  for( i in 1:500) {
    set.seed(i)
    zb <- boot.sequence(mat[[1]],mat[[2]],u) > u
    eib[i] <- exi.intervals(zb)
  }
}
```

```
    } # end of for 'i' loop.  
    # Obtain bootstrapped 95th percentile confidence intervals.  
    conf.int <- quantile( eib, c((1-0.95)/2, (1+0.95)/2))  
  } # end of if run length not zero stmt.
```

---

clearlog

*Clear file extRemes.log*

---

### Description

Overwrites the file extRemes.log with the date and time that the file is cleared.

### Usage

```
clearlog(base.txt)
```

### Arguments

base.txt      Unused argument for compatibility with GUI windows.

### Details

As the GUI windows are used, the major commands invoked by the GUI windows are written to the file extRemes.log, and this file can get large with use. This function is invoked when 'Clear log file' is selected from the File menu of the main toolkit dialog.

The file is cleared, but still exists with a message indicating the time the file was last cleared. If no extRemes.log file exists, then one is created with the date and time message.

### Value

No value returned. Has the effect of erasing any information contained in extRemes.log, or of creating this file if it does not exist; and adds the date and time when the file was cleared/created.

### Author(s)

Eric Gilleland

---

 damage

---

*Hurricane Damage Data*


---

**Description**

Estimated economic damage (billions USD) caused by hurricanes.

**Usage**

```
data(damage)
```

**Format**

A data frame with 144 observations on the following 3 variables.

**obs** a numeric vector that simply gives the line numbers.

**Year** a numeric vector giving the years in which the specific hurricane occurred.

**Dam** a numeric vector giving the total estimated economic damage in billions of U.S. dollars.

**Details**

More information on these data can be found in Pielke and Landsea (1998) or Katz (2002). Also see Gilleland et al. (2005) for examples using **extRemes**.

**Source**

[http://sciencepolicy.colorado.edu/pielke/hp\\_roger/hurr\\_norm/data.html](http://sciencepolicy.colorado.edu/pielke/hp_roger/hurr_norm/data.html)

**References**

Gilleland, Eric and Katz, Richard W. Tutorial for the 'Extremes Toolkit: Weather and Climate Applications of Extreme Value Statistics.' <http://www.assessment.ucar.edu/toolkit>, 2005.

Katz, Richard W., Stochastic modeling of hurricane damage. *Journal of Applied Meteorology*, 41:754–762, 2002.

Pielke, Roger A. and Landsea, CW., Normalized hurricane damages in the United States: 1925-95. *Weather and Forecasting*, 13 (3):621–631, 1998.

**Examples**

```
data(damage)
plot( damage[,1], damage[,3], xlab="", ylab="Economic Damage", type="l", lwd=2)

# Fig. 3 of Katz (2002).
plot( damage[, "Year"], log( damage[, "Dam"]), xlab="Year", ylab="ln(Damage)", ylim=c(-10,5))

# Fig. 4 of Katz (2002).
qqnorm( log( damage[, "Dam"]), ylim=c(-10,5))
```

---

dclust                      *Decluster data by runs declustering.*

---

### Description

Decluster data by assuming that exceedances belong to the same cluster if they are separated by fewer than 'r' (run length) values below a given threshold.

### Usage

```
dclust(xdat, u, r, cluster.by = NULL, verbose=getOption("verbose"))
```

### Arguments

xdat	a single numeric vector of data to be declustered.
u	single number or vector of thresholds.
r	run length
cluster.by	If there are blocks implying a natural clustering that is to be preserved (e.g., if data cover several years, but only for a single season), this is a vector defining the blocks to ensure that clusters do not cross over from one block to another.
verbose	logical whether to field progress information to screen or not.

### Details

This function applies runs declustering to automatically decluster a dataset. To ensure that clusters do not cross natural or decided boundaries, use the 'cluster.by' option. That is, suppose data are measured only in the summer, say from June 1 through August 1. In such a case, it is perhaps not desired to have a value from August 1, 2003 and June 1, 2004 in the same cluster. To account for this, create a 'cluster.by' vector defining years in order to keep clusters within years. For the example of data from June 1 to August 1 (62 days), a vector like `c(rep(1, 62), rep(2, 62), ..., rep(n, 62))` should be used for the 'cluster.by' argument.

This function will return a vector of the same length as the original data vector, but with maximums from each cluster followed by 'filler' numbers that are below the given threshold, 'u'.

Missing values are not handled. The function will still run, but the results will be questionable.

### Value

A list with components:

xdat.dc	Maximums from each cluster with additional filler values below the given threshold 'u' in order to maintain the same length as the original data vector 'xdat'. This is for compatibility with extRemes GUI data object of class "ev.data".
ncluster	The number of clusters found by runs declustering.
clust	numeric vector giving the clusters.

**Author(s)**

Eric Gilleland

**References**

Coles, Stuart (2001). An Introduction to Statistical Modeling of Extreme Values. Springer-Verlag, London.

Gilleland, Eric and Katz, Richard W. Tutorial for the 'Extremes Toolkit: Weather and Climate Applications of Extreme Value Statistics.' <http://www.assessment.ucar.edu/toolkit>, 2005.

**Examples**

```
# Load a dataset.
data(Tphap)

plot(Tphap[, "MaxT"])
abline(h=115)

# Decluster using a threshold of 115 degrees and a run length of 'r=1'.
temp <- dclust(xdat=Tphap[, "MaxT"], u=115, r=1, cluster.by = Tphap[, "Year"])
temp[["ncluster"]] # See how many clusters were found.

