

Package ‘hamlet’

January 27, 2015

Type Package

Title Hierarchical Optimal Matching and Machine Learning Toolbox

Version 0.7

Date 2014-05-28

Author Teemu Daniel Laajala <teelaa@utu.fi>

Maintainer Teemu Daniel Laajala <teelaa@utu.fi>

Depends R (>= 3.0.0)

Suggests lme4, nlme, amap, nbpMatching, lattice, lmerTest

Description This package offers helpful functions and algorithms for solving optimal matching tasks in the context of preclinical cancer studies. Furthermore, various helper and plotting functions are provided for unsupervised and supervised machine learning and longitudinal modeling of tumor response patterns.

License GPL (>= 2)

NeedsCompilation no

Repository CRAN

Date/Publication 2014-05-29 10:17:17

R topics documented:

hamlet-package	2
extendsymrange	4
hmap	5
match.allocate	6
match.bb	7
match.dummy	9
match.mat2vec	10
match.vec2mat	12
mem.getcomp	13
mem.plotran	14
mem.plotresid	15
mix.binary	16
mix.fun	17

mixplot	18
smartjitter	19
vcaplong	20
vcapwide	21

Index	25
--------------	-----------

hamlet-package	<i>Hierarchical Optimal Matching and Machine Learning Toolbox</i>
----------------	---

Description

This package offers helpful functions and algorithms for solving optimal matching tasks in the context of preclinical cancer studies. Furthermore, various helper and plotting functions are provided for unsupervised and supervised machine learning.

Details

Package: hamlet
 Type: Package
 Version: 0.7
 Date: 2014-05-27
 License: GPL (>= 2)

The package 'hamlet' offers useful functions for optimal matching, randomization, and mixed-effects modeling in preclinical cancer studies. The functions are divided to 'match'-prefix indicating optimal matching intended functions, 'mem' indicating mixed-effects modeling, 'mix' for mixed type (numerical and categorical) data analysis, and rest that are various plotting and helper functions for various tasks.

Author(s)

Teemu Daniel Laajala

Maintainer: Teemu Daniel Laajala <teelaa@utu.fi>

References

Knuutila M, Yatkin E, Kallio J, Savolainen S, Laajala TD, et al. (2014) Castration induces upregulation of intratumoral androgen biosynthesis and androgen receptor expression in orthotopic VCaP human prostate cancer xenograft model. *Am J Pathol*. Accepted

Laajala TD, Kaur A, Knuutila M, Westermarck J, et al. (2014) Optimal Matching, Randomization and Statistical Modeling of Hierarchical Baseline Variables in Preclinical Cancer Studies. Submitted

Examples

```

##
## Exploring the VCaP dataset provided alongside the 'hamlet' package
##

data(vcapwide)
data(vcaplong)

# VCaP Castration-resistant prostate cancer (CRPC) PSA-measurements (and body weight) in wide-format
mixplot(vcapwide[,c("PSAWeek10", "PSAWeek14", "BWWeek10", "Group")], pch=16)
anv <- aov(PSA ~ Group, data.frame(PSA = vcapwide[, "PSAWeek14"], Group = vcapwide[, "Group"]))
summary(anv)
TukeyHSD(anv)
summary(aov(BW ~ Group, data.frame(BW = vcapwide[, "BWWeek14"], Group = vcapwide[, "Group"])))

# VCaP Castration-resistant prostate cancer (CRPC) PSA-measurements (and body weight) in long-format
library(lattice)
xyplot(log2PSA ~ DrugWeek | Group, data = vcaplong, type="l", group=ID, layout=c(3,1))
xyplot(BW ~ DrugWeek | Group, data = vcaplong, type="l", group=ID, layout=c(3,1))

##
## Example multigroup (g=3) nbp-matching using the branch and bound algorithm,
## and subsequent random allocation of submatches to 3 arms
##

# Construct an Euclidean distance example distance matrix using 15 observations from the VCaP study
d <- as.matrix(dist(vcapwide[1:15,c("PSAWeek10", "BWWeek10")]))
# Matching using the b&b algorithm to submatches of size 3
# (which will result in 3 intervention groups)
bb3 <- match.bb(d, g=3)
str(bb3)

solvec <- bb3$solution
# matching vector, where each element indicates to which submatch each observation belongs to

# Perform an example random allocation of the above submatches,
# these will be randomly allocated to 3 arms based on the submatches
set.seed(1)
groups <- match.allocate(solvec)

# Illustrate randomization, no baseline differences in these three artificial groups
by(vcapwide[1:15,c("PSAWeek10", "BWWeek10")], INDICES=groups, FUN=function(x) x)

summary(aov(PSAWeek10 ~ groups, data = data.frame(PSAWeek10 = vcapwide[1:15, "PSAWeek10"], groups)))
summary(aov(BWWeek10 ~ groups, data = data.frame(BWWeek10 = vcapwide[1:15, "BWWeek10"], groups)))

##
## Example mixed-effects modeling of the longitudinal PSA profiles using
## the actual experimental groups
##

