Package: DoOR.functions (via r-universe)

October 28, 2024

Type Package

Title Integrating Heterogeneous Odorant Response Data into a Common Response Model: A DoOR to the Complete Olfactome

Version 2.0.2

URL https://docs.ropensci.org/DoOR.functions, http://neuro.uni.kn/DoOR, http://dx.doi.org/10.1038/srep21841, http://dx.doi.org/10.1093/chemse/bjq042, https://github.com/ropensci/DoOR.functions

BugReports https://github.com/ropensci/DoOR.functions/issues

Description This is a function package providing functions to perform data manipulations and visualizations for DoOR.data. See the URLs for the original and the DoOR 2.0 publication.

License GPL-3

Encoding UTF-8

Imports utils

Depends DoOR.data (>= 2.0.1), R (>= 3.4.1)

Remotes Dahaniel/DoOR.data@v2.0.1

Suggests ggplot2, grid, gridExtra (>= 2.0.0), knitr, class, rmarkdown, testthat

VignetteBuilder knitr

RoxygenNote 6.0.1

Repository https://ropensci.r-universe.dev

RemoteUrl https://github.com/ropensci/DoOR.functions

RemoteRef master

RemoteSha 14728b933e5829a558687ae04b3f33c366ccede7

Contents

`back_project` *back_project*

Description

project the model response values back into your scale of interest (spike, deltaF/F...)

Usage

```
back_project(template.data, responding.unit,
  response_matrix = door_default_values("door_response_matrix"))
```

Arguments

`template.data`	data frame, the data that provides the scale to transform to, containing InChIKeys in a column called "odorants" and responses in a column called "responses"
`responding.unit`	character, the name of the receptor/OSN/glomerulus which responses should be transformed
`response_matrix`	DoOR response matrix, the source data is picked from here

Details

The process of back projection is the following:

- 1. rescale both data sets to [0,1],
- 2. find the best fitting model between "bp.data" and "cons.data" (lowest MD value),
- 3. project the consensus data onto the fitted curv, these are now our normalized, back projected responses
- 4. rescale all responses to the scale of the original data via a linear fit.

Value

Output of back_project is a list containing a data frame with the back_projected data, the original data, the data used as a template and the data that resulted from fitting source and template (before rescaling to the template scale). additionaly the parameters of the linear fit between the source and template response scale is returned.

Author(s)

Daniel Münch <<daniel.muench@uni-konstanz.de>>

Shouwen Ma <<shouwen.ma@uni-konstanz.de>>

Examples

```
# load some data sets
data(Or22a)
data(door_response_matrix)

# create example data we are going to back project
template.data <- data.frame(odorants = Or22a$InChIKey,
                            responses = Or22a$Hallem.2004.EN)

# run back_project and plot the results
bp <- back_project(template.data, "Or22a")
```

```
plot(bp$back_projected$original.data,
     bp$back_projected$back_projected.data,
     xlab = "DoOR consensus response",
     ylab = "back_projected data [spikes, Hallem.2004.EN]"
)
```

---

calculate_model *' select the best model function*

Description

calculate_model is used to return the best model function that represent the relationship between responses from study x and y.

Usage

```
calculate_model(x, y, select.MD = door_default_values("select.MD"))
```

Arguments

x, y	data vectors from study x and y (can contain NA)
select.MD	logical, if TRUE, only the best model function (in terms of MD) will be returned.

Details

calculate_model chooses the best model function from following: linear, exponential function, sigmoid, asymptotic model with x intercept, asympototic model with y intercept and their inverse versions. (If your are interested in these functions please check the sources at https://github.com/Dahaniel/DoOR.functions)

Author(s)

Shouwen Ma <<shouwen.ma@uni-konstanz.de>>

Daniel Münch <<daniel.muench@uni-konstanz.de>>

Examples

```
# load a data set
library(DoOR.data)
data(Or35a)

# pick 2 data sets for Or35a and rescale the data [0,1]
x <- door_norm(Or35a[,6])
y <- door_norm(Or35a[,9])
# run calculate_model
calM_xy <- calculate_model(x,y, select.MD = door_default_values("select.MD"))
```

count_studies *count_studies*

Description

returns a matrix indiating how many studies have recorded individual receptor-odorant combinations

Usage

```
count_studies(ors = door_default_values("ORs"),
  odor_data = door_default_values("odor"),
  char.columns = door_default_values("num.charColumns"),
  ident = door_default_values("ident"))
```

Arguments

ors	data.frame containing all receptors exidting in DoOR.
odor_data	data.frame containing information about the odorants in DoOR.
char.columns	number of character columns in each receptor data.frame.
ident	odorant identifier to be used as rownames.

Value

matrix

Examples

```
# load some data
library(DoOR.data)
load_door_data(nointeraction = TRUE)

#run count studies and plot the result
count <- count_studies()
image(count)
head(count)
```

create_door_database *Compose a Response Matrix of All Odor Receptors*

Description

computes the complete response model for all receptors in the database (calls model_response for all receptors). Overwrites response_matrix, door_response_matrix_non_normalized and door_excluded_data.

Usage

```
create_door_database(tag = door_default_values("tag"),
  select.MDValue = door_default_values("select.MDValue"),
  overlapValues = door_default_values("overlapValues"), ...)
```

Arguments

tag	character string, format for rownames, possibilities: "InChIKey", CAS", "CID", "Name"
select.MDValue	a numeric, threshold on the MD, this is used to reject studies that do not align sufficiently well to the response model
overlapValues	numeric, a criterion using to refuse a data set that has not enough overlap value.
...	pass more parameters to model_response

Author(s)

Shouwen Ma <<shouwen.ma@uni-konstanz.de>>

Daniel Münch <<daniel.muench@uni-konstanz.de>>

See Also

model_response

Examples

```
## Not run:
# load DoOR data
library(DoOR.data)
load_door_data()

# create a new consensus matrix
create_door_database()

## End(Not run)
```

DoOR.functions.package

DoOR Functions

Description

Functions package providing manipulation and application of the DoOR.

