

# Package ‘clusterfly’

February 14, 2012

**Type** Package

**Title** Explore clustering interactively using R and GGobi

**Version** 0.3

**Author** Hadley Wickham <h.wickham@gmail.com>

**Maintainer** Hadley Wickham <h.wickham@gmail.com>

**Description** Visualise clustering algorithms with GGobi. Contains both general code for visualising clustering results and specific visualisations for model-based, hierarchical and SOM clustering.

**URL** <http://had.co.nz/clusterfly>

**Depends** ggplot2, reshape, rggobi

**Imports** e1071

**Suggests** som, mclust, kohonen

**License** MIT

**LazyData** true

**Collate** ‘clusterfly.r’ ‘data.r’ ‘ellipsoid.r’ ‘extract.r’ ‘ggplot.r’ ‘hclust.r’ ‘hulls.r’ ‘model-based.r’ ‘som-iteration.r’ ‘som.r’ ‘utils.r’

**Repository** CRAN

**Date/Publication** 2010-09-03 07:31:45

## R topics documented:

addhull	2
as.data.frame.clusterfly	3
cfly_animate	3
cfly_clarify	4
cfly_cluster	4

cfly_dist . . . . .	5
cfly_fluct . . . . .	6
cfly_pcp . . . . .	6
cfly_show . . . . .	7
clarify . . . . .	8
clusterfly . . . . .	8
cut.hierfly . . . . .	9
ggobi.hierfly . . . . .	10
ggobi.som . . . . .	11
hierarchical . . . . .	12
hierfly . . . . .	12
mefly . . . . .	13
olive_example . . . . .	13
<b>Index</b>	<b>14</b>

---

addhull	<i>Add convex hulls...</i>
---------	----------------------------

---

## Description

Add convex hulls Add conver hulls using the tool qconvex

## Usage

```
addhull(gd, g, by)
```

## Arguments

gd	ggobi dataset
g	ggobi reference
by	grouping variable

## Details

To use this command you must have qconvex installed and available on your path. I'm not sure if this will work on windows (probably not) but it's not a big loss, because the technique isn't very useful anyway.

---

```
as.data.frame.clusterfly
      Convert clusterfly object to data...
```

---

**Description**

Convert clusterfly object to data.frame. Concatenates data and cluster assignments into one data.frame. Cluster assignments are prefixed with cl\_.

**Usage**

```
## S3 method for class 'clusterfly'
as.data.frame(x, ...)
```

**Arguments**

x	clusterfly object
...	ignored

---

```
cfly_animate      Dynamic plot: Animate glyph colours...
```

---

**Description**

Dynamic plot: Animate glyph colours

**Usage**

```
cfly_animate(cf, clusters=seq_along(cf$clusters), pause=1, print=TRUE,
             max_iterations=100)
```

**Arguments**

cf	list of cluster ids that you want to animate across
clusters	clusters to display
pause	clusters number of seconds to pause between each change
print	print current cluster to screen?
max_iterations	maximum number of iterations

**Details**

This function will animate until you manually break the loop using Ctrl-Break or Ctrl-C.

**Examples**

```
# Press Ctrl-Break or Ctrl-C to exit
o <- olive_example()
cfly_animate(cfly_clarify(o), max = 5)
if (!interactive()) close(o)
```

---

cfly_clarify	<i>Match all cluster indices to common reference.</i>
--------------	---

---

**Description**

Match all cluster indices to common reference.

**Usage**

```
cfly_clarify(cf, reference=1, method="rowmax")
```

**Arguments**

cf	clusterfly object
reference	index to reference clustering
method	method to use, see <a href="#">clarify</a>

**Details**

It's a good idea to run this before running any animation sequences so that unnecessary colour changes are minimised.

**Examples**

```
o <- olive_example()
o <- cfly_clarify(o, "Region")
```

---

cfly_cluster	<i>Add clustering.</i>
--------------	------------------------

---

**Description**

Add clustering.