```

```
exdat <- vcaplong[vcaplong[, "Group"] %in% c("Vehicle", "ARN"),]

library(lme4)
# Model fitting using lme4-package
f1 <- lmer(log2PSA ~ 1 + DrugWeek + DrugWeek:ARN + (1 + DrugWeek|ID), data = exdat)

mem.getcomp(f1)

library(lmerTest)
# Model term testing using the lmerTest-package
summary(f1)
```

extendsymrange	<i>Extend range of variable limits while retaining a point of symmetry</i>
----------------	--

Description

This function serves as an alternative to the R function 'extendrange', when user wishes to conserve a point of symmetry for the range. For example, this might be desired when the plot should be symmetric around the origin $x=0$, but that the sides need to extend beyond the actual range of values.

Usage

```
extendsymrange(x, r = range(x, na.rm = T), f = 0.05, sym = 0)
```

Arguments

x	Vector of values to compute the range for
r	The range of values
f	The factor by which the range is extended beyond the extremes
sym	The defined point of symmetry

Value

A vector of 2 values for the lower and higher limit of the symmetric extended range

Author(s)

Teemu Daniel Laajala <teelaa@utu.fi>

See Also

[extendrange](#)

Examples

```
set.seed(1)
ex <- rnorm(10)+2

hist(ex, xlim=extendsymrange(ex, sym=0), breaks=100)
```

hmap

Plot-region based heatmap

Description

This function plots heatmap figure based on the normal plot-region. This is useful if the image-based function 'heatmap' is not suitable, i.e. when multiple heatmaps should be placed in a single device.

Usage

```
hmap(x, add = F, xlim = c(0, 1), ylim = c(0, 1), col = heat.colors(10),
border = matrix(NA, nrow = nrow(x), ncol = ncol(x)),
lty = matrix("solid", nrow = nrow(x), ncol = ncol(x)),
lwd = matrix(1, nrow = nrow(x), ncol = ncol(x)), ...)
```

Arguments

x	Matrix to be plotted
add	Should the figure be added to the plotting region of an already existing figure
xlim	The x limits in which the heatmap is placed horizontally in the plotting region
ylim	The y limits in which the heatmap is placed vertically in the plotting region
col	Color palette for the heatmap colors
border	A matrix of border color definitions (rectangles in the heatmap)
lty	A matrix of line type definitions (rectangles in the heatmap)
lwd	A matrix of line width definitions (rectangles in the heatmap)
...	Additional parameters provided to the rectangle plotting function

Note

The function does not yet implement the hclust-functionality of the normal R heatmap-function.

Author(s)

Teemu Daniel Laajala <teelaa@utu.fi>

See Also

[heatmap](#)

Examples

```

set.seed(1)
ex <- matrix(rnorm(100), nrow=10, ncol=10)
highlight <- matrix("black", nrow=10, ncol=10)
highlight[ex>1] <- "red"
thick <- matrix(1, nrow=10, ncol=10)
thick[ex>1] <- 3

plot.new()
plot.window(xlim=c(0,3), ylim=c(0,1))
hmap(ex, add=TRUE, xlim=c(0,1))
hmap(ex, add=TRUE, xlim=c(2,3), border=highlight, lwd=thick)

```

match.allocate	<i>Allocation of matched units to intervention arms</i>
----------------	---

Description

This function allocates units belonging to a single submatch to separate intervention arms. This ensures that the resulting intervention groups are homogeneous in respect to the variables that were used to construct the distance/dissimilarity matrix for the non-bipartite matching. The number of resulting intervention groups is equal to the 'g' (i.e. submatch size) used in the multigroup non-bipartite matching.

Usage

```
match.allocate(xmat)
```

Arguments

xmat A binary matching matrix or a matching vector given by match.bb-function.

Value

A vector where each element indicates to which group the observation was randomized to. The group names are "Group_A", "Group_B", "Group_C", ... until 'g' letters, where 'g' was the size of submatches.

Author(s)

Teemu Daniel Laajala <teelaa@utu.fi>

References

Laajala TD, Kaur A, Knuutila M, Westermarck J, et al. (2014) Optimal Matching, Randomization and Statistical Modeling of Hierarchical Baseline Variables in Preclinical Cancer Studies. Submitted

See Also

[match.bb](#) [match.mat2vec](#) [match.vec2mat](#) [match.dummy](#)

