

# Package ‘Xeva’

April 1, 2025

**Type** Package

**Title** Analysis of patient-derived xenograft (PDX) data

**Version** 1.23.2

**Date** 2025-03-20

**Maintainer** Benjamin Haibe-Kains <benjamin.haibe.kains@utoronto.ca>

**Description** The Xeva package provides efficient and powerful functions for patient-derived xenograft (PDX) based pharmacogenomic data analysis.

This package contains a set of functions to perform analysis of patient-derived xenograft data. This package was developed by the BHKLab, for further information please see our documentation.

**License** GPL-3

**Encoding** UTF-8

**RoxygenNote** 7.3.2

**VignetteBuilder** knitr

**Suggests** BiocStyle, knitr, rmarkdown

**Imports** methods, stats, utils, BBmisc, Biobase, grDevices, ggplot2, scales, ComplexHeatmap, parallel, doParallel, Rmisc, grid, nlme, PharmacoGx, downloader

**Depends** R (>= 3.6)

**biocViews** GeneExpression, Pharmacogenetics, Pharmacogenomics, Software, Classification

**BugReports** <https://github.com/bhklab/Xeva/issues>

**git\_url** <https://git.bioconductor.org/packages/Xeva>

**git\_branch** devel

**git\_last\_commit** 7418d16

**git\_last\_commit\_date** 2025-03-20

**Repository** Bioconductor 3.21

**Date/Publication** 2025-03-31

**Author** Arvind Mer [aut],  
Benjamin Haibe-Kains [aut, cre]

## Contents

ABC	2
addExperimentalDesign	3
angle	4
AUC	5
batchInfo	6
brca	7
createXevaSet	8
dosePlot	9
downloadXevaSet	10
drugInform	11
drugSensitivitySig	12
getExperiment	13
getMolecularProfiles	15
lmm	16
modelInfo	16
mRECIST	17
PDXMI	18
plotmRECIST	18
plotPDX	19
print.batchResponse	21
print.modelResponse	22
print.pdxBatch	22
repx	23
response	23
selectModelIds	25
sensitivity	25
setResponse	26
slope	27
subsetXeva	28
summarizeMolecularProfiles	29
summarizeResponse	30
TGI	31
waterfall	32
<b>Index</b>	<b>34</b>

---

ABC	<i>area between curves Computes the area between two time-volume curves.</i>
-----	--

---

### Description

area between curves Computes the area between two time-volume curves.

**Usage**

```
ABC(  
  contr.time = NULL,  
  contr.volume = NULL,  
  treat.time = NULL,  
  treat.volume = NULL  
)
```

**Arguments**

contr.time	Time vector for control.
contr.volume	Volume vector for control.
treat.time	Time vector for treatment.
treat.volume	Volume vector for treatment.

**Value**

Returns batch response object.

**Examples**

```
contr.time <- treat.time <- c(0, 3, 7, 11, 18, 22, 26, 30, 32, 35)  
contr.volume<- contr.time * tan(60*pi/180)  
treat.volume<- treat.time * tan(15*pi/180)  
abc <- ABC(contr.time, contr.volume, treat.time, treat.volume)  
par(pty="s")  
xylimit <- range(c(contr.time, contr.volume, treat.time, treat.volume))  
plot(contr.time, contr.volume, type = "b", xlim = xylimit, ylim = xylimit)  
lines(treat.time, treat.volume, type = "b")  
polygon(c(treat.time, rev(treat.time)), c(contr.volume, rev(treat.volume)),  
  col = "#fa9fb5", border = NA)
```

---

addExperimentalDesign *Add a new experimental design*

---

**Description**

Add a new experimental design in the expDesign slot.

**Usage**

```
addExperimentalDesign(  
  object,  
  treatment = NULL,  
  control = NULL,  
  batch.id = NULL,
```

```

    replace = FALSE
  )

  ## S4 method for signature 'XevaSet'
  addExperimentalDesign(
    object,
    treatment = NULL,
    control = NULL,
    batch.id = NULL,
    replace = FALSE
  )

```

### Arguments

object	The Xeva dataset.
treatment	The model . id of treatment.
control	The model . id of control.
batch.id	The batch . id for a new batch.
replace	If TRUE, replace an old batch with new values.

### Value

Returns Xeva dataset with new experimental design added.

### Examples

```

data(brca)
brca <- addExperimentalDesign(object=brca, treatment=c("X.6047.LL71"),
                             control=c("X.6047.uned"), batch.id="new.batch", replace=FALSE)

```

---

angle	<i>compute angle Computes the angle between two time-volume curves.</i>
-------	---

---

### Description

compute angle Computes the angle between two time-volume curves.