# Now do the same as above, but with a run length of 3 for comparison.
# Note: 'r=2' gives same clusters as 'r=1' for these data.
temp2 <- dclust(xdat=Tphap[, "MaxT"], u=115, r=3, cluster.by = Tphap[, "Year"])
temp2[["ncluster"]]

# See Gilleland et al. (2005) for more.
```

---

decluster.runs

*Declustering Extremes*

---

**Description**

Performs runs/intervals declustering.

**Usage**

```
decluster.runs(z, r)
decluster.intervals(z, ei)
```

**Arguments**

z	Logical vector indicating which positions correspond to extreme values.
r	Integer run length.
ei	Estimate of the extremal index.

**Details**

Runs declustering: Extremes separated by fewer than 'r' non-extremes belong to the same cluster. Setting 'r' < 1 causes each extreme to form a separate cluster.

Intervals declustering: Extremes separated by fewer than 'r' non-extremes belong to the same cluster, where 'r' is the 'nc'-th largest interexceedance time and 'nc', the number of clusters, is estimated from the extremal index, 'ei', and the times between extremes. Setting 'ei' = 1 causes each extreme to form a separate cluster.

**Value**

A list containing

scheme	Name of declustering scheme.
par	Value of declustering parameter (i.e., run length).
nc	Number of clusters.
size	Vector of cluster sizes.
s	Vector of times of extremes.
cluster	Vector of numbers identifying clusters to which extremes belong.
t	Vector of times between extremes.
inter	Vector of intercluster time indicators (logical).
intra	Vector of intracluster time indicators (logical).

**Note**

Maintained by Eric Gilleland.

**Author(s)**

Chris Ferro

**References**

Smith RL (1989) Extreme value analysis of environmental time series: an application to trend detection in ground-level ozone. *Statistical Science* 4, 367-393.

Ferro CAT and Segers J (2003) Inference for clusters of extreme values. *Journal of the Royal Statistical Society B* 65, 545-556.

**See Also**

[exi.intervals](#)

**Examples**

```

# Simulate a dependent series of random variables.
x <- runif(1000,-1,1)
x[2:1000] <- x[1:999]*0.6
# -- DON'T RUN
# pacf( x)

# use runs and intervals declustering using the 90th percentile as the threshold.
u <- quantile(x, 0.9)
z <- x > u
exi.intervals(z)
tmp1 <- decluster.runs(z, 1)
tmp2 <- decluster.intervals( z, exi.intervals(z))

```

---

 eiAnalyze

*Extremal index analysis*


---

**Description**

Estimate the extremal index (and confidence limits) using intervals estimation.

**Usage**

```
eiAnalyze(x, thresholds = quantile(x, seq(0.9, 0.995, by = 0.005)), conf = 0.95, it
```

**Arguments**

<code>x</code>	'n X 1' vector of data observations.
<code>thresholds</code>	Single number, or 'm X 1' vector of thresholds above which to estimate the extremal index.
<code>conf</code>	The <code>conf</code> *100 percent confidece level.
<code>iter</code>	Number of iterations desired for bootstrapping.
<code>plot</code>	Logical whether or not to plot the extremal indices against the thresholds (w/ confidence bounds). Only used if the length of 'thresholds' is greater than 1.

**Details**

Uses several functions written by Chris Ferro for estimating the extremal index by intervals estimation (Ferro and Segers (2003)). Specifically, it calls the functions: `exi.intervals`, `decluster.intervals`, `boot.matrix` and `boot.sequence`.

**Value**

Returns a list with the following components.

<code>ei</code>	'm X 1' vector of the estimate(s) of the extremal index.
<code>ci</code>	An 'm X 2' matrix giving the <code>conf*100</code> percent confidence limits for <code>ei</code> .
<code>u</code>	The threshold(s) used.
<code>nc</code>	An 'n X 1' vector giving the number of clusters for each threshold choice.
<code>run.length</code>	An 'm X 1' vector giving the estimated run lengths.

**Note**

Maintained by Eric Gilleland.

**Author(s)**

Chris Ferro and Eric Gilleland

**References**

Ferro CAT and Segers J (2003) Inference for clusters of extreme values. *Journal of the Royal Statistical Society B* 65, 545-556.

**See Also**

[exi.intervals](#), [decluster.intervals](#), [boot.matrix](#), [boot.sequence](#)

**Examples**

```
## The function is currently defined as
# function(x, thresholds=quantile(x, seq(0.900,0.995,by=0.005)), conf=0.95, iter=500)
```

---

`exi.intervals`      *Extremal index estimator*

---

**Description**

Evaluates the intervals estimator for the extremal index.

**Usage**

```
exi.intervals(z)
```

**Arguments**

`z`                      Logical vector indicating which positions correspond to extreme values.

**Value**

Estimate of the extremal index.

**Warning**

The estimator is not constrained to lie in  $[0,1]$  and a default value of 1 is returned if there are fewer than two extreme values.

**Note**

Maintained by Eric Gilleland.

**Author(s)**

Chris Ferro

**References**

Ferro CAT and Segers J (2003) Inference for clusters of extreme values. *Journal of the Royal Statistical Society B* 65, 545-556.

**See Also**

[decluster.intervals](#)

**Examples**