Examples

```
data(vcapwide)

# Construct an Euclidean distance example distance matrix using 15 observations from the VCaP study
d <- as.matrix(dist(vcapwide[1:15,c("PSAWeek10", "BWWeek10")]))
# Matching using the b&b algorithm to submatches of size 3
# (which will result in 3 intervention groups)
bb3 <- match.bb(d, g=3)
str(bb3)

solvec <- bb3$solution
# matching vector, where each element indicates to which submatch each observation belongs to

# Perform an example random allocation of the above submatches,
# these will be randomly allocated to 3 arms based on the submatches
set.seed(1)
groups <- match.allocate(solvec)

# Illustrate randomization, no baseline differences in these three artificial groups
by(vcapwide[1:15,c("PSAWeek10", "BWWeek10")], INDICES=groups, FUN=function(x) x)

summary(aov(PSAWeek10 ~ groups, data = data.frame(PSAWeek10 = vcapwide[1:15,"PSAWeek10"], groups)))
summary(aov(BWWeek10 ~ groups, data = data.frame(BWWeek10 = vcapwide[1:15,"BWWeek10"], groups)))
```

match.bb

Branch and Bound algorithm implementation for performing multi-group non-bipartite matching

Description

This function performs multigroup non-bipartite matching of observations based on a provided distance/dissimilarity matrix 'd'. The number of elements in each submatch is defined by the parameter 'g'.

Usage

```
match.bb(d, g = 2, presort = "complete", progress = 1e+05,
bestknown = Inf, maxbranches = Inf, verb = 0)
```

Arguments

d A distance matrix with NxN elements

g Number of elements per each submatch, i.e. how many observations are always matched together

presort	If hierarchical clustering should be used for an initial guess, hclust method-options are valid options ("complete", "single", "ward", "average")
progress	How many branching operations are done before outputting information to the user
bestknown	If a best known solution already exists, this can be used to bound branches of the tree before initiation. The default Inf value causes whole search tree to be potential solution space.
maxbranches	Maximum number of branching operations before returning current best solution, by default no cutoff is defined.
verb	Level of verbosity

Details

See further details in the reference Laajala et al.

Value

The function returns a list of objects, where elements are

branches	Number of branching operations during the branch and bound algorithm
bounds	Number of bounding operations during the branch and bound algorithm
ends	Number of end leaf nodes visited during the branch and bound algorithm
matrix	The resulting binary matching matrix where rows and columns sum to g
solution	The resulting matching vector where each element indicates the submatch where the observation was placed
cost	Final cost value of the target function in the minimization task

Note

Notice that the solution submatch vector in \$solution is not the same as the intervention group allocation. Submatches should be randomly allocated to intervention arms using the match.allocate-function.

The package 'nbpMatching' provides a FORTRAN implementation for computation of paired non-bipartite matching case (g=2).

Computation may be heavy if the number of observations is high, or the number of within-submatch pairwise distances to consider is high (increases quadratically as a function of 'g').

Author(s)

Teemu Daniel Laajala <teelaa@utu.fi>

References

Laajala TD, Kaur A, Knuutila M, Westermarck J, et al. (2014) Optimal Matching, Randomization and Statistical Modeling of Hierarchical Baseline Variables in Preclinical Cancer Studies. Submitted

See Also

[match.allocate](#) [match.mat2vec](#) [match.vec2mat](#) [match.dummy](#)

Examples

```
data(vcapwide)

# Construct an Euclidean distance example distance matrix using 15 observations from the VCaP study
d <- as.matrix(dist(vcapwide[1:15,c("PSAWeek10", "BWWeek10")]))

bb3 <- match.bb(d, g=3)
str(bb3)

mat <- bb3$matrix
# binary matching matrix
solvec <- bb3$solution
# matching vector, where each element indicates to which submatch each observation belongs to

mixplot(data.frame(vcapwide[1:15,c("PSAWeek10", "BWWeek10")],
  submatch=as.factor(paste("Submatch_",solvec, sep=""))), pch=16, col=rainbow(5))
```

match.dummy

Create dummy individuals or sinks to a data matrix or a distance/dissimilarity matrix

Description

Dummy observations are allowed in order to make the number of observations dividable by the number of elements in each submatch, i.e. for pairwise matching the number of observations should be paired, for triangular matching the number of observations should be dividable by 3, etc. This can be done either by adding column averaged individuals to the original data frame (parameter 'dat'), or by adding zero distance sinks to the distance/dissimilarity matrix (parameter 'd'). The latter approach favors dummies being matched to real extreme observations, while the former favors dummies being matched to close-to-mean real observations.

Usage

```
match.dummy(dat, d, g = 2)
```

Arguments

dat	A data.frame of the original observations, to which column averaged new dummy observations are added
d	N times N distance/dissimilarity matrix, to which zero distance sinks are added
g	The desired number of elements per each submatch, i.e. the size of the clusters. The number of added dummies is the smallest number of additions that fulfills (N+dummy)

Value

Depending on if the `dat` or the `d` parameter was provided, the function either: `dat`: adds new averaged individuals according to column means and then returns the data matrix `d`: adds zero distance sinks to the distance/dissimilarity matrix and returns the new distance/dissimilarity matrix

Note

Adding zero distance sinks to the distance matrix or averaged individuals to the original data frame produce different results and affect the optimal matching task differently.

