

# Package: antibioticR (via r-universe)

September 20, 2026

**Type** Package

**Title** Analysis of Antibiotic Resistance Data

**Version** 0.3.4

**Date** 2026-08-05

**Author** Thomas Petzoldt

**Encoding** UTF-8

**Maintainer** Thomas Petzoldt <thomas.petzoldt@tu-dresden.de>

**Description** Antibiotic resistant bacteria are a serious threat to human health. This package aims to collect tools and functions to analyze environmental resistance data, especially minimal inhibitory concentration ('MIC') and inhibition zone diameters ('ZD').

**License** GPL (>= 2)

**Depends** reshape2, methods

**Imports** bbmle, evmix

**Suggests** knitr, dplyr, ggplot2, shiny, rhandsontable, rmarkdown

**VignetteBuilder** knitr

**Collate** 'antibioticR-package.R' 'ECOFFfinder.R' 'aaa\_oldclasses.R'  
'abr\_baseclass.R' 'abr\_densfunctions.R' 'abr\_methods.R'  
'abr\_quantile.R' 'abr\_summary.R' 'antibioticR-internal.R'  
'ecofffinder-internal-plot-decoff.R'  
'ecofffinder-internal-plot-pecoff.R' 'ecofffinder-plot-methods.R'  
'ecofffinder\_nls.R' 'ecofffinder\_startpar.R' 'fit\_unimix.R'  
'micdata.R' 'mixParmDf2parmlist.R' 'mxObj-class.R'  
'mxObj-methods.R' 'mxObj-coerce.R' 'mx\_compare.R'  
'mx\_diagnostics.R' 'mx\_guess\_components.R' 'mx\_metafit.R'  
'mx\_parmlist\_sort.R' 'mx\_plot.R' 'mx\_results.R' 'parmlist.R'  
'peakwindow.R' 'reformat\_eucast.R' 'turnpoints.R' 'unbin.R'  
'unimix-distributions-internal.R' 'unimix-methods.R'  
'utilities-internal.R' 'weighted.sd.R' 'zd\_main\_components.R'

**Config/roxygen2/version** 8.0.0

**Config/pak/sysreqs** libgs10-dev libicu-dev

**Repository** <https://tpetzoldt.r-universe.dev>

**Date/Publication** 2026-08-04 22:34:20 UTC

**RemoteUrl** <https://github.com/tpetzoldt/antibioticR>

**RemoteRef** HEAD

**RemoteSha** 76d52990a110f074f764bbe0d4ea496522b1c2d3

## Contents

|                                          |    |
|------------------------------------------|----|
| antibioticR-package . . . . .            | 3  |
| abr_baseclass . . . . .                  | 4  |
| abr_density . . . . .                    | 4  |
| abr_quantile . . . . .                   | 7  |
| coerce . . . . .                         | 7  |
| diagnostics . . . . .                    | 8  |
| dunimix . . . . .                        | 9  |
| ECOFFinder . . . . .                     | 10 |
| ecoffinder_nls . . . . .                 | 11 |
| ecoffinder_startpar . . . . .            | 12 |
| fit . . . . .                            | 14 |
| fit_unimix . . . . .                     | 15 |
| micdata . . . . .                        | 16 |
| mixParmDf2parmlist . . . . .             | 17 |
| mx_compare . . . . .                     | 17 |
| mx_guess_components . . . . .            | 18 |
| mx_metafit . . . . .                     | 19 |
| mx_parmlist_sort . . . . .               | 20 |
| mx_plot . . . . .                        | 21 |
| mxMle-class . . . . .                    | 22 |
| mxObj-class . . . . .                    | 23 |
| nls-class . . . . .                      | 23 |
| obs . . . . .                            | 23 |
| parmList . . . . .                       | 24 |
| peakwindow . . . . .                     | 25 |
| plot,abr_ecoffinder,ANY-method . . . . . | 26 |
| reformat_eucast . . . . .                | 27 |
| results . . . . .                        | 28 |
| summary,abr_baseclass-method . . . . .   | 29 |
| turnpoints . . . . .                     | 29 |
| unbin . . . . .                          | 30 |
| weighted.sd . . . . .                    | 31 |
| zd_main_components . . . . .             | 32 |

|              |           |
|--------------|-----------|
| <b>Index</b> | <b>33</b> |
|--------------|-----------|

## Description

The package aims to improve accesibility to statistical methods for analysing populations of resistant and non-resistant bacteria from an environmental, i.e. non-clinical perspective. The methods are intended to describe sensitivity, tolerance and resistance on a sub-acute level in order to compare populations of different origin on gradual scales.

## Details

The package supports currently three methods: (1) Kernel density smoothing for getting mean values and multiple modes from the distributions, (2) an implementation of the ECOFFinder algorithm (Turnidge, 2006) with automatic start value estimation and a shiny app for interactive use, and (3) Maximum likelihood estimation of multi-modal normal and exponential-normal mixtures.

## Author(s)

Thomas Petzoldt

## References

Bolker, Ben and R Development Core Team (2017) bbmle: Tools for General Maximum Likelihood Estimation. R package version 1.0.20. <https://CRAN.R-project.org/package=bbmle>

Gruen, Bettina and Leisch, Friedrich (2008) FlexMix Version 2: Finite mixtures with concomitant variables and varying and constant parameters. Journal of Statistical Software, 28(4), 1-35 [doi:10.18637/jss.v028.i04](https://doi.org/10.18637/jss.v028.i04)

The European Committee on Antimicrobial Susceptibility Testing (2018). Breakpoint tables for interpretation of MICs and zone diameters, version 8.0, [http://www.eucast.org/clinical\\_breakpoints/](http://www.eucast.org/clinical_breakpoints/) accessed: 2018-07-09

Turnidge, J., Kahlmeter, G., Kronvall, G. (2006) Statistical characterization of bacterial wild-type MIC value distributions and the determination of epidemiological cut-off values. Clin Microbial Infect 12: 418-425 [doi:10.1111/j.14690691.2006.01377.x](https://doi.org/10.1111/j.14690691.2006.01377.x)

Venables, W. N. and Ripley, B. D. (2002) Modern Applied Statistics with S. Fourth Edition. Springer, New York. ISBN 0-387-95457-0

---

|               |                                                  |
|---------------|--------------------------------------------------|
| abr_baseclass | <i>Classes of package <b>antibioticR</b> ...</i> |
|---------------|--------------------------------------------------|

---

### Description

The classes define ...

### Details

The abr\_ecofffinder: class represents a distribution component fitted with the ECOFFinder method.