### Usage

```

angle(
  contr.time = NULL,
  contr.volume = NULL,
  treat.time = NULL,
  treat.volume = NULL,
  degree = TRUE
)

```

**Arguments**

<code>contr.time</code>	Time vector for control.
<code>contr.volume</code>	Volume vector for control.
<code>treat.time</code>	Time vector for treatment.
<code>treat.volume</code>	Volume vector for treatment.
<code>degree</code>	Default TRUE will give angle in degrees and FALSE will return in radians.

**Value**

Returns batch response object.

**Examples**

```

contr.time <- treat.time <- c(0, 3, 7, 11, 18, 22, 26, 30, 32, 35)
contr.volume <- contr.time * tan(60*pi/180)
treat.volume <- treat.time * tan(15*pi/180)
ang <- angle(contr.time, contr.volume, treat.time, treat.volume)
print(ang)
par(pty="s")
xylim = range(c(contr.time, contr.volume, treat.time, treat.volume))
plot(contr.time, contr.volume, type = "b", xlim = xylim, ylim = xylim)
lines(treat.time, treat.volume, type = "b")
abline(lm(contr.volume~contr.time))
abline(lm(treat.volume~treat.time))

```

---

AUC

*area under the curve AUC Returns area under the curve*


---

**Description**

area under the curve AUC Returns area under the curve

**Usage**

```
AUC(time, volume)
```

**Arguments**

<code>time</code>	A vector of time points recorded for the experiment.
<code>volume</code>	First vector of volume.

**Value**

Returns angle and slope object.

**Examples**

```

time <- c(0, 3, 7, 11, 18, 22, 26, 30, 32, 35)
volume1<- time * tan(30*pi/180)
volume2<- time * tan(45*pi/180)
auc1 <- AUC(time, volume1)
auc2 <- AUC(time, volume2)
par(pty="s")
xylim = range(c(time, volume1, volume2))
plot(time, volume1, type = "b", xlim = xylim, ylim = xylim)
lines(time, volume2, type = "b")
abline(lm(volume1~time))
abline(lm(volume2~time))

```

---

batchInfo

*Get batch information*


---

**Description**

Get batch information from a Xeva dataset.

**Usage**

```

batchInfo(
  object,
  batch = NULL,
  model.id = NULL,
  model.id.type = c("any", "control", "treatment")
)

```

```
## S4 method for signature 'XevaSet'
```

```

batchInfo(
  object,
  batch = NULL,
  model.id = NULL,
  model.id.type = c("any", "control", "treatment")
)

```

**Arguments**

object	The Xeva object from which batch information is obtained.
batch	Name of the batch. Default NULL.
model.id	Model ID for which need to be searched in the batches. Default NULL.
model.id.type	Type of the model ID in a batch. See the Details section below.

## Details

By default this function will return the names of all the batches present in the dataset. If a batch specified, it will return the experiment design (control and treatment model IDs) of that particular batch. If `model.id` is specified, it will return the names of all the batches where this particular `model.id` is present. If both `batch` and `model.id` are specified, `batch` will take precedent.

For `model.id.type`, the default value 'any' will return all the batch IDs where the given model ID is present in any arm (ie. control or treatment) of the batch. It can also be restricted to look only for treatment (or control) arm by specifying the type.

## Value

A Vector with batch names.

## Examples

```
data(brca)
##to get all the batch names
batch.name <- batchInfo(brca)

##to get a specific batch
batch.design <- batchInfo(brca, batch=batch.name[1])

##to get all the batches where a model.id is present
batchInfo(brca, model.id="X.6047.uned")
```

---

brca

*PDXE breast cancer dataset*

---

## Description

A Xeva object containing only breast cancer PDXs from the PDXE dataset For details about PDX-MI, see: Gao et al. High-throughput screening using patient-derived tumor xenografts to predict clinical trial drug response. Nature medicine, 21(11):1318, 2015.

## Usage

```
data(brca)
```

## Format

An object of class XevaSet of length 1.

## Source

<https://www.nature.com/articles/nm.3954?draft=journal>

---

createXevaSet	<i>XevaSet constructor</i>
---------------	----------------------------

---

### Description

A constructor to create XevaSet. Only objects returned by this constructor are expected to work with the XevaSet methods.