Details

Package:	DoOR.functions
Type:	Package
Version:	2.0.0
Date:	2016-01-25
License:	GPL-3
LazyLoad:	yes

Type `help(package = DoOR.functions)` **to see a complete list of datasets and functions. Below is what you need for a quick start.**

First, load the DoOR packages, data and function package:

```
library(DoOR.functions):
library(DoOR.data):
```

then, load all datasets including the precomputed response matrix:

`load_door_data`: Load all data into current active environment (function comes with DoOR.data) .

or, load all odorant reseponse data into a list:

`load2list`: Load odorant response data only and compose them as a list.

Try some visualizations (e.g. producing the plots from the paper):

`dplot_al_map`:	response to a chemical mapped onto an image of the antennal lobe.
`dplot_compare_profiles`:	compare the results of two studies.
`dplot_response_matrix`:	Dot Plot of Odorant Responses Across Receptors.
`dplot_response_profile`:	bar plot: one receptor, all chemicals.
`dplot_tuningCurve`:	pyramid diagram depicting a receptor's tuning breadth.

Try some queries:

get_responses:	given a chemical, get original responses from all studies in the database.
get_normalized_responses:	given a chemical, get normalised responses from all studies in the database.

In case you wish to create your own response model (e.g. because you want to include your own datasets):

create_door_database:	compute the complete response model for all receptors in the database (calls model_response for
model_response:	run the DoOR algorithm, that merges all measurements for one receptor.

Estimate odorant responses:

estimate_missing_value:	estimate NA entries in a consensus response data.

Project the model response values back to tested values:

back_project:	project the model response values back to tested values.

Introduce new data into DoOR and update the supported data sets:

import_new_data:	import new data into DoOR, and update the weight, response range and receptor names.
update_door_database:	update response matrix by calculating new consensus response data for a given receptor.

See the Vignettes and the help pages for more documentation.

Author(s)

C. Giovanni Galizia
Daniel Muench
Martin Strauch
Anja Nissler
Shouwen Ma

Maintainer: Daniel Münch <daniel.muench@uni-konstanz.de>

References

http://neuro.uni-konstanz.de/DoOR

See Also

DoOR.data

door_default_values *default values for DoOR functions*

Description

door_default_values is used to return default values for DoOR functions.

Usage

```
door_default_values(DoOR_default)
```

Arguments

DoOR_default	a character string, indicating which argument is to be returned for DoOR functions.

Details

There are six categories for default value. real number, integer, logical, NULL, character string and character vector.

Author(s)

Shouwen Ma <<shouwen.ma@uni-konstanz.de>>

Examples

```
# extract DoOR default values
door_default_values(DoOR_default = "select.MD")
```

door_norm *rescale the data values from 0 to 1*

Description

door_norm is used to normalize the data in values from 0 to 1.

Usage

```
door_norm(x)
```

Arguments

x	a numeric vector

Author(s)

Shouwen Ma <<shouwen.ma@uni-konstanz.de>>

Daniel Münch <<daniel.muench@uni-konstanz.de>>

Examples

```
# create example data
x <- rnorm(10)

# run door_norm on it
door_norm(x)
```

`dplot_across_osns` *dplot_across_osns*

Description

plot the activation patterns of one or several odorants across OSNs

Usage

```
dplot_across_osns(odorants,
  response_matrix = door_default_values("door_response_matrix"),
  odor_data = door_default_values("odor"),
  door_mappings = door_default_values("door_mappings"),
  zero = door_default_values("zero"), tag = "Name", sub, plot.type = 1,
  base_size = 12)
```

Arguments

`odorants`	character vector, one or several InChIKeys
`response_matrix`	DOOR response matrix, contains the data to plot
`odor_data`	data frame, contains the odorant information.
`door_mappings`	data frame, containing the mappings of response profiles to morphological structures.
`zero`	character, InChIKey of the odorant that should be set to 0 (e.g. SFR)
`tag`	character, the chemical identifier to use in the plot, one of `colnames(odor)`
`sub`	character vector, specify one or several classes of sensilla the plot should be restricted to. One or several of: "ab" "ac", "at", "ai", "pb", "sacI", "sacII"
`plot.type`	interger, 1 or 2, defining the type of plot (1: facet_grid of odorants * sensillae, 2: facet_wrap across OSNs)
`base_size`	numeric, the base font size for the ggplot2 plot

Details

DoOR response profiles will be selected and ordered according to the OSNs they are related to. Several DoOR response profiles might exist for a given OSN (e.g. one for the OSN itself and one for the OSNs misexpressed receptor protein) but only one will be shown. Which DoOR profile is mapped to which OSN is controlled via the "code.OSN" column in `DoORmapings`.

Value

a ggplot2 object

Author(s)

Daniel Münch <<daniel.muench@uni-konstanz.de>>

Examples

```
# load DoOR data & functions
library(DoOR.data)
library(DoOR.functions)

# pick example odorants by name ans transform their ID to InChIKey
odorants <- trans_id(c("1-butanol", "isopentyl acetate", "carbon dioxide", "water"),
from = "Name", to = "InChIKey")

# plot
dplot_across_osns(odorants)
# plot only across ac and at sensillae
dplot_across_osns(odorants, sub = c("ac", "at"))
# plot across sensory neurons
dplot_across_osns(odorants, plot.type = 2)
```

---

dplot_across_ru *dplot_across_ru*

Description

barplot of DoOR responses of a set of odorant across all responding units in DoOR

Usage

```
dplot_across_ru(odorant,
  response_matrix = door_default_values("door_response_matrix"),
  odor_data = door_default_values("odor"),
  zero = door_default_values("zero"), tag = "Name", limits,
  base_size = 12)
```

Arguments

odorant	character vecto, one or several InChIKeys
response_matrix	DoOR response matrix, a DoOR response matrix as data source
odor_data	data frame, contains the odorant information.
zero	character, an InChIKey of the odorant that should be set to 0
tag	character, the chemical identifier to plot as odorant name (one of colnames(odor))
limits	numeric of length 2, if provided the ylim will range accordingly
base_size	numeric, the base font size for the ggplot2 plot

Value

a ggplot object

Author(s)

Daniel Münch <<daniel.muench@uni-konstanz.de>>

Examples

```
# load data
library(DoOR.data)
library(DoOR.functions)
data(odor)

# plot activation pattern of one or several odorants
dplot_across_ru(trans_id("123-92-2"), tag = "CAS")
dplot_across_ru(odor$InChIKey[4:10])
```

---

dplot_al_map *dplot_al_map*

Description

Plot an antennal lobe map with color coded odorant responses.