**Usage**

```
cfly_cluster(cf, method, ..., name=deparse(substitute(method)))
```

**Arguments**

cfly	clusterfly object
method	clusterfing method (function)
...	arguments passed to clustering method
name	name of clustering

**Details**

Clustering method needs to respond to `clusters`, if the default does not work, you will need to write your own to extract clusters.

**Examples**

```
o <- olive_example()
cfly_cluster(o, kmeans, 4)
cfly_cluster(o, kmeans, 4, name="blah")
```

---

cfly_dist	<i>Static plot: Variable distribution.</i>
-----------	--

---

**Description**

Static plot: Variable distribution. Draw a density plot for each continuous variable, faceted across clustering.

**Usage**

```
cfly_dist(cfly, index, scale="range")
```

**Arguments**

cfly	clusterfly object
index	clustering to use
scale	scaling to use

**Details**

This allows you to quickly visualise how the cluster vary in a univariate manner. Currently, it is a bit of a hack, because `ggplot` does not support plots with different scales, so the variables are manually rescaled prior to plotting.

This plot is inspired by Gaguin <http://www.rosuda.org/gaguin>.

**Examples**

```
o <- olive_example()
cfly_dist(o, "kmeans")
cfly_dist(o, "kmeans") + scale_y_continuous(limit=c(0, 2))
if (!interactive()) close(o)
```

---

cfly\_fluct                      *Static plot: Fluctuation diagram.*

---

**Description**

Static plot: Fluctuation diagram. Draw a fluctuation diagram comparing two clusterings.

**Usage**

```
cfly_fluct(cfly, a, b, clarify=TRUE, ...)
```

**Arguments**

cfly	clusterfly object
a	first clustering, will be reordered to match b if clarify=TRUE
b	second clustering
clarify	use <code>clarify</code> to rearranged cluster indices?
...	other arguments passed on to <code>ggfluctuation</code>

**Examples**

```
o <- olive_example()
cfly_fluct(o, "kmeans", "Region", clarify=TRUE)
if (!interactive()) close(o)
```

---

cfly\_pcp                      *Static plot: Parallel coordinates.*

---

**Description**

Static plot: Parallel coordinates. Draw a parallel coordinates plot, faceted across clustering.

**Usage**

```
cfly_pcp(cfly, index, ...)
```

**Arguments**

cfly	clusterfly object
index	clustering to use
...	other arguments passed to <code>geom_line</code>

## Details

This really only a proof of concept, a truly useful PCP needs interaction, especially to move the variables around.

## Examples

```
o <- olive_example()
cfly_pcp(o, "kmeans")
if (!interactive()) close(o)
```

---

cfly\_show

*Show in ggobi.*

---

## Description

Show in ggobi. Opens an instance ggobi for this dataset (if not already open), and colours the points according the cluster assignment.

## Usage

```
cfly_show(cf, idx="true", hulls=FALSE)
```

## Arguments

cf	clusterfly object
idx	clustering to display
hulls	add convex hull? see <a href="#">addhull</a> for details

## Examples

```
o <- olive_example()
cfly_show(o, 1)
cfly_show(o, "Region")
if (!interactive()) close(o)
```

---

clarify	<i>Clarify matrix...</i>
---------	--------------------------

---

**Description**

Clarify matrix Clarify matrix ordering to minimize off diagonals

**Usage**

```
clarify(a, b, method="greedy")
```

**Arguments**

a	cluster assignments to reassign
b	matrix b
method	clarification method

**Value**

vector of reassigned cluster a

**See Also**

[matchClasses](#)

---

clusterfly	<i>Creates a convenient data structure for dealing with a dataset and a number...</i>
------------	---

---

**Description**

Creates a convenient data structure for dealing with a dataset and a number of alternative clusterings.

