

# Package: nullabor (via r-universe)

November 1, 2024

**Version** 0.3.12

**Description** Tools for visual inference. Generate null data sets and null plots using permutation and simulation. Calculate distance metrics for a lineup, and examine the distributions of metrics.

**Title** Tools for Graphical Inference

**Maintainer** Di Cook <dicook@monash.edu>

**License** GPL (>= 2)

**URL** <https://github.com/dicook/nullabor>

**BugReports** <https://github.com/dicook/nullabor/issues>

**Imports** MASS, moments, fpc, ggplot2, dplyr, purrr, tidyr, tibble, magrittr

**Suggests** forecast, viridis, knitr

**LazyData** true

**Type** Package

**LazyLoad** false

**VignetteBuilder** knitr

**RoxygenNote** 7.3.1

**Encoding** UTF-8

**NeedsCompilation** no

**Author** Hadley Wickham [aut, ctb]  
(<<https://orcid.org/0000-0003-4757-117X>>), Niladri Roy Chowdhury [aut, ctb], Di Cook [aut, cre]  
(<<https://orcid.org/0000-0002-3813-7155>>), Heike Hofmann [aut, ctb] (<<https://orcid.org/0000-0001-6216-5183>>)

**Repository** CRAN

**Date/Publication** 2024-07-21 06:40:02 UTC

## Contents

aud . . . . .	2
bin_dist . . . . .	3
box_dist . . . . .	3
calc_diff . . . . .	4
calc_mean_dist . . . . .	5
decrypt . . . . .	6
distmet . . . . .	6
distplot . . . . .	8
electoral . . . . .	9
lal . . . . .	9
lineup . . . . .	10
null_dist . . . . .	11
null_lm . . . . .	12
null_permute . . . . .	13
null_ts . . . . .	13
opt_bin_diff . . . . .	14
pvisual . . . . .	15
reg_dist . . . . .	16
resid_boot . . . . .	17
resid_pboot . . . . .	18
resid_rotate . . . . .	18
resid_sigma . . . . .	19
rorschach . . . . .	19
sample_size . . . . .	20
sep_dist . . . . .	20
theme_strip . . . . .	21
tips . . . . .	22
turk_results . . . . .	22
uni_dist . . . . .	23
visual_power . . . . .	23
wasps . . . . .	24
<b>Index</b>	<b>25</b>

---

 aud

---

*Conversion rate of 1 Australian Dollar (AUD) to 1 US Dollar*


---

## Description

The dataset consists of the daily exchange rates of 1 Australian Dollar to 1 US Dollar between Jan 9 2018 and Feb 21 2018.

---

bin_dist	<i>Binned Distance</i>
----------	------------------------

---

**Description**

Data X is binned into X.bin bins in x-direction and Y.bins in y-direction. The number of points in each cell is then counted. Same is done for data PX. An euclidean distance is calculated between the number of points in each cell between X and PX.

**Usage**

```
bin_dist(X, PX, lineup.dat = lineup.dat, X.bin = 5, Y.bin = 5)
```

**Arguments**

X	a data.frame with two variables, the first two columns are used
PX	another data.frame with two variables, the first two columns are used
lineup.dat	lineup data so that the binning is done based on the lineup data and not the individual plots, by default lineup.dat = lineup.dat ; if one wishes to calculate the binned distance between two plots, one should use lineup.dat = NULL
X.bin	number of bins on the x-direction, by default X.bin = 5
Y.bin	number of bins on the y-direction, by default Y.bin = 5

**Value**

distance between X and PX

**Examples**

```
with(mtcars, bin_dist(data.frame(wt, mpg), data.frame(sample(wt), mpg),
lineup.dat = NULL))
```

---

box_dist	<i>Distance based on side by side Boxplots</i>
----------	--

---

**Description**

Assuming that data set X consists of a categorical group variable a numeric value, a summary of the first quartile, median and third quartile of this value is calculated for each group. The extent (as absolute difference) of the minimum and maximum value across groups is computed for first quartile, median and third quartile. Same is done for data PX. Finally an euclidean distance is calculated between the absolute differences of X and PX.