Usage

```
dplot_al_map(InChIKey,
  response_matrix = door_default_values("door_response_matrix"),
  odor_data = door_default_values("odor"),
  door_mappings = door_default_values("door_mappings"),
  zero = door_default_values("zero"),
  tag = door_default_values("tag.ALmap"), main = "Name",
  scalebar = door_default_values("scalebar"),
```

```
door_AL_map = door_default_values("door_AL_map"),
colors = door_default_values("colors"), legend = TRUE, limits,
base_size = 12)
```

Arguments

InChIKey	InChIKey specifying the odorant to plot
response_matrix	the input data (e.g. door_response_matrix or door_response_matrix_non_normalized)
odor_data	data frame, contains the odorant information.
door_mappings	the data frame containing the mapping information
zero	the odorant to set to zero (defaults to "SFR")
tag	the labels to plot on top of the glomeruli (one of the following door_mappings columns: "receptor", "sensillum", "ORN", "glomerulus" or "co.receptor")
main	the title, one column of odor, defaults to "Name"
scalebar	whether or not to add a scalebar
door_AL_map	a list containing the AL model
colors	a vector containing 6 color values (2 for values below 0, 1 0 value and 3 steps between 0 and 1)
legend	logical, plot a legend?
limits	the limits for the color scale, if empty the range of the response matrix is taken (after setting "zero" to 0)
base_size	numeric, the base font size for the ggplot plot

Details

Normalized, color coded odor responses across receptors are mapped onto a map of the *Drosophila* antennal lobe. The antennal lobe map was a kind gift from Veit Grabe.

Value

a ggplot2 object

Author(s)

Daniel Münch <daniel.muench@uni-konstanz.de>

Daniel Münch <<daniel.muench@uni-konstanz.de>>

References

Grabe, V., Strutz, A., Baschwitz, A., Hansson, B.S., Sachse, S., 2014. A digital in vivo 3D atlas of the antennal lobe of Drosophila melanogaster. J. Comp. Neurol. n/a–n/a. doi:10.1002/cne.23697

See Also

get_normalized_responses, **ggplot2**, **grid**

Examples

```
# load data
library(DoOR.data)
library(DoOR.functions)

# map responses on antennal lobe scheme
dplot_al_map("MLFHJEHSLIIPHL-UHFFFAOYSA-N", scalebar = FALSE)

# change colors
dplot_al_map("MLFHJEHSLIIPHL-UHFFFAOYSA-N", tag = "Ors",
   color = c("magenta", "pink", "white", "yellow", "orange", "red"))

# pass some ggplot2 theming parameters
dplot_al_map(trans_id("123-92-2"), scalebar = FALSE) +
ggplot2::theme(legend.position  = "bottom",
     panel.background = ggplot2::element_rect(fill = "grey90", color = NA)) +
ggplot2::ggtitle("responses elicited by isopentyl acetate")

# export as pdf
## Not run:
p <- dplot_al_map(trans_id("123-92-2"))
ggplot2::ggsave("AL.response.pdf", p, width = 6, height = 2, scale = 2)

## End(Not run)
```

---

dplot_compare_profiles

Compare two response profiles

Description

Orderdered bar plots for two studies, allowing for an easy comparison of the two studies / response profiles'.

Usage

```
dplot_compare_profiles(x, y, by.x, by.y, tag = "Name", base_size = 12)
```

Arguments

x	the input data frame the first response profile will be taken from
y	the input data frame the second response profile will be taken from (x will be taken if y is missing)
by.x	character string, specifying a column in x
by.y	character string, specifying a column in y
tag	character, the chemical identifier that will be used as odorant label.
base_size	numeric, the base font size for the ggplot2 plot

Author(s)

Daniel Münch <<daniel.muench@uni-konstanz.de>>

Examples

```
# load data
library(DoOR.data)
library(DoOR.functions)
data(Or22a)
data(door_response_range)
data(door_response_matrix)

# compare the Hallem and the Pelz data set for Or22a
dplot_compare_profiles(x = Or22a, y = Or22a,
                         by.x = "Hallem.2006.EN",
                         by.y = "Pelz.2006.AntEC50")

# comparedata from two different sensory neurons and add odorant labels
dplot_compare_profiles(x = cbind(door_response_matrix, InChIKey =
rownames(door_response_matrix)), y = cbind(door_response_matrix, InChIKey =
rownames(door_response_matrix)), by.x = "Or22a", by.y = "Or10a")
```

---

dplot_response_matrix *dplot_response_matrix*

Description

plot DoOR responses as a point matrix

Usage

```
dplot_response_matrix(data, odor_data = door_default_values("odor"),
  tag = door_default_values("tag"), colors = door_default_values("colors"),
  flip = FALSE, fix = TRUE, bw = FALSE, point = FALSE, limits,
  base_size = 12)
```

Arguments

data	a subset of e.g. door_response_matrix
odor_data	data frame, contains the odorant information.
tag	the chemical identfier to plot (one of colnames(odor))
colors	the colors to use if negative values are supplied (range of 5 colors, 2 for negative values, 1 for 0 and 3 for positive values)
flip	logical, if TRUE the x and y axes will be flipped
fix	logical, whether to fix the ratio of the tiles when plotting as a heatmap

bw	logical, whether to plot b&w or colored
point	logical, if TRUE a point matrix instead of a heatmap will be returned (the default if you supply only positive values)
limits	the limits of the scale, will be calculated if not set
base_size	numeric, the base font size for the ggplot2 plot