### Usage

```
createXevaSet(
  name,
  model = data.frame(),
  drug = data.frame(),
  experiment = data.frame(),
  expDesign = list(),
  modelSensitivity = data.frame(),
  batchSensitivity = data.frame(),
  molecularProfiles = list(),
  modToBiobaseMap = data.frame()
)
```

### Arguments

name	A character string detailing the name of the dataset.
model	A data.frame containing the annotations for all the models used in the experiment.
drug	A data.frame containing the annotations for all the drugs profiled in the dataset, across all data types.
experiment	A data.frame containing all experiment information.
expDesign	A list containing name of the batch, control and treatment model.id
modelSensitivity	A data.frame containing sensitivity for each model
batchSensitivity	A data.frame containing sensitivity for each batch
molecularProfiles	A list of ExpressionSet objects containing different molecular profiles.
modToBiobaseMap	A data.frame containing model.id corresponding Biobase object id and name of the molecularProfiles

### Details

This function creates a XevaSet object. It takes different model information and genomic data as input. For detailed discription of all varaibles please see Xeva vignette section "**Creating new Xeva object**"



**Value**

Returns Xeva object

**Examples**

```
## read raw data files containg PDX experiment information and genomic data
model = read.csv(system.file("extdata", "model.csv", package = "Xeva"))
drug = read.csv(system.file("extdata", "drug.csv", package = "Xeva"))
experiment= read.csv(system.file("extdata", "experiments.csv", package = "Xeva"))
expDesign=readRDS(system.file("extdata", "batch_list.rds", package = "Xeva"))
RNASeq=readRDS(system.file("extdata", "rnaseq.rds", package = "Xeva"))
modToBiobaseMap=read.csv(system.file("extdata", "modelToExpressionMap.csv", package = "Xeva"))

## create Xeva object
xeva.set = createXevaSet(name="example xevaSet", model=model, drug=drug,
                        experiment=experiment, expDesign=expDesign,
                        molecularProfiles=list(RNASeq = RNASeq),
                        modToBiobaseMap = modToBiobaseMap)

print(xeva.set)
```

---

dosePlot

*plot dose data*

---

**Description**

plot data for dose in model.id

**Usage**

```
dosePlot(
  object,
  model.id,
  max.time = NULL,
  treatment.only = FALSE,
  vol.normal = FALSE,
  concurrent.time = FALSE,
  point.shape = 21,
  point.size = 3,
  line.size = 4,
  point.color = "#878787",
  line.color = "#bababa",
  fill.col = c("#f5f5f5", "#E55100"),
  modify.x.axis = FALSE
)
```

**Arguments**

<code>object</code>	Xeva object.
<code>model.id</code>	one or multiple model.id
<code>max.time</code>	Maximum time point of the plot. Default NULL will plot complete data
<code>treatment.only</code>	Default FALSE. Given full data <code>treatment.only=TRUE</code> will plot data only during treatment
<code>vol.normal</code>	Default FALSE. If TRUE, volume will be normalized
<code>concurrent.time</code>	Default FALSE. If TRUE, cut the batch data such that control and treatment will end at the same time point
<code>point.shape</code>	shape of the point
<code>point.size</code>	size of the point
<code>line.size</code>	size of the line
<code>point.color</code>	color for point
<code>line.color</code>	color for line
<code>fill.col</code>	a vector with color to fill
<code>modify.x.axis</code>	Default FALSE

**Value**

A ggplot2 plot

**Examples**

```
data(brca)
dosePlot(brca, model.id=c("X.6047.LJ16", "X.6047.LJ16.trab"), fill.col=c("#f5f5f5", "#993404"))
```

---

downloadXevaSet

*Download a XevaSet object or table of available XevaSet objects*

---

**Description**

This function allows you to see the available XevaSet object and download them for use with this package. The XevaSet have been extensively curated and organised within a XevaSet class, enabling use with all the analysis tools provided in Xeva.

**Usage**

```
downloadXevaSet(
  name = NULL,
  saveDir = file.path(".", "XevaSet"),
  XevaSetFileName = NULL,
  verbose = TRUE
)
```

**Arguments**

name	Character string, the name of the XevaSet to download.
saveDir	Character string with the folder path where the XevaSet should be saved. Defaults to './XevaSet/'. Will create directory if it does not exist.
XevaSetFileName	character string, the file name to save the dataset under
verbose	bool Should status messages be printed during download. Defaults to TRUE.

