Package ‘swfscMisc’

May 9, 2026

Type Package

Title Miscellaneous Functions for Southwest Fisheries Science Center

Description Collection of conversion, analytical, geodesic, mapping, and plotting functions. Used to support packages and code written by researchers at the Southwest Fisheries Science Center of the National Oceanic and Atmospheric Administration.

Version 1.7

URL `https://github.com/EricArcher/swfscMisc`

BugReports `https://github.com/EricArcher/swfscMisc/issues`

Depends R (>= 4.3.0)

Imports abind, dplyr, ggplot2, ggrepel, HDInterval, kknn, methods, modeest, tibble, rlang, sf, spatstat.geom, stats, tidyr

Suggests coda, runjags, rstan

License GPL (>= 2)

Encoding UTF-8

RoxygenNote 7.3.2

NeedsCompilation no

Author Eric Archer [aut, cre]

Maintainer Eric Archer <`eric.ivan.archer@gmail.com`>

Repository CRAN

Date/Publication 2025-06-04 08:00:02 UTC

Contents

`affin.prop` *Affinity Propagation*

Description

Runs the Affinity Propagation clustering algorithm of Frey and Dueck, 2007.

Usage

```
affin.prop(
  sim.mat,
  num.iter = 100,
  stable.iter = 10,
  shared.pref = "min",
  lambda = 0.5
)
```

Arguments

`sim.mat`	a similarity matrix between individuals to be clustered.
`num.iter`	maximum number of iterations to attempt.
`stable.iter`	number of sequential iterations for which consistent clustering is considered acceptable.
`shared.pref`	type of shared preference to use. Can be one of "min", "median", or a numeric value.
`lambda`	damping factor.

Value

A matrix with one row per sample in 'sim.mat' and one column for each iteration. Values in columns indicate cluster assignment (arbitrary numbers) for each sample.

Author(s)

Eric Archer <`eric.archer@noaa.gov`>

References

Frey, B.J., and D. Dueck. 2007. Clustering by passing messages between data points. Science 315:972-976

Examples

```
data(iris)

# Take 75 random iris rows for example
iris <- iris[sample(1:nrow(iris), 75), ]
iris <- droplevels(iris)

iris.sim <- -dist(iris[, -5])

iris.affin <- affin.prop(iris.sim, stable.iter = 5)
table(iris$Species, iris.affin[, ncol(iris.affin)])
```

`autoUnits` *Auto Time Interval Units*

Description

Convert time interval units to natural values based on magnitude of difference.

Usage

```
autoUnits(x)
```

Arguments

`x`	an object inheriting from class `difftime`

Author(s)

Eric Archer <`eric.archer@noaa.gov`>

Examples

```
autoUnits(as.difftime("0:3:35"))
autoUnits(as.difftime("15:3:35"))
autoUnits(ISOdate(2000, 5, 1) - ISOdate(2000, 4, 20))
```

`bearing` *Calculate Bearing Between Two Positions*

Description

Calculates the bearing between two points, given each point's latitude and longitude coordinates

Usage

```
bearing(lat1, lon1, lat2, lon2)
```

Arguments

`lat1, lon1`	numeric. The latitude and longitude of the starting coordinate in decimal degrees.
`lat2, lon2`	numeric. The latitude and longitude of the ending coordinate in decimal degrees.

Value

vector with initial and final bearings.

Author(s)

Eric Archer <`eric.archer@noaa.gov`>

Examples

```
# What is the bearing from San Diego, CA to Honolulu, HI?
bearing(32.87, -117.25, 21.35, -157.98)
```

`betaParams` *Calculate Beta parameters*

Description

Calculate Beta shape parameters and variance from mode and concentration.

Usage

```
betaParams(w, k)
```

Arguments

`w`	mode
`k`	concentration

`box.area` *Area of a Box*

Description

Calculate the area of a square on the earth.

Usage

```
box.area(lat, lon, edge, units = "nm")
```

Arguments

`lat`, `lon`	The latitude and longitude of the lower right corner of the box in decimal degrees.
`edge`	The length of one side of the square in decimal degrees.
`units`	units of distance. Can be "km" (kilometers), "nm" (nautical miles), or "mi" (statute miles).

Author(s)

Eric Archer <`eric.archer@noaa.gov`>

Examples

```
#What is the area of a 5 degree grid off of San Diego, CA?
box.area(32.87, -117.25, edge = 1, units = "nm")
box.area(32.87, -117.25, edge = 1, units = "km")
box.area(32.87, -117.25, edge = 1, units = "mi")
```

`braces` *Braces*

Description

Adds curly braces to a plot.

Usage

```
braces(
  xfrom,
  xto,
  yfrom,
  yto,
  radius = 1,
  col = par("fg"),
  lty = par("lty"),
  lwd = par("lwd")
)
```

Arguments

`xfrom`, `xto`, `yfrom`, `yto`	start and end points of braces. Direction of brace determined by from and to arguments.
`radius`	radius of curve in brace.
`col`, `lty`, `lwd`	color, line type, and line width of braces. See `par` for more details.