Value

a dotplot if limits[1] >= 0 or a heatmap if limits[1] < 0

Author(s)

Daniel Münch <<daniel.muench@uni-konstanz.de>>

Examples

```
# load data
library(DoOR.data)
data(door_response_matrix)

# reset the spontaneous firing rate to 0
tmp <- reset_sfr(door_response_matrix, "SFR")

# plot heatmap / coloured tiles
dplot_response_matrix(tmp[10:50,], tag = "Name",
 limits = range(tmp, na.rm = TRUE))

# plot dotplot
dplot_response_matrix(door_response_matrix[10:50,], tag = "Name",
                        limits = range(door_response_matrix, na.rm = TRUE))
```

---

dplot_response_profile

dplot_response_profile

Description

create a barplot of a DoOR response profile

Usage

```
dplot_response_profile(receptor,
  response_matrix = door_default_values("door_response_matrix"),
  odor_data = door_default_values("odor"), tag = door_default_values("tag"),
  colored = TRUE, colors = door_default_values("colors"), limits,
  zero = door_default_values("zero"),
  scalebar = door_default_values("scalebar"), base_size = 12)
```

Arguments

receptor	character, receptor name, any of colnames(door_response_matrix)
response_matrix	a DoOR door_response_matrix
odor_data	data frame, contains the odorant information.
tag	character, chemical identifier for annotation
colored	logical, color code the bars according to the response value?
colors	character vector, a vector of 5 colors (2 for values < 0, 1 value for 0 and 3 values > 0)
limits	numeric of length 2, the limits for the colorscale and the x axis, global range of data will be used if empty
zero	character, the odorant response that is set to 0, defaults to "SFR"
scalebar	logical, add or suppress scalebars
base_size	numeric, the base font size for the ggplot2 plot

Value

ggplot2 plot

Author(s)

Daniel Münch <<daniel.muench@uni-konstanz.de>>

Examples

```
# load data
library(DoOR.data)
data(door_response_matrix)

# plot with default parameters
dplot_response_profile("Or22a", door_response_matrix)

# plot wit odorant names
dplot_response_profile("Or22a", door_response_matrix, tag = "Name")
```

---

dplot_tuningCurve *dplot_tuningCurve*

Description

plot a receptor or odorant tuning curve

Usage

```
dplot_tuningCurve(receptor, odorant, response.vector,
  response_matrix = door_default_values("door_response_matrix"),
  odor_data = door_default_values("odor"),
  zero = door_default_values("zero"),
  fill.receptor = door_default_values("color.receptor"),
  fill.odorant = door_default_values("color.odorant"), odor.main = "Name",
  limits, base_size = 12)
```

Arguments

receptor	character, a receptor name (one of ORS$OR)
odorant	character, an odorant name (InChIKey)
response.vector	numerical vector, a vector with responses, if empty this is taken from door_response_matrix
response_matrix	DoOR response matrix, response vector will be taken from here, not needed if response.vector is given
odor_data	data frame, contains the odorant information.
zero	InChIKey, will be set to zero, default is SFR, ignored when data is provided via response.vector, set to "" if you don't want to subtract anything
fill.receptor	color code, bar color for receptor tuning curve
fill.odorant	color code, bar color for odorant tuning curve
odor.main	the odor identifier to plot, one of colnamed(odor)
limits	the numerical vector of length 2, y limits for the tuning curve
base_size	numeric, the base font size for the ggplot2 plot

Value

a ggplot object

Author(s)

Daniel Münch <<daniel.muench@uni-konstanz.de>>

Examples

```
# load data
library(DoOR.data)
data(door_response_matrix)
data(odor)

# plot a tuning curve for an odorant
dplot_tuningCurve(odorant = odor$InChIKey[2])
# or for a receptor
dplot_tuningCurve(receptor = "Or22a")
```

```
# adjust the plotting range
range <- range(reset_sfr(door_response_matrix, "SFR"), na.rm = TRUE)
dplot_tuningCurve(receptor = "Or10a", limits = range,
                  fill.receptor = "magenta")

# plot with manual input data as receptor tuning curve
dplot_tuningCurve(receptor = "receptor X", response.vector = c(1:100))
# or as odor tuning curve
dplot_tuningCurve(odorant = "odor X", response.vector = rnorm(200))
```

`estimate_missing_value` *Estimate the missing entries in a response data*

Description

Estimate the missing entries in a response data

Usage

```
estimate_missing_value(data, nodor, method = "PC")
```

Arguments

`data`	a data frame or matrix, contaning the consensus response values
`nodor`	a numeric value, specifying the number of the selected odors
`method`	character string, specifying the method ("PC" (Pearson's coefficient) and "Knn" (k nearest neighbors)) for estimation, the default is "PC".

Details

A wrapper programe for using Pearson Correlation or k-nearest neighbors to estimate the missing entries in a response matrix.

Author(s)

Shouwen Ma <<shouwen.ma@uni-konstanz.de>>

References

Kim, H., Golub, G. H. & Park, H., Missing value estimation for DNA microarray gene expression data: local least squares imputation., 2005, Bioinformatics, 21, 187-198

Examples

```
## Not run:
# load data
library(DoOR.data)
data(door_response_matrix)

# pick an example subset
subset <- door_response_matrix[1:100, 11:30]

# estimate missing values
est.data <- estimate_missing_value(data = subset, nodor = 6)

## End(Not run)
```

`export_door_data` *export data*

Description

export odor response data and supported data

Usage

```
export_door_data(file.format, directory,
  odorantReceptors = door_default_values("ORs"),
  response_matrix = door_default_values("door_response_matrix"),
  responseRange = door_default_values("door_response_range"),
  unglobalNorm_RM = door_default_values("door_response_matrix_non_normalized"),
  weightGlobNorm = door_default_values("door_global_normalization_weights"),
  all.data = TRUE)
```

Arguments

`file.format`	character string, the format of given file, either ".txt" or ".csv"
`directory`	character string, naming a directory for writing. If missing, the exported data are saved in current working directory.
`odorantReceptors`	data frame, receptor names and expressions
`response_matrix`	data matrix, an global unnormalized responses matrix
`responseRange`	data frame, response ranges for each study
`unglobalNorm_RM`	data matrix, an unnormalized responses matrix
`weightGlobNorm`	data frame, weight matrix for global normalizazion
`all.data`	logical, if TRUE, export odorant response data and supported data "door_response_matrix", "door_response_range", "door_response_matrix_non_normalized", "door_response_matrix", "door_global_normalization_weights" and "ORs".