**Usage**

```
box_dist(X, PX)
```

**Arguments**

X	a data.frame with one factor variable and one continuous variable
PX	a data.frame with one factor variable and one continuous variable

**Value**

distance between X and PX

**Examples**

```
if(require('dplyr')) {
  with(mtcars,
    box_dist(data.frame(as.factor(am), mpg),
      data.frame(as.factor(sample(am)), mpg))
  )
}
```

---

calc_diff	<i>Calculating the difference between true plot and the null plot with the maximum distance.</i>
-----------	--

---

**Description**

Distance metric is used to calculate the mean distance between the true plot and all the null plots in a lineup. The difference between the mean distance of the true plot and the maximum mean distance of the null plots is calculated.

**Usage**

```
calc_diff(lineup.dat, var, met, pos, dist.arg = NULL, m = 20)
```

**Arguments**

lineup.dat	lineup data to get the lineup
var	a vector of names of the variables to be used to calculate the difference
met	distance metric needed to calculate the distance as a character
pos	position of the true plot in the lineup
dist.arg	a list or vector of inputs for the distance metric met; NULL by default
m	number of plots in the lineup, by default m = 20

**Value**

difference between the mean distance of the true plot and the maximum mean distance of the null plots

**Examples**

```

if(require('dplyr')){
  lineup.dat <- lineup(null_permute('mpg'), mtcars, pos = 1)
  calc_diff(lineup.dat, var = c('mpg', 'wt'), met = 'bin_dist',
    dist.arg = list(lineup.dat = lineup.dat, X.bin = 5, Y.bin = 5), pos = 1, m = 8)}

if(require('dplyr')){
  calc_diff(lineup(null_permute('mpg'), mtcars, pos = 1), var = c('mpg', 'wt'), met = 'reg_dist',
    dist.arg = NULL, pos = 1, m = 8)}

```

---

calc_mean_dist	<i>Calculating the mean distances of each plot in the lineup.</i>
----------------	---

---

**Description**

Distance metric is used to calculate the mean distance between the true plot and all the null plots in a lineup. The mean distances of each null plot to all the other null plots are calculated. The mean distances are returned for all the plots in the lineup.

**Usage**

```
calc_mean_dist(lineup.dat, var, met, pos, dist.arg = NULL, m = 20)
```

**Arguments**

lineup.dat	lineup data of the lineup
var	a vector of names of the variables to be used to calculate the mean distances
met	distance metric needed to calculate the distance as a character
pos	position of the true plot in the lineup
dist.arg	a list or vector of inputs for the distance metric met; NULL by default
m	number of plots in the lineup, by default m = 20

**Value**

the mean distances of each plot in the lineup

**Examples**

```

if(require('dplyr')){
  calc_mean_dist(lineup(null_permute('mpg'), mtcars, pos = 1), var = c('mpg', 'wt'),
    met = 'reg_dist', pos = 1, m = 10)}

```

---

 decrypt

*Use decrypt to reveal the position of the real data.*


---

### Description

The real data position is encrypted by the lineup function, and writes this out as a text string. Decrypt, decrypts this text string to reveal which where the real data is.

### Usage

```
decrypt(...)
```

### Arguments

... character vector to decrypt

### Examples

```
decrypt('0uXR2p rut L202')
```

---

 distmet

*Empirical distribution of the distance*


---

### Description

The empirical distribution of the distance measures is calculated based on the mean distance of each of the null plots from the other null plots in a lineup. At this moment this method works only for [null\\_permute](#) method. This function helps get some assessment of whether the actual data plot is very different from the null plots.

### Usage

```
distmet(
  lineup.dat,
  var,
  met,
  method,
  pos,
  repl = 1000,
  dist.arg = NULL,
  m = 20
)
```

**Arguments**

<code>lineup.dat</code>	lineup data
<code>var</code>	a vector of names of the variables to be used
<code>met</code>	distance metric needed to calculate the distance as a character
<code>method</code>	method for generating null data sets
<code>pos</code>	position of the observed data in the lineup
<code>repl</code>	number of sets of null plots selected to obtain the distribution; 1000 by default
<code>dist.arg</code>	a list or vector of inputs for the distance metric <code>met</code> ; NULL by default
<code>m</code>	the number of plots in the lineup; <code>m = 20</code> by default

**Value**

`lineup` has the data used for the calculations

`null_values` contains new null samples from which to compare nulls in `lineup`

`diff` difference in distance between nulls and actual data and that of the null that is most different from other nulls. A negative value means that the actual data plot is similar to the null plots.