**Value**

A data.frame if name is NULL, showing all the available XevaSet objects. If name is specified, it will download the dataset from our server

---

drugInform	<i>Get drug information Get the drug information slot from a XevaSet object.</i>
------------	--

---

**Description**

Get drug information Get the drug information slot from a XevaSet object.

**Usage**

```
drugInform(object)

## S4 method for signature 'XevaSet'
drugInform(object)
```

**Arguments**

object            The XevaSet to retrieve drug information from.

**Value**

A data.frame with the drug annotations.

**Examples**

```
data(brca)
head(drugInform(brca))
```

---

drugSensitivitySig     *get drug sensitivity values*

---

### Description

Given a Xeva object and drug name, this function will return sensitivity values for all the genes/features.

### Usage

```
drugSensitivitySig(
  object,
  drug,
  mDataType = NULL,
  molData = NULL,
  features = NULL,
  model.ids = NULL,
  model2bidMap = NULL,
  sensitivity.measure = "slope",
  fit = c("lm", "CI", "pearson", "spearman", NA),
  standardize = c("SD", "rescale", "none"),
  nthread = 1,
  tissue = NULL,
  verbose = TRUE
)
```

### Arguments

object	The Xeva dataset.
drug	Name of the drug.
mDataType	Molecular data type.
molData	External data matrix. Rows as features and columns as samples.
features	Set which molecular data features to use. Default NULL will use all features.
model.ids	Set which model.id to use from the dataset. Default NULL will use all model.ids.
model2bidMap	A data.frame with model.id and biobase.id. Default NULL will use internal mapping.
sensitivity.measure	Name of the sensitivity measure.
fit	Association method to use, can be 'lm', 'CI', 'pearson' or 'spearman'. If 'NA' only the data will be return. Default lm.
standardize	Default SD. Name of the method to use for data standardization before fitting.
nthread	number of threads
tissue	tissue type. Default NULL uses 'tissue' from object.
verbose	Default TRUE will show information

## Details

Method to compute association can be specified by `fit`. It can be one of the:

- "lm" for linear models
- "CI" for concordance index
- "pearson" for Pearson correlation
- "spearman" for Spearman correlation

If `fit` is set to NA, processed data (an ExpressionSet) will be returned.

A matrix of values can be directly passed to `molData`. In case where a `model.id` maps to multiple `biobase.ids`, the first `biobase.id` in the `data.frame` will be used.

## Value

A `data.frame` with features and values.

## Examples

```
data(brca)
senSig <- drugSensitivitySig(object=brca, drug="tamoxifen",
                           mDataType="RNASeq", features=c(1,2,3,4,5),
                           sensitivity.measure="slope", fit = "lm")

## example to compute the Pearson correlation between gene expression and PDX response
senSig <- drugSensitivitySig(object=brca, drug="tamoxifen",
                           mDataType="RNASeq", features=c(1,2,3,4,5),
                           sensitivity.measure="slope", fit = "pearson")
```

---

getExperiment

*Get PDX experiment data*

---

## Description

For a given `model.id`, `getExperiment` will

## Usage

```
getExperiment(
  object,
  model.id = NULL,
  batch = NULL,
  patient.id = NULL,
  drug = NULL,
  control.name = NULL,
  treatment.only = FALSE,
  max.time = NULL,
```

```

    vol.normal = FALSE,
    log.volume = FALSE,
    return.list = FALSE,
    impute.value = FALSE,
    concurrent.time = FALSE
  )

## S4 method for signature 'XevaSet'
getExperiment(
  object,
  model.id = NULL,
  batch = NULL,
  patient.id = NULL,
  drug = NULL,
  control.name = NULL,
  treatment.only = FALSE,
  max.time = NULL,
  vol.normal = FALSE,
  log.volume = FALSE,
  return.list = FALSE,
  impute.value = FALSE,
  concurrent.time = FALSE
)

```

### Arguments

<code>object</code>	The XevaSet object.
<code>model.id</code>	The model . id for which data is required, multiple IDs are allowed.
<code>batch</code>	Batch name from the XevaSet or experiment design.
<code>patient.id</code>	Patient id from the XevaSet. Default NULL.
<code>drug</code>	Name of the drug.
<code>control.name</code>	Name of drug used as control. Default NULL.
<code>treatment.only</code>	Default FALSE. If TRUE, give data for non-zero dose periods only (if dose data are available).
<code>max.time</code>	Maximum time for data.
<code>vol.normal</code>	If TRUE it will normalize the volume. Default FALSE.
<code>log.volume</code>	If TRUE log of the volume will be used. Default FALSE.
<code>return.list</code>	Default FALSE will return a data . frame.
<code>impute.value</code>	Default FALSE. If TRUE, impute the missing values.
<code>concurrent.time</code>	Default FALSE. If TRUE, cut the batch data such that control and treatment will end at same time point.

