Package ‘mulgar’

August 26, 2023

Title  Functions for Pre-Processing Data for Multivariate Data
       Visualisation using Tours

Version  1.0.2

Description  This is a companion to the book Cook, D. and Laa, U. (2023) <https://dicook.github.io/mulgar_book/>
``Interactively exploring high-dimensional data and models in R''.
by Cook and Laa. It contains useful functions for processing data in preparation for
visualising with a tour. There are also several sample data sets.

Depends  R (>= 4.0)
Imports  geozoo, tibble, ggplot2, tidyr, dplyr, purrr, stats, methods
Suggests  tourr, ggdendro, colorspace, mclust, kohonen
License  MIT + file LICENSE
Encoding  UTF-8

LazyData  true
LazyDataCompression  bzip2
RoxygenNote  7.2.3


BugReports  https://github.com/dicook/mulgar/issues

NeedsCompilation  no

Author  Dianne Cook [aut, cre] (<https://orcid.org/0000-0002-3813-7155>),
        Ursula Laa [aut] (<https://orcid.org/0000-0002-0249-6439>)

Maintainer  Dianne Cook <dicook@monash.edu>
Repository  CRAN
Date/Publication  2023-08-25 22:00:02 UTC

R topics documented:

  aflw  .................................................................  2
  box  .................................................................  3
Description

This is data from the 2021 Women’s Australian Football League. These are average player statistics across the season, with game statistics provided by the fitzRoy package. If you are new to the game of AFL, there is a nice explanation on Wikipedia. The primary analysis is to summarise the variation using principal component analysis, which gives information about relationships between the statistics or skills sets common in players. One also might be tempted to cluster the players, but there are no obvious clusters so it could be frustrating. At best one could partition the players into groups, while recognising there are no absolutely distinct and separated groups.

Format

A dataset with 381 rows and 35 columns
**Details**

- `id, given_name, surname, number, position, team` player identification details
- `time_pct, ..., clearances` player statistics for the match

**Examples**

```r
require(dplyr)
data(aflw)
glimpse(aflw)
```

---

**box**

*3D plane in 5D*

---

**Description**

This data is simulated to use for testing. It has three dimensions of variability and two of noise. It is created from a 3 factor model. All variables are linearly associated.

**Format**

A dataset with 200 rows and 5 columns

**Details**

- `x1, x2, x3, x4, x5` five numeric variables

**See Also**

plane

**Examples**

```r
box_pca <- prcomp(box)
ggscree(box_pca)
```
Description

This data was collated by Weihao (Patrick) Li as part of his Honours research at Monash University. It contains fire ignitions as detected from satellite hotspots, and processed using the `spotoroo` package, augmented with measurements on weather, vegetation, proximity to human activity. The cause variable is predicted based on historical fire ignition data collected by County Fire Authority personnel.

Format

A dataset with 1021 rows and 60 columns

Details

- `id, lon, lat, time` unique id, and spatiotemporal information for each fire ignition
- `FOR_CODE, FOR_TYPE, COVER, HEIGHT, FOREST` vegetation variables
- `rf, arf7-arf720` average rainfall, on that day, and over last 7, ..., 720 days
- `se, asese7-ase720` solar exposure, on that day, and over last 7, ..., 720 days
- `maxt, amaxt7-amaxt720` max temperature, on that day, and over last 7, ..., 720 days
- `mint, amint7-amint720` min temperature, on that day, and over last 7, ..., 720 days
- `ws, aws_m0-aws_m24` average wind speed, on that day, and for last 1-24 months
- `dist_road, log_dist_road` distance to nearest road
- `dist_cfa, log_dist_cfa` distance to nearest county fire authority facility
- `dist_camp, log_dist_camp` distance to nearest camp site
- `cause` predicted ignition cause, accident, arson, burning_off, lightning

Examples

```r
require(dplyr)
data(bushfires)
glimpse(bushfires)
```
**Description**

Simulated data with different structure

**Format**

A dataset with differing number of rows and columns

**Details**

$x_1, x_2, ...$ numeric variables

**Examples**

```r
require(ggplot2)
ggplot(c1, aes(x=x1, y=x2)) + geom_point() + theme(aspect.ratio=1)
```

---

**calc_mv_dist**

*Compute Mahalanobis distances between all pairs of observations*

**Description**

For a data matrix, compute the sample variance-covariance, which is used to compute the Mahalanobis distance.