`closest` list of the five closest nulls to the actual data plot

`pos` position of the actual data plot in the lineup

**Examples**

```
# Each of these examples uses a small number of nulls (m=8), and a small number of
# repeated sampling from the null distribution (repl=100), to make it faster to run.
# In your own examples you should think about increasing each of these, at least to the defaults.
## Not run:
if (require('dplyr')) {
  d <- lineup(null_permute('mpg'), mtcars, pos = 1)
  dd <- distmet(d, var = c('mpg', 'wt'),
    'reg_dist', null_permute('mpg'), pos = 1, repl = 100, m = 8)
  distplot(dd, m=8)
}

## End(Not run)

## Not run:
d <- lineup(null_permute('mpg'), mtcars, pos=4, n=8)
library(ggplot2)
ggplot(d, aes(mpg, wt)) + geom_point() + facet_wrap(~ .sample, ncol=4)
if (require('dplyr')) {
  dd <- distmet(d, var = c('mpg', 'wt'), 'bin_dist', null_permute('mpg'),
    pos = 4, repl = 100, dist.arg = list(lineup.dat = d, X.bin = 5,
    Y.bin = 5), m = 8)
  distplot(dd, m=8)
}

## End(Not run)
```

```

# Example using bin_dist
## Not run:
if (require('dplyr')) {
  d <- lineup(null_permute('mpg'), mtcars, pos = 1)
  library(ggplot2)
  ggplot(d, aes(mpg, wt)) + geom_point() + facet_wrap(~ .sample, ncol=5)
  dd <- distmet(d, var = c('mpg', 'wt'),
    'bin_dist', null_permute('mpg'), pos = 1, repl = 500,
    dist.arg = list(lineup.dat = d, X.bin = 5, Y.bin = 5))
  distplot(dd)
}

## End(Not run)

# Example using uni_dist
## Not run:
mod <- lm(wt ~ mpg, data = mtcars)
resid.dat <- data.frame(residual = mod$resid)
d <- lineup(null_dist('residual', dist = 'normal'), resid.dat, pos=19)
ggplot(d, aes(residual)) + geom_histogram(binwidth = 0.25) + facet_wrap(~ .sample, ncol=5)
if (require('dplyr')) {
  dd <- distmet(d, var = 'residual', 'uni_dist', null_dist('residual',
    dist = 'normal'), pos = 19, repl = 500)
  distplot(dd)
}

## End(Not run)

```

---

distplot

*Plotting the distribution of the distance measure*


---

## Description

The permutation distribution of the distance measure is plotted with the distances for the null plots. Distance measure values for the null plots and the true plot are overlaid.

## Usage

```
distplot(dat, m = 20)
```

## Arguments

dat	output from <a href="#">distmet</a>
m	the number of plots in the lineup; m = 20 by default

**Examples**

```
## Not run:
if (require('dplyr')) {
  d <- lineup(null_permute('mpg'), mtcars, pos = 1)
  library(ggplot2)
  ggplot(d, aes(mpg, wt)) + geom_point() + facet_wrap(~.sample)
  distplot(d, var = c('mpg', 'wt'), 'reg_dist', null_permute('mpg'),
    pos = 1, repl = 100, m = 8, m = 8)
}

## End(Not run)
```

electoral

*Polls and election results from the 2012 US Election***Description**

Polls and election results from the 2012 US Election

**Format**

A list with two data frames: polls is a data frame of 51 rows and 4 variables

**State** State name

**Electoral.vote** Number of electoral votes in the 2012 election

**Margin** Margin between the parties with the highest number of votes and second highest number of votes. These margins are based on polls.

**Democrat** logical vector True, if the democratic party is the majority party in this state.

‘election’ is a data frame of 51 rows and 5 variables

**State** State name

**Candidate** character string of the winner: Romney or Obama

**Electoral.vote** Number of electoral votes in the 2012 election

**Margin** Margin between the parties with the highest number of votes and second highest number of votes. These margins are based on the actual election outcome

**Democrat** logical vector True, if the democratic party is the majority party in this state.

lal

*Los Angeles Lakers play-by-play data.***Description**

Play by play data from all games played by the Los Angeles lakers in the 2008/2009 season.

lineup

*The line-up protocol.***Description**

In this protocol the plot of the real data is embedded amongst a field of plots of data generated to be consistent with some null hypothesis. If the observe can pick the real data as different from the others, this lends weight to the statistical significance of the structure in the plot. The protocol is described in Buja, Cook, Hofmann, Lawrence, Lee, Swayne, Wickham (2009) Statistical inference for exploratory data analysis and model diagnostics, Phil. Trans. R. Soc. A, 367, 4361-4383.