Note

Orientation of brace is either horizontal or vertical, with axis along largest range of x or y in plotting units.

Author(s)

Tim Gerrodette <`tim.gerrodette@noaa.gov`>

Examples

```
plot(x = c(0, 1), y = c(0, 1000), type = "n", xlab= "", ylab = "")
braces(xfrom = 0.2, xto = 0.8, yfrom = c(400, 600), yto = c(300, 700))
plot(x = c(0, 100), y = c(0, 17), type = "n", xlab = "x", ylab = "y")
text(10, 16, "radius =")
for (i in 1:8) {
  braces(xfrom = 10 * i + 10, xto = 10 * i + 18, yfrom = 1,
         yto = 15, radius = i / 4, lwd = 2)
  text(10 * i + 12, 16, round(i / 4, 2))
}
plot(c(0, 100), c(0, 17), type = "n", xlab = "x", ylab = "y")
braces(30, 80, 13, 11, 1)

plot(c(0, 100), c(0, 17), type = "n", xlab = "x", ylab = "y")
braces(c(20, 80, 30), c(10,75,40), 1, 15, radius = c(0.2, 0.5, 0.1),
       lwd = c(1, 2, 3), col = 1:2, lty = 1)

plot(c(0, 100), c(0, 17), type = "n")
braces(20, 80, 7, 5, 1)
braces(20, 80, 13, 15, 1)
```

`catSpatInterp` *Categorical Spatial Interpolation*

Description

Create a raster of probability of categorical values interpolated across a 2-dimensional space given a set of points where each is assigned to one of several classes.

Usage

```
catSpatInterp(
  data,
  x.col = "x",
  y.col = "y",
  group.col = "group",
  num.grid = 100,
  knn = 10,
  hull.buffer = 0.1,
  num.cores = 1,
  num.batches = NULL
)
```

Arguments

`data`	matrix or data.frame containing points and grouping designation.
`x.col`, `y.col`, `group.col`	numbers or characters identifying which columns in `data` are the x and y values and grouping designation.
`num.grid`	number of grid cells for k-nearest neighbor interpolation.
`knn`	number of nearest neighbors to consider for interpolation.
`hull.buffer`	percent increase of convex hull to use as spatial area to interpolate over.
`num.cores`	number of cores to distribute interpolations over.
`num.batches`	number of batches to divide grid cell interpolations into.

Value

A list containing a raster and points of buffered convex hull.

Author(s)

Eric Archer `<eric.archer@noaa.gov>`

References

Adapted from code originally presented in a blog post on Categorical Spatial Interpolation by Timo Grossenbacher https://timogrossenbacher.ch/2018/03/categorical-spatial-interpolation-with-r/

Examples

```
## Not run:
iris.mds <- stats::cmdscale(dist(iris[, 1:4]), k = 2)
mds.df <- setNames(
  cbind(iris.mds, data.frame(iris$Species)),
  c("dim1", "dim2", "Species")
)

result <- catSpatInterp(
  mds.df, x.col = "dim1", y.col = "dim2", group.col = "Species",
  num.grid = 300, knn = 20, hull.buffer = 0.05,
  num.cores = 5, num.batches = NULL
)

library(ggplot2)
ggplot(mapping = aes(dim1, dim2)) +
  geom_raster(
    aes(fill = Species, alpha = prob),
    data = result$raster
  ) +
  geom_polygon(data = result$hull.poly, fill = NA, col = "black") +
  geom_hline(yintercept = 0, col = "white") +
  geom_vline(xintercept = 0, col = "white") +
  geom_point(
    aes(fill = Species),
    data = mds.df,
    col = "black",
    shape = 21,
    size = 4
  ) +
  theme(
    axis.ticks = element_blank(),
    axis.text = element_blank(),
    axis.title = element_blank(),
    legend.position = "top",
    panel.grid = element_blank(),
    panel.background = element_blank()
  )

## End(Not run)
```

`central.quantile` *Central Quantile*

Description

Upper and lower values of central quantile

Usage

```
central.quantile(x, pct = 0.95)
```

Arguments

`x`	numeric vector.
`pct`	central percentile desired.

Value

a two element vector giving the lower and upper quantiles.

Author(s)

Eric Archer <`eric.archer@noaa.gov`>

Examples

```
x <- runif(1000)
central.quantile(x)
central.quantile(x, pct = 0.75)
```

`circle.polygon` *Circle Polygon (on Earth)*

Description

Creates a circular polygon (optionally on the earth) centered at a given point with a constant radius.