```
x <- rnorm(1000)
exi.intervals(x > quantile(x, 0.9))
```

---

extRemes internal *extRemes internal and secondary functions*

---

**Description**

Listed below are supporting functions for the major methods in extRemes. Most of the functions control GUI interfaces for the R package ismev.

**Usage**

```
DataSummaryGUI(base.txt)
extremalind.gui(base.txt)
gev.ret(A, b, c, p = 100, ...)
gev.rlci(z, m, prange = NULL, conf = 0.95, nint = 100)
gpd.ret(z, p)
gpd.rlci(z, m, prange=NULL, npy = 365, conf = 0.95, nint = 100)
gpd.rl2(a, u, la, n, npy, mat, dat, xdat, ci=0.05, add.ci=FALSE)
indicatorTransform.gui(base.txt)
```

```
llhrt.gui(base.txt)
## S3 method for class 'gev.fit':
plot(x, ...)
## S3 method for class 'gpd.fit':
plot(x, ...)
## S3 method for class 'gum.fit':
plot(x, ...)
## S3 method for class 'pp.fit':
plot(x, ...)
## S3 method for class 'rlarg.fit':
plot(x, ...)
rl.gev.fit(a, mat, dat)
rl.gpd.fit(a, u, la, n, npy, mat, dat, xdat)
scatterplot.gui(base.txt)
affine.gui(base.txt)
ppfit.gui(base.txt)
ppfitrange.gui(base.txt)
gevf.gui(base.txt)
gpdsim.gui(base.txt)
gevparamCI.gui(base.txt)
hist.gev.fit(x, breaks.method="Sturges", ...)
hist.gpd.fit(x, breaks.method="Sturges", ...)
hist.pp.fit(x, breaks.method="Sturges", ...)
rlargf.gui(base.txt)
rlplot.gui(base.txt)
decluster.gui(base.txt)
deviancestat(l1, l2, v, alpha=0.05)
histogram.gui(base.txt)
## S3 method for class 'gev.fit':
summary(object, ...)
## S3 method for class 'gpd.fit':
summary(object, ...)
load.data(base.txt)
## S3 method for class 'pp.fit':
summary(object, ...)
diagnostic.plots
describe2(x)
gevsim.gui(base.txt)
logdr.gui(base.txt)
stats2(x, by)
## S3 method for class 'rlarg.fit':
summary(object, ...)
logtrans.gui(base.txt)
trigtrans.gui(base.txt)
mrlplot.gui(base.txt)
view.data(base.txt)
gpdf.gui(base.txt)
negtrans.gui(base.txt)
```

```

zzz
fitdiag.gui(base.txt)
gpdfitrangle.gui(base.txt)
fitsummary.gui(base.txt)
gpdparamCI.gui(base.txt)
poisson.gui(base.txt)
scrubber.gui(base.txt)
write.extRemesMainMessage(txt)

```

---

extRemes

*extRemes – The Extremes Toolkit: Weather and Climate Applications  
of Extreme-Value Statistics*


---

## Description

**extRemes** is a graphical user interface (GUI) for modeling extreme values with the purpose of providing a pedagogical tool for scientists who are unfamiliar with extreme value theory or extreme value software. It is not designed to teach the R programming language; although, R command lines invoked by the GUI dialogs are printed to a log file (`extRemes.log`) in the current working directory. This package relies heavily on the **ismev** package, and most of the underlying functions are from this package. **ismev** is an R port of Stuart Coles' S-Plus extreme value statistical routines.

For help using **extRemes**, please see the tutorial at:

<http://www.assessment.ucar.edu/toolkit>

Also, please consider registering at this site. We need to determine if enough people are using the package to continue with its maintenance and development.

Extreme Value Statistics:

Extreme value statistics are used primarily to quantify the stochastic behavior of a process at unusually large (or small) values. Particularly, such analyses usually require estimation of the probability of events that are more extreme than any previously observed. Many fields have begun to use extreme value theory and some have been using it for a very long time including meteorology, hydrology, finance and ocean wave modeling to name just a few.

Example Datasets:

There are several example datasets included with this toolkit. In each case, it is possible to load these datasets into R using the `data` function. Use the `as.extRemesDataObject` function to coerce these data to the form that the GUI windows can recognize when loading data in this way. Generally, it is possible to load all of these datasets using the GUI windows (File → Read Data → browse to data and select → choose R source → OK). The example datasets will be in the data directory of the **extRemes** file structure. It is not necessary to unzip any files to do this; if you find yourself unzipping files, you have chosen the wrong file! The example datasets included with **extRemes** are:

Denmint – Denver daily minimum temperature.

Flood.dat – U.S. Flood damage (in terms of monetary loss) ('dat' file used as example of reading in common data using the `extRemes` dialog).

ftcanmax – Annual maximum precipitation amounts at one rain gauge in Fort Collins, Colorado.

HEAT – Summer maximum (and minimum) temperature at Phoenix Sky Harbor airport.

Ozone4H.dat – Ground-level ozone order statistics from 1997 from 513 monitoring stations in the eastern United States.

PORTw – Maximum and minimum temperature data (and some covariates) for Port Jervis, New York.

Rsum – Frequency of Hurricanes.

SEPTsp – Maximum and minimum temperature data (and some covariates) for Sept-Iles, Quebec.

damage – Hurricane monetary damage.

Denversp – Denver precipitation.

Flood – R source version of the above mentioned 'Flood.dat' dataset.

FtCoPrec – Precipitation amounts at one rain gauge in Fort Collins, Colorado.

Peak – Salt River peak stream flow.

Potomac – Potomac River peak stream flow.

Tphap – Daily maximum and minimum temperatures at Phoenix Sky Harbor Airport.

For more information on any of these datasets, type