**Usage**

```
lineup(method, true = NULL, n = 20, pos = sample(n, 1), samples = NULL)
```

**Arguments**

method	method for generating null data sets
true	true data set. If NULL, <a href="#">find_plot_data</a> will attempt to extract it from the current ggplot2 plot.
n	total number of samples to generate (including true data)
pos	position of true data. Leave missing to pick position at random. Encrypted position will be printed on the command line, <a href="#">decrypt</a> to understand.
samples	samples generated under the null hypothesis. Only specify this if you don't want lineup to generate the data for you.

**Details**

Generate  $n - 1$  null datasets and randomly position the true data. If you pick the real data as being noticeably different, then you have formally established that it is different to with p-value  $1/n$ .

**Examples**

```
library(ggplot2)
ggplot(lineup(null_permute('mpg'), mtcars), aes(mpg, wt)) +
  geom_point() +
  facet_wrap(~ .sample)
ggplot(lineup(null_permute('cyl'), mtcars),
  aes(mpg, .sample, colour = factor(cyl))) +
  geom_point()
```

---

null_dist	<i>Generate null data with a specific distribution.</i>
-----------	---

---

## Description

Null hypothesis: variable has specified distribution

## Usage

```
null_dist(var, dist, params = NULL)
```

## Arguments

var	variable name
dist	distribution name. One of: beta, cauchy, chisq, exp, f, gamma, geom, lnorm, logis, nbinom, binom, norm, pois, t, unif, weibull
params	list of parameters of distribution. If NULL, will use <a href="#">fitdistr</a> to estimate them.

## Value

a function that given data generates a null data set. For use with [lineup](#) or [rorschach](#)

## See Also

[null\\_permute](#), [null\\_lm](#)

## Examples

```
dframe <- data.frame(x = rnorm(150))
library(ggplot2)
# three histograms of normally distributed values
ggplot(
  data=rorschach(method=null_dist("x", "norm"), n = 3, true=dframe)
) +
  geom_histogram(aes(x=x, y=..density..), binwidth=0.25) +
  facet_grid(.~.sample) +
  geom_density(aes(x=x), colour="steelblue", size=1)

# uniform distributions are not as easy to recognize as such
dframe$x = runif(150)
ggplot(
  data=rorschach(method=null_dist("x", "uniform",
    params=list(min=0, max=1)),
    n = 3, true=dframe)) +
  geom_histogram(aes(x=x, y=..density..), binwidth=0.1) +
  facet_grid(.~.sample) +
  geom_density(aes(x=x), colour="steelblue", size=1)
```

---

null\_lm

---

*Generate null data with null residuals from a model.*


---

## Description

Null hypothesis: variable is linear combination of predictors

## Usage

```
null_lm(f, method = "rotate", ...)
```

## Arguments

f	model specification formula, as defined by <a href="#">lm</a>
method	method for generating null residuals. Built in methods 'rotate', 'pboot' and 'boot' are defined by <a href="#">resid_rotate</a> , <a href="#">resid_pboot</a> and <a href="#">resid_boot</a> respectively
...	other arguments passed onto method.

## Value

a function that given data generates a null data set. For use with [lineup](#) or [rorschach](#)

## See Also

[null\\_permute](#), [null\\_dist](#)

## Examples

```
data(tips)
x <- lm(tip ~ total_bill, data = tips)
tips.reg <- data.frame(tips, .resid = residuals(x), .fitted = fitted(x))
library(ggplot2)
ggplot(lineup(null_lm(tip ~ total_bill, method = 'rotate'), tips.reg)) +
  geom_point(aes(x = total_bill, y = .resid)) +
  facet_wrap(~ .sample)
```

---

null_permute	<i>Generate null data by permuting a variable.</i>
--------------	--

---

**Description**

Null hypothesis: variable is independent of others

**Usage**

```
null_permute(var)
```

**Arguments**

var	name of variable to permute
-----	-----------------------------

**Value**

a function that given data generates a null data set. For use with [lineup](#) or [rorschach](#)

**See Also**

[null\\_lm](#), [null\\_dist](#)

**Examples**

```
data(mtcars)
library(ggplot2)
ggplot(data=rorschach(method=null_permute("mpg"), n = 3, true=mtcars)) +
  geom_boxplot(aes(x=factor(cyl), y=mpg, fill=factor(cyl))) +facet_grid(.~.sample) +
  theme(legend.position="none", aspect.ratio=1)
```

---

null_ts	<i>Generate null data by simulating from a time series model.</i>
---------	---

---

**Description**

Null hypothesis: data follows a time series model using auto.arima from the forecast package

**Usage**

```
null_ts(var, modelfn)
```

**Arguments**

var	variable to model as a time series
modelfn	method for simulating from ts model.