**Usage**

calc_mv_dist(x)

**Arguments**

- **x** multivariate data set

**Details**

This is useful for checking distance arise from a multivariate normal sample.

**Value**

vector of length n
Examples

```r
require(ggplot2)
require(tibble)
data(aflw)
aflw_std <- apply(aflw[,7:35], 2, function(x)
  (x-mean(x, na.rm=TRUE))/
  sd(x, na.rm=TRUE))
d <- calc_mv_dist(aflw_std[,c("goals","behinds",
    "kicks","disposals")])
d <- as_tibble(d, .name_repair="minimal")
ggplot(d, aes(x=value)) + geom_histogram()
```

---

**calc_norm**  
*Calculate the norm of a vector*

**Description**

Returns the square root of the sum of squares of a vector

**Usage**

```r
calc_norm(x)
```

**Arguments**

- **x**  
  numeric vector

**Value**

numeric value

**Examples**

```r
x <- rnorm(5)
calc_norm(x)
```

---

**clusters**  
*Three clusters in 5D*

**Description**

This data is simulated to use for testing. It has three elliptical clusters in mostly variables 2 and 4. They are not equidistant.

**Format**

A dataset with 300 rows and 6 columns
clusters_nonlin

Details

\textbf{x1, x2, x3, x4, x5} five numeric variables
\textbf{cl} class variable

See Also

simple_clusters

Examples

clusters_pca <- prcomp(clusters[,1:5])
ggscreec(clusters_pca)

\begin{verbatim}
clusters_nonlin Four unusually shaped clusters in 4D
\end{verbatim}

Description

This data is simulated to use for testing. It has two small spherical clusters, and a curve cluster and a sine wave cluster.

Format

A dataset with 300 rows and 6 columns

Details

\textbf{x1, x2, x3, x4} five numeric variables

See Also

clusters

Examples

require(ggplot2)
ggplot(clusters_nonlin, aes(x=x1, y=x2)) +
  geom_point() +
  theme(aspect.ratio=1)
### convert_proj_tibble

*This function turns a projection sequence into a tibble*

**Description**

Take an array of a projection sequence, and turn into a tibble with numbered projections.

**Usage**

```r
convert_proj_tibble(t1)
```

**Arguments**

- `t1`: tour projection sequence

**Value**

- `tbl1` tibble

**Examples**

```r
require(tourr)
t1 <- interpolate(save_history(flea[, 1:6], grand_tour(4), max = 2))
tbl1 <- convert_proj_tibble(t1)
```

---

### gen_vc_ellipse

*Generate points on the surface of an ellipse*

**Description**

This function generates points by transforming points on the surface of a sphere.

**Usage**

```r
gen_vc_ellipse(vc, xm = rep(0, ncol(vc)), n = 500)
```

**Arguments**

- `vc`: symmetric square matrix describing the variance-covariance matrix which defines the shape of the ellipse.
- `xm`: center of the ellipse, a vector of length equal to the dimension of `vc`
- `n`: number of points to generate

**Value**

- matrix of size n x p
Examples

```r
require(ggplot2)
require(tibble)
ell2d <- gen_vc_ellipse(vc = matrix(c(4, 2, 2, 6),
                        ncol=2, byrow=TRUE),
xm = c(1,1))
ell2d <- as_tibble(eill2d)
ggplot(eill2d, aes(x = V1, y = V2)) + geom_point() +
theme(aspect.ratio=1)
```

---

**gen_xvar_ellipse**

*Ellipse matching data center and variance*

**Description**

This function generates points on the surface of an ellipse with the same center and variance-covariance of the provided data.

**Usage**

```r
gen_xvar_ellipse(x, n = 100, nstd = 1)
```

**Arguments**

- **x** multivariate data set.
- **n** number of points to generate
- **nstd** scale factor for size of ellipse, in terms of number of standard deviations

**Details**

This is useful for checking the equal variance-covariance assumption from linear discriminant analysis.