```
help( [name_of_dataset] )
```

from the R prompt.

Primary Functions for extRemes:

As mentioned above, **extRemes** is primarily a pedagogical tool that provides a GUI interface to the **ismev** package. Listed below is the function to invoke the main toolkit dialog window, followed by a few underlying functions provided with **extRemes** that are not provided with **ismev**.

```
extremes.gui:
```

The main toolkit dialog window should appear upon loading extRemes with `library(extRemes)`. If this window is closed while the package is still loaded, use `extremes.gui()` from the R command prompt to re-open the main dialog window.

————— The following is targeted at the more advanced R user. It is hoped that a person unfamiliar with R can use this toolkit without learning more than a very few R commands; and the tutorial at the above website should be sufficient to get started for such a user.

Extremes Data Object:

For ease of operation, it is convenient to have a special class of objects for the GUI windows to operate on. This class, `extRemesDataObject`, might be considered an orphaned object in R lingo because it does not have the usual method functions associated with it—they generally do not make sense for this object. The object simply stores information so that the GUI windows know which elements of the object to use in particular situations. It would not, for example, make sense to do `plot(tmp)` (where `tmp` is an object of class `extRemesDataObject` because there are numerous possible plots depending on what is included in the object; it would be tantamount to plotting the entire workspace (in some cases). The `extRemesDataObject` is a list object with, at the very least, a `data` component that includes either a data frame or matrix. Often it will also have `name` and `file.path` components (whenever data is loaded via the GUI windows) which give the name of the original file, which can be different from the name of the `extRemesDataObject` list object, and the path to the original file.

---

The Extremes Toolkit is funded by the National Science Foundation (NSF) through the NCAR Weather and Climate Impact Assessment Science Initiative with additional support from the NCAR Geophysical Statistics Project (GSP).

---



---

extremalindex      *Estimate the extremal index (theta)*

---

### Description

Estimates the extremal index based on the intervals estimator due to Ferro and Segers (2003).

### Usage

```
extremalindex(xdat, u)
```

### Arguments

xdat	n X 1 numeric vector of the data.
u	User defined threshold. May be a single number or a numeric n X 1 vector.

### Details

The extremal index is a useful indicator of how much clustering of exceedances of a threshold occurs in the limit of the distribution. If  $\theta = 1$ , then the data is independent and if  $\theta < 1$ , then there is some dependency (clustering) in the limit.

There are many possible estimators of the extremal index. The one used here is the intervals estimator described in Ferro and Segers (2003). It is unbiased in the mean and can be used to estimate the number of clusters, which is also done by this function.

### Value

A list with components:

theta	The intervals estimate of the extremal index (theta).
msg	A message stating whether any interexceedance times were above 2 and possibly other pertinent information.
C	The estimated number of clusters (handles ties).
run.length	The estimated run length for runs declustering.

### Author(s)

Eric Gilleland

## References

Ferro, C.A.T. and Segers, J. (2003): Inference for clusters of extreme values, J.R. Statist. Soc. B, 65 (2): 545-556.

Gilleland, Eric and Katz, Richard W. Tutorial for the 'Extremes Toolkit: Weather and Climate Applications of Extreme Value Statistics.' <http://www.assessment.ucar.edu/toolkit>, 2005.

## See Also

dclust

## Examples

```
data( Tphap)
temp <- -Tphap[, "MinT"]
look <- extremalindex( temp, -70)
look

# See Gilleland et al. (2005) for more.
```

---

extremes.gui            *open the toolkit main dialog window*

---

## Description

Simply opens the toolkit dialog window.

## Usage

```
extremes.gui ()
```

## Value

Opens a Tcl/Tk dialog window.

## Note

This function is called on start-up (i.e., when `library( extRemes)` is invoked). This function may be used to re-open the main dialog window if it is closed while the package is still loaded.

## Author(s)

Greg Young and Eric Gilleland

## References

Gilleland, Eric and Katz, Richard W. Tutorial for the 'Extremes Toolkit: Weather and Climate Applications of Extreme Value Statistics.' <http://www.assessment.ucar.edu/toolkit>, 2005.

**Examples**

```
## Don't run
# extremes.gui()
```

fpp

*Point Process***Description**

Fit data to a point process model (allows for covariates in each of the parameters).

**Usage**