---

|             |                                                    |
|-------------|----------------------------------------------------|
| abr_density | <i>Kernel Density Estimate for ZD and MIC data</i> |
|-------------|----------------------------------------------------|

---

### Description

Estimate kernel density for antibiotic resistance data using either standard density estimation with Gaussian kernels or boundary corrected density estimation with function pbckden from package **evmix**.

### Usage

```
abr_density(
  x,
  cutoff = 5.5,
  method = c("density", "evmix"),
  control = abr_density.control()
)

abr_cumdens(
  x,
  cutoff = 5.5,
  method = c("evmix", "density", "spline", "fmm"),
  control = abr_density.control()
)

abr_density.control(bw = 1, cut = 3, n=512, bcmethod = "renorm", proper = TRUE)

abr_cumdensquant(
  x,
  p = 0.5,
  cutoff = 5.5,
  method = c("density", "evmix", "spline", "fmm"),
  control = abr_density.control()
)
```

```

maxdens(
  x,
  cutoff = 5.5,
  from = min(x),
  to = max(x),
  method = c("density", "evmix"),
  control = abr_density.control()
)

```

## Arguments

|          |                                                                                                                                                                                                                                                                                                                                                                                                |
|----------|------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| x        | vector of antibiotic resistance observations, given as zone diameter (zd) or log of minimum inhibitory concentration (mic). Note that for <code>abr_density</code> x must contain raw observations, not frequencies, cf. <a href="#">unbin</a> , while for <code>abr_cumdens</code> x and y contain binned data. If y is omitted, x must contain a two-column matrix with value and frequency. |
| cutoff   | cutoff value (disc diameter) in case of zd data                                                                                                                                                                                                                                                                                                                                                |
| method   | character which density method to use: 'density', 'evmix' (function <code>pbckden</code> ), 'spline', or 'fmm' (monotone spline)                                                                                                                                                                                                                                                               |
| control  | options passed to the density estimation methods, see <a href="#">abr_density.control</a>                                                                                                                                                                                                                                                                                                      |
| bw       | the smoothing bandwidth to be used, see <a href="#">density</a> for details.                                                                                                                                                                                                                                                                                                                   |
| cut      | by default, the values of from and to are cut bandwidths beyond the extremes of the data, see <a href="#">density</a> for details                                                                                                                                                                                                                                                              |
| n        | the number of equally spaced points at which the density is to be estimated, see <a href="#">density</a> for details                                                                                                                                                                                                                                                                           |
| bcmethod | character, boundary correction method                                                                                                                                                                                                                                                                                                                                                          |
| proper   | logical, whether density is renormalised to integrate to unity                                                                                                                                                                                                                                                                                                                                 |
| p        | vector of probability quantiles                                                                                                                                                                                                                                                                                                                                                                |
| from, to | search range for maximum (= mode) search                                                                                                                                                                                                                                                                                                                                                       |

## Value

data frame with

- x zone diameter resp. mic value
- y density estimate

## See Also

[stats::density], [evmix::dbckden]

## Examples

```

opar <- par(no.readonly = TRUE)
## ===== minimum inhibitory concentrations (MIC) data =====
data("micdata")
micdata$freq <- with(micdata, ifelse(is.na(freq), 0, freq))

par(las = 2, mar=c(9,4,4,2)+.1)

##' relative frequencies
y <- micdata$freq / sum(micdata$freq)
x <- log2(micdata$conc)

plot(y ~ x, type = "h", axes = FALSE, xlab="", ylab="rel. frequency")
axis(2)
axis(1)
axis(1, line=3, at = log2(micdata$conc), label = round(micdata$conc, 4))
mtext("log2(conc)", side=1, line=1.5, las=1)
mtext("conc", side=1, line=6, las=1)

##' standard R function with weights
dens <- density(x, weights=y, bw=2/3)
lines(dens)

## ===== zone diameter (ZD) data =====
freq <- c(36, 0, 2, 3, 4, 8, 9, 14, 10, 9, 3, 1, 8, 2, 4, 8, 20,
          45, 40, 54, 41, 22, 8, 3, 3, 0, 0, 0, 0, 2)

relfreq <- freq/sum(freq)
bins <- 5 +(1:length(freq))
mids <- bins + 0.5

plot(mids, relfreq, type="h")

## ----- standard R function -----
edf <- density(mids, weights=relfreq, bw=2/3, from=0, to=40)
lines(edf)

## ----- same, but with different interface -----
edf2 <- abr_density(unbin(mids, freq), control=list(bw=2/3))
lines(edf2, col="red", lty="dotted")

## ----- boundary corrected density from package evmix -----
bcd <- abr_density(unbin(mids, freq), method="evmix")
lines(bcd, col="blue", lty="dotted")

## ===== maximum of density function (uses optimizer) =====
maxdens(unbin(mids, freq))

## ----- cumulative density -----
plot(abr_cumdens(unbin(mids, freq), method="fmm"), type="l")

## ----- quantiles -----

```

```

prob <- c(0.5, 0.95, 0.99)
abline(h=prob, lty="dotted")
abline(v=abr_cumdensquant(unbin(mids, freq),
                           p=prob, method="fmm"), lty="dotted")

par(opar)

```

abr\_quantile

*Normal Quantiles of Wild Type Population Identified with Ecoffinder***Description**

Applies [qnorm](#) to an object returned by `ecoffinder_nls`.

**Usage**

```
abr_quantile(obj, q, ...)
```

**Arguments**

|                  |                                                 |
|------------------|-------------------------------------------------|
| <code>obj</code> | object of class <a href="#">abr_ecoffinder</a>  |
| <code>q</code>   | vector of quantiles                             |
| <code>...</code> | other arguments passed to <a href="#">qnorm</a> |

**Details**

If `log2` was used for MIC data, the results are back-transformed by  $2^x$ .

**Value**

vector of quantiles

coerce

*Coercion Methods for 'mxObj' Univariate Mixture Objects***Description**

Conversion methods of univariate mixture distributions, composed of exponential, normal and gamma distributions.

**Usage**

```

as.mxObj(from, ...)

## S4 method for signature 'list'
as.mxObj(from, ...)

## S4 method for signature 'vector'
as.mxObj(from, ...)

## S4 method for signature 'mxObj'
as.list(x, ...)

## S4 method for signature 'mxObj'
as.vector(x)

```

**Arguments**

|       |                                                   |
|-------|---------------------------------------------------|
| from  | an object that is to be converted to another type |
| ...   | reserved for future extension                     |
| x     | an object that is to be converted to another type |
| mxObj | an object of class 'mxObj'                        |

---

diagnostics

*Diagnostics of Mixture Objects Fitted with 'mle2'*


---

**Description**

Return diagnostics after on a mixture distribution object fitted with 'mx\_metafit'

**Usage**

```

diagnostics(obj, ...)

## S4 method for signature 'mxMle'
diagnostics(obj, breaks = NULL, counts = NULL, ...)