**Value**

matrix of size n x p

**Examples**

```r
data(aflw)
aflw_vc <- gen_xvar_ellipse(aflw[,c("goals","behinds",
                           "kicks","disposals")], n=500)
require(ggplot2)
ggplot(aflw_vc, aes(x=goals, y=behinds)) + geom_point() +
theme(aspect.ratio=1)
if (interactive()) {
  require(tourr)
animate_slice(aflw_vc, rescale=TRUE, v_rel=0.02)
```
aflw_all <- rbind(aflw_vc, aflw[,c("goals","behinds", 
"kicks","disposals")])
clrs <- c(rep("orange", 500), rep("black", nrow(aflw)))
animate_xy(aflw_all, col=clrs)
}

---

**ggmcbic**

*Produces an mclust summary plot with ggplot*

**Description**

Takes data returned by `mclustBIC()`, converts to a tibble for plotting.

**Usage**

`ggmcbic(mc, cl = 1:nrow(mc), top = ncol(mc))`

**Arguments**

- `mc` : mclustBIC object
- `cl` : subset of clusters to show
- `top` : number to indicate how many models to show, default "all"

**Value**

`mc_bic` a ggplot object

**Examples**

```r
require(mclust)
data(clusters)
clusters_BIC <- mclustBIC(clusters[,1:5], G=2:6)
ggmcbic(clusters_BIC)
ggmcbic(clusters_BIC, top=4)

data(simple_clusters)
clusters_BIC <- mclustBIC(simple_clusters[,1:2])
ggmcbic(clusters_BIC, cl=2:5, top=3)
```
This function produces a simple scree plot

**Description**

Takes a PCA object returned by prcomp(), extracts the standard deviations of the principal components (PC), and plots these against the PC number. The guidance line assumes that all of the variables have been standardised prior to PCA.

**Usage**

```r
ggscreene(pc, q = 2, guide = TRUE, cumulative = FALSE)
```

**Arguments**

- `pc`: PCA object
- `q`: number of principal components to show, default 2 (you should change)
- `guide`: logical whether to compute and add a typical value of the variance, if the data was full-dimensional
- `cumulative`: logical whether to draw cumulative variance

**Value**

- `scree`: a ggplot object

**Examples**

```r
data(aflw)
aflw_std <- apply(aflw[,7:35], 2, function(x)
  (x-mean(x, na.rm=TRUE))/
  sd(x, na.rm=TRUE))
aflw_pca <- prcomp(aflw_std[,c("goals","behinds","kicks","disposals")])
ggscreene(aflw_pca, q=3)
```

---

**ggslice**

Generate an axis-parallel slice display

**Description**

Following the slice definition available in tourr this function returns a ggplot2 display of a slice defined via the projection onto two of the variables. Note that because the underlying function works with any projection, the axis labels need to be set by the user.
Usage

```
ggslice(data, h, v1 = 1, v2 = 2, center = NULL, col = NULL)
```

Arguments

- `data`: data frame containing only variables used for the display
- `h`: slice thickness
- `v1`: column number of variable mapped to x-axis
- `v2`: column number of variable mapped to y-axis
- `center`: center point vector used for anchoring the slice, if NULL the mean of the data is used
- `col`: grouping vector mapped to color in the display

Value

ggplot2 object showing the sliced data

See Also

`ggslice_projection`

Examples

```
d <- geozoo::sphere.hollow(4, 1000)$points
ggslice(d, 0.3, 1, 2)
ggslice(d, 0.3, 1, 2, center = c(0, 0, 0.7, 0))
```

---

**ggslice_projection**  
*Generate slice display*

Description

Generate slice display

Usage

```
ggslice_projection(data, h, proj, center = NULL, col = NULL)
```

Arguments

- `data`: data frame containing only variables used for the display
- `h`: slice thickness
- `proj`: projection matrix from p to 2 dimensions
- `center`: center point vector used for anchoring the slice, if NULL the mean of the data is used
- `col`: grouping vector mapped to color in the display
**hierfly**

**Value**

ggplot2 object showing the sliced data

**See Also**

ggslice

**Examples**

```r
d <- geozoo::sphere.hollow(4, 1000)$points
ggslice_projection(d, 0.3, tourr::basis_random(4))
ggslice_projection(d, 0.3, tourr::basis_random(4),
  center = c(0.4, 0.4, 0.4, 0.4))
```

---

**hierfly** *Generate a dendrogram to be added to data*

**Description**

Supplements a data set with information needed to draw a dendrogram. Intermediate cluster nodes are added as needed, and positioned at the centroid of the combined clusters. Note that categorical variables need to be factors.