Usage

```
circle.polygon(
  x,
  y,
  radius,
  brng.limits = 0,
  sides = 1,
  by.length = TRUE,
  units = "nm",
  ellipsoid = datum(),
  dist.method = "lawofcosines",
  destination.type = "ellipsoid",
  poly.type = "cart.earth"
)
```

Arguments

`x, y`	number specifying the coordinates of the center of the circle in decimal degrees. If `poly.type` is "simple.earth" or "complex.earth", this will be longitude and latitude respectively.
`radius`	radius of sphere.
`brng.limits`	number, or vector of two numbers. If one value is given, it is used as the starting bearing in degrees for the first point of the circle. If a vector of two values is given, then they are used as the start and end bearings of arc.
`sides`	number that represents either length of sides or number of sides, as specified by the 'by.length' argument.
`by.length`	logical. If `TRUE`, then `sides` is the length of sides, if `FALSE`, then `sides` is number of sides.
`units`	character for units of distance: Can be "km" (kilometers), "nm" (nautical miles), "mi" (statute miles).
`ellipsoid`	ellipsoid model parameters as returned from a call to `datum`.
`dist.method`	character specifying method for calculating distance for `type` = "cart.earth". See `method` argument of `distance` for more information.
`destination.type`	character specifying type of surface for `type` = "gc.earth". See `type` argument of `destination` for more information.
`poly.type`	character specifying the type of polygon calculation to use. Can be one of "cartesian" using basic cartesian coordinates, "cart.earth" for a simple polygon on the earth's surface treating latitude and longitude as cartesian coordinates, or "gc.earth" for a more precise calculation keeping a constant great-circle radius.

Value

A matrix representing the desired circle polygon centered at lat, lon of radius.

Author(s)

Eric Archer <`eric.archer@noaa.gov`>

Examples

```
cart.earth <- circle.polygon(-117.24, 32.86, 40, poly.type = "cart.earth")

lat.range <- c(32, 34)
lon.range <- c(-118.5, -116)

op <- par(mar = c(3, 5, 5, 5) + 0.1, oma = c(1, 1, 1, 1))

plot.new()
plot.window(xlim = lon.range, ylim = lat.range)
points(-117.24, 32.86, pch = 19, col = "red")
polygon(cart.earth, border = "red", lwd = 3)
lat.lon.axes(n = 3)
```

```
box(lwd = 2)
mtext("poly.type = 'cart.earth'", line = 3)

par(op)
```

`color.name` *Color Name*

Description

Return the name of a color listed given the number.

Usage

```
color.name(i)
```

Arguments

`i`	integer specifying color .

Value

character value of 'i' color.

Author(s)

Eric Archer <`eric.archer@noaa.gov`>

`convert.angle` *Angle Conversion*

Description

Converts angles between radians and degrees.

Usage

```
convert.angle(x, from = c("degrees", "radians"), to = c("radians", "degrees"))
```

Arguments

`x`	numeric. The angle to be converted.
`from, to`	character. Units to convert from and to. Can be "radians" or "degrees" or any partial match (case-sensitive).

Author(s)

Eric Archer <`eric.archer@noaa.gov`>

Examples

```
convert.angle(45, "deg", "rad")
convert.angle(4.5, "r", "d")
```

`convert.distance` *Distance Conversion*

Description

Convert distances between kilometers, nautical miles, and statute miles.

Usage

```
convert.distance(x, from = c("nm", "km", "mi"), to = c("km", "nm", "mi"))
```

Arguments

`x`	numeric. The distance to be converted.
`from`, `to`	character. Units to convert from and to. Can be "km" (kilometers), "nm" (nautical miles), or "mi" (statute miles), or any partial match thereof (case sensitive).

Author(s)

Eric Archer <`eric.archer@noaa.gov`>

`copy.tri` *Copy Matrix Triangles*

Description

Copy between lower left and upper right triangles of a matrix.

Usage

```
copy.tri(x, from = "lower")
```

Arguments

`x`	a matrix.
`from`	triangle to copy from. Can be "lower" or "upper".

Value

a matrix.

Author(s)

Eric Archer `<eric.archer@noaa.gov>`

Examples

```
x <- matrix(1:9, nrow = 3)
print(x)
copy.tri(x)
```

`crossing.point` *Crossing Point*

Description

Return point where two lines cross

Usage

```
crossing.point(l1, l2)
```

Arguments

`l1, l2`	matrices representing two lines, where first two columns are x and y values respectively

Value

a data.frame of x and y values of points where lines cross

Author(s)

Eric Archer `<eric.archer@noaa.gov>`

Examples

```
x <- 1:100
line1 <- cbind(x, 3 + 3 * x)
line2 <- cbind(x, 10 - 3 * x)
plot(line1[, 1], line1[, 2], type = "l", col = "red")
lines(line2[, 1], line2[, 2], col = "blue")
cr.pt <- crossing.point(line1, line2)
print(cr.pt)
```

`datum` *Datum*

Description

Return parameters specifying ellipsoid datum model.

Usage

```
datum(model = c("wgs84", "grs80", "airy", "international", "clarke", "grs67"))
```

Arguments

`model`	character, specifying which model to use for ellipsoid model. Options are: "wgs84", "grs80", "airy", "international", "clarke", "grs67", or partial matches thereof (case-sensitive).

Value

vector of a, b, and f parameters.

Note

Model parameters are based on distances in km.

Author(s)

Eric Archer `<eric.archer@noaa.gov>`

`destination` *Destination on Sphere or Ellipsoid*

Description

Calculates latitude and longitude of the destination along a sphere or ellipsoid.