```

**Arguments**

|        |                                |
|--------|--------------------------------|
| obj    | object of class mxMle          |
| ...    | reserved for future extensions |
| breaks | class boundaries of the data   |
| counts | frequency of observations      |

**Value**

list of diagnostic measures

**Examples**

```

breaks <- 0:28
counts <- c(36, 0, 2, 3, 4, 8, 9, 14, 10, 9, 3, 1, 1, 2,
           4, 8, 20, 45, 40, 54, 41, 22, 8, 3, 3, 0, 0,0)

observations <- unbin(breaks[-1], counts) # upper class boundaries

(comp <- mx_guess_components(observations, bw=2/3, mincut=0.9))
obj <- mxObj(comp, left="e")
ret <- mx_metafit(breaks, counts, obj)

summary(ret)          # general summary
cov2cor(vcov(ret))    # correlation matrix of parameters
diagnostics(ret)      # additional diagnostics

```

---

|         |                                                                           |
|---------|---------------------------------------------------------------------------|
| dunimix | <i>Univariate Mixtures of Exponential, Normal and Gamma Distributions</i> |
|---------|---------------------------------------------------------------------------|

---

**Description**

Density, distribution function, quantile function and random generation for univariate mixture distributions, composed of exponential, normal and gamma distributions.

**Usage**

```

dunimix(x, obj, full.out, ...)

punimix(q, obj, full.out, ...)

runimix(n, obj, ...)

## S4 method for signature 'numeric,list'
dunimix(x, obj, full.out, ...)

## S4 method for signature 'numeric,mxObj'
dunimix(x, obj, full.out = FALSE, ...)

## S4 method for signature 'numeric,list'
punimix(q, obj, full.out = FALSE, ...)

## S4 method for signature 'numeric,mxObj'
punimix(q, obj, full.out, ...)

## S4 method for signature 'numeric,list'
runimix(n, obj, ...)

```

```
## S4 method for signature 'numeric,mxObj'
runimix(n, obj, ...)
```

### Arguments

|          |                                                            |
|----------|------------------------------------------------------------|
| x        | vector of quantiles                                        |
| obj      | object of class mxObj with mixture distribution parameters |
| full.out | return total density only or a matrix with all components  |
| ...      | reserved for future extensions                             |
| q        | vector of quantiles                                        |
| n        | number of random values                                    |

### Value

dunimix gives the density, punimix gives the distribution function, qunimix gives the quantile function, and runimix generates random deviates. The length of the result is determined by the length of x. Argument full.out determines whether only a vector with the totals of the distribution or a data frame with all components is returned.

### Note

A quantile function qunimix is not yet implemented. Automatic coercion of arguments may still have bugs.

---

ECOFFinder

*Launch antibioticR App ECOFFinder*


---

### Description

Launch a shiny app to estimate MIC concentrations interactively.

### Usage

```
ECOFFinder(...)
```

### Arguments

|     |                                                             |
|-----|-------------------------------------------------------------|
| ... | optional parameters passed to the app (not yet implemented) |
|-----|-------------------------------------------------------------|

### Details

The app is still experimental.

### Value

This function has no return value.

**See Also**

[ecoffinder\\_nls](#) for a command line version of this function.

**Examples**

```
## Not run:
ECOFFinder()

## End(Not run)
```

---

ecoffinder\_nls

*Find Cutoff Value by Nonlinear Regression to Truncated Data*


---

**Description**

Identifies the wild-type sub-population by fitting a cumulative normal distribution to subsets of MIC or ID data.

**Usage**

```
ecoffinder_nls(
  conc,
  count,
  startpar = "mode",
  concentrations = NA,
  log2 = TRUE,
  plot = TRUE
)
```

**Arguments**

|                |                                                                                                  |
|----------------|--------------------------------------------------------------------------------------------------|
| conc           | concentration of the antibiotic                                                                  |
| count          | raw frequency                                                                                    |
| startpar       | start parameters for the nonlinear search, or "mode" resp. "mean" for an automatic determination |
| concentrations | which concentrations are tested                                                                  |
| log2           | logical determining if conc are log-transformed or not                                           |
| plot           | logical, switch visualization on or off                                                          |

**Details**

Start values for the nonlinear regression can be automatically determined with function [ecoffinder\\_startpar](#). The default search interval starts one concentration level above the mode resp. mean.

**Value**

an object of class `abr_ecoffinder-class` containing the fitted parameters and statistics of the final and intermediate fits.

**References**

Turnidge, J., Kahlmeter, G., Kronvall, G. (2006) Statistical characterization of bacterial wild-type MIC value distributions and the determination of epidemiological cut-off values. Clin Microbial Infect 12: 418-425 doi:10.1111/j.14690691.2006.01377.x

**See Also**

`ecoffinder_startpar` for heuristic methods to guess start parameters  
`ECOFFinder` for an interactive shiny app

**Examples**

```
## raw data contain NA values
data(micdata)
plot(freq ~ log2(conc), data=micdata, type="h")

## discard NA values
measured <- na.omit(micdata)

## cumulative plot
plot(cumsum(freq) ~ log2(conc), data=measured, type="l")

x <- log2(measured$conc)
y <- measured$freq

## heuristic start values
pstart <- ecoffinder_startpar(x, y)
pstart

## nonlinear regression
p <- ecoffinder_nls(x, y, pstart)
summary(p)
```

---

`ecoffinder_startpar`     *Start Values for ECOFFinder*

---

**Description**

A heuristic function function to estimate start values for the ECOFFinder.

**Usage**

```
ecoffinder_startpar(x, y, method = c("peak1", "mode", "mean"), bw = "nrd0")
```

## Arguments

|        |                                                                     |
|--------|---------------------------------------------------------------------|
| x      | x-values (log2 MIC or inhibition zone diameter)                     |
| y      | y-values (frequencies)                                              |
| method | select heuristic "peak1", "mode" or "mean"                          |
| bw     | bandwidth of kernel density smoother (see <a href="#">density</a> ) |

## Details

Start values for the nonlinear regression are estimated from the mode and median absolute deviation, from mean and standard deviation of the data or from a heuristic peak detection method, borrowed from package **carddates**.