**Usage**

```
clusterfly(df, extra, rescale=TRUE)
```

**Arguments**

df	data frame to be clustered
extra	extra variables to be included in output, but not clustered
rescale	rescale, if true each variable will be scaled to have mean 0 and variance 1.

**Details**

Once you have created a clusterfly object, you can add clusterings to it with `cfly_cluster`, and visualise then in GGobi with `cfly_show` and `cfly_animate`. Static graphics are also available: `cfly_pcp` will produce a parallel coordinates plot, `cfly_dist` will show the distribution of each variable in each cluster, and `cfly_fluct` compares two clusterings with a fluctuation diagram.

If you want to standardise the cluster labelling to one group, look at `clarify` and `cfly_clarify`

**See Also**

`vignette("introduction")`

**Examples**

```
olives <- read.csv(ggobi_find_file("data", "olive.csv"))
ol <- clusterfly(olives[, -(1:3)], olives[, 2:3])
ol <- cfly_cluster(ol, kmeans, 4, name="k4-1")
ol <- cfly_cluster(ol, kmeans, 4, name="k4-2")
ol <- cfly_cluster(ol, kmeans, 4, name="k4-3")

ggobi(ol)
cfly_show(ol, "k4-1")
cfly_animate(ol, max = 5)
if (!interactive()) close(ol)
```

---

cut.hierfly

*Cut hierfly object into k clusters/colours.*

---

**Description**

Cut hierfly object into k clusters/colours.

**Usage**

```
## S3 method for class 'hierfly'
cut(x, k=2, g=ggobi(x), ...)
```

**Arguments**

x	hierfly object to colour
k	number of clusters
g	GGobi instance displaying x, will create new if not specified
...	ignored

### Examples

```
h <- hierfly(iris)
hfly <- ggobi(h)
cut(h, 2, hfly)
h <- hierfly(iris, method="ward")
g <- ggobi(h)
cut(h, 2, g)
```

---

ggobi.hierfly

*Visualise hierarchical clustering with GGobi.*

---

### Description

Visualise hierarchical clustering with GGobi. Displays both data and dendrogram in original high-d space.

### Usage

```
## S3 method for class 'hierfly'
ggobi(data, ...)
```

### Arguments

data	hierfly object to visualise in GGobi
...	ignored

### Details

This adds four new variables to the original data set:

- ORDER, the order in which the clusters are joined
- HEIGHT, the height of the branch, ie. the dissimilarity between the branches
- LEVEL, the level of the branch
- POINTS, the number of points in the branch

Make sure to select "attach edge set (edges)" in the in the edges menu on the plot window, when you create a new plot.

A tour over the original variables will show how the clusters agglomerate in space. Plotting order vs height, level or points will give various types of dendograms. A correlation tour with height/level/points on the y axis and the original variables on the x axis will show a mobile blowing in the wind.

### See Also

[cut.hierfly](#)

**Examples**

```
h <- hierfly(iris)
ggobi(h)
h <- hierfly(iris, method="single")
```

ggobi.som

*Visualise Kohonen self organising maps with GGobi...***Description**

Visualise Kohonen self organising maps with GGobi Displays both data, and map in original high-d space.

**Usage**

```
ggobi.som(data, ...)
```

**Arguments**

data	SOM object
...	ignored

**Details**

Map variables added as map1 and map2. Plot these to get traditional SOM plot. Tour over all other variables to see how well the map fits the original data.

**Examples**

```
## Not run:
d.music <- read.csv("http://www.ggobi.org/book/data/music-all.csv")

music <- rescaler(d.music)[complete.cases(d.music), 1:10]
music.som <- som::som(music[,-(1:3)], 6, 6, neigh="bubble", rlen=1000)
ggobi(music.som)

## End(Not run)
## Not run:
d.music <- read.csv("http://www.ggobi.org/book/data/music-all.csv")

music <- rescaler(d.music)[complete.cases(d.music), 1:10]
music.hex <- kohonen::som(music[,-(1:3)], grid = somgrid(3, 3, "hexagonal"), rlen=1000)
music.rect <- kohonen::som(music[,-(1:3)], grid = somgrid(6, 6, "rectangular"), rlen=1000)
ggobi(music.rect)

## End(Not run)
```