Details

Please load ORs from data package DoOR.data by typing (`data(ORs)`) before use.

Author(s)

Shouwen Ma <<shouwen.ma@uni-konstanz.de>>

Examples

```
## Not run:
# load data
library(DoOR.data)
library(DoOR.functions)
load_door_data()

# export odorant response data only
export_door_data(".txt", all.data = FALSE)

## End(Not run)
```

`get_dataset` *getDataset*

Description

aggregates original data from a given study

Usage

```
get_dataset(study, na.rm = FALSE)
```

Arguments

`study`	character, the name of the study you want to aggregate the dta from
`na.rm`	logical, whether or not to exclude odorants that were not measured in the study

Value

returns a data frame containing all the odorant responses measured in `study`

Author(s)

Daniel Münch <<daniel.muench@uni-konstanz.de>>

Examples

```
# load data
library(DoOR.data)
load_door_data(nointeraction = TRUE)

# get all recordings from the Hallem.2004.EN data set
get_dataset("Hallem.2004.EN", na.rm = TRUE)
```

`get_normalized_responses` *Find normalised receptor responses*

Description

given a chemical, get normalised receptor responses from all studies in the database.

Usage

```
get_normalized_responses(odors, zero = door_default_values("zero"),
  response_matrix = door_default_values("door_response_matrix"), round = 3,
  na.rm = FALSE)
```

Arguments

`odors`	character vector, one or more odors provided as InChIKey.
`zero`	InChIKey of background that should be set to zero. The default is "SFR", i.e. the spontaneous firing rate.
`response_matrix`	a data frame, as e.g. "door_response_matrix" that is loaded by `model_response`. It is also possible to create this frame manually using `model_response`.
`round`	numeric, round to this amount of digits, set to NA if you do not want to round
`na.rm`	logical, remove NAs?

Author(s)

Daniel Münch <daniel.muench@uni-konstanz.de>

See Also

`model_response`,`create_door_database`

Examples

```
# load data
library(DoOR.data)
data(door_response_matrix)

# define a list of odorants
odors <- c("MLFHJEHSLIIPHL-UHFFFAOYSA-N",
           "OBNCKNCVKJNDBV-UHFFFAOYSA-N",
           "IKHGUXGNUITLKF-UHFFFAOYSA-N")

# get the normalized responses for these odorants
result <- get_normalized_responses(odors,
                                   response_matrix = door_response_matrix)
```

`get_responses` *Find receptor responses*

Description

given a chemical, get original receptor responses from all studies in the database.

Usage

```
get_responses(odorant,
  responseRange = door_default_values("door_response_range"),
  Or.list = load2list())
```

Arguments

`odorant`	a single odor provided as InChIKey
`responseRange`	data frame, response ranges of studies
`Or.list`	a list contains reponse data of all available receptors. It can be loaded using `load2list`.

Details

output is a data frame containing response values of given odor across receptors from all available studies.

Author(s)

Daniel Münch <daniel.muench@uni-konstanz.de>

Examples

```
# load data
library(DoOR.data)
load_door_data(nointeraction = TRUE)

# get raw responses for odorant MLFHJEHSLIIPHL-UHFFFAOYSA-N
responses <- get_responses(odorant = 'MLFHJEHSLIIPHL-UHFFFAOYSA-N')
```

`identify_sensillum` *identify_sensillum*

Description

correlates the result from a SSR recording of several odorants against all DoOR response profiles

Usage

```
identify_sensillum(recording,
  response_matrix = door_default_values("door_response_matrix"),
  odor_data = door_default_values("odor"),
  door_mappings = door_default_values("door_mappings"), tag = "Name",
  min.cor = 0.9, nshow = 10, method = "cor", sub, base_size = 12,
  plot = TRUE, use = "everything")
```

Arguments

`recording`	data frame, a data frame with the following columns "odorants" containing InChIKeys of the tested odorrant, and one column called "unit1" etc. for each unit, containing responses (or estimates) scaled between 0 and 1 (see examples)
`response_matrix`	DoOR response matrix, the data to compair against
`odor_data`	data frame, contains the odorant information.
`door_mappings`	the data frame containing the mapping information
`tag`	character, the chemical identifier to use in plots, one of `colnames(odor)`
`min.cor`	numeric, a minimum correlation value, the function will check wether there is a higher correlation for all units within a single sensillum
`nshow`	numeric, the number of plots to nshow, plot e.g. only the top 10 matches
`method`	character, the method for similarity calculations: correlation ("cor") or Euclidean distances ("dist")
`sub`	character, if you know the class of sensillum you were recording from you can restrict the search to this subset here ("ab", "ac", "at", "pb", "sac")
`base_size`	numeric, the base font size of the ggplot plots
`plot`	logical, if TRUE returns the plot, else returns the data frame with the correlations/distances

use — character, the "use" option from the cor function, "all" returns NA when pairs are incomplete, "na.or.complete" only uses complete observations to calculate correlations; see cor for details

Value

eithe& Carolin G.(†27)r a plot (gtable) with responses sorted by highest correlations or lowest distances, or a data frame containing all calculated correlations or Euclidean distances

Author(s)

Daniel Münch <<daniel.muench@uni-konstanz.de>>

Examples

```
# load data
library(DoOR.data)

# create an example recording
recording <- data.frame(
   odorants = c(trans_id(c("BEDN", "ETAS"), "Code"),
   trans_id("carbon dioxide", "Name")),
   unit1 = c(.9,.1,.1),
   unit2 = c(0, .1, 1)
)

# run the identification
identify_sensillum(recording)
identify_sensillum(recording, method = "dist", nshow = 5)
```

---

import_new_data *Import new data into DoOR*

Description

Import or update new data and update door_global_normalization_weights, door_response_range, odor, ORs and receptor data frames.