**Value**

a function that given data generates a null data set. For use with [lineup](#) or [rorschach](#)

**See Also**

`null_model`

**Examples**

```
require(forecast)
require(ggplot2)
require(dplyr)
data(aud)
l <- lineup(null_ts("rate", auto.arima), aud)
ggplot(l, aes(x=date, y=rate)) + geom_line() +
  facet_wrap(~.sample, scales="free_y") +
  theme(axis.text = element_blank()) +
  xlab("") + ylab("")
l_dif <- l %>%
  group_by(.sample) %>%
  mutate(d=c(NA,diff(rate))) %>%
  ggplot(aes(x=d)) + geom_density() +
  facet_wrap(~.sample)
```

---

opt\_bin\_diff

*Finds the number of bins in x and y direction which gives the maximum binned distance.*

---

**Description**

This function finds the optimal number of bins in both x and y direction which should be used to calculate the binned distance. The binned distance is calculated for each combination of provided choices of number of bins in x and y direction and finds the difference using `calc_diff` for each combination. The combination for which the difference is maximum should be used.

**Usage**

```
opt_bin_diff(
  lineup.dat,
  var,
  xlow,
  xhigh,
  ylow,
  yhigh,
  pos,
  plot = FALSE,
  m = 20
)
```

**Arguments**

lineup.dat	lineup data to get the lineup
var	a list of names of the variables to be used to calculate the difference
xlow	the lowest value of number of bins on the x-direction
xhigh	the highest value of number of bins on the x-direction
ylow	the lowest value of number of bins on the y-direction
yhigh	the highest value of number of bins on the y-direction
pos	position of the true plot in the lineup
plot	LOGICAL; if true, returns a tile plot for the combinations of number of bins with the differences as weights
m	number of plots in the lineup, by default $m = 20$

**Value**

a dataframe with the number of bins and differences the maximum mean distance of the null plots

**Examples**

```
if(require('dplyr')){
  opt_bin_diff(lineup(null_permute('mpg'), mtcars, pos = 1), var = c('mpg', 'wt'),
    2, 5, 4, 8, pos = 1, plot = TRUE, m = 8)
}
```

---

pvisual

*P-value calculations.*


---

**Description**

These set of functions allow the user to calculate a p-value from the lineup after it has been evaluated by  $K$  independent observers. The different functions accommodate different lineup construction and showing to observers. Details are in the papers Majumder et al (2012) JASA, and Hofmann et al (2015). We distinguish between three different scenarios:

- Scenario I: in each of  $K$  evaluations a different data set and a different set of  $(m-1)$  null plots is shown.
- Scenario II: in each of  $K$  evaluations the same data set but a different set of  $(m-1)$  null plots is shown.
- Scenario III: the same lineup, i.e. same data and same set of null plots, is shown to  $K$  different observers.

**Usage**

```
pvisual(
  x,
  K,
  m = 20,
  N = 10000,
  type = "scenario3",
  xp = 1,
  target = 1,
  upper.tail = TRUE
)
```

**Arguments**

x	number of observed picks of the data plot
K	number of evaluations
m	size of the lineup
N	MC parameter: number of replicates on which MC probabilities are based. Higher number of replicates will decrease MC variability.
type	type of simulation used: scenario 3 assumes that the same lineup is shown in all K evaluations
xp	exponent used, defaults to 1
target	integer value identifying the location of the data plot
upper.tail	compute probabilities $P(X \geq x)$ . Be aware that the use of this parameter is not consistent with the other distribution functions in base. There, a value of $P(X > x)$ is computed for upper.tail=TRUE.

**Value**

Vector/data frame. For comparison a p value based on a binomial distribution is provided as well.

**Examples**

```
pvisual(15, 20, m=3) # triangle test
```

---

reg\_dist

*Distance based on the regression parameters*


---

**Description**

Dataset X is binned into 5 bins in x-direction. A regression line is fitted to the data in each bin and the regression coefficients are noted. Same is done for dataset PX. An euclidean distance is calculated between the two sets of regression parameters. If the relationship between X and PX looks linear, number of bins should be equal to 1.