## Value

vector with estimates for location, scale and total frequency

## References

Rolinski, S., Sachse, S. and Petzoldt, T. (2007) carddates: Identification of Cardinal Dates in Ecological Time Series. R package version 0.4.8. <https://cran.r-project.org/package=carddates>

Turnidge, J., Kahlmeter, G., Kronvall, G. (2006) Statistical characterization of bacterial wild-type MIC value distributions and the determination of epidemiological cut-off values. Clin Microbial Infect 12: 418-425 [doi:10.1111/j.14690691.2006.01377.x](https://doi.org/10.1111/j.14690691.2006.01377.x)

## See Also

[ecoffinder\\_nls](#)

## Examples

```
## raw data contain NA values
data(micdata)
plot(freq ~ log2(conc), data=micdata, type="h")

## discard NA values
measured <- na.omit(micdata)

## cumulative plot
plot(cumsum(freq) ~ log2(conc), data=measured, type="l")

x <- log2(measured$conc)
y <- measured$freq

## heuristic start values
(pstart <- ecoffinder_startpar(x, y, method = "mean"))
(pstart <- ecoffinder_startpar(x, y, method = "mode"))
(pstart <- ecoffinder_startpar(x, y, method = "peak1"))
```

fit

*Methods for 'mxObj' Univariate Mixture Objects***Description**

Constructor and accessor functions of univariate mixture distributions, composed of exponential, normal and gamma distributions.

The methods leverage methods from class 'mle2' of package 'bbmle'

**Usage**

```
fit(object, ...)

ecoff(object, ...)

rsquared(object, ...)

startpar(object, ...)

models(object, ...)

## S4 method for signature 'mxObj'
coef(object, i = NULL, ...)

pstart(obj, ...)

## S4 method for signature 'mxObj'
pstart(obj, ...)

## S4 method for signature 'mxMle'
summary(object, ...)

## S4 method for signature 'mxMle'
vcov(object, ...)

## S4 method for signature 'mxMle'
AIC(object, ..., k = 2)

## S4 method for signature 'mxMle'
logLik(object, ...)

get_data(object, ...)

## S4 method for signature 'mxMle'
get_data(object, ...)

mxObj(x, type = NA, left = c("e", "n"))
```

**Arguments**

|        |                                                                                                                                         |
|--------|-----------------------------------------------------------------------------------------------------------------------------------------|
| object | an object of class 'mxObj'                                                                                                              |
| ...    | optional arguments passed to the methods                                                                                                |
| i      | index of an individual distribution component                                                                                           |
| obj    | an object of class 'mxObj'                                                                                                              |
| k      | penalty parameter (nearly always left at its default value of 2)                                                                        |
| x      | object (vector, list, data.frame) containing mixture distributions in one of several known formats. (thpe-todo: document known formats) |
| type   | type of the mixture distribution, e.g. "n", "en", "nnn", etc.                                                                           |
| left   | whether leftmost distribution is exponential ("e") or normal ("n")                                                                      |

**See Also**

[bbmle::mle2-class]

---

fit\_unimix

*Fit of Normal and Exponential-Normal Mixtures to Binned Data*


---

**Description**

Fit mixture distributions to binned data with a maximum likelihood method, inspired by Venables and Ripley (2002)

**Usage**

```
fit_unimix(
  breaks,
  counts,
  parms,
  type = c("en", "enn", "ennn", "ennnn", "ennnnn", "n", "nn", "nnn", "nnnn", "nnnnn"),
  sd_min = 0,
  ...
)

fit_n(breaks, counts, parms, sd_min = 0, ...)

fit_en(breaks, counts, parms, sd_min = 0, ...)

fit_nn(breaks, counts, parms, sd_min = 0, ...)

fit_enn(breaks, counts, parms, sd_min = 0, ...)

fit_nnn(breaks, counts, parms, sd_min = 0, ...)
```

```
fit_nnnn(breaks, counts, parms, sd_min = 0, ...)
```

```
fit_ennn(breaks, counts, parms, sd_min = 0, ...)
```

```
fit_ennnn(breaks, counts, parms, sd_min = 0, ...)
```

```
fit_nnnnn(breaks, counts, parms, sd_min = 0, ...)
```

```
fit_ennnnn(breaks, counts, parms, sd_min = 0, ...)
```

### Arguments

|        |                                                                       |
|--------|-----------------------------------------------------------------------|
| breaks | upper class limits of the data                                        |
| counts | frequency of observations                                             |
| parms  | list of initial parameters for the mixture                            |
| type   | of the mixture distribution, e.g. 'enn' for exponential-normal-normal |
| sd_min | lower boundary value for standard deviation and rate parameter        |
| ...    | additional arguments passed to mle2                                   |

### References

Venables, W. N. and Ripley, B. D. (2002) Modern Applied Statistics with S. Fourth Edition. Springer, New York. ISBN 0-387-95457-0

Bolker, Ben and R Development Core Team (2017) bbmle: Tools for General Maximum Likelihood Estimation. R package version 1.0.20. <https://CRAN.R-project.org/package=bbmle>

---

micdata

*Minimum Inhibitory Concentration (MIC) Data*

---

### Description

Example data set of a MIC experiment, taken from Turnidge et al. (2006)

### Format

Data frame with two columns:

**conc** vector of antibiotic concentrations

**freq** frequency of detected MIC concentrations

### Source

<https://clsi.org/education/microbiology/ecoffinder/>

## References

Turnidge, J., Kahlmeter, G., Kronvall, G. (2006) Statistical characterization of bacterial wild-type MIC value distributions and the determination of epidemiological cut-off values. Clin Microbial Infect 12: 418-425 doi: 10.1111/j.1469-0691.2006.01377.x

## Examples

```
data(micdata)
plot(freq ~ log2(conc), data=micdata, type="h")

## replace NA values with zero
micdata$freq[is.na(micdata$freq)] <- 0
plot(cumsum(freq) ~ log2(conc), data=micdata, type="l")
```

---

|                    |                                                                        |
|--------------------|------------------------------------------------------------------------|
| mixParmDf2parmlist | <i>Convert Parameters in data frame format to a parameter list ...</i> |
|--------------------|------------------------------------------------------------------------|

---

## Description

Convert Parameters in data frame format to a parameter list ...

## Usage

```
mixParmDf2parmlist(df)
```

## Arguments

df                      data frame with parameters

## Details

deprecated !!!

---

|            |                                                    |
|------------|----------------------------------------------------|
| mx_compare | <i>Compare Fitted Mixture Distribution to Data</i> |
|------------|----------------------------------------------------|

---

## Description

Compare Fitted Mixture Distribution to Data

## Usage

```
mx_compare(parms, breaks, counts, plot = FALSE, ...)
```

**Arguments**

|        |                                                |
|--------|------------------------------------------------|
| parms  | parameter list of a ..... mixture distribution |
| breaks | upper class limits of the data                 |
| counts | frequency of observations                      |
| plot   | boolean, whether to show a diagnostic plot     |
| ...    | additional arguments passed to plot            |

---

|                     |                                                       |
|---------------------|-------------------------------------------------------|
| mx_guess_components | <i>Initial Guess of Univariate Mixture Components</i> |
|---------------------|-------------------------------------------------------|

---

**Description**

Heuristic method to guess number of components and their parameters (mean, standard deviation and weight) from observation data.