Usage

```
import_new_data(file.name,
  dataFormat = door_default_values("door_data_format"),
  odor_data = door_default_values("odor"),
  weightGlobNorm = door_default_values("door_global_normalization_weights"),
  responseRange = door_default_values("door_response_range"),
  receptors = door_default_values("ORs"),
  ident = door_default_values("ident"), round = 3)
```

Arguments

file.name	character string, the name of given file that contains response values of one or more odorant receptors, either a .csv or .txt file.
dataFormat	data frame, a data frame does not contain any response value but odorant information.
odor_data	data frame, contains the odorant information.
weightGlobNorm	data matrix, indicates whether given receptor has been measured by given study.
responseRange	data frame, contains the information about response range of each study and how many odors have been measured in each study.
receptors	data frame, contains the receptor and OSN names and their expression.
ident	the identifier used for matching, usually the InChIKey is used.
round	the number of digits the imported values are rounded to.

Details

import_new_data is used to import new data into database. If the data contains a new receptor or ORN, then build a new data frame for this receptor or ORN. If the data contains a receptor that is already present in database, then merge the imported data into old data frame. The information (e.g. response range, how many receptors and odors were measured from given study) will be integrated into data door_response_range, odor, ORs and door_global_normalization_weights. If an existing study is imported, remove_study will be run first in order to perform an update.

Author(s)

Shouwen Ma <<shouwen.ma@uni-konstanz.de>>

Daniel Münch <<daniel.muench@uni-konstanz.de>>

Examples

```
## Not run:
import new data named "odorantResponses_Orx.txt" into database and update the
support data.
library(DoOR.data)
import_new_data(file.name = "odorantResponses_Orx.csv")

## End(Not run)
```

---

load2list *load2list*

Description

returns all original DoOR response data as a list

Usage

```
load2list()
```

Value

a list

Author(s)

Daniel Münch <<daniel.muench@uni-konstanz.de>>

Examples

```
# load DoOR.data
library(DoOR.data)
load_door_data(nointeraction = TRUE)

# write the data into a list
lst <- load2list()
```

---

map_receptor *map_receptor*

Description

Identifying the source of unknown response data by correlating it agains all DoOR responding units.

Usage

```
map_receptor(data,
  response_matrix = door_default_values("door_response_matrix"), sub,
  threshold.p, threshold.cor, nshow)
```

Arguments

data	data frame, containing two columns, one called "odorants" and one "responses" providing InChIKeys and odorant responses respectively.
response_matrix	output is a numeric vector that contains the Pearson Correlation Coefficient between given data and selected consensus data in
sub	character, a subset of responding units returned response matrix
threshold.p	numeric, a p-value threshold, only correlations below will be returned
threshold.cor	numeric, a correlation-coefficient threshold, only correlations above will be returned
nshow	numeric, if defined, only this number of results will be

Author(s)

Shouwen Ma <<shouwen.ma@uni-konstanz.de>>

Daniel Münch <<daniel.muench@uni-konstanz.de>>

Examples

```
# load data
load_door_data(nointeraction = TRUE)

# pick example data
data <- data.frame(odorants  = Or22a$InChIKey,
                   responses = Or22a$Hallem.2004.EN)
data <- na.omit(data)

# find the corresponding receptor / responding unit
map_receptor(data = data)
```

`model_response` *Generates a model response*

Description

Runs the DoOR algorithm, that merges all measurements for one receptor into a common response model.

Usage

```
model_response(da, select.MDValue = door_default_values("select.MDValue"),
  overlapValues = door_default_values("overlapValues"),
  responseRange = door_default_values("door_response_range"),
  weightGlobNorm = door_default_values("door_global_normalization_weights"),
  glob.normalization = door_default_values("glob.normalization"),
  plot = door_default_values("plot"))
```

Arguments

`da`	data frame, a selected receptor containing measured responses from studies.
`select.MDValue`	numeric, threshold on the MD for rejecting a fit.
`overlapValues`	numeric, a criterion using to refuse a data set that has not enough overlap value.
`responseRange`	data frame, contains response ranges for all studies.
`weightGlobNorm`	data frame, a binary data matrix, 1 indicates given odor has been measured in given study, NA indicates NOT.
`glob.normalization`	logical, default is `TRUE`, performs a global normalization for the model response. Otherwise (`FALSE`) response values will be given in value from 0 to 1.
`plot`	logical, If `FALSE`, plotting is suppressed. Default is `FALSE`.

Details

Merging a data is processed by following:

1. Normalize all response data in value [0,1].
2. Compute the correlation between studies and selected the best pair using `select_model`.
3. Merge the first pair using function `project_points`.
4. Add other datasets if the correlation between the growing model response and the new dataset is below the correlation threshold (select.MDValue). Datasets excluded based on this criterion will be appended in a separate list.

Author(s)

Shouwen Ma <<shouwen.ma@uni-konstanz.de>>

Examples

```
# load data
library(DoOR.data)
data(Or35a)
data(door_global_normalization_weights)
data(door_response_range)

# merge all existing data sets for Or35a into a consensus model response
model_response_Or35a <- model_response(Or35a, plot = TRUE)
```

---

model_response_seq *model_response_seq*

Description

generates a model response and merge data in given sequence

Usage

```
model_response_seq(data, SEQ,
  overlapValues = door_default_values("overlapValues"),
  select.MDValue = door_default_values("select.MDValue"), strict = TRUE,
  plot = FALSE)
```

Arguments

`data`	data frame, odorant response data, e.g. Or22a.
`SEQ`	character vector, containing the names of studies indicating given sequence for merging data.
`overlapValues`	minimum overlap between studies to perfom a merge

select.MDValue	the minimum mean distance between studies to perfom a merge
strict	logical, if TRUE merging a permutation will be stopped once a single merge has a mean distance above select.MDValue
plot	logical