**Usage**

```r
hierfly(data, h = NULL, metric = "euclidean", method = "ward.D2", scale = TRUE)
```

**Arguments**

- `data`: data set
- `h`: an hclust object
- `metric`: distance metric to use, see `dist` for list of possibilities
- `method`: cluster distance measure to use, see `hclust` for details
- `scale`: logical value whether to scale data or not, default TRUE

**Value**

list with data and edges and segments
Examples

```r
data(clusters)
cl_dist <- dist(clusters[,1:5])
cl_hw <- hclust(cl_dist, method="ward.D2")
require(ggdendro)
ggdendrogram(cl_hw, type = "triangle", labels = FALSE)
clusters$clw <- factor(cutree(cl_hw, 3))
cl_hfly <- hierfly(clusters, cl_hw, scale=FALSE)
if (interactive()) {
  require(tourr)
  glyphs <- c(16, 46)
pch <- glyphs[cl_hfly$data$node+1]
  require(colorspace)
  clrs <- heat_hcl(length(unique(cl_hfly$data$clw)))
pcol <- clrs[cl_hfly$data$clw]
  ecol <- clrs[cl_hfly$data$clw[cl_hfly$edges[,1]]]
  animate_xy(cl_hfly$data[,1:5], edges=cl_hfly$edges,
             col=pcol, pch=pch, edges.col=ecol,
             axes="bottomleft")
}
```

---

### mc_ellipse

**Computes the ellipses of an mclust model**

**Description**

Takes data returned by `Mclust()`, extracts parameter estimates, and computes points on ellipses.

**Usage**

```r
mc_ellipse(mc, npts = 100)
```

**Arguments**

- `mc`: Mclust object
- `npts`: Number of points to simulate for each cluster, default 100

**Value**

`mc_ellipses` data frame

**Examples**

```r
require(mclust)
data(simple_clusters)
clusters_mc <- Mclust(simple_clusters[,1:2],
                      G=2,
                      modelname="EEI")
mce <- mc_ellipse(clusters_mc, npts=400)
```
require(ggplot2)
sc <- simple_clusters
sc$cl <- factor(clusters_mc$classification)
ggplot() +
  geom_point(data=sc, aes(x=x1, y=x2, colour=cl)) +
  geom_point(data=mce$ell, aes(x=x1, y=x2, colour=cl), shape=4) +
  geom_point(data=mce$mn, aes(x=x1, y=x2, colour=cl), shape=3) +
  theme(aspect.ratio=1, legend.position="none")

multicluster

Multiple clusters of different sizes, shapes and distance from each other

Description
This data is originally from http://ifs.tuwien.ac.at/dm/download/multiChallenge-matrix.txt, and provided as a challenge for non-linear dimension reduction. It was used as an example in Lee, Laa, Cook (2023) https://doi.org/10.52933/jdssv.v2i3.

Format
A dataset with 400 rows and 11 columns

Details

group  cluster label
x1, ... x10  numeric variables

See Also

clusters

Examples

require(ggplot2)
ggplot(multicluster, aes(x=x1, y=x2)) +
  geom_point() + theme(aspect.ratio=1)
norm_vec

Normalize a vector to have length 1

Description

Returns the normalised vector, where the sum of squares is equal to 1

Usage

norm_vec(x)

Arguments

x numeric vector

Value

d numeric vector

Examples

x <- rnorm(5)
norm_vec(x)

pca_model

Create wire frame of PCA model

Description

This function takes the PCA and produces a wire frame of the PCA to examine with the data in a tour. The purpose is to see how well the variance is explained. The model will be centered at the mean, and extend 3 SDs towards the edge of the data, which is assuming that the data is standardised.