Author(s)

Teemu Daniel Laajala <teelaa@utu.fi>

See Also

[match.allocate](#) [match.mat2vec](#) [match.vec2mat](#) [match.bb](#)

Examples

```
data(vcapwide)

exdat <- vcapwide[1:10,c("PSAWeek10", "BWWeek10")]
dim(exdat)
avgdummies <- match.dummy(dat=exdat, g=3)
dim(avgdummies)
# Construct an Euclidean distance matrix after adding two dummy individuals
# (averaged individuals to the original data matrix)
bb3 <- match.bb(as.matrix(dist(avgdummies)), g=3)
str(bb3)

# Construct an Euclidean distance matrix after adding two dummy distances (zero distance sinks)
exd <- as.matrix(dist(vcapwide[1:10,c("PSAWeek10", "BWWeek10")]))
dim(exd)
d <- match.dummy(d=exd, g=3)
dim(d)
# 10 is not dividable by 3, 2 sinks are added to make d 12x12
bb3 <- match.bb(d, g=3)
str(bb3)

# Notice that sinks produce a lot smaller target function costs than averaged individuals
```

Description

This function transforms a binary matching matrix to a matching vector. A matching vector is of length N where each element indicates the submatch to which the observation belongs to. Notice that this is not the same as the group allocation vector that is provided by the `match.allocate` function. The binary matching matrix is of size $N \times N$ where 0 indicates that the observations have been part of a different submatch, and 1 indicates that the observations have been part of the same submatch. Diagonal is always 0 although an observation is always in the same submatch with its self.

Usage

```
match.mat2vec(xmat)
```

Arguments

`xmat` A binary matching matrix 'xmat'

Value

A matching vector where each element indicates submatch the observation belongs to

Note

Notice that the particular index numbers produced by `match.mat2vec` may be different to that of the branch and bound solution vector, but that the submatches shared by observations are common.

Author(s)

Teemu Daniel Laajala <teelaa@utu.fi>

See Also

[match.allocate](#) [match.bb](#) [match.vec2mat](#) [match.dummy](#)

Examples

```
data(vcapwide)

# Construct an Euclidean distance example distance matrix using 15 observations from the VCaP study
d <- as.matrix(dist(vcapwide[1:15,c("PSAWeek10", "BWWeek10")]))

bb3 <- match.bb(d, g=3)
str(bb3)

mat <- bb3$matrix
# matching vector, where each element indicates to which submatch each observation belongs to

mat
solvec <- match.mat2vec(mat)
which(mat[1,] == 1)
# E.g. the first, third and thirteenth observation are part of the same submatch
```

```
which(solvec == solvec[1])
# Similarly
```

match.vec2mat	<i>Transform a matching vector to a binary matching matrix</i>
---------------	--

Description

This function allows transforming a matching vector to a binary matching matrix. A matching vector is of length N where each element indicates the submatch to which the observation belongs to. Notice that this is not the same as the group allocation vector that is provided by the `match.allocate` function. The binary matching matrix is of size $N \times N$ where 0 indicates that the observations have been part of a different submatch, and 1 indicates that the observations have been part of the same submatch. Diagonal is always 0 although an observation is always in the same submatch with its self.

Usage

```
match.vec2mat(x)
```

Arguments

`x` A matching vector 'x'

Value

N times N binary matching matrix, where 0 indicates that the observations have been part of a different submatch, and 1 indicates that the observations have been part of the same submatch.

Author(s)

Teemu Daniel Laajala <teelaa@utu.fi>

See Also

[match.allocate](#) [match.mat2vec](#) [match.bb](#) [match.dummy](#)

Examples

```
data(vcapwide)

# Construct an Euclidean distance example distance matrix using 15 observations from the VCaP study
d <- as.matrix(dist(vcapwide[1:15,c("PSAWeek10", "BWWeek10")]))

bb3 <- match.bb(d, g=3)
str(bb3)

solvec <- bb3$solution
# matching vector, where each element indicates to which submatch each observation belongs to
```

```

solvec
mat <- match.vec2mat(solvec)
mat
which(mat[1,] == 1)
# E.g. the first, third and thirteenth observation are part of the same submatch
which(solvec == solvec[1])
# Similarly

```

mem.getcomp	<i>Extract per-observation components for fixed and random effects of a mixed-effects model</i>
-------------	---

Description

Assuming a mixed-effects model of form $y_{\text{fit}} = Xb + Zu + e$, where X is the model matrix for fixed effects, Z is the model matrix for random effects, and b and u are the fixed and random effects respectively, this function returns these components per each fitted value y . These may be useful for model inference and/or diagnostic purposes.

Usage

```
mem.getcomp(fit)
```

Arguments

<code>fit</code>	A fitted mixed-effects model generated either through the lme4 or the nlme package.
------------------	---

Details

Notice that per-observation model error is $e = Xb + Zu - y_{\text{observation}}$. Similarly, the components Xb and Zu are extracted.