### Value

a data . fram will all the the values stored in experiment slot

**Examples**

```
data(brca)

getExperiment(brca, model.id="X.6047.uned", treatment.only=TRUE)

getExperiment(brca, model.id=c("X.6047.uned", "X.6047.pael"), treatment.only=TRUE)

getExperiment(brca, batch="X-6047.paclitaxel", treatment.only=TRUE)

ed <- list(batch.name="myBatch", treatment=c("X.6047.LJ16","X.6047.LJ16.trab"),
          control=c("X.6047.uned"))

getExperiment(brca, batch=ed)
```

---

getMolecularProfiles *Get molecular profiles from a XevaSet object*

---

**Description**

This function serves to get molecular profiles from a XevaSet object.

**Usage**

```
getMolecularProfiles(object, data.type)
```

**Arguments**

object	The XevaSet.
data.type	character, where one of the molecular data types is needed.

**Value**

An ExpressionSet where sample names are the biobase.id of the model.

**Examples**

```
data(brca)
brca.RNA <- getMolecularProfiles(brca, data.type="RNASeq")
```

lmm *linear mixed model*

---

**Description**

Comput the linear mixed model (lmm) statistics for a PDX batch

**Usage**

```
lmm(data)
```

**Arguments**

data            a data.frame containg a batch data

**Details**

The input data.frame (data) must contain these columns: model.id, volume, time, exp.type

**Value**

Returns a fit object

**Examples**

```
data(repdx)
data <- getExperiment(repdx, batch = "P1")$model
lmm(data)
```

---

modelInfo *modelInfo Generic Generic for modelInfo method*

---

**Description**

modelInfo Generic Generic for modelInfo method

**Usage**

```
modelInfo(object, mDataType = NULL)
```

```
## S4 method for signature 'XevaSet'
modelInfo(object, mDataType = NULL)
```



**Arguments**

object            Xeva object  
mDataType        Molecular data type.

**Value**

A data.frame with the model annotations.

**Examples**

```
data(brca)
mid <- modelInfo(brca)
head(mid)
```

---

mRECIST

*Computes the mRECIST*

---

**Description**

mRECIST Returns the mRECIST for given volume response.

**Usage**

```
mRECIST(time, volume, min.time = 10, return.detail = FALSE)
```

**Arguments**

time            Value of best response.  
volume         Value of best average response.  
min.time        Minimum time after which tumor volume will be considered.  
return.detail   Default FALSE. If TRUE, return all intermediate values.

**Value**

Returns the mRECIST.

**Examples**

```
time <- c(0, 3, 7, 11, 18, 22, 26, 30, 32, 35)
volume<- c(250.8, 320.4, 402.3, 382.6, 384, 445.9, 460.2, 546.8, 554.3, 617.9)
mRECIST(time, volume, min.time=10, return.detail=FALSE)
```

---

PDXMI

*PDX-MI data*

---

**Description**

A dataset containing PDX models minimal information (PDX-MI) standard and corresponding Xeva variable.

**Usage**

```
data(PDXMI)
```

**Format**

An object of class `data.frame` with 45 rows and 4 columns.

**Details**

For details about PDX-MI, see:

Meehan, Terrence F, et al. "PDX-MI: minimal information for patient-derived tumor xenograft models." *Cancer research* 77.21 (2017): e62-e66.

**Source**

<http://cancerres.aacrjournals.org/lookup/doi/10.1158/0008-5472.CAN-17-0582>

---

plotmRECIST

*To plot mRECIST values*

---

**Description**

plotmRECIST plots the mRECIST matrix obtained from `summarizeResponse`.

**Usage**

```
plotmRECIST(  
  mat,  
  control.name = NA,  
  control.col = "#238b45",  
  drug.col = "black",  
  colPalette = NULL,  
  name = "Drug & Models",  
  sort = TRUE,  
  row_fontsize = 12,  
  col_fontsize = 12,  
  draw_plot = TRUE  
)
```

**Arguments**

mat	The mRECIST matrix where rows are drugs and columns are patients.
control.name	Name of the control.
control.col	Color of the control.
drug.col	Color of the drug names.
colPalette	Color palette for mRECIST values.
name	Title of the plot.
sort	If matrix should be sorted before plotting.
row_fontsize	Size of the row name font.
col_fontsize	Size of the column name font.
draw_plot	Default TRUE will plot the figure. If FALSE, return an object.