Usage

pca_model(pc, d = 2, s = 1)

Arguments

pc PCA object
d number of dimensions to use, default=2
s scale model, default=1
Value

a list of points and edges

Examples

```r
data(plane)
plane_pca <- prcomp(plane)
plane_m <- pca_model(plane_pca)
plane_m_d <- rbind(plane_m$points, plane)
if (interactive()) {
  require(tourr)
  animate_xy(plane_m_d, edges=plane_m$edges, axes="bottomleft")
}
```

pisa

PISA scores

Description

This is data from the 2018 testing, available from https://webfs.oecd.org/pisa2018/SPSS_STU_QQQ.zip. A subset of the data containing only Australia and Indonesia, and the simulated scores for math, reading and science.

Format

A data set with 26371 rows and 31 columns

Details

CNT  Country (Australia, Indonesia)
PV1MATH-PV10SCIE  simulated scores for math, reading and science

Examples

```r
require(dplyr)
data(pisa)
pisa %>% count(CNT)
```
plane

2D plane in 5D

Description
This data is simulated to use for testing. It has two dimensions of variability and three of noise. It is created from a 2 factor model, where all variables are related.

Format
A data set with 100 rows and 5 columns

Details
x1, x2, x3, x4, x5 five numeric variables

See Also
box

Examples

```r
plane_pca <- prcomp(plane)
ggscreene(plane_pca)
```

plane_nonlin

Non-linear relationship in 5D

Description
This data is simulated to use for testing. It has three dimensions of variability and two of noise. It is created from a 2 factor non-linear model. All variables are associated.

Format
A dataset with 100 rows and 5 columns

Details
x1, x2, x3, x4, x5 five numeric variables

See Also
plane, box
pooled_vc

Examples

```r
plane_nonlin_pca <- prcomp(plane_nonlin)
ggscreene(plane_nonlin_pca)
```

pooled_vc

*Compute pooled variance-covariance matrix*

Description

This function computes the group variance-covariance matrices, and produces a weighted average. It is useful for examining the linear discriminant analysis model.

Usage

```r
pooled_vc(x, cl, prior = rep(1/length(unique(cl)), length(unique(cl))))
```

Arguments

- `x`: multivariate data set, matrix.
- `cl`: class variable
- `prior`: prior probability for each class, must sum to 1, default all equal

Value

matrix

Examples

```r
data(clusters)
pooled_vc(clusters[,1:5], clusters$cl)
```

rmvn

*Generate a sample from a multivariate normal*

Description

This function generates a sample of size n from a multivariate normal distribution

Usage

```r
rmvn(n = 100, p = 5, mn = rep(0, p), vc = diag(rep(1, p)))
```
simple_clusters

Arguments

- `n` number of points to generate
- `p` dimension
- `mn` mean of the distribution, a vector of length equal to the dimension of `vc`
- `vc` symmetric square matrix describing the variance-covariance matrix which defines the shape of the ellipse.

Value

matrix of size n x p

Examples

```r
require(ggplot2)
d <- mulgar::rmvn(n=100, p=2, mn = c(1,1),
                 vc = matrix(c(4, 2, 2, 6),
                            ncol=2, byrow=TRUE))
ggplot(data.frame(d), aes(x = x1, y = x2)) +
       geom_point() + theme(aspect.ratio=1)
```

Description

This data is simulated to use for testing. It has two spherical clusters, and two variables.

Format

A dataset with 137 rows and 3 columns

Details

- `x1, x2` two numeric variables
- `cl` class variable

See Also

clusters

Examples

```r
require(ggplot2)
ggplot(simple_clusters, aes(x=x1, y=x2)) +
       geom_point() + theme(aspect.ratio=1)
```
sketches_test

Images of sketches for testing

Description

This data is a subset of images from https://quickdraw.withgoogle.com. The subset was created using the quickdraw R package at https://huizezhang-sherry.github.io/quickdraw/. It has 6 different groups: banana, boomerang, cactus, flip flops, kangaroo. Each image is 28x28 pixels.

Format

A data frame with 1200 rows and 786 columns.