Details

model_response_seq.R:

merges studies in a given sequence (determined by the user or by exhaustive enumeration and choosing the optimal sequence)

input parameters:

data : data frame, odorant response data for a given receptor, e.g. Or22a # SEQ : character vector, contains the names of studies that measured this receptor in a specific order (the merging sequence)

output is a numeric vector: response values

Author(s)

Shouwen Ma <<shouwen.ma@uni-konstanz.de>>

Examples

```
# load data
library(DoOR.data)
data(Or35a)
data(door_response_range)

# specify a sequence of merging
SEQ <- c("Hallem.2006.EN","Kreher.2008.EN","Hallem.2006.EN")

# perform the merging
selected_merg <- model_response_seq(Or35a, SEQ = SEQ, plot = TRUE)
```

---

private_odorant *private_odorant*

Description

return an odorant that activates the receptor of interest exclusively

Usage

```
private_odorant(receptor, sensillum = FALSE,
  response_matrix = door_default_values("door_response_matrix"),
  door_mappings = door_default_values("door_mappings"),
  zero = door_default_values("zero"), nshow = 5, tag)
```

Arguments

receptor	character, name of a DoOR responding unit (one of ORs$Or)
sensillum	logical, restrict the search to the sensillum the receptor is expressed in?
response_matrix	DoOR response matrix, the input data to perform the search on
door_mappings	the data frame containing the mapping information
zero	character, an odorant that should be set to 0
nshow	numeric, the number of private odorants to return
tag	character, the chemical identifier to give the odorant names in (on of colnames(odor))

Value

a data.frame containing odorants and the response in the receptor of interest as well as the maximum response of the remaining receptors and their difference

Examples

```
# load data
library(DoOR.data)

# find a private odorant for Gr21a.Gr63a (the carbon dioxide receptor)
# private_odorant("Gr21a.Gr63a", tag = "Name")

# now find an odorant that within the ab3 sensillum specifically activates
# Or22a
private_odorant("Or22a", tag = "Name", sensillum = TRUE)
```

---

project_points *project_points*

Description

projects data points onto the curve defined by the model function

Usage

```
project_points(x, y, xylim, best.model, plot = door_default_values("plot"),
  points_cex = door_default_values("points.cex"),
  title = door_default_values("title"), ...)
```

Arguments

x, y	numeric vectors of data values, coordinate vectors of points to plot, the coordinates can contain NA values.
xylim	numeric vectors, x, y limits of the plot.
best.model	a list, containing the parameters, function, inverse function, Leibniz's notation for distance calculation and MD value. if missing, the best model will be generated automatically.
plot	logical, If FALSE, plotting is suppressed. Default is FALSE.
points_cex	a numerical value, giving the magnification level of symbols relative to the default size.
title	logical, If TRUE, title is shown. Default is FALSE.
...	further graphical parameters

Details

For internal use in the merging process (see also model_response). The model function is choosen by calculate_model. project_points then projects the data points from the datasets to be merged onto the curve defined by the model function. It computes the closest distance from a data point to a point on the curve by numerical optimisation.

A list with two data frames "double.observations" and "single.observations" is returned, which give the coordinates of double observations (defined as (x,y)) and coordinates of single observation (defined as (x,NA) or (NA,y)). Both data frames contain seven columns: "ID" indicating the original position of data x and y, "x", "y" indicating the coordinate of observation, "X", "Y" indicating the coordinate of projected point on the function, "distance" indicating the distances between (xmin, f(xmin)) and all points on the functional line, "NDR" indicating the normalized distances across all the distance values.

Author(s)

Shouwen Ma <<shouwen.ma@uni-konstanz.de>>

See Also

calculate_model, optimize, integrate

Examples

```
# load data
library(DoOR.data)
data(Or23a)

# normalize two example data sets
x <- door_norm(Or23a[,'Hallem.2004.EN'])
y <- door_norm(Or23a[,'Hallem.2006.EN'])

# find the best fitting function and project the remaining points (measured
# only in one of the data sets) onto the fit.
project_points(x = x, y = y, plot = TRUE)
```

remove_study *Remove a study from DoOR*

Description

Use this function to remove a study from the DoOR database. `import_new_data.R` uses this function when it detects an existing study during the import process (e.g. because you imported updated data).

Usage

```
remove_study(study, receptors = door_default_values("ORs"),
  responseRange = door_default_values("door_response_range"),
  weightGlobNorm = door_default_values("door_global_normalization_weights"))
```

Arguments

`study`	a string containing the name of the study you want to remove (e.g. 'Bruyne.2001.WT')
`receptors`	a vector of all the receptors to be checked. Defaults to all receptors exidting in DoOR.
`responseRange`	the dataframe containing the info about the response ranges of all studies (`door_response_range`)
`weightGlobNorm`	the dataframe containing the info about the relative weights between receptors (`door_global_normalization_weights`)

Author(s)

Daniel Münch <<daniel.muench@uni-konstanz.de>>

See Also

`import_new_data`

Examples

```
# load data
library(DoOR.data)
load_door_data(nointeraction = TRUE)

# remove Bruyne.2001.WT from DoOR
remove_study('Bruyne.2001.WT')
```

reset_sfr *reset SFR*

Description

A function for reseting SFR to zero

Usage

```
reset_sfr(x, sfr)
```

Arguments

x	numeric or DoOR response matrix, input values
sfr	numeric or character, either a value to subtract if x is a vector or an InChIKey if x is a DoOR response matrix

Details

Performs a simple subtraction of the SFR value.

Author(s)

Daniel Münch <<daniel.muench@uni-konstanz.de>>

Examples

```
# load data
library(DoOR.data)
data(door_response_matrix)

# create a response matrix with the SFR reset to 0
door_response_matrix_SFRreset <- reset_sfr(door_response_matrix, "SFR")
```

select_model *compute the data pairwise and and selects a pair with the lowest "MD" value.*

Description

compute the data pairwise using function calculate_model and selects a pair with the lowest "MD" value.