```
fpp(xdat, threshold, npy = 365, ydat = NULL, mul = NULL, sigl = NULL, shl = NULL, n
```

**Arguments**

xdat	'n X 1' vector of observed data.
threshold	The threshold; a single number or a numeric 'n X 1' vector.
npy	Number of points/observations per year.
ydat	Optional 'n X p' matrix of covariate information.
mul, sigl, shl	Numeric vectors of integers, giving the columns of 'ydat' that contain covariates for generalized linear modelling of the location, scale and shape parameters respectively (or 'NULL' (the default) if the corresponding parameter is stationary).
mulink, siglink, shlink	Inverse link functions for generalized linear modelling of the location, scale and shape parameters respectively.
show	Logical; if 'TRUE' (the default), print details of the fit.
method	The optimization method (see 'optim' for details).
maxit	The maximum number of iterations.
...	Other control parameters for the optimization. These are passed to components of the 'control' argument of 'optim'.

**Details**

For non-stationary fitting it is recommended that the covariates within the generalized linear models are (at least approximately) centered and scaled (i.e. the columns of 'ydat' should be approximately centered and scaled).

This function is a modification of the **ismev** function `pp.fit`, but uses the estimation  $\sum_i \lambda_i$  instead of  $\sum_i \lambda_i \cdot I_{x>u}$  (where  $\lambda$  is the exceedance rate and  $u$  the threshold) for the rate approximation of the point process likelihood.

**Value**

A list containing the following components. A subset of these components are printed after the fit. If 'show' is 'TRUE', then assuming that successful convergence is indicated, the components 'nexc', 'nllh', 'mle' and 'se' are always printed.

trans: An logical indicator for a non-stationary fit.

model: A list with components 'mul', 'sigl' and 'shl'.

link: A character vector giving inverse link functions.

threshold: The threshold, or vector of thresholds.

npy: The number of observations per year/block.

nexc: The number of data points above the threshold.

data: The data that lie above the threshold. For non-stationary models, the data is standardized.

conv: The convergence code, taken from the list returned by 'optim'. A zero indicates successful convergence.

nllh: The negative logarithm of the likelihood evaluated at the maximum likelihood estimates.

vals: A matrix with four columns containing the maximum likelihood estimates of the location, scale and shape parameters, and the threshold, at each data point. model: A list with components 'mul', 'sigl' and 'shl'.

link: A character vector giving inverse link functions.

threshold: The threshold, or vector of thresholds.

npy: The number of observations per year/block.

nexc: The number of data points above the threshold.

data: The data that lie above the threshold. For non-stationary models, the data is standardized.

conv: The convergence code, taken from the list returned by 'optim'. A zero indicates successful convergence.

nllh: The negative logarithm of the likelihood evaluated at the maximum likelihood estimates.

vals: A matrix with four columns containing the maximum likelihood estimates of the location, scale and shape parameters, and the threshold, at each data point.

gpd: A matrix with three rows containing the maximum likelihood estimates of corresponding GPD location, scale and shape parameters at each data point.

mle: A vector containing the maximum likelihood estimates.

cov: The covariance matrix.

se: A vector containing the standard errors.

**Warning**

Different optimization methods may result in wildly different parameter estimates.

**Note**

This is adapted from code originally written for S-Plus by Stuart Coles, and ported to R by Alec Stephenson. See details section above.

**Author(s)**

Eric Gilleland

**References**

Beirlant J, Goegebeur Y, Segers J and Teugels J. (2004). *Statistics of Extremes*, Wiley, Chichester, England.

Coles, Stuart (2001). *An Introduction to Statistical Modeling of Extreme Values*. Springer-Verlag, London.

**See Also**

pp.fit, pp.diag, [optim](#), pp.fitrange, mrl.plot, gpd.fit

**Examples**

```
# load Fort Collins, CO precipitation dataset.
data(FtCoPrec)

# Perform a simple point process model fit.
x <- FtCoPrec[, "Prec"]
fit <- fpp( x, 0.395)
pp.diag( fit)

# Add seasonal covariates.
Time <- FtCoPrec[, "obs"]
angle <- (2*pi*Time)/365.25
s <- cbind( sin( angle), cos( angle))
fit <- fpp( xdat=x, threshold=0.395, npy=365.25, ydat=s, mul=1:2, sigl=1:2, siglink=exp)
pp.diag( fit)
```

---

ftcanmax

*Fort Collins, Colorado annual maximum precipitation.*


---

**Description**

Annual maximum precipitation (inches) for one rain gauge in Fort Collins, Colorado from 1900 through 1999. See Katz et al. (2002) Sec. 2.3.1 for more information and analyses.

**Usage**

```
data(ftcanmax)
```

**Format**

A data frame with 100 observations on the following 2 variables.

**Year** a numeric vector giving the Year.

**Prec** a numeric vector giving the annual maximum precipitation amount in inches.

**Source**

Colorado Climate Center, Colorado State University (<http://ulysses.atmos.colostate.edu>). The annual maximum precipitation data is taken from daily precipitation data.

**References**

Gilleland, Eric and Katz, Richard W. Tutorial for the 'Extremes Toolkit: Weather and Climate Applications of Extreme Value Statistics.' <http://www.assessment.ucar.edu/toolkit>, 2005.

Katz, Richard W., Parlange, Marc B. and Naveau, Philippe. Statistics of extremes in hydrology. *Advances in Water Resources*, 25:1287–1304, 2002.

**Examples**

```
data(ftcanmax)
str(ftcanmax)
plot(ftcanmax, type="l", lwd=2)

# See Gilleland et al. (2005) for more examples using these data with extRemes.
```

---

gen.gev

---

*Simulate data from a generalized extreme value (GEV)*


---

**Description**

Generates data from a GEV (GPD) using 'runif' (and 'rexp') function. May also incorporate a linear trend in location parameter of GEV.

**Usage**