**Usage**

```
mx_guess_components(x, bw = 1, minpeak = 0, mincut = 0.9, ...)
```

**Arguments**

|         |                                                                                                                                                                                           |
|---------|-------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| x       | vector of (non-binned) observation data                                                                                                                                                   |
| bw      | bandwidth of kernel density, cf. <a href="#">density</a>                                                                                                                                  |
| minpeak | minimum value of the total maximum which is regarded as peak                                                                                                                              |
| mincut  | minimum relative height of a pit compared to the lower of the two neighbouring maxima at which these maxima are regarded as separate peaks (default value is derived from golden section) |
| ...     | further arguments passed to <a href="#">density</a>                                                                                                                                       |

**Details**

The function guesses approximate start values of parameters that then need subsequent identification, e.g. with a maximum likelihood or EM (expectation maximization) method.

**Value**

data frame with mean (mean), standard deviation sd and mixing proportions {L}

**See Also**

[peakwindow](#)

**Examples**

```
set.seed(123)
x <- c(rnorm(20, 5, 1), rnorm(30, 10, 1), rnorm(50, 20, 2))
hist(x, breaks=20)

mx_guess_components(x)
```

mx\_metafit

*Fit Mixture Distribution to Data and Evaluate Results***Description**

Fit Mixture Distribution to Data and Evaluate Results

**Usage**

```
mx_metafit(breaks, counts, parms, sd_min = 0, save.data = TRUE, ...)
```

**Arguments**

|           |                                                                                                          |
|-----------|----------------------------------------------------------------------------------------------------------|
| breaks    | class boundaries of the data                                                                             |
| counts    | frequency of observations                                                                                |
| parms     | list of initial parameters for the mixture, or a suitable <a href="#">mxObject</a> with start parameters |
| sd_min    | lower boundary value for standard deviation and rate parameter                                           |
| save.data | if data should be included in the returned object                                                        |
| ...       | additional arguments passed to <a href="#">mle2</a> and <a href="#">optim</a>                            |

**Examples**

```
## === preparation of data ===

zd <- 6:35
cutoff <- 6.0

counts <- c(155, 0, 8, 12, 17, 35, 37, 66, 42, 39, 13, 4, 4, 8, 19, 36,
  80, 205, 188, 219, 170, 104, 32, 13, 12, 0, 3, 0, 0, 0)

breaks <- c(zd, (max(zd+1))) - cutoff

observations <- unbin(zd, counts) - cutoff

## === quick example ===

(comp <- mx_guess_components(observations, bw=2/3, mincut=0.9))
obj <- mxObj(comp, left="e")
```

```

ret <- mx_metafit(breaks, counts, obj)
summary(ret)

## === details ===

comp <- mx_guess_components(observations, bw=2/3, mincut=0.9)
comp # parameters of components in form of a data frame

obj <- mxObj(comp, left="n") ## all components normal
obj <- mxObj(comp, left="e") ## left component exponential (= default)

## base function, needs parms in correct format
ret <- fit_unimix(breaks, counts, parms=pstart(obj), type="enn")

## for more difficult cases, slower
ret <- fit_unimix(breaks, counts, parms=pstart(obj), type="enn",
  method="BFGS",
  control=list(maxit=1000, ndeps=rep(1e-4, length(pstart(obj)))))

## higher level function, determines type from start parameters
ret <- mx_metafit(breaks, counts, obj)

## --- evaluation of results ---
coef(ret)          # top-level mxObj-object, contains all weights
coef(ret@fit)      # included mle2-object, last weight missing (sums up to 1.0)

summary(ret)       # from bbe::mle

AIC(ret)
logLik(ret)
vcov(ret)          # covariance matrix

cov2cor(vcov(ret)) # correlation matrix

```

---

mx\_parmlist\_sort

---

*Sort and Select Distribution Components*


---

## Description

Sort and Select Distribution Components

## Usage

```
mx_parmlist_sort(p)
```

```
mx_parmlist_select(p, N = 1, item = c("L", "mean", "sd"), ascending = TRUE)
```

```
mx_intermediate(p, as.vector = TRUE)
```

```

mx_wildtype(p, as.vector = TRUE)

mx_select(
  obj,
  type = c("resistant", "wildtype", "intermediate", "hypersensitive", "nonresistant")
)

mx_subset(
  p,
  type = c("resistant", "wildtype", "intermediate", "hypersensitive", "nonresistant"),
  as.vector = TRUE
)

```

### Arguments

|           |                                                                        |
|-----------|------------------------------------------------------------------------|
| p         | mixture distribution object or parameter list                          |
| N         | number of components                                                   |
| item      | parameter name used for sorting and selection (e.g. 'L', 'mean', 'sd') |
| ascending | ascending (TRUE) or descending (FALSE) order                           |
| as.vector | if TRUE, result is returned as a vector, otherwise as list             |
| obj       | mixture distribution object                                            |
| type      | kind of subset                                                         |

---

mx\_plot

---

*Plot Components of Mixture Distribution*


---

### Description

Plot Components of Mixture Distribution

### Usage

```

mx_plot(
  obj,
  breaks,
  counts,
  disc = 0,
  ecoff.prob = 0.01,
  main = "Subpopulations",
  length.out = 200,
  ...
)

```

**Arguments**

|            |                                                        |
|------------|--------------------------------------------------------|
| obj        | Object of class <code>mxObj</code> .                   |
| breaks     | class boundaries                                       |
| counts     | frequency of observations                              |
| disc       | disc diameter (defaults to zero)                       |
| ecoff.prob | probability threshold for the ecoff                    |
| main       | main title of the plot                                 |
| length.out | number of points used for plotting component densities |
| ...        | other arguments passed to <code>hist</code>            |

**Value**

ecoff value (normal quantile)

**Examples**

```
breaks <- 0:28
counts <- c(36, 0, 2, 3, 4, 8, 9, 14, 10, 9, 3, 1, 1, 2,
            4, 8, 20, 45, 40, 54, 41, 22, 8, 3, 3, 0, 0,0)

observations <- unbin(breaks[-1], counts) # upper class boundaries

(comp <- mx_guess_components(observations, bw=2/3, mincut=0.9))

obj <- mxObj(comp, left="e")

obj2 <- mx_metafit(breaks, counts, obj)

mx_plot(obj2, breaks, counts, disc=5.5)

## simplification for fitted objects with save.data = TRUE
mx_plot(obj2, disc=5.5)
```

---

mxMle-class

*Mixture Distribution Object Containing an 'mle2' fit as Subobject*


---

**Description**

The classes define ...