Usage

```
select_model(candidate, data_candidate, merged_data,
  overlapValues = door_default_values("overlapValues"),
  merged = door_default_values("merged"))
```

Arguments

candidate	a character vector, contains the names of studies.
data_candidate	a data frame, odorant response data that only contains value columns.
merged_data	numeric vector, merged data
overlapValues	numeric, a criterion using to refuse a data set that has not enough overlap value.
merged	logical, if merged is TRUE, calculate models between merged_data and candidate data. If FALSE, calculate models between candidates.

Details

This function is used in model_response to select the first pair or next data set for merging. The output is a list containing "selected.x" and "selected.y" that specify which data plots against another, and "best.model" that is used in function project_points.

Author(s)

Shouwen Ma <<shouwen.ma@uni-konstanz.de>>

See Also

project_points,model_response

Examples

```
# load data
library(DoOR.data)
data(ac3B)

# split into data and header
studies <- names(ac3B)[c(7:8)]
data_candidate <- ac3B[,c(7:8)]

# rescale data
norm_data_candidate <- apply(data_candidate, 2, door_norm)

# find the best fitting model
select_model(candidate = studies, data_candidate = norm_data_candidate,
             merged = FALSE)
```

sparse *Calculate the sparseness of a vector.*

Description

Sparseness can be calculated as lifetime kurtosis (LTK, Willmore and Tolhurst, 2001) or as lifetime sparseness (LTS, Bhandawat et al., 2007).

Usage

```
sparse(x, method = "ltk")
```

Arguments

x	numerical input vector
method	type of sparseness measure, either 'ltk' for lifetime kurtosis or 'lts' for lifetime sparseness (see references).

Details

LTS scales between \[0,1\] while LTK is not restricted. LTS only takes positive values.

Author(s)

Daniel Münch <<daniel.muench@uni-konstanz.de>>

References

Bhandawat, V., Olsen, S.R., Gouwens, N.W., Schlief, M.L., Wilson, R.I., 2007. Sensory processing in the Drosophila antennal lobe increases reliability and separability of ensemble odor representations. Nature neuroscience 10, 1474–82. doi:10.1038/nn1976

Willmore, B., Tolhurst, D.J., 2001. Characterizing the sparseness of neural codes. Network 12, 255–270. doi:10.1080/net.12.3.255.270

trans_id *trans_id*

Description

Translate chemical identifiers from one to the other.

Usage

```
trans_id(x, from = "CAS", to = "InChIKey",
  odor_data = door_default_values("odor"))
```

Arguments

x	character vector, one or many chemical identifiers
from	character, the type of identifier to translate from (one of the column names of "odor")
to	character, the type of identifier to translate from (one of the column names of "odor")
odor_data	the data frame containing the odor information (defaults to "odor").

Value

character vector of translated chemical identifiers

Examples

```
# load data
library(DoOR.data)

# transform CAS to InChIKey
trans_id("123-92-2")

# transform Name to InChIKey
trans_id("isopentyl acetate", "Name")

# transform SMILE to InChIKey
trans_id("C(C(C)C)COC(=O)C", "SMILES", "Name")
```

---

update_door_database *update response matrix*

Description

update the globally response matrix and the unglobally normalized response matrix door_response_matrix_non_normali
by introducing new consensus response data of given receptor.

Usage

```
update_door_database(receptor, permutation = TRUE, perm,
 response_matrix_nn = door_default_values("door_response_matrix_non_normalized"),
  response_matrix = door_default_values("door_response_matrix"),
  responseRange = door_default_values("door_response_range"),
  weightGlobNorm = door_default_values("door_global_normalization_weights"),
  select.MDValue = door_default_values("select.MDValue"), strict = TRUE,
  overlapValues = door_default_values("overlapValues"),
  door_excluded_data = door_default_values("door_excluded_data"),
  plot = FALSE)
```

Arguments

receptor	character string, name of given odorant receptor.
permutation	logical, if TRUE, the sequence is chosen from permutation, if FALSE, sequence is chosen by the routine process.
perm	a matrix with one sequence of study names per row, if empty, all possible permutations of study names will be provided.
response_matrix_nn	data frame, response data that has not been globally normalized.
response_matrix	data frame, globally normalized response data.
responseRange	data frame, response range of studies.
weightGlobNorm	data frame, weight matrix for global normalization.
select.MDValue	the minimum mean distance between studies to perfom a merge (used if permutation == FALSE or if permutation == TRUE AND strict == TRUE)
strict	logical, if TRUE merging a permutation will be stopped once a single merge has a mean distance above select.MDValue (only valid if permutation == TRUE)
overlapValues	minimum overlap between studies to perfom a merge
door_excluded_data	the data frame that contains the list of excluded data sets.
plot	logical

Details

The merging sequence could be arranged by the routine process (using model_response or taking the optimized sequence that is chosen from permutations. The mean correlation between merged responses and each original recording will be computed for each permutation, the optimozed sequence is with the highest correlation.

Author(s)

Shouwen Ma <<shouwen.ma@uni-konstanz.de>>

Shouwen Ma <<daniel.muench@uni-konstanz.de>>

See Also

model_response,model_response_seq

Examples

```
## Not run:
# load data
library(DoOR.data)
load_door_data()
# update the entry "Or67b" of data "door_response_matrix" and
# "door_response_matrix_non_normalized" with permutations.
 update_door_database(receptor="Or67b", permutation = TRUE)

## End(Not run)
```

update_door_odorinfo *update_door_odorinfo*

Description

Update the DoOR odor data with info from odor. For the function to work, all DoOR data has to be loaded to the current environment.

Usage

```
update_door_odorinfo()
```

Author(s)

Daniel Münch, <daniel@muench.bio>

Examples

```
# load data
load_door_data(nointeraction = TRUE)
# modify odor
odor[1,1] <- "acid"

# run
update_door_odorinfo()

# check that data sets have been updated
head(Or22a)
```

Index