Usage

```
destination(
  lat,
  lon,
  brng,
  distance,
  units = c("nm", "km", "mi"),
  ellipsoid = datum(),
  radius = convert.distance(6371, "km", "nm"),
  type = c("ellipsoid", "sphere", "vincenty")
)
```

Arguments

`lat, lon`	numeric. The latitude and longitude of the coordinate in decimal degrees.
`brng`	numeric. The bearing, ranging from 0 to 360 degrees.
`distance`	numeric. The distance travelled, in units specified by `units`.
`units`	units of distance. Can be "km" (kilometers), "nm" (nautical miles), or "mi" (statute miles), or any partial match thereof (case sensitive).
`ellipsoid`	ellipsoid model parameters as returned from a call to `datum`.
`radius`	numeric. Define the radius for `type` = "sphere". In units of `units`.
`type`	Character defining type of surface. Can be "sphere", "ellipsoid", "vincenty", or partial match thereof (case-sensitive).

Value

latitude and longitude of destination.

Author(s)

Eric Archer `<eric.archer@noaa.gov>`

References

Ellipsoid code adapted from JavaScript by Larry Bogan.
Vincenty code adapted from JavaScript by Chris Veness.
Vincenty, T. 1975. Direct and inverse solutions of geodesics on the ellipsoid with application of nested equations. Survey Review 22(176):88-93.

Examples

```
destination(32.87, -117.25, 262, 4174, units = "km", type = "sphere")
destination(32.87, -117.25, 262, 4174, units = "km", type = "ellipsoid")
destination(32.87, -117.25, 262, 4174, units = "km", type = "vincenty")
```

`distance` *Distance Between Coordinates*

Description

Calculates the distance between two coordinates using the Law of Cosines, Haversine, or Vincenty methods.

Usage

```
distance(
  lat1,
  lon1,
  lat2,
  lon2,
  radius = convert.distance(6371, "km", "nm"),
  units = c("nm", "km", "mi"),
  ellipsoid = datum(),
  iter.limit = 20,
  method = c("lawofcosines", "haversine", "vincenty")
)
```

Arguments

`lat1`, `lon1`, `lat2`, `lon2`	The latitude and longitude of the first and second points in decimal degrees.
`radius`	radius of sphere.
`units`	units of distance. Can be "km" (kilometers), "nm" (nautical miles), or "mi" (statute miles), or any partial match thereof (case sensitive).
`ellipsoid`	ellipsoid model parameters as returned from a call to `datum`.
`iter.limit`	An integer value defining the limit of iterations for Vincenty method.
`method`	Character defining the distance method to use. Can be "lawofcosines", "haversine", "vincenty", or any partial match thereof (case sensitive).

Author(s)

Eric Archer <`eric.archer@noaa.gov`>

References

Code adapted from JavaScript by Chris Veness
Vincenty, T. 1975. Direct and inverse solutions of geodesics on the ellipsoid with application of nested equations. Survey Review 22(176):88-93.

Examples

```
# What is the distance from San Diego, CA to Honolulu, HI?
distance(32.87, -117.25, 21.35, -157.98, method = "lawofcosines")
distance(32.87, -117.25, 21.35, -157.98, method = "haversine")
distance(32.87, -117.25, 21.35, -157.98, method = "vincenty")
```

`distSmry` *Distribution summary*

Description

Summarize a numerical distribution.

Usage

```
distSmry(x, p = 0.95, ...)
```

Arguments

`x`	vector of numerical values.
`p`	percent of distribution to summarized by quantile interval (ci) and highest posterior density interval (hdi).
`...`	arguments passed to `mlv` to estimate the mode if `use.mlv` is `TRUE`.

Author(s)

Eric Archer <`eric.archer@noaa.gov`>

`fisher.p` *Fisher's Method p-value*

Description

Calculate Fisher's method p-value to summarize a vector of p-values based on a chi-squared distribution.

Usage

```
fisher.p(p)
```

Arguments

`p`	vector of p-values.

Author(s)

Eric Archer <`eric.archer@noaa.gov`>

`gammaParams` *Calculate Gamma parameters*

Description

Calculate Gamma rate and shape parameters from mode and variance.

Usage

```
gammaParams(m, v)
```

Arguments

`m`	mode
`v`	variance

`geometric.mean` *Geometric Mean*

Description

Calculates the geometric mean of a vector.

Usage

```
geometric.mean(x, w = NULL, na.rm = FALSE)
```

Arguments

`x`	a numeric vector.
`w`	an optional numerical vector of weights the same length as `x`.
`na.rm`	a logical value indicating whether NA values should be stripped before the computation proceeds.

Author(s)

Eric Archer <`eric.archer@noaa.gov`>

Examples

```
x <- rlnorm(100)
mean(x)
median(x)
geometric.mean(x)
```

`ggBiplot` *ggBiplot*

Description

Plot a biplot of a Principal Components Analysis using ggplot2.

Usage

```
ggBiplot(pca, x = 1, y = 2, mult.fac = 0.8, arrow.size = 1.5, label.size = 6)
```

Arguments

`pca`	result from a call to `princomp`.
`x, y`	the number or column names of the components to plot.
`mult.fac`	multiplier factor for lengths of arrows from `0:1`.
`arrow.size`	thickness of arrow lines.
`label.size`	size of labels.