---

hierarchical	<i>Hierachical clustering...</i>
--------------	----------------------------------

---

**Description**

Hierachical clustering Convenient methods for hierachical clustering

**Usage**

```
hierarchical(df, method="complete", metric="euclidean", n=5)
```

**Arguments**

df	data frame
method	method to use, see <a href="#">hclust</a>
metric	distance metric to use, see <a href="#">dist</a>
n	number of clusters to retrieve, see <a href="#">cut</a>

---

hierfly	<i>Visualisig hierarchical clustering.</i>
---------	--

---

**Description**

Visualisig hierarchical clustering. This method supplements a data set with information needed to draw a dendrogram

**Usage**

```
hierfly(data, metric="euclidean", method="average")
```

**Arguments**

data	data set
metric	distance metric to use, see <a href="#">dist</a> for list of possibilities
method	cluster distance measure to use, see <a href="#">hclust</a> for details

**Details**

Intermediate cluster nodes are added as needed, and positioned at the centroid of the combined clusters.

**Value**

object of type, hierfly

**See Also**

[cut.hierfly](#), [ggobi.hierfly](#)

**Examples**

```
h <- hierfly(iris)
ggobi(h)
h <- hierfly(iris, method="single")
```

---

mefly

*Display model based clustering with mvn ellipses.*

---

**Description**

Display model based clustering with mvn ellipses. Displays the results of model based clustering with an ellipse drawn from the multivariate normal model for each group.

**Usage**

```
mefly(model, data)
```

**Arguments**

model	output from me function
data	input data frame to me

**Examples**

```
if(require("mclust")) {
  eei <- me(modelName = "EEI", data = iris[,-5], z = unmap(iris[,5]))
  vvv <- me(modelName = "VVV", data = iris[,-5], z = unmap(iris[,5]))
  vvi <- me(modelName = "VVI", data = iris[,-5], z = unmap(iris[,5]))
  mefly(eei, iris[,-5])
  mefly(vvi, iris[,-5])
  mefly(vvv, iris[,-5])
}
```

---

olive\_example

*Example clusterfly object created with olives data...*

---

**Description**

Example clusterfly object created with olives data

# Index

## \*Topic **cluster**

- cut.hierfly, 9
- ggobi.hierfly, 10
- ggobi.som, 11
- hierarchical, 12
- hierfly, 12
- mefly, 13

## \*Topic **dataset**

- olive\_example, 13

## \*Topic **dynamic**

- cfly\_animate, 3
- cfly\_show, 7
- clusterfly, 8
- ggobi.hierfly, 10
- ggobi.som, 11
- mefly, 13

## \*Topic **hplot**

- addhull, 2
- cfly\_dist, 5
- cfly\_fluct, 6
- cfly\_pcp, 6

## \*Topic **manip**

- as.data.frame.clusterfly, 3
- cfly\_clarify, 4
- cfly\_cluster, 4
- clarify, 8

addhull, 2, 7

as.data.frame.clusterfly, 3

cfly\_animate, 3, 9  
cfly\_clarify, 4, 9  
cfly\_cluster, 4, 9  
cfly\_dist, 5, 9  
cfly\_fluct, 6, 9  
cfly\_pcp, 6, 9  
cfly\_show, 7, 9  
clarify, 4, 6, 8, 9  
clusterfly, 8  
clusters, 5

cut, 12

cut.hierfly, 9, 10, 13

dist, 12

geom\_line, 6

ggobi.hierfly, 10, 13

ggobi.som, 11

ggplot, 5

hclust, 12

hierarchical, 12

hierfly, 12

matchClasses, 8

mefly, 13

olive\_example, 13

package-clusterfly (clusterfly), 8