Value

The function returns per-observation model fit components for a mixed-effects model. The return fields are

<code>Xb</code>	Fixed effects component Xb
<code>Zu</code>	Random effects component Zu
<code>XbZu</code>	Full model fit by summing the above two $Xb+Zu$
<code>e</code>	Model error e
<code>y</code>	Original observations y

Author(s)

Teemu Daniel Laajala <teelaa@utu.fi>

See Also

[mem.plotran](#) [mem.plotresid](#)

Examples

```
data(vcaplong)

exdat <- vcaplong[vcaplong[,"Group"] %in% c("Vehicle", "ARN"),]

library(lme4)
f1 <- lmer(log2PSA ~ 1 + DrugWeek + DrugWeek:ARN + (1 + DrugWeek|ID), data = exdat)

mem.getcomp(f1)
```

mem.plotran

Plot random effects histograms for a fitted mixed-effects model

Description

This plot creates histogram plots for the columns extracted from random effects from a model fit. This is useful for model diagnostics, such as observing deviations from normality in the random effects.

Usage

```
mem.plotran(fit, breaks = 100)
```

Arguments

fit	A fitted mixed-effects model generated either through the lme4 or the nlme package.
breaks	Number of breaks in the histograms (passed to the 'hist'-function)

Author(s)

Teemu Daniel Laajala <teelaa@utu.fi>

See Also

[mem.getcomp](#), [mem.plotresid](#)

Examples

```

data(vcaplong)

exdat <- vcaplong[vcaplong[,"Group"] %in% c("Vehicle", "ARN"),]

library(lme4)
f1 <- lmer(log2PSA ~ 1 + DrugWeek + DrugWeek:ARN + (1 + DrugWeek|ID), data = exdat)

ranef(f1) # Histograms are plotted for these columns
mem.plotran(f1)

```

mem.plotresid *Plot residuals of a mixed-effects model along with trend lines*

Description

This function plots stylized residuals of a mixed-effects model. It is possible to obtain fitted values versus errors ($XbZu$ vs e), or original values versus errors (y vs e) in order to obtain different views to the errors in connection to the observations.

Usage

```
mem.plotresid(fit, linear = T, type = "XbZu", main, xlab, ylab)
```

Arguments

fit	A fitted mixed-effects model generated either through the lme4 or the nlme package.
linear	Should linear trend lines be drawn to the residual plot
type	Type of residual plot; should fitted values (value "XbZu") or original observations (value "y") be plotted against "e" errors
main	Main title
xlab	x-axis label
ylab	y-axis label

Details

Notice that the typical residual plot is fitted values (type="XbZu") versus errors ("e").

Author(s)

Teemu Daniel Laajala <teelaa@utu.fi>

See Also

[mem.getcomp](#), [mem.plotran](#)

Examples

```

data(vcaplong)

exdat <- vcaplong[vcaplong[,"Group"] %in% c("Vehicle", "ARN"),]

library(lme4)
f0 <- lmer(log2PSA ~ 1 + DrugWeek + (1 + DrugWeek|ID), data = exdat)
f1 <- lmer(log2PSA ~ 1 + DrugWeek + DrugWeek:ARN + (1 + DrugWeek|ID), data = exdat)
f2 <- lmer(log2PSA ~ 1 + DrugWeek + DrugWeek:ARN + (1|ID) + (0 + DrugWeek|ID), data = exdat)
f3 <- lmer(log2PSA ~ 1 + DrugWeek + DrugWeek:ARN + (1|Submatch) + (0 + DrugWeek|ID), data = exdat)

par(mfrow=c(2,2))
mem.plotresid(f0)
mem.plotresid(f1)
mem.plotresid(f2)
mem.plotresid(f3)

```

mix.binary

Binary coding of categorical variables

Description

This function encodes categorical variables (e.g. columns of type 'factor' or 'character'). U new columns are created per each such column, where U is the number of unique instances of that column. The new columns are named OriginalColumnName_U1, OriginalColumnName_U2, etc.

Usage

```
mix.binary(x)
```

Arguments

x A data.frame or a matrix where categorical columns are to be binary coded. Categorical columns are assumed to be all non-numeric fields.

Details

A function that codes categorical variables in a dataset into binary variables. This is done in the following manner: e.g. x = red, green, blue, green -> x_new = 1,0,0, 0,1,0, 0,0,1, 0,1,0 where the dimensions in x_new are is_red, is_green and is_blue

Value

The function returns a data.frame, where categorical variables have been replaced with 0/1-binary fields, and numeric fields have been left untouched. Notice that the order of the columns may not be the original.