```
gen.gev(p, n, trend = NULL)
gen.gpd(n, sigma, xi, u)
```

**Arguments**

p	A $1 \times 3$ vector indicating the mean, scale and shape of the GEV, respectively.
n	The sample size to generate.
trend	Slope of the location parameter trend (if desired).
sigma	Scale parameter of GPD.
xi	Shape parameter of GPD.
u	Threshold for GPD.

**Details**

Value returned (with no trend) is derived from the following formula (GEV).  $\mu + \sigma \frac{X^{-\xi-1}}{\xi}$ , where  $X$  is a uniform random variable.

For GPD the formula is:  $\frac{\sigma}{\xi} \cdot ((1 - \text{runif}(n))^{-\xi-1})$  for  $\xi \neq 0$  and  $\text{rexp}(n, \text{rate} = \frac{1}{\sigma})$  for  $\xi = 0$ .

**Value**

Returns a vector of simulated data.

**Author(s)**

Functions written by Eric Gilleland and Greg Young.

**References**

Coles, Stuart. "An introduction to statistical modeling of extreme values", Springer-Verlag (London), 2001.

Gilleland, Eric and Katz, Richard W. Tutorial for the 'Extremes Toolkit: Weather and Climate Applications of Extreme Value Statistics.' <http://www.assessment.ucar.edu/toolkit>, 2005.

**See Also**

From ismev package: `gev.fit`, `gev.diag`, `gpd.fit`, `gpd.diag`

**Examples**

```
# obtain a GEV with mean, 4, scale 1.5 and shape of -0.1
mu <- 4 # location parameter
sigma <- 1.5 # scale parameter
xi <- -0.1 # shape parameter

params <- c(mu, sigma, xi)

# generate a sample of size 25
gen1 <- gen.gev(p=params, n=25)

# Now generate one with a trend.
gen2 <- gen.gev(p=params, n=25, trend=0.1)

# Fit 'gen1' to a GEV distribution and plot the diagnostics.
gen1.fit <- gev.fit(gen1)
class(gen1.fit) <- "gev.fit"
plot(gen1.fit)

# Fit 'gen2' to a GEV distribution and plot the diagnostics.
gen2.fit1 <- gev.fit(gen2)
class(gen2.fit1) <- "gev.fit"
plot(gen2.fit1)
```

---

gev.parameterCI      *Return level and shape parameter confidence intervals for GEV (or GP) distribution.*

---

### Description

Computes confidence intervals for return levels and/or shape parameters of GEV (or GP) using the profile likelihood approach.

### Usage

```
gev.parameterCI(z, m, rl.xlow, rl.xup, xi.xlow=NULL, xi.xup=NULL, conf = 0.95,
               nint = 100, rl.only=FALSE, xi.only=FALSE, make.plot=FALSE)

gpd.parameterCI(z, m, conf = 0.95, nint = 100, rl.xup=NULL, rl.xlow=NULL,
               xi.xup=NULL, xi.xlow=NULL, rl.only=FALSE,
               xi.only=FALSE, make.plot=FALSE)
```

### Arguments

z	object from <code>gev.fit</code> or <code>gpd.fit</code> functions of the <code>ismev</code> package.
m	m-year return level for which to calculate confidence interval.
rl.xlow	initial guess of lower limit for return level (generally should be lower than actual limit). If <code>NULL</code> , this function will make its own guess—it is recommended to plot the profile likelihood when finding these limits automatically ( <code>make.plot=TRUE</code> ).
rl.xup	initial guess of upper limit for return level (generally should be higher than actual limit). If <code>NULL</code> , this function will make its own guess—it is recommended to plot the profile likelihood when finding these limits automatically ( <code>make.plot=TRUE</code> ).
xi.xlow	analogous to <code>rl.xlow</code> , but for shape parameter.
xi.xup	analogous to <code>rl.xup</code> , but for shape parameter.
conf	desired confidence.
nint	number of values (ret. level and/or shape parameter) to compute in trying to find confidence bounds.
rl.only	logical, if <code>TRUE</code> calculate only the return level confidence intervals.
xi.only	logical, if <code>TRUE</code> calculate only the shape parameter confidence intervals.
make.plot	logical, if <code>TRUE</code> plots profile likelihoods.

### Details

This is a fairly rough routine, and is designed mainly to work internally with the `extRemes` GUI dialogs. However, it can be used externally with care. It makes use of a bisection search and spline fitting in order to find where the profile likelihood crosses the horizontal line through `c`; the maximum log-likelihood less the associated chi-square quantile.

**Value**

A list with components:

<code>upcross.level</code>	the maximum log-likelihood value less the associated chi-square quantile.
<code>rl</code>	a list object containing the return level characteristics. Including: the maximum likelihood estimate for the m-year return level (mle), the spline function used to estimate the profile likelihood (sfun), and the upper (up) and lower (dn) confidence limits.
<code>xi</code>	Shape parameter characteristics—analogue to <code>rl</code> , but no mle component.

**Author(s)**

Eric Gilleland

**References**

Coles, Stuart (2001). An Introduction to Statistical Modeling of Extreme Values. Springer-Verlag, London.

Gilleland, Eric and Katz, Richard W. Tutorial for the 'Extremes Toolkit: Weather and Climate Applications of Extreme Value Statistics.' <http://www.assessment.ucar.edu/toolkit>, 2005.

**See Also**

`gev.fit`, `gpd.fit`, `gev.diag`, `gpd.diag`, `gev.prof`, `gev.profxi`, `gpd.prof`, `gpd.profxi`

**Examples**