Value

the ggplot2 object is invisibly returned.

Author(s)

Eric Archer `<eric.archer@noaa.gov>`

Examples

```
pc.cr <- princomp(USArrests, cor = TRUE)
ggBiplot(pc.cr)
```

`harmonic.mean` *Harmonic Mean*

Description

Calculate the harmonic mean of a set of numbers.

Usage

```
harmonic.mean(x, w = NULL, na.rm = FALSE)
```

Arguments

`x`	a numeric vector.
`w`	an optional numerical vector of weights the same length as `x`.
`na.rm`	a logical value indicating whether NA values should be stripped before the computation proceeds.

Note

If zeroes are present in `x`, function will return approximation with a warning. In this case, weights will not be used.

Author(s)

Eric Archer <`eric.archer@noaa.gov`>

Examples

```
x <- rlnorm(100)
mean(x)
median(x)
harmonic.mean(x)
```

`imdo` *Iterative Missing Data Optimization (IMDO)*

Description

Identify optimal combination of variables to minimize number of samples with missing data.

Usage

```
imdo(x, groups = NULL, plot = TRUE)

imdoPlot(opt.smry, equal.axes = FALSE)
```

Arguments

`x`	data.frame or matrix to optimize.
`groups`	vector of groups as long as number of rows in `x`.
`plot`	generate a plot of the optimization results.
`opt.smry`	data.frame of optimization summary results from run of `imdo` in (`$opt.smry` element).
`equal.axes`	show imdo plot with both axes on same scale?

Author(s)

Eric Archer <`eric.archer@noaa.gov`>

`intersectingPoint` *Intersecting Point*

Description

Calculates the perpendicular point and distance to a line for a series of points.

Usage

```
intersectingPoint(pts, p1 = NULL, p2 = NULL, intercept = NULL, slope = NULL)
```

Arguments

`pts`	two element vector or two column matrix of x and y values of points.
`p1, p2`	two element vectors of two points laying on line.
`intercept, slope`	the intercept and slope of the line.

Value

A matrix containing columns giving the x and y values of the intersecting point on the line, and the distance to each point.

Note

The line can be specified by providing either `p1` and `p2` or `intercept` and `slope`. If `intercept` and `slope` are specified, then `p1` and `p2` will be ignored.

Author(s)

Eric Archer `<eric.archer@noaa.gov>`

Examples

```
pts <- cbind(x = runif(5, 0, 10), y = runif(5, 0, 10))

intersectingPoint(pts, p1 = c(-1, -1), p2 = c(60, 60))

intersectingPoint(pts, intercept = 0, slope = 1)
```

`isBetween` *Between*

Description

Is a numeric value between two other values?

Usage

```
isBetween(x, a, b = NULL, include.ends = FALSE, na.convert = TRUE)
```

Arguments

`x`	vector of numeric values to check.
`a, b`	numeric values describing range.
`include.ends`	logical. Should test include `a` and `b`? Is test > and < or >= and <= ?
`na.convert`	logical. If `TRUE` and result of test is `NA` because either `x`, `a`, or `b` is `NA`, return `FALSE`, otherwise return `NA`.

Details

Order of `a` and `b` does not matter. If `b` is `NULL` the range will be taken from values in `a`.

Author(s)

Eric Archer <`eric.archer@noaa.gov`>

`lab.wid` *Label Width*

Description

Calculate width of labels for plots.

Usage

```
lab.wid(labels)
```

Arguments

`labels`	vector of labels to be used on plots

`lat.lon.axes` *Latitude and Longitude axes*

Description

Add latitude and longitude axes to a map.

Usage

```
lat.lon.axes(n = 5, lon.n = n, lat.n = n)
```

Arguments

`n`, `lon.n`, `lat.n`	the number of tick marks desired. Can be specified separately for longitude (`lon.n`) or latitude (`lat.n`). See `pretty` for more details.

Author(s)

Eric Archer <`eric.archer@noaa.gov`>

`mcmc2list` *Convert mcmc.list posterior to list*

Description

Convert ‘mcmc.list‘ posterior to named list of vectors or arrays.

Usage

```
mcmc2list(x, pars, collapse.chains = TRUE)
```

Arguments

`x`	object of class mcmc.list.
`pars`	vector of parameter names to extract.
`collapse.chains`	return array with dimension for each chain?

Note

If `collapse.chains` = `TRUE`, the last dimension of arrays will always be samples from the posterior. If `collapse.chains` = `FALSE`, the last dimension of arrays will be individual chains, and the one prior to that will be samples from the posterior for each chain.

See Also

aperm to transpose the array if necessary. as.data.frame.table to convert arrays to data.frames.