**Slots**

coef list with the coefficients of the mixture distribution  
fit complete results of the mle2-fit  
data optionally, data used for the fit

---

|             |                                                                               |
|-------------|-------------------------------------------------------------------------------|
| mxObj-class | <i>Class of Univariate Mixture Distribution Components and its Parameters</i> |
|-------------|-------------------------------------------------------------------------------|

---

**Description**

The classes define ...

**Slots**

coef coefficients of the mixture distribution, specified by a one-character ID of the distribution and the appropriate parameters

---

|           |                       |
|-----------|-----------------------|
| nls-class | <i>Helper Classes</i> |
|-----------|-----------------------|

---

**Description**

Classes that import base functionality from other packages.

**See Also**

[abr\\_baseclass](#), [abr\\_ecoffinder-class](#)

---

|     |                                                        |
|-----|--------------------------------------------------------|
| obs | <i>Accessor Methods of Package <b>antibioticR</b>.</i> |
|-----|--------------------------------------------------------|

---

**Description**

Functions to access data and results of fitted antibioticR objects: summary, obs coef, ...

**Usage**

```
obs(object, ...)

## S4 method for signature 'abr_baseclass'
obs(object, ...)

## S4 method for signature 'abr_baseclass'
coef(object, ...)

## S4 method for signature 'abr_baseclass'
fit(object, ...)
```

```
## S4 method for signature 'abr_baseclass'
deviance(object, ...)

## S4 method for signature 'abr_baseclass'
residuals(object, ...)

## S4 method for signature 'abr_baseclass'
df.residual(object, ...)

## S4 method for signature 'abr_ecofffinder'
startpar(object, ...)

## S4 method for signature 'abr_ecofffinder'
models(object, ...)

## S4 method for signature 'abr_ecofffinder'
ecoff(object, ...)

## S4 method for signature 'abr_ecofffinder'
rsquared(object, ...)
```

### Arguments

|        |                                                                     |
|--------|---------------------------------------------------------------------|
| object | name of a 'antibioticR' object. parameters is returned and printed. |
| ...    | other arguments passed to the methods.                              |

---

parmList

*Convert Parameter Vector to List*

---

### Description

deprecated! use as(vector, mxObj instead)

### Usage

```
parmList(p)
```

### Arguments

|   |                                               |
|---|-----------------------------------------------|
| p | parameter vector of the following form: ..... |
|---|-----------------------------------------------|

### Details

works for exponential, normal and gamma distributions

### Value

list structure of mixture distribution parameters

peakwindow

*Identify Peaks in Multimodal Density Curves and Time Series***Description**

This function identifies peaks in multimodal empirical density functions or in time series and helps to identify the time window of the first maximum according to a given heuristics.

**Usage**

```
peakwindow(
  x,
  y = NULL,
  xstart = 0,
  xend = max(x),
  minpeak = 0.1,
  mincut = 0.382
)
```

**Arguments**

|         |                                                                                                                                                                                           |
|---------|-------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| x       | x coordinate of a set of points.                                                                                                                                                          |
| y       | y coordinate of a set of points. Alternatively, a single argument x can be provided.                                                                                                      |
| xstart  | start before the maximum value of the searched peak (this is a “weak” limit)                                                                                                              |
| xend    | maximum of the end of the searched peak (this is a “hard” maximum)                                                                                                                        |
| minpeak | minimum value of the total maximum which is regarded as peak                                                                                                                              |
| mincut  | minimum relative height of a pit compared to the lower of the two neighbouring maxima at which these maxima are regarded as separate peaks (default value is derived from golden section) |

**Value**

A list with the following elements:

|                 |                                                                                                |
|-----------------|------------------------------------------------------------------------------------------------|
| peaks           | a data frame with the characteristics (index, xleft, x, xright and y) of all identified peaks, |
| first.max.index | index of the maximum value of the “specified” peak,                                            |
| first.max.x     | x-value of the maximum of the “specified” peak,                                                |
| first.indices   | indices (data window) of all data belonging to the “specified” peak,                           |
| first.x         | x-values (time window) of all data belonging to the “specified” peak,                          |
| first.y         | corresponding y-values of all data belonging to the “specified” peak,                          |
| peakid          | vector with peak-id-numbers for all data.                                                      |

**Examples**

```
## generate test data with 3 peaks
set.seed(123)
x <- seq(0, 360, length = 20)
y <- abs(rnorm(20, mean = 1, sd = 0.1))
y[5:10] <- c(2, 4, 7, 3, 4, 2)
y <- c(y, 0.8 * y, 1.2 * y)
x <- seq(0, 360, along = y)
y[6] <- y[7] # test case with 2 neighbouring equal points
## plot the test data
plot(x, y, type="b")
## identify the first peak
peaks <- peakwindow(x, y)
ind <- peaks$smd.indices
lines(x[ind], y[ind], col="red", lwd=2)

## some more options ...
peaks <- peakwindow(x, y, xstart=150, mincut = 0.455)
ind <- peaks$first.indices
lines(x[ind], y[ind], col = "blue")
points(x, y, col = peaks$peakid + 1, pch = 16) # all peaks

## work with indices only
peaks <- peakwindow(y)

## test case with disturbed sinus
x<- 1:100
y <- sin(x/5) + 1.5 + rnorm(x, sd = 0.2)
peaks <- peakwindow(x, y)
plot(x, y, type = "l", ylim = c(0, 3))
points(x, y, col = peaks$peakid + 2, pch = 16)

## test case: only one peak
yy <- c(1:10, 11:1)
peakwindow(yy)

## error handling test case: no turnpoints
# yy <- rep(1, length(x))
# peakwindow(x, yy)
```

---

plot,abr\_ecofffinder,ANY-method

*Plot Fitted abrECOFFinder Object*


---

**Description**

Plot Fitted abrECOFFinder Object

**Usage**

```
## S4 method for signature 'abr_ecoffinder,ANY'
plot(x, cumulative = TRUE, fits = c("all", "best"), ...)
```

**Arguments**

|            |                                                 |
|------------|-------------------------------------------------|
| x          | an object fitted by <code>ecoffinder_nls</code> |
| cumulative | plot cumulative or density function             |
| fits       | whether to plot all fits or only the best       |
| ...        | further parameters passed to plot               |