**Usage**

```
reg_dist(X, PX, nbins = 1, intercept = TRUE, scale = TRUE)
```

**Arguments**

X	a data.frame with two variables, the first column giving the explanatory variable and the second column giving the response variable
PX	another data.frame with two variables, the first column giving the explanatory variable and the second column giving the response variable
nbins	number of bins on the x-direction, by default nbins = 1
intercept	include the distances between intercepts?
scale	logical value: should the variables be scaled before computing regression coefficients?

**Value**

distance between X and PX

**Examples**

```
with(mtcars, reg_dist(data.frame(wt, mpg), data.frame(sample(wt), mpg)))
```

---

resid\_boot

*Bootstrap residuals.*

---

**Description**

For use with [null\\_lm](#)

**Usage**

```
resid_boot(model, data)
```

**Arguments**

model	to extract residuals from
data	used to fit model

---

resid_pboot	<i>Parametric bootstrap residuals.</i>
-------------	--

---

**Description**

For use with [null\\_lm](#)

**Usage**

```
resid_pboot(model, data)
```

**Arguments**

model	to extract residuals from
data	used to fit model

---

resid_rotate	<i>Rotation residuals.</i>
--------------	----------------------------

---

**Description**

For use with [null\\_lm](#)

**Usage**

```
resid_rotate(model, data)
```

**Arguments**

model	to extract residuals from
data	used to fit model

---

resid_sigma	<i>Residuals simulated by a normal model, with specified sigma</i>
-------------	--

---

**Description**

For use with [null\\_lm](#)

**Usage**

```
resid_sigma(model, data, sigma = 1)
```

**Arguments**

model	to extract residuals from
data	used to fit model
sigma	a specific sigma to model

---

rorschach	<i>The Rorschach protocol.</i>
-----------	--------------------------------

---

**Description**

This protocol is used to calibrate the eyes for variation due to sampling. All plots are typically null data sets, data that is consistent with a null hypothesis. The protocol is described in Buja, Cook, Hofmann, Lawrence, Lee, Swayne, Wickham (2009) Statistical inference for exploratory data analysis and model diagnostics, Phil. Trans. R. Soc. A, 367, 4361-4383.

**Usage**

```
rorschach(method, true = NULL, n = 20, p = 0)
```

**Arguments**

method	method for generating null data sets
true	true data set. If NULL, <a href="#">find_plot_data</a> will attempt to extract it from the current ggplot2 plot.
n	total number of samples to generate (including true data)
p	probability of including true data with null data.

---

sample_size	<i>Sample size calculator</i>
-------------	-------------------------------

---

### Description

This function calculates a table of sample sizes for with an experiment, given a lineup size, and estimates of the detection rate.

### Usage

```
sample_size(n = 53:64, m = 20, pA = seq(1/20, 1/3, 0.01), conf = 0.95)
```

### Arguments

n	range of sample sizes to check, default is 53:64
m	linup size, default 20
pA	range of estimated detection rates to consider, default is seq(1/20, 1/3, 0.01)
conf	confidence level to use to simulate from binomial

### Examples

```
pow <- sample_size()
pow
library(ggplot2)
library(viridis)
ggplot(pow, aes(x=n, y=pA, fill=prob, group=pA)) +
  geom_tile() +
  scale_fill_viridis_c("power") +
  ylab("detect rate (pA)") + xlab("sample size (n)") +
  theme_bw()
```

---

sep_dist	<i>Distance based on separation of clusters</i>
----------	---

---

### Description

The separation between clusters is defined by the minimum distances of a point in the cluster to a point in another cluster. The number of clusters are provided. If not, the hierarchical clustering method is used to obtain the clusters. The separation between the clusters for dataset X is calculated. Same is done for dataset PX. An euclidean distance is then calculated between these separation for X and PX.