`month2Season` *Convert Months to Seasons*

Description

Convert numeric month to season: Winter = Dec-Feb, Spring = Mar-May, Summer = Jun-Aug, Fall = Sep-Nov

Usage

```
month2Season(x)
```

Arguments

`x`	a vector of months from 1:12

Author(s)

Eric Archer <`eric.archer@noaa.gov`>

Examples

```
months <- sample(1:12, 10, rep = TRUE)
months
month2Season(months)
```

`na.count` *Count NAs*

Description

Counts NAs in an object.

Usage

```
na.count(x)
```

Arguments

`x`	a vector, data.frame, or matrix.

Author(s)

Eric Archer <`eric.archer@noaa.gov`>

Examples

```
x <- sample(c(1:10, NA), 30, replace = TRUE)
na.count(x)
x.df <- do.call(data.frame, lapply(1:4, function(i) sample(c(1:10, NA), 30, replace = TRUE)))
colnames(x.df) <- paste("X", 1:4, sep = "")
na.count(x.df)
```

`odds` *Odds Conversion*

Description

`odds`	converts probability to odds
`logOdds`	converts odds to log-odds
`invOdds`	converts odds to probability
`invLogOdds`	converts log-odds to odds

Usage

```
odds(x)

logOdds(x)

invOdds(x)

invLogOdds(x)
```

Arguments

`x`	a numeric vector of probabilities (0 to 1), odds (0 to Inf), or log.odds (-Inf to Inf).

Author(s)

Eric Archer `<eric.archer@noaa.gov>`

Examples

```
x <- sort(runif(10))
odds.df <- data.frame(x = x, odds = odds(x), logOdds = logOdds(x))
odds.df
invOdds(odds.df$odds)
invLogOdds(odds.df$logOdds)
```

`one.arg` *One Argument*

Description

Does the function have just one argument?

Usage

```
one.arg(f)
```

Arguments

`f`	a function.

Author(s)

Eric Archer <`eric.archer@noaa.gov`>

Examples

```
one.arg(mean)
one.arg(one.arg)
```

`perpDist` *Perpendicular Distance*

Description

Calculate the perpendicular distance of a matrix of points to a line.

Usage

```
perpDist(pts, line)
```

Arguments

`pts`	two column matrix of points.
`line`	either a 2x2 matrix of points defining line or two element vector giving intercept and slope of line.

Author(s)

Eric Archer <`eric.archer@noaa.gov`>

Examples

```
ran.pts <- matrix(runif(10), ncol = 2)
x <- perpDist(ran.pts, c(0, 1))
x

plot.new()
plot.window(xlim = c(0, 1), ylim = c(0, 1), asp = 1)
abline(a = 0, b = 1)
points(ran.pts[, 1], ran.pts[, 2])
segments(ran.pts[, 1], ran.pts[, 2], x[, 1], x[, 2], lty = "dashed")
points(x[, 1], x[, 2], col = "red")
axis(1, pos = 0)
axis(2, pos = 0)
```

`perpPoint` *Perpendicular Point*

Description

Compute the perpendicular point between points and a line specified by an intercept and slope.

Usage

```
perpPoint(pts, line)
```

Arguments

`pts`	two column matrix of points.
`line`	two element vector giving intercept and slope of a line.

Author(s)

Eric Archer <`eric.archer@noaa.gov`>

`plotAssignments` *Plot assignment distributions*

Description

Plot individual assignment probability distributions.

Usage

```
plotAssignments(
  probs,
  orig,
  type = NULL,
  ylab = NULL,
  freq.sep.line = TRUE,
  plot = TRUE
)
```

Arguments

`probs`	matrix or data.frame of individual assignment probabilities. Each column represents probability of assignment to that group and rows sum to one.
`orig`	vector of original group assignments
`type`	either `area` for stacked continuous area plot or `bar` for discrete stacked bar chart. The latter is prefered for small numbers of cases. If not specified, a bar chart will be used if all classes have <= 30 cases.
`ylab`	label for y-axis
`freq.sep.line`	put frequency of original group on second line in facet label? If `FALSE`, labels are single line. If `NULL` frequencies will not be included in labels.
`plot`	display the plot?

Value

the `ggplot` object is invisibly returned.

Author(s)

Eric Archer <`eric.archer@noaa.gov`>

Examples

```
n <- 40
probs <- abs(c(rnorm(n, 80, 10), rnorm(n, 20, 10)))
probs <- (probs - min(probs)) / max(probs)
probs <- cbind(probs, 1 - probs)
colnames(probs) <- NULL
orig <- rep(c("Group.1", "Group.2"), each = n)

plotAssignments(probs, orig)

n <- 15
probs <- abs(c(rnorm(n, 80, 10), rnorm(n, 20, 10)))
probs <- (probs - min(probs)) / max(probs)
probs <- cbind(probs, 1 - probs)
colnames(probs) <- NULL
orig <- rep(c("Group.1", "Group.2"), each = n)
```

```
plotAssignments(probs, orig)
```

`pVal` *Permutation Test P-value*

Description

Calculate the p-value for a permutation test.

Usage

```
pVal(obs, null.dist)
```

Arguments

`obs`	observed value.
`null.dist`	vector of values from permutation null distribution.