**Details**

: to do: documentation and examples

---

|                 |                                                                |
|-----------------|----------------------------------------------------------------|
| reformat_eucast | <i>Reformat EUCAST crosstabe format into data base formats</i> |
|-----------------|----------------------------------------------------------------|

---

**Description**

Reformat EUCAST crosstabe format into data base formats

**Usage**

```
reformat_eucast(data, type = c("ZD", "MIC"), melt = TRUE, unbin = FALSE)
```

**Arguments**

|       |                                                                                                 |
|-------|-------------------------------------------------------------------------------------------------|
| data  | data frame representing an EUCAST cross table or a result from this function with "unbin=FALSE" |
| type  | either "ZD" or "MIC", the latter is not implemented yet                                         |
| melt  | if TRUE, convert EUCAST crosstable to data base format                                          |
| unbin | expand binned data to single observations                                                       |

**Details**

Reformatting of from crosstable to data base format is done with function `melt` from package **reshape2**. In contrast, `unbin` is a quite memory consuming workaround for R functions that do not support a `weight` argument. Please consider to use `weights` and avoid `unbin` whenever possible.

**Value**

reformatted data frame

**Note**

This function is under development, name and functionality may change.

---

 results

*Quantiles and Results of Mixture Objects Fitted with 'mle2'*


---

## Description

Quantiles and Results of Mixture Objects Fitted with 'mle2'

## Usage

```
results(obj, ...)

## S4 method for signature 'mxMle'
results(obj, breaks = NULL, counts = NULL, ...)
```

## Arguments

|        |                                |
|--------|--------------------------------|
| obj    | object of class mxMle          |
| ...    | reserved for future extensions |
| breaks | class boundaries of the data   |
| counts | frequency of observations      |

## Value

list of results

## Examples

```
breaks <- 0:28
counts <- c(36, 0, 2, 3, 4, 8, 9, 14, 10, 9, 3, 1, 1, 2,
           4, 8, 20, 45, 40, 54, 41, 22, 8, 3, 3, 0, 0,0)

observations <- unbin(breaks[-1], counts) # upper class boundaries

(comp <- mx_guess_components(observations, bw=2/3, mincut=0.9))
obj <- mxObj(comp, left="e")
ret <- mx_metafit(breaks, counts, obj)

summary(ret)          # general summary
cov2cor(vcov(ret))    # correlation matrix of parameters
results(ret)          # additional diagnostics
```

---

summary,abr\_baseclass-method

*Summary of an 'abr\_ecoffinder' Object*


---

## Description

Returns parameters and quantiles of the normal distribution fitted to the wild type subset of the data.

## Usage

```
## S4 method for signature 'abr_baseclass'
summary(object, ...)

## S4 method for signature 'abr_ecoffinder'
summary(object, ...)
```

## Arguments

|        |                                                      |
|--------|------------------------------------------------------|
| object | abr_ecoffinder object                                |
| ...    | additional arguments affecting the summary produced. |

---

|            |                                                                                                                                                              |
|------------|--------------------------------------------------------------------------------------------------------------------------------------------------------------|
| turnpoints | <i>Find turning points (peaks or pits) Determine the number and the position of extrema (turning points, either peaks or pits) in a regular time series.</i> |
|------------|--------------------------------------------------------------------------------------------------------------------------------------------------------------|

---

## Description

Find turning points (peaks or pits) Determine the number and the position of extrema (turning points, either peaks or pits) in a regular time series.

## Usage

```
turnpoints(x)
```

## Arguments

|   |                         |
|---|-------------------------|
| x | vector or a time series |
|---|-------------------------|

**Value**

An object of type 'abrTurnpoints' is returned, with:

|             |                                                                                  |
|-------------|----------------------------------------------------------------------------------|
| data        | The dataset to which the calculation is done                                     |
| n           | The number of observations                                                       |
| points      | The value of the points in the series, after elimination of ex-aequos            |
| pos         | The position of the points on the time scale in the series (including ex-aequos) |
| exaequos    | Location of exaequos (1), or not (0)                                             |
| nturns      | Total number of turning points in the whole time series                          |
| firstispeak | Is the first turning point a peak (TRUE), or not (FALSE)                         |
| peaks       | Logical vector. Location of the peaks in the time series without ex-aequos       |
| pits        | Logical vector. Location of the pits in the time series without ex-aequos        |
| tppos       | Position of the turning points in the initial series (with ex-aequos)            |

**Author(s)**

This is a stripped-down version taken from package **pastecs**, original authors are Frédéric Ibanez, Philippe Grosjean. The original code and the derived version are licensed under the GPL (>=2).

**References**

Ibanez, F. (1982) Sur une nouvelle application de la theorie de l'information a la description des series chronologiques planctoniques. J. Exp. Mar. Biol. Ecol., 4:619-632

Grosjean, P. and Ibanez, F. (2014) pastecs: Package for Analysis of Space-Time Ecological Series. R package version 1.3-18. <https://CRAN.R-project.org/package=pastecs>

---

unbin

---

*Unbin Data Frame*


---

**Description**

Unbin a data frame, i.e. replicate rows according to frequencies given in a vector. The function is intended to reconstruct raw data from data binned to classes.

**Usage**

```
unbin(x, freq)
```

**Arguments**

|      |                                         |
|------|-----------------------------------------|
| x    | a vector or data frame with binned data |
| freq | vector of frequencies                   |

**Value**

Vector or data frame with the same structure as `x` with replicated elements or rows.

**Examples**

```
x <- data.frame(a=1:3, b=c("a", "b", "c"), freq=c(2, 5, 3))
unbin(x, x$freq)

zd <- 6:35

counts <- c(155, 0, 8, 12, 17, 35, 37, 66, 42, 39, 13, 4, 4, 8, 19, 36,
            80, 205, 188, 219, 170, 104, 32, 13, 12, 0, 3, 0, 0, 0)

observations <- unbin(zd, counts)
```

---

 weighted.sd

*Weighted Standard Deviation*


---

**Description**

Computes the standard deviation of values in `x` using weights.

**Usage**

```
weighted.sd(x, w = rep(1, length(x)), df = sum(w) - 1)
```

**Arguments**

|                 |                           |
|-----------------|---------------------------|
| <code>x</code>  | numeric vector of data    |
| <code>w</code>  | numeric vector of weights |
| <code>df</code> | degrees of freedom        |

**Value**

scalar value

**Examples**

```
x <- c(1.5, 4.1, 3.7, 8.2)
w <- c(4, 5, 4, 1)
weighted.sd(x, w)
```

---

|                    |                                                                             |
|--------------------|-----------------------------------------------------------------------------|
| zd_main_components | <i>Extract Main Components from Zone Diameter (ZD) Mixture Distribution</i> |
|--------------------|-----------------------------------------------------------------------------|

---

### Description

Extract np-1 main components and an additional first potentially exponentially distributed component from a data frame of distribution parameters with columns mean, sd, L.