### Usage

```
sep_dist(X, PX, clustering = FALSE, nclust = 3, type = "separation")
```

**Arguments**

X	a data.frame with two or three columns, the first two columns providing the dataset
PX	a data.frame with two or three columns, the first two columns providing the dataset
clustering	LOGICAL; if TRUE, the third column is used as the clustering variable, by default FALSE
nclust	the number of clusters to be obtained by hierarchical clustering, by default nclust = 3
type	character string to specify which measure to use for distance, see ?cluster.stats for details

**Value**

distance between X and PX

**Examples**

```
if(require('fpc')) {
  with(mtcars, sep_dist(data.frame(wt, mpg, as.numeric(as.factor(mtcars$cyl))),
    data.frame(sample(wt), mpg, as.numeric(as.factor(mtcars$cyl))),
    clustering = TRUE))
}

if (require('fpc')) {
  with(mtcars, sep_dist(data.frame(wt, mpg, as.numeric(as.factor(mtcars$cyl))),
    data.frame(sample(wt), mpg, as.numeric(as.factor(mtcars$cyl))),
    nclust = 3))
}
```

---

 theme\_strip

---

*A theme to minimally strip away the context*


---

**Description**

Note this is not a complete theme hence why there are no arguments.

**Usage**

```
theme_strip()
```

**Examples**

```
library(ggplot2)
ggplot(cars, aes(dist, speed)) + theme_strip()
```

---

tips	<i>Tipping data</i>
------	---------------------

---

**Description**

One waiter recorded information about each tip he received over a period of a few months working in one restaurant. He collected several variables:

**Usage**

tips

**Format**

A data frame with 244 rows and 7 variables

**Details**

- tip in dollars,
- bill in dollars,
- sex of the bill payer,
- whether there were smokers in the party,
- day of the week,
- time of day,
- size of the party.

In all he recorded 244 tips. The data was reported in a collection of case studies for business statistics (Bryant & Smith 1995).

**References**

Bryant, P. G. and Smith, M (1995) *Practical Data Analysis: Case Studies in Business Statistics*. Homewood, IL: Richard D. Irwin Publishing:

---

turk_results	<i>Sample turk results</i>
--------------	----------------------------

---

**Description**

Subset of data from a Turk experiment, used to show how to compute power of a lineup

---

uni_dist	<i>Distance for univariate data</i>
----------	-------------------------------------

---

**Description**

The first four moments is calculated for data X and data PX. An euclidean distance is calculated between these moments for X and PX.

**Usage**

```
uni_dist(X, PX)
```

**Arguments**

X	a data.frame where the first column is only used
PX	another data.frame where the first column is only used

**Value**

distance between X and PX

**Examples**

```
if(require('moments')){uni_dist(rnorm(100), rpois(100, 2))}
```

---

visual_power	<i>Power calculations.</i>
--------------	----------------------------

---

**Description**

This function simply counts the proportion of people who selected the data plot, in a set of lineups. It adjusts for multiple picks by the same individual, by weighting by the total number of choices.

**Usage**

```
visual_power(data, m = 20)
```

**Arguments**

data	summary of the results, containing columns id, pic_id, response, detected
m	size of the lineup

**Value**

vector of powers for each pic\_id

**Examples**

```
data(turk_results)
visual_power(turk_results)
```

---

wasps	<i>Wasp gene expression data.</i>
-------	-----------------------------------

---

**Description**

Data from Toth et al (2010) used in Niladri Roy et al (2015)

# Index

- \* **datasets**
  - tips, [22](#)
- aud, [2](#)
- bin\_dist, [3](#)
- box\_dist, [3](#)
- calc\_diff, [4](#)
- calc\_mean\_dist, [5](#)
- decrypt, [6](#), [10](#)
- distmet, [6](#), [8](#)
- distplot, [8](#)
- electoral, [9](#)
- find\_plot\_data, [10](#), [19](#)
- fitdistr, [11](#)
- lal, [9](#)
- lineup, [10](#), [11–14](#)
- lm, [12](#)
- null\_dist, [11](#)
- null\_lm, [12](#), [17–19](#)
- null\_permute, [6](#), [13](#)
- null\_ts, [13](#)
- opt\_bin\_diff, [14](#)
- pvisual, [15](#)
- reg\_dist, [16](#)
- resid\_boot, [12](#), [17](#)
- resid\_pboot, [12](#), [18](#)
- resid\_rotate, [12](#), [18](#)
- resid\_sigma, [19](#)
- rorschach, [11–14](#), [19](#)
- sample\_size, [20](#)
- sep\_dist, [20](#)
- theme\_strip, [21](#)
- tips, [22](#)
- turk\_results, [22](#)
- uni\_dist, [23](#)
- visual\_power, [23](#)
- wasps, [24](#)