Author(s)

Eric Archer `<eric.archer@noaa.gov>`

Examples

```
null.dist <- rnorm(1000)
obs <- rnorm(1, mean = 1)

plot(density(null.dist), xlim = range(c(obs, null.dist)), main = "")
abline(v = obs)
print(obs)
pVal(obs, null.dist)
```

`round` *Rounding Numbers for Data Frames*

Description

Rounds numeric columns in data.frames

Usage

```
## S3 method for class 'data.frame'
ceiling(x)

## S3 method for class 'data.frame'
floor(x)

## S3 method for class 'data.frame'
trunc(x, ...)

## S3 method for class 'data.frame'
round(x, digits = 0)

## S3 method for class 'data.frame'
signif(x, digits = 6)
```

Arguments

`x`	a data.frame with numeric columns.
`...`	arguments to be passed to methods.
`digits`	integer indicating the number of decimal places (`round`) or significant digits (`signif`) to be used. See `round` for more details.

Details

Takes a data.frame and returns a data.frame with the specified function applied to each numeric column.

Author(s)

Eric Archer <`eric.archer@noaa.gov`>

See Also

`Round`

Examples

```
data(mtcars)

round(mtcars, 0)

signif(mtcars, 2)
```

`row.col.page.fit` *Number of Rows and Columns on Page*

Description

Return the number of rows and columns for `n` that best fits on a page of size `width` x `height`.

Usage

```
row.col.page.fit(n, width = 8.5, height = 11)
```

Arguments

`n`	number of items (e.g., plots) to fit on page.
`width`, `height`	dimensions of page.

Value

A vector listing the number of rows and columns to use.

Author(s)

Eric Archer <`eric.archer@noaa.gov`>

Examples

```
# 9 frames on US letter paper
row.col.page.fit(9)

# 9 frames on a square
row.col.page.fit(9, width = 10, height = 10)
```

`runjags2list` *Convert runjags posterior to list*

Description

Convert ‘runjags‘ posterior to named list of vectors or arrays.

Usage

```
runjags2list(x, collapse.chains = TRUE)
```

Arguments

`x`	list of class ‘runjags‘. The output from a call to run.jags.
`collapse.chains`	return array with dimension for each chain?

Note

If `collapse.chains` = `TRUE`, the last dimension of arrays will always be samples from the posterior. If `collapse.chains` = `FALSE`, the last dimension of arrays will be individual chains, and the one prior to that will be samples from the posterior for each chain.

See Also

aperm to transpose the array if necessary. as.data.frame.table to convert arrays to data.frames.

`scatterdens` *Scatter Plot with Density Margins*

Description

Produce a scatter plot with a histogram or density plot in the margins

Usage

```
scatterdens(x, y, dens.frac = 1/5, ...)

scatterhist(x, y, xlab = "", ylab = "", dens.frac = 1/5, ...)
```

Arguments

`x, y`	vectors of points to plot.
`dens.frac`	fraction of screen to be taken up by density plots on margins.
`...`	Arguments to be passed to plot.
`xlab, ylab`	labels for x and y axes.

Author(s)

Eric Archer `<eric.archer@noaa.gov>`

References

Original code by Ken Kleiman

Examples

```
x <- rnorm(100)
y <- rlnorm(100)
op <- par(ask = TRUE)
scatterdens(x, y, xlab = "x", ylab = "y")
par(op)
```

`setupClusters` *Setup Clusters*

Description

Setup parallel clusters for different operating systems.

Usage

```
setupClusters(num.cores = 1, max.cores = NULL)
```

Arguments

`num.cores`	number of cores for multithreading. If `NULL`, the number used is set to the value of `parallel::detectCores() - 1`.
`max.cores`	maximum number of cores to use.

Value

an object of class `c("SOCKcluster", "cluster")`.

Author(s)

Eric Archer `<eric.archer@noaa.gov>`

`sex.symbols` *Sex Symbols*

Description

Plots male and female symbols on current plot.

Usage

```
sex.symbols(x, y, sex = 1, col = par("fg"), lwd = par("lwd"), cex = 1)
```

Arguments

`x, y`	the `x` and `y` coordinates on the current plot.
`sex`	a numeric vector containing the values 1 (male) or 2 (female). If of length one, then value is recycled for all symbols.
`col, lwd, cex`	color, line width, and character expansion for each point. `lwd` and `col` are recycled as necessary to cover all points. See `par` for more details.

Author(s)

Tim Gerrodette <`tim.gerrodette@noaa.gov`>

Examples

```
x <- runif(20, 0, 10)
y <- runif(20, 0, 200)
plot(x, y, type = "n")
sex.symbols(x, y, sex = 1:2, cex = 1.5, lwd = c(1.5, 4), col = c("blue", "red"))
```

`sn.params` *Skew-Normal parameter computation*

Description

Compute parameters and moments of skew normal distribution.

Usage

```
sn.location(mode, scale, shape)

sn.mean(dp)

sn.mode(dp)

sn.variance(scale, shape)

sn.skewness(shape)

sn.delta(shape)

sn.m0(shape)
```

Arguments

`mode`	mode of skew normal distribution.
`scale`	skew normal scale parameter.
`shape`	skew normal shape parameter.
`dp`	3 element vector of (in order) location, scale, and shape parameters.