### Usage

```
zd_main_components(p, np = 3, near_zero = 3)
```

### Arguments

|           |                                                                                                                                                                             |
|-----------|-----------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| p         | data frame of distribution parameters (mean, sd, L) returned by <a href="#">mx_guess_components</a>                                                                         |
| np        | number of desired components                                                                                                                                                |
| near_zero | heuristic upper boundary of the zone diameter interval that is considered as close to zero. Components with mean value below will be considered as resistant subpopulation. |

### Details

The algorithm is specifically intended to zone diameter (ZD) distributions where the leftmost (resistant) subpopulation is considered exponentially distributed, while the others (intermediate, wild-type) are considered approximately normal.

### Examples

```
set.seed(123)
x <- c(rexp(20, rate=1), rnorm(30, 10, 1), rnorm(10,15,.2), rnorm(50, 20, 2))
hist(x, breaks=20)
p <- mx_guess_components(x)

(ret <- zd_main_components(p, np=3))
```

# Index

- \* **data**
  - micdata, [16](#)
- \* **nls**
  - nls-class, [23](#)
- \* **package**
  - antibioticR-package, [3](#)
- abr\_baseclass, [4](#), [23](#)
- abr\_cumdens (abr\_density), [4](#)
- abr\_cumdensquant (abr\_density), [4](#)
- abr\_density, [4](#)
- abr\_density.control, [5](#)
- abr\_ecofffinder, [7](#)
- abr\_ecofffinder-class (abr\_baseclass), [4](#)
- abr\_quantile, [7](#)
- AIC, mxMle-method (fit), [14](#)
- antibioticR (antibioticR-package), [3](#)
- antibioticR-package, [3](#)
- as.list, mxObj-method (coerce), [7](#)
- as.mxObj (coerce), [7](#)
- as.mxObj, list-method (coerce), [7](#)
- as.mxObj, vector-method (coerce), [7](#)
- as.vector, mxObj-method (coerce), [7](#)
- coef, abr\_baseclass-method (obs), [23](#)
- coef, mxObj-method (fit), [14](#)
- coerce, [7](#)
- density, [5](#), [13](#), [18](#)
- deviance, abr\_baseclass-method (obs), [23](#)
- df.residual, abr\_baseclass-method (obs), [23](#)
- diagnostics, [8](#)
- diagnostics, mxMle-method (diagnostics), [8](#)
- dunimix, [9](#)
- dunimix, numeric, list-method (dunimix), [9](#)
- dunimix, numeric, mxObj-method (dunimix), [9](#)
- ecoff (fit), [14](#)
- ecoff, abr\_ecofffinder-method (obs), [23](#)
- ECOFFfinder, [10](#), [12](#)
- ecofffinder\_nls, [11](#), [11](#), [13](#), [27](#)
- ecofffinder\_startpar, [11](#), [12](#), [12](#)
- fit, [14](#)
- fit, abr\_baseclass-method (obs), [23](#)
- fit\_en (fit\_unimix), [15](#)
- fit\_enn (fit\_unimix), [15](#)
- fit\_ennn (fit\_unimix), [15](#)
- fit\_ennnn (fit\_unimix), [15](#)
- fit\_ennnnn (fit\_unimix), [15](#)
- fit\_n (fit\_unimix), [15](#)
- fit\_nn (fit\_unimix), [15](#)
- fit\_nnn (fit\_unimix), [15](#)
- fit\_nnnn (fit\_unimix), [15](#)
- fit\_nnnnn (fit\_unimix), [15](#)
- fit\_unimix, [15](#)
- get\_data (fit), [14](#)
- get\_data, mxMle-method (fit), [14](#)
- hist, [22](#)
- logLik, mxMle-method (fit), [14](#)
- maxdens (abr\_density), [4](#)
- micdata, [16](#)
- mixParmDf2parmlist, [17](#)
- mle2, [19](#)
- models (fit), [14](#)
- models, abr\_ecofffinder-method (obs), [23](#)
- mx\_compare, [17](#)
- mx\_guess\_components, [18](#), [32](#)
- mx\_intermediate (mx\_parmlist\_sort), [20](#)
- mx\_metafit, [19](#)
- mx\_parmlist\_select (mx\_parmlist\_sort), [20](#)
- mx\_parmlist\_sort, [20](#)
- mx\_plot, [21](#)
- mx\_select (mx\_parmlist\_sort), [20](#)

`mx_subset` (`mx_parmlist_sort`), 20  
`mx_wildtype` (`mx_parmlist_sort`), 20  
`mxMle-class`, 22  
`mxObj`, 19, 22  
`mxObj` (`fit`), 14  
`mxObj-class`, 23  
  
`nls` (`nls-class`), 23  
`nls-class`, 23  
  
`obs`, 23  
`obs`, `abr_baseclass-method` (`obs`), 23  
`optim`, 19  
  
`parmList`, 24  
`peakwindow`, 18, 25  
`plot`, `abr_ecoffinder`, `ANY-method`, 26  
`pstart` (`fit`), 14  
`pstart`, `mxObj-method` (`fit`), 14  
`punimix` (`dunimix`), 9  
`punimix`, `numeric`, `list-method` (`dunimix`), 9  
`punimix`, `numeric`, `mxObj-method` (`dunimix`), 9  
  
`qnorm`, 7  
  
`reformat_eucast`, 27  
`residuals`, `abr_baseclass-method` (`obs`), 23  
`results`, 28  
`results`, `mxMle-method` (`results`), 28  
`rsquared` (`fit`), 14  
`rsquared`, `abr_ecoffinder-method` (`obs`), 23  
`runimix` (`dunimix`), 9  
`runimix`, `numeric`, `list-method` (`dunimix`), 9  
`runimix`, `numeric`, `mxObj-method` (`dunimix`), 9  
  
`startpar` (`fit`), 14  
`startpar`, `abr_ecoffinder-method` (`obs`), 23  
`summary`, `abr_baseclass-method`, 29  
`summary`, `abr_ecoffinder-method`  
    (`summary`, `abr_baseclass-method`), 29  
`summary`, `mxMle-method` (`fit`), 14  
  
`turnpoints`, 29  
  
`unbin`, 5, 30  
  
`vcov`, `mxMle-method` (`fit`), 14  
  
`weighted.sd`, 31  
  
`zd_main_components`, 32