Author(s)

Teemu Daniel Laajala <teelaa@utu.fi>

Examples

```
data(vcapwide)

ex <- mix.binary(vcapwide[,c("Group", "CastrationDate")])
apply(ex, MARGIN=1, FUN=sum)
# Notice that each row sums to 2, as two categorical variables were binary coded
# and no missing values were present

mix.binary(vcapwide[,c("PSAWeek4", "Group", "CastrationDate")])
# Binary coding is only applied to non-numeric fields
```

mix.fun	<i>Apply function to numerical columns of a mixed data.frame while ignoring non-numeric fields</i>
---------	--

Description

This function is intended for applying functions to numeric fields of a mixed type data.frame. Namely, the function ignores fields that are e.g. factors, and returns FUN function applied to only the numeric fields.

Usage

```
mix.fun(x, FUN = scale, ...)
```

Arguments

x	Data.frame x with mixed type fields
FUN	Function to apply, for example 'scale', 'cov', or 'cor'
...	Additional parameters passed on to FUN

Value

Return values of FUN when applied to numeric columns of 'x'

Author(s)

Teemu Daniel Laajala <teelaa@utu.fi>

See Also

[apply](#)

Examples

```

data(vcapwide)

mix.fun(vcapwide[,c("Group", "PSAWeek4", "PSAWeek10", "PSAWeek14")], FUN=scale)
# Column 'Group' is ignored
mix.fun(vcapwide[,c("Group", "PSAWeek4", "PSAWeek10", "PSAWeek14")], FUN=cov, use="na.or.complete")
# ... is used to pass the 'use' parameter to the 'cov'-function

```

mixplot

Scatterplot for mixed type data

Description

This function plots a scatterplot similar to the default plot-function, with the difference that factor/character fields in input data.frame are handled as categorical variables. These categorical variables are color-coded and handled separately in marginal distributions.

Usage

```

mixplot(x, main = NA, legend = T, col = palette(), na.lines = T,
origin = F, marginal = F, lhei, lwid, verb = 0, ...)

```

Arguments

x	A data.frame or a matrix of observations. Typically x should be a data.frame, where columns are of different types, e.g. some of 'numeric' and some of 'factor' class.
main	Main title plotted on top of the figure
legend	Should an automated legend be generated
col	Colors per observation
na.lines	Should lines be drawn to represent one of the variables if the other one is missing in a 2-dim scatterplot
origin	Should the origin x=0, y=0 be separately indicated using lines
marginal	Should marginal distributions be drawn in sides of each scatterplot
lhei	Heights for bins in the layout
lwid	Widths for bins in the layout
verb	Level of verbosity: -1<= (no verbosity), 0/FALSE (warnings) or >=1/TRUE (additional information)
...	Additional parameters given to the plot-function

Value

An invisible return of the measurements and plot layout structure (matrix, heights, and widths)

Author(s)

Teemu Daniel Laajala <teelaa@utu.fi>

Examples

```
data(vcapwide)

mixplot(vcapwide[,c("Group", "PSAWeek4", "PSAWeek10", "PSAWeek14")], marginal=TRUE, pch=16,
  main="PSA at weeks 4, 10 and 14 per intervention group")
```

smartjitter	<i>Smart jittering function for deterministic shifting of overlapping values</i>
-------------	--

Description

This function takes in a vector of measurements and computes overlapping bins of observations, and applies a jittering function within each overlapping bin.

Usage

```
smartjitter(x, q = seq(from = 0, to = 1, length.out = 10), type = 1,
  amount = 0.1, jitterfuncs = list(function(n) {
    (1:n)/(1/amount)
  }, function(n) {
    (((-1)^c(0:(n - 1))) * (0:(n - 1)))/(1/amount)
  }), jits = jitterfuncs[[type]])
```

Arguments

x	The values that should be jittered. Notice that these are used to determine which are overlapping, and should not be thought of as x-axis positions (see example).
q	Probability quantiles where the ends of the bins should be placed
type	Type of jittering, by default it is used to choose which element (1 or 2) of the list of jittering functions is chosen as the final jittering function. Customized functions may be provided to the jitterfuncs-parameter.
amount	Amount of jittering (here deterministic shifting) for the jittering function
jitterfuncs	List of possible jittering functions for n overlapping values. The jittering function at list position 'type' is chosen
jits	Final jittering function from the jitterfuncs-list

Details

The smart jittering is applied to the x-parameter values, and returns a vector of shifting amounts per each observation. Notice that in the typical case, parameter 'x' are the desired response values e.g. among the y-axis, and the returned value of smartjitter are the amounts of jittering done on the x-axis of a plot.

Value

The function returns a vector of values with same length as `x`. The values in this vector indicate what should be the shifting per each observation, if the observations should be jittered along an another axis.

Author(s)

Teemu Daniel Laajala <teelaa@utu.fi>

Examples

```
data(vcapwide)

plot.new()
plot.window(xlim=extendrange(c(0,1)), ylim=extendrange(vcapwide[, "PSAWeek4"]))
y1 <- vcapwide[vcapwide[, "CastrationDate"]=="100413", "PSAWeek4"]
y2 <- vcapwide[vcapwide[, "CastrationDate"]=="170413", "PSAWeek4"]
points(x=0+smartjitter(y1, type=2, amount=0.02), y=y1)
points(x=1+smartjitter(y2, type=2, amount=0.02), y=y2)
axis(1, at=c(0,1), labels=c("10.04.13", "17.04.13"))
axis(2); box()
title(ylab="PSA at week 4", xlab="Castration batches")
```

vcaplong

Long-format data of the Castration-resistant Prostate Cancer experiment using the VCaP cell line.