Value

`sn.location`	location parameter computed from mode, scale, and shape.
`sn.mean`	mean of the skew normal distribution.
`sn.mode`	mode of the skew normal distribution.
`sn.variance`	variance of the skew normal distribution.
`sn.skewness`	skewness of the skew normal distribution.
`sn.delta`	value used in other moment computations.
`sn.m0`	value used in mode computation.

Author(s)

Eric Archer `<eric.archer@noaa.gov>`

References

`https://en.wikipedia.org/wiki/Skew_normal_distribution`

See Also

`sn` package by Adelchi Azzalini for skew normal PDF and CDF functions.
Azzalini, A. with the collaboration of Capitanio, A. (2014). The Skew-Normal and Related Families. Cambridge University Press, IMS Monographs series.

`stan2list` *Convert STAN posterior to list*

Description

Convert ‘stan‘ posterior to named list of vectors or arrays.

Usage

```
stan2list(x, collapse.chains = TRUE)
```

Arguments

`x`	list of class ‘stan‘. The output from a call to stan.
`collapse.chains`	return array with dimension for each chain?

Note

If `collapse.chains` = `TRUE`, the last dimension of arrays will always be samples from the posterior. If `collapse.chains` = `FALSE`, the last dimension of arrays will be individual chains, and the one prior to that will be samples from the posterior for each chain.

See Also

aperm to transpose the array if necessary. as.data.frame.table to convert arrays to data.frames.

`stouffer.p` *Stouffer's Method p-value*

Description

Calculate Fisher's method p-value to summarize a vector of p-values based on a chi-squared distribution.

Usage

```
stouffer.p(p, w = NULL)
```

Arguments

`p`	vector of p-values.
`w`	vector weights.

Author(s)

Eric Archer <`eric.archer@noaa.gov`>

`swfscMisc` `swfscMisc` *package*

Description

SWFSC Miscellaneous Functions

Author(s)

Maintainer: Eric Archer <`eric.ivan.archer@gmail.com`>

See Also

Useful links:

- `https://github.com/EricArcher/swfscMisc`
- Report bugs at `https://github.com/EricArcher/swfscMisc/issues`

`transparent` *Transparent Colors*

Description

Return transparent form of a named color.

Usage

```
transparent(col, percent = 50)
```

Arguments

`col`	vector of colors as name, hexadecimal, or positive integer (see `col2rgb`).
`percent`	percent of transparency (0 = solid, 100 = transparent).

Author(s)

Eric Archer `<eric.archer@noaa.gov>`

Examples

```
pct <- seq(0, 100, by = 10)
plot(pct, pct, bg = transparent("red", pct), pch = 21, cex = 4, xlab = "X", ylab = "Y")
```

`uniform.test` *Uniform Distribution Test*

Description

Tests whether a histogram is significantly different from a uniform distribution.

Usage

```
uniform.test(hist.output, B = NULL)
```

Arguments

`hist.output`	output from a call to `hist`.
`B`	number of replicates for chi-squared permutation.

Value

result of chi-squared test.

Author(s)

Eric Archer <`eric.archer@noaa.gov`>

Examples

```
x.unif <- runif(100)
uniform.test(hist(x.unif), B = 1000)
x.lnorm <- rlnorm(100)
uniform.test(hist(x.lnorm), B = 1000)
```

`weighted.fisher.p` *Weighted Fisher's Method p-value*

Description

Calculate weighted Fisher's method p-value to summarize a vector of p-values based on a chi-squared distribution.

Usage

```
weighted.fisher.p(p, w = NULL)
```

Arguments

`p`	vector of p-values.
`w`	vector weights.

Author(s)

Eric Archer <`eric.archer@noaa.gov`>

`which.nearest` *Which Nearest*

Description

Find values of one vector that are nearest to values in another vector.

Usage

```
which.nearest(x, y)
```

Arguments

`x`	vector of values to be compared against.
`y`	vector of values to examine relative to `x`. May be of length 1. @return For each value in `y`, returns index of value of `x` which is nearest to `y` in absolute value. In the case of ties, the function returns the first index of `x`. If nearest value is `min(x)` or `max(x)`, a warning is issued. `NA`s and `NaN`s in `x` are ignored; `NA`s and `NaN`s in `y` are returned.

Author(s)

Tim Gerrodette <`tim.gerrodette@noaa.gov`>

Examples

```
x <- sort(sample(1:100, 20))
y <- sort(sample(min(x):max(x), 5))
i <- which.nearest(x, y)
x
y
x[i]
```

`zero.pad` *Zero Pad Integers*

Description

Return character representation of integers that are zero-padded to the left so all are the same length.

Usage

```
zero.pad(x)
```

Arguments

`x`	a vector of integers.

Author(s)

Eric Archer <`eric.archer@noaa.gov`>

Examples

```
x <- c(0, 1, 3, 4, 10)
zero.pad(x)
x <- c(x, 11, 12, 100, 1000)
zero.pad(x)
```

Index