```
# See extRemes tutorial (Gilleland et al. (2005)) for examples using the GUI interface.
```

---

<code>return.level</code>	<i>Computes m-year return levels for GEV and GPD with confidence bounds.</i>
---------------------------	------------------------------------------------------------------------------

---

**Description**

Computes confidence limits for m-year return levels for GEV and GPD using profile likelihoods and a cubic spline ('splinefun'). It also computes the confidence levels using the "delta" method.

**Usage**

```
return.level(z, conf = 0.05, rperiods= c(10,100,210,510,810,980), make.plot = TRUE)
```

**Arguments**

<code>z</code>	An object of class "gev.fit" or "gpd.fit" from the <code>gev.fit</code> or <code>gpd.fit</code> functions from the <code>ismev</code> library of Stuart Coles (ported to R by Alec Stephenson).
<code>conf</code>	The confidence level for confidence bounds.
<code>rperiods</code>	Return periods at which to compute confidence limits with profile likelihood plots—the rest will be interpolated with a spline function using the R function <code>splinefun</code>
<code>make.plot</code>	logical, if 'TRUE' plots the return levels with confidence bounds.

**Details**

This function is in part a modification of the functions 'gev.rl' and 'gpd.rl' from the `ismev` library. It allows for the plotted values to be returned (invisibly) as well as other confidence levels besides 97.5%.

Confidence levels are computed using both the profile likelihood method (for return levels given by 'rlevels') with a cubic spline (to interpolate for many values not given by 'rlevels') and by the delta method described in Coles (2001).

**Value**

Creates a plot if 'make.plot' is TRUE and returns a list invisibly with components.

<code>return.level</code>	The m-year return levels.
<code>return.period</code>	The return periods, m.
<code>confidence</code>	The confidence bounds estimated from the profile likelihood.
<code>confidence.delta</code>	The confidence bounds computed by the delta method.
<code>conf.level</code>	The confidence level.

**Author(s)**

Eric Gilleland

**References**

Coles, Stuart. "An Introduction to Statistical Modeling of Extreme Values", Springer 2001. ISBN: 1852334592

Gilleland, Eric and Katz, Richard W. Tutorial for the 'Extremes Toolkit: Weather and Climate Applications of Extreme Value Statistics.' <http://www.assessment.ucar.edu/toolkit>, 2005.

**See Also**

From the `ismev` package: `gev.fit`, `gpd.fit`, `gev.diag`, `gpd.diag`, `gev.rl`, `gpd.rl`

**Examples**

```
# Must have the 'ismev' package loaded.
require( ismev)
data( ftcanmax)
fit <- gev.fit( ftcanmax[, "Prec"])
class( fit) <- "gev.fit" # 'gev.fit' does not actually assign this class,
                        # but it must be done.

return.level( fit)
```

---

rlplot

---

*Create a return level plot for a fitted object of an extreme-value distribution.*


---

**Description**

Plots several return levels against the return period for a fitted object from one of the **ismev** functions: `gev.fit` and `gpd.fit`.

**Usage**

```
rlplot(z, ci = 0.05, add.ci = FALSE)
```

**Arguments**

<code>z</code>	A list object as returned by one of <code>gev.fit</code> or <code>gpd.fit</code> (with appropriate class attribute added).
<code>ci</code>	The $(1-ci)*100$ confidence value.
<code>add.ci</code>	logical if true will add confidence bounds to plot.

**Details**

Given a fitted list object from `gev.fit` or `gpd.fit`—attributed with the class "gev.fit" or "gpd.fit", respectively—the return level plot is generated. Confidence bounds, if included, are found by the delta method, which is generally appropriate for shorter return periods, but not for longer return periods because the return level distribution is generally skewed. Therefore, if a plot with better estimates of the confidence bounds are desired, use `add.ci=FALSE`, and use the R function `lines` to add different bounds (e.g., using values obtained from the `gev.parameterCI` or `gpd.parameterCI` functions).

This function is simply a modification of Stuart Coles' functions `gpd.rl` and `gev.rl` (Coles, 2001).

**Value**

A plot is created. If assigned to an object, a list will be returned with the following items.

<code>period</code>	The return periods used for calculating the return levels.
<code>level</code>	The estimated return level for each return period.

lower	If add.ci is TRUE, then this is a vector of lower 1-ci confidence bounds. Otherwise the value is NULL.
upper	If add.ci is TRUE, then this is a vector of upper 1-ci confidence bounds. Otherwise the value is NULL.

**Author(s)**

Eric Gilleland

**References**

Coles, Stuart. "An introduction to statistical modeling of extreme values", Springer-Verlag (London), 2001.

Gilleland, Eric and Katz, Richard W. Tutorial for the 'Extremes Toolkit: Weather and Climate Applications of Extreme Value Statistics.' <http://www.assessment.ucar.edu/toolkit>, 2005.

**See Also**

[gev.parameterCI](#), [gpd.parameterCI](#), [gev.diag](#), [gpd.diag](#)

**Examples**