Description

The long-format of the VCaP experiment PSA-measurements may be used to model longitudinal measurements during interventions (Vehicle, ARN, or MDV). Body weights and PSA were measured weekly during the experiment. PSA concentrations were log₂-transformed to make data better normally distributed.

Usage

```
data(vcaplong)
```

Format

A data frame with 225 observations on the following 10 variables.

log2PSA Log₂-transformed PSA (prostate-specific antigen) measurements with unit (log₂ ug/l)

BW Body weights (g)

Submatch A grouping factor for indicating which measurements belong to individuals that were part of the same submatch prior to interventions

ID A character vector indicating unique animal IDs

Week Week of the experiment, notice that this is not the same as the week of drug administration (see below)

DrugWeek Week since beginning administration of the drugs

Group Grouping factor for intervention groups of the observations

Vehicle Binary indicator for which observations belonged to the group 'Vehicle'

ARN Binary indicator for which observations belonged to the group 'ARN-509'

MDV Binary indicator for which observations belonged to the group 'MDV3100'

Details

Notice that the long-format is suitable for modeling longitudinal measurements. The grouping factors ID or Submatch could be used to group observations belonging to a single individual or matched individuals.

Source

Knuutila M, Yatkin E, Kallio J, Savolainen S, Laajala TD, et al. (2014) Castration induces upregulation of intratumoral androgen biosynthesis and androgen receptor expression in orthotopic VCaP human prostate cancer xenograft model. *Am J Pathol*. Accepted

References

Laajala TD, Kaur A, Knuutila M, Westermarck J, et al. (2014) Optimal Matching, Randomization and Statistical Modeling of Hierarchical Baseline Variables in Preclinical Cancer Studies. Submitted

Examples

```
data(vcaplong)

str(vcaplong)
head(vcaplong)

library(lattice)
xyplot(log2PSA ~ DrugWeek | Group, data = vcaplong, type="l", group=ID, layout=c(3,1))
xyplot(BW ~ DrugWeek | Group, data = vcaplong, type="l", group=ID, layout=c(3,1))
```

vcapwide	<i>Wide-format data of the Castration-resistant Prostate Cancer experiment using the VCaP cell line.</i>
----------	--

Description

VCaP cancer cells were injected orthotopically into the prostate of mice and PSA (prostate-specific antigen) was followed. The animals were castrated on two subsequent weeks, after which the castration-resistant tumors were allowed to emerge. Since PSA reached pre-castration levels, the animals were non-bipartite matched and allocated to separate intervention arms (at week 10). 3 different interventions are presented here, with 'Vehicle' as a comparison point and MDV3100 and ARN-509 tested for reducing PSA and its correlated tumor size.

Usage

```
data(vcapwide)
```

Format

A data frame with 45 observations on the following 34 variables.

CastrationDate A numeric vector indicating week when the animal was castrated, resulting in steep decrease in PSA and subsequent castration-resistant tumors to emerge.

CageAtAllocation A factorial vector indicating cage labels for each animal at the intervention allocation.

Group A character vector indicating which intervention group the animal was allocated to in the actual experiment (3 alternatives).

TreatmentInitiationWeek A character vector indicating at which week the intervention was started.

Submatch A character vector indicating which submatch the individual was part of the original non-bipartite matching task.

ID A unique character vector indicating the animals.

PSAWeek2 Numeric vector(s) indicating PSA concentration (ug/l) per each week (2 to 14) of the experiment.

PSAWeek3 Numeric vector(s) indicating PSA concentration (ug/l) per each week (2 to 14) of the experiment.

PSAWeek4 Numeric vector(s) indicating PSA concentration (ug/l) per each week (2 to 14) of the experiment.

PSAWeek5 Numeric vector(s) indicating PSA concentration (ug/l) per each week (2 to 14) of the experiment.

PSAWeek6 Numeric vector(s) indicating PSA concentration (ug/l) per each week (2 to 14) of the experiment.

PSAWeek7 Numeric vector(s) indicating PSA concentration (ug/l) per each week (2 to 14) of the experiment.

PSAWeek8 Numeric vector(s) indicating PSA concentration (ug/l) per each week (2 to 14) of the experiment.

PSAWeek9 Numeric vector(s) indicating PSA concentration (ug/l) per each week (2 to 14) of the experiment.

PSAWeek10 Numeric vector(s) indicating PSA concentration (ug/l) per each week (2 to 14) of the experiment.

PSAWeek11 Numeric vector(s) indicating PSA concentration (ug/l) per each week (2 to 14) of the experiment.