Details

V1-V784 grey scale 0-255

word all NA, you need to predict this

id unique id for each sketch

See Also

sketches_train

Examples

```
require(ggplot2)
data("sketches_test")
x <- sketches_test[sample(1:nrow(sketches_test), 1), ]
xm <- data.frame(gry=t(as.matrix(x[,1:784])),
    x=rep(1:28, 28),
    y=rep(28:1, rep(28, 28))
ggplot(xm, aes(x=x, y=y, fill=gry)) +
geom_tile() +
  scale_fill_gradientn(colors = gray.colors(256, start = 0, end = 1, rev = TRUE )) +
  theme_void() + theme(legend.position="none")
```

sketches_train

Images of sketches for training

Description

This data is a subset of images from https://quickdraw.withgoogle.com. The subset was created using the quickdraw R package at https://huizezhang-sherry.github.io/quickdraw/. It has 6 different groups: banana, boomerang, cactus, flip flops, kangaroo. Each image is 28x28 pixels. This data would be used to train a classification model.
Format

A data frame with 5998 rows and 786 columns

Details

**V1-V784** grey scale 0-255
**word** what the person was asked to draw
**id** unique id for each sketch

Examples

```r
require(ggplot2)
data("sketches_train")
x <- sketches_train[sample(1:nrow(sketches_train), 1), ]
# print(x$word)
xm <- data.frame(gry=t(as.matrix(x[,1:784]))),
  x=rep(1:28, 28),
  y=rep(28:1, rep(28, 28)))
ggplot(xm, aes(x=x, y=y, fill=gry)) +
  geom_tile() +
  scale_fill_gradientn(colors = gray.colors(256, start = 0, end = 1, rev = TRUE )) +
  theme_void() + theme(legend.position="none")
```

---

**som_model**

*Process the output from SOM to display the map and data*

Description

This function generates a grid of points to match the nodes from the self-organising map (SOM), and jitters points from the data so they can be seen relative to the grid. This allows the clustering of points by SOM to be inspected.

Usage

```
som_model(x_som, j_val = 0.5)
```

Arguments

- **x_som** object returned by `kohonen::som`
- **j_val** amount of jitter, should range from 0-1, default 0.3
som_model

Value

• data this object contains
  – original variables from the data
  – map1, map2 location of observations in 2D som map, jittered
  – distance distances between observations and the closest node
  – id row id of data

• net this object contains
  – values of the nodes in the high-d space
  – map1, map2 nodes of the som net
  – distance distances between observations and the closest node
  – id row id of net

• edges from, to specifying row ids of net to connect with lines
• edges_s x, xend, y, yend for segments to draw lines to form 2D map

Examples

require(kohonen)
data(clusters)
c_grid <- kohonen::somgrid(xdim = 5, ydim = 5,
  topo = 'rectangular')
c_som <- kohonen::som(as.matrix(clusters[,1:5]), grid = c_grid)
c_data_net <- som_model(c_som)
require(ggplot2)
ggplot() +
  geom_segment(data=c_data_net$edges_s,
    aes(x=x, xend=xend, y=y, yend=yend)) +
  geom_point(data=c_data_net$data, aes(x=map1, y=map2),
    colour="orange", size=2, alpha=0.5)
Index

* cluster
  hierfly, 13

* datasets
  aflw, 2
  box, 3
  bushfires, 4
  c1, 5
  clusters, 6
  clusters_nonlin, 7
  multicluster, 15
  pisa, 17
  plane, 18
  plane_nonlin, 18
  simple_clusters, 20
  sketches_test, 21
  sketches_train, 21

aflw, 2
box, 3
bushfires, 4
c1, 5
c2 (c1), 5
c3 (c1), 5
c4 (c1), 5
c5 (c1), 5
c6 (c1), 5
c7 (c1), 5
calc_mv_dist, 5
calc_norm, 6
clusters, 6
clusters_nonlin, 7
convert_proj_tibble, 8
dist, 13
gen_vc_ellipse, 8
gen_xvar_ellipse, 9
ggmcbic, 10
ggscreene, 11
ggslice, 11
ggslice_projection, 12
hclust, 13
hierfly, 13
mc_ellipse, 14
multicluster, 15
norm_vec, 16
pca_model, 16
pisa, 17
plane, 18
plane_nonlin, 18
pooled_vc, 19
rmvn, 19
simple_clusters, 20
sketches_test, 21
sketches_train, 21
som_model, 22