```
data(ftcanmax)
fit <- gev.fit(ftcanmax[, "Prec"])
class(fit) <- "gev.fit"
rlplot(fit)
```

# Index

## \*Topic **datasets**

- damage, 19
- Denmint, 1
- Denversp, 2
- Flood, 3
- ftcanmax, 33
- FtCoPrec, 4
- HEAT, 5
- Ozone4H, 6
- Peak, 9
- PORTw, 7
- Potomac, 10
- Rsum, 11
- Tphap, 12

## \*Topic **distribution**

- gen.gev, 34

## \*Topic **hplot**

- gev.parameterCI, 36
- return.level, 37
- rlplot, 39

## \*Topic **internal**

- extRemes internal, 25

## \*Topic **manip**

- boot.matrix, 15
- dclust, 20
- gev.parameterCI, 36

## \*Topic **math**

- bisearch, 14

## \*Topic **misc**

- as.extRemesDataObject, 13
- boot.sequence, 17
- clearlog, 18
- decluster.runs, 21
- eiAnalyze, 23
- exi.intervals, 24
- extRemes, 27
- extRemes internal, 25
- extremes.gui, 30
- fpp, 31

## \*Topic **univar**

- extremalindex, 29

- affine.gui (*extRemes internal*), 25
- as.extRemesDataObject, 13

- bisearch, 14
- boot.matrix, 15, 17, 24
- boot.sequence, 16, 17, 24

- clearlog, 18

- damage, 19

- DataSummaryGUI (*extRemes internal*), 25

- dclust, 20

- decluster.gui (*extRemes internal*), 25

- decluster.intervals, 16, 24, 25

- decluster.intervals  
(*decluster.runs*), 21

- decluster.runs, 16, 21

- Denmint, 1

- Denversp, 2

- describe2 (*extRemes internal*), 25

- deviancestat (*extRemes internal*),  
25

- diagnostic.plots (*extRemes internal*), 25

- eiAnalyze, 23

- exi.intervals, 23, 24, 24

- extremalind.gui (*extRemes internal*), 25

- extremalindex, 29

- extRemes, 27

- extRemes internal, 25

- extremes.gui, 14, 30

- fitdiag.gui (*extRemes internal*),  
25

- fitsummary.gui(*extRemes internal*), 25
- Flood, 3
- fpp, 31
- ftcanmax, 33
- FtCoPrec, 4
- gen.gev, 34
- gen.gpd(*gen.gev*), 34
- gev.parameterCI, 15, 36, 40
- gev.ret(*extRemes internal*), 25
- gev.rlci(*extRemes internal*), 25
- gevf.gui(*extRemes internal*), 25
- gevparamCI.gui(*extRemes internal*), 25
- gevsim.gui(*extRemes internal*), 25
- gpd.parameterCI, 15, 40
- gpd.parameterCI(*gev.parameterCI*), 36
- gpd.ret(*extRemes internal*), 25
- gpd.rl2(*extRemes internal*), 25
- gpd.rlci(*extRemes internal*), 25
- gpdf.gui(*extRemes internal*), 25
- gpdfitrage.gui(*extRemes internal*), 25
- gpdparamCI.gui(*extRemes internal*), 25
- gpdsim.gui(*extRemes internal*), 25
- HEAT, 5
- hist.gev.fit(*extRemes internal*), 25
- hist.gpd.fit(*extRemes internal*), 25
- hist.pp.fit(*extRemes internal*), 25
- histogram.gui(*extRemes internal*), 25
- indicatorTransform.gui(*extRemes internal*), 25
- llhrt.gui(*extRemes internal*), 25
- load.data(*extRemes internal*), 25
- logdr.gui(*extRemes internal*), 25
- logtrans.gui(*extRemes internal*), 25
- mrlplot.gui(*extRemes internal*), 25
- negtrans.gui(*extRemes internal*), 25
- optim, 33
- Ozone4H, 6
- Peak, 9
- plot.gev.fit(*extRemes internal*), 25
- plot.gpd.fit(*extRemes internal*), 25
- plot.gum.fit(*extRemes internal*), 25
- plot.pp.fit(*extRemes internal*), 25
- plot.rlarg.fit(*extRemes internal*), 25
- poisson.gui(*extRemes internal*), 25
- PORTw, 7
- Potomac, 10
- ppfit.gui(*extRemes internal*), 25
- ppfitrange.gui(*extRemes internal*), 25
- return.level, 37
- rl.gev.fit(*extRemes internal*), 25
- rl.gpd.fit(*extRemes internal*), 25
- rlargf.gui(*extRemes internal*), 25
- rlplot, 39
- rlplot.gui(*extRemes internal*), 25
- Rsum, 11
- scatterplot.gui(*extRemes internal*), 25
- scrubber.gui(*extRemes internal*), 25
- SEPTsp(*PORTw*), 7
- stats2(*extRemes internal*), 25
- summary.gev.fit(*extRemes internal*), 25
- summary.gpd.fit(*extRemes internal*), 25
- summary.pp.fit(*extRemes internal*), 25
- summary.rlarg.fit(*extRemes internal*), 25
- Tphap, 12

`trigtrans.gui(extRemes  
internal)`, 25

`view.data(extRemes internal)`, 25

`write.extRemesMainMessage  
(extRemes internal)`, 25

`zzz(extRemes internal)`, 25