PSAWeek12 Numeric vector(s) indicating PSA concentration (ug/l) per each week (2 to 14) of the experiment.

PSAWeek13 Numeric vector(s) indicating PSA concentration (ug/l) per each week (2 to 14) of the experiment.

- PSAWeek14 Numeric vector(s) indicating PSA concentration (ug/l) per each week (2 to 14) of the experiment.
- BWWeek0 Numeric vector indicating body weight (g) of the animals per each week (0 to 14) of the experiment.
- BWWeek1 Numeric vector indicating body weight (g) of the animals per each week (0 to 14) of the experiment.
- BWWeek2 Numeric vector indicating body weight (g) of the animals per each week (0 to 14) of the experiment.
- BWWeek3 Numeric vector indicating body weight (g) of the animals per each week (0 to 14) of the experiment.
- BWWeek4 Numeric vector indicating body weight (g) of the animals per each week (0 to 14) of the experiment.
- BWWeek5 Numeric vector indicating body weight (g) of the animals per each week (0 to 14) of the experiment.
- BWWeek6 Numeric vector indicating body weight (g) of the animals per each week (0 to 14) of the experiment.
- BWWeek7 Numeric vector indicating body weight (g) of the animals per each week (0 to 14) of the experiment.
- BWWeek8 Numeric vector indicating body weight (g) of the animals per each week (0 to 14) of the experiment.
- BWWeek9 Numeric vector indicating body weight (g) of the animals per each week (0 to 14) of the experiment.
- BWWeek10 Numeric vector indicating body weight (g) of the animals per each week (0 to 14) of the experiment.
- BWWeek11 Numeric vector indicating body weight (g) of the animals per each week (0 to 14) of the experiment.
- BWWeek12 Numeric vector indicating body weight (g) of the animals per each week (0 to 14) of the experiment.
- BWWeek13 Numeric vector indicating body weight (g) of the animals per each week (0 to 14) of the experiment.
- BWWeek14 Numeric vector indicating body weight (g) of the animals per each week (0 to 14) of the experiment.

Details

The wide-format here presented the longitudinal measurements for PSA and Body Weight per each column. For modeling the PSA growth longitudinally e.g. using mixed-effects models, see the [vcaplong](#) dataset where the data has been readily transposed into the long-format.

Source

Knuuttila M, Yatkin E, Kallio J, Savolainen S, Laajala TD, et al. (2014) Castration induces upregulation of intratumoral androgen biosynthesis and androgen receptor expression in orthotopic VCaP human prostate cancer xenograft model. *Am J Pathol*. Accepted

References

Laajala TD, Kaur A, Knuutila M, Westermarck J, et al. (2014) Optimal Matching, Randomization and Statistical Modeling of Hierarchical Baseline Variables in Preclinical Cancer Studies. Submitted

See Also

[vcaplong](#)

Examples

```
data(vcapwide)

str(vcapwide)
head(vcapwide)

mixplot(vcapwide[,c("PSAWeek10", "PSAWeek14", "BWWeek10", "Group")], pch=16)
anv <- aov(PSA ~ Group, data.frame(PSA = vcapwide[, "PSAWeek14"], Group = vcapwide[, "Group"]))
summary(anv)
TukeyHSD(anv)
summary(aov(BW ~ Group, data.frame(BW = vcapwide[, "BWWeek14"], Group = vcapwide[, "Group"])))
```


Index

- *Topic **aplot**
 - hmap, 5
 - *Topic **cluster**
 - match.bb, 7
 - *Topic **datasets**
 - vcaplong, 20
 - vcapwide, 21
 - *Topic **design**
 - match.allocate, 6
 - match.bb, 7
 - *Topic **dplot**
 - extendsymrange, 4
 - smartjitter, 19
 - *Topic **hplot**
 - hmap, 5
 - mixplot, 18
 - *Topic **manip**
 - match.dummy, 9
 - match.mat2vec, 10
 - match.vec2mat, 12
 - mix.binary, 16
 - mix.fun, 17
 - *Topic **package**
 - hamlet-package, 2
 - *Topic **regression**
 - mem.getcomp, 13
 - mem.plotran, 14
 - mem.plotresid, 15
- apply, 17
- extendrange, 4
- extendsymrange, 4
- hamlet (hamlet-package), 2
- hamlet-package, 2
- heatmap, 5
- hmap, 5
- match.allocate, 6, 9–12
- match.bb, 7, 7, 10–12
- match.dummy, 7, 9, 9, 11, 12
- match.mat2vec, 7, 9, 10, 10, 12
- match.vec2mat, 7, 9–11, 12
- mem.getcomp, 13, 14, 15
- mem.plotran, 14, 14, 15
- mem.plotresid, 14, 15
- mix.binary, 16
- mix.fun, 17
- mixplot, 18
- smartjitter, 19
- vcaplong, 20, 23, 24
- vcapwide, 21