Package 'mulgar'

August 31, 2026

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

Version 1.0.8

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 (>= 4.1.0)

Imports geozoo, tibble, ggplot2, tidyr, dplyr, purrr, stats, methods

Suggests tourr, ggdendro, colorspace, mclust, kohonen, GGally

License MIT + file LICENSE

Encoding UTF-8

LazyData true

LazyDataCompression bzip2

URL https://dicook.github.io/mulgar/, https://github.com/dicook/mulgar

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

Config/roxygen2/version 8.0.0

NeedsCompilation no

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

Maintainer Dianne Cook <dicook@monash.edu>

Repository CRAN

Date/Publication 2026-08-31 15:30:02 UTC

Contents

aflw 2
anomalies 3

associations . . . 4
box . . . 4
bushfires . . . 5
calc_mv_dist . . . 6
calc_norm . . . 6
clusterchallenges . . . 7
clusters . . . 8
clusters_nonlin . . . 8
convert_proj_tibble . . . 9
copula . . . 10
fake_trees . . . 10
gen_vc_ellipse . . . 11
gen_xvar_ellipse . . . 12
ggmcbic . . . 13
ggscree . . . 13
ggslice . . . 14
ggslice_projection . . . 15
hierfly . . . 16
mc_ellipse . . . 17
multicluster . . . 17
norm_vec . . . 18
pca_model . . . 19
pisa . . . 19
plane . . . 20
plane_nonlin . . . 21
pooled_vc . . . 21
rmvn . . . 22
simple_clusters . . . 23
sketches_test . . . 23
sketches_train . . . 24
som_model . . . 25

Index **26**

`aflw` *AFLW player statistics*

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

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

---

anomalies *Data sets with anomalies*

Description

Simulated data with anomalies

Usage

```
anomaly1

anomaly2

anomaly3

anomaly4

anomaly5
```

Format

A datasets with anomalies

Details

x1, x2, x3, x4 numeric variables

Source

Created by Di Cook.

Examples

```
require(GGally)
data(anomaly1)
data(anomaly2)
ggscatmat(anomaly1)
ggscatmat(anomaly2)
```

`associations` *Data sets with different types of association*

Description

Simulated data with various associations

Usage

```
assoc1

assoc2

assoc3
```

Format

A datasets with various association

Details

x1, x2, x3, x4 numeric variables

Examples

```
require(GGally)
data(assoc1)
ggscatmat(assoc1)
```

---

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

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

---

bushfires *Australian bushfires 2019-2020*

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, ase7-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

```
require(dplyr)
data(bushfires)
glimpse(bushfires)
```

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

```
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

```
calc_norm(x)
```

Arguments

x	numeric vector

Value

numeric value

Examples

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

---

clusterchallenges *Cluster challenge data sets*

Description

Simulated data with different structures

Usage

```
c1

c2

c3

c4

c5

c6

c7
```

Format

A datasets with differing number of rows and columns

Details

x1, x2, ... numeric variables

Source

Created by Di Cook.

Examples

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

`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

Details

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

cl class variable

See Also

simple_clusters

Examples

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

`clusters_nonlin` *Four unusually shaped clusters in 4D*

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

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

```
convert_proj_tibble(t1)
```

Arguments

`t1`	tour projection sequence

Value

tbl1 tibble

Examples

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

copula *Data sets generated using copulas*

Description

Simulated data from covsim, using different copula models

Usage

```
copclayton

copjoe

copfrank

copnorm
```

Format

A datasets with various association

Details

x1, x2, x3, x4, x5 numeric variables

Examples

```
require(GGally)
data(copclayton)
ggscatmat(copclayton)
```

fake_trees *A high-dimensional tree data structure with 10 branching points.*

Description

A high-dimensional tree data structure with 10 branching points.

Usage

```
fake_trees
```

Format

An object of class `data.frame` with 3000 rows and 101 columns.

Details

Data are obtained from diffusion limited aggregation tree simulation in the `phate` python and `phateR` packages, but reconstructed as a wide data.frame rather than a list.

There are 3000 rows and 101 columns, the first 100 columns are labelled dim1 - dim100, and are numeric, while the final column is a factor representing the branch id.

Source

PHATE

`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

```
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

```
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(ell2d)
ggplot(ell2d, 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

```
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

```
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

```
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)
```

ggscree *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

```
ggscree(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

```
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")])
ggscree(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

Value

ggplot2 object showing the sliced data

See Also

ggslice

Examples

```
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

```
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

```
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

```
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

```
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 *Normalise 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

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

```
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 samples from Australia and Indonesia, and the simulated scores for math, reading and science.

Format

A data set with 10548 rows and 31 columns

Details

CNT Country (Australia, Indonesia)

PV1MATH-PV10SCIE simulated scores for math, reading and science

Examples

```
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

```
plane_pca <- prcomp(plane)
ggscree(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

Examples

```
plane_nonlin_pca <- prcomp(plane_nonlin)
ggscree(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

```
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

```
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

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

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

```
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)
```

simple_clusters *Two clusters in 2D*

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

```
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

```
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

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, 16
∗ **datasets**
  aflw, 2
  anomalies, 3
  associations, 4
  box, 4
  bushfires, 5
  clusterchallenges, 7
  clusters, 8
  clusters_nonlin, 8
  copula, 10
  fake_trees, 10
  multicluster, 17
  pisa, 19
  plane, 20
  plane_nonlin, 21
  simple_clusters, 23
  sketches_test, 23
  sketches_train, 24

aflw, 2
anomalies, 3
anomaly1 *(*anomalies*)*, 3
anomaly2 *(*anomalies*)*, 3
anomaly3 *(*anomalies*)*, 3
anomaly4 *(*anomalies*)*, 3
anomaly5 *(*anomalies*)*, 3
assoc1 *(*associations*)*, 4
assoc2 *(*associations*)*, 4
assoc3 *(*associations*)*, 4
associations, 4

box, 4
bushfires, 5

c1 *(*clusterchallenges*)*, 7
c2 *(*clusterchallenges*)*, 7
c3 *(*clusterchallenges*)*, 7
c4 *(*clusterchallenges*)*, 7
c5 *(*clusterchallenges*)*, 7
c6 *(*clusterchallenges*)*, 7
c7 *(*clusterchallenges*)*, 7
calc_mv_dist, 6
calc_norm, 6
clusterchallenges, 7
clusters, 8
clusters_nonlin, 8
convert_proj_tibble, 9
copclayton *(*copula*)*, 10
copfrank *(*copula*)*, 10
copjoe *(*copula*)*, 10
copnorm *(*copula*)*, 10
copula, 10

dist, *16*

fake_trees, 10

gen_vc_ellipse, 11
gen_xvar_ellipse, 12
ggmcbic, 13
ggscree, 13
ggslice, 14
ggslice_projection, 15

hclust, *16*
hierfly, 16

mc_ellipse, 17
multicluster, 17

norm_vec, 18

pca_model, 19
pisa, 19
plane, 20
plane_nonlin, 21
pooled_vc, 21

rmvn, 22

simple_clusters, 23
sketches_test, 23
sketches_train, 24
som_model, 25