**Value**

mRECIST plot.

**Examples**

```
data(brca)
brca.mr <- summarizeResponse(brca, response.measure = "mRECIST", group.by="patient.id")
plotmRECIST(as.matrix(brca.mr), control.name = "untreated")
```

---

plotPDX

*Plot batch data*

---

**Description**

Plot data for a batch.id, experiment design or model.id

**Usage**

```
plotPDX(
  object,
  batch = NULL,
  patient.id = NULL,
  drug = NULL,
  model.id = NULL,
  model.color = NULL,
  control.name = NULL,
  max.time = NULL,
  treatment.only = FALSE,
  vol.normal = FALSE,
  impute.value = TRUE,
  concurrent.time = FALSE,
```

```

control.col = "#e41a1c",
treatment.col = "#377eb8",
title = "",
xlab = "Time",
ylab = "Volume",
log.y = FALSE,
SE.plot = c("all", "none", "errorbar", "ribbon"),
aspect.ratio = c(1, NULL),
minor.line.size = 0.5,
major.line.size = 0.7
)

plotBatch(
  object,
  batch = NULL,
  patient.id = NULL,
  drug = NULL,
  control.name = NULL,
  max.time = NULL,
  treatment.only = FALSE,
  vol.normal = FALSE,
  impute.value = TRUE,
  concurrent.time = FALSE,
  control.col = "#6baed6",
  treatment.col = "#fc8d59",
  title = "",
  xlab = "Time",
  ylab = "Volume",
  log.y = FALSE,
  SE.plot = c("all", "none", "errorbar", "ribbon"),
  aspect.ratio = c(1, NULL),
  minor.line.size = 0.5,
  major.line.size = 0.7
)

```

### Arguments

<code>object</code>	Xeva object.
<code>batch</code>	Batch name or experiment design list.
<code>patient.id</code>	Patient id from the XevaSet. Default NULL.
<code>drug</code>	Name of the drug. Default NULL.
<code>model.id</code>	One or multiple model.id. Default NULL.
<code>model.color</code>	Color for model.id. Default NULL.
<code>control.name</code>	Name of the control sample.
<code>max.time</code>	Maximum time point of the plot. Default NULL will plot complete data.
<code>treatment.only</code>	Default FALSE. Given full data <code>treatment.only=TRUE</code> will plot data only during treatment.

vol.normal	Default FALSE. If TRUE, volume will be normalized.
impute.value	Default TRUE will impute values if missing.
concurrent.time	Default FALSE. If TRUE, cut the batch data such that control and treatment will end at the same time point.
control.col	Color for control plots.
treatment.col	Color for treatment plots.
title	Title of the plot.
xlab	Title of the x-axis.
ylab	Title of the y-axis.
log.y	Default FALSE. If TRUE, y-axis will be log-transformed.
SE.plot	Plot type. Default "all" will plot all plots and average curves. Possible values are "all", "none", "errorbar", and "ribbon".
aspect.ratio	Default 1 will create a plot of equal width and height.
minor.line.size	Line size for minor lines. Default 0.5.
major.line.size	Line size for major lines. Default 0.7.

**Value**

A ggplot2 plot with control and treatment batch data.

**Examples**

```
data(brca)
plotPDX(brca, model.id=c("X.6047.LJ16", "X.6047.LJ16.trab"))

plotPDX(brca, batch="X-1004.BGJ398", vol.normal=TRUE)
expDesign <- list(batch.name="myBatch", treatment=c("X.6047.LJ16", "X.6047.LJ16.trab"),
                 control=c("X.6047.uned"))
plotBatch(brca, batch=expDesign, vol.normal=TRUE)
plotBatch(brca, batch=expDesign, vol.normal=FALSE, SE.plot = "errorbar")
```

---

print.batchResponse    *Print the batch response*

---

**Description**

Print the batch response

**Usage**

```
## S3 method for class 'batchResponse'
print(x, ...)
```

**Arguments**

x                   batchResponse object  
...                  Other arguments

**Value**

prints the batchResponse

---

print.modelResponse    *Print the model response*

---

**Description**

Print the model response

**Usage**

```
## S3 method for class 'modelResponse'  
print(x, ...)
```

**Arguments**

x                   modelResponse object  
...                  Other arguments

**Value**

prints the modelResponse

---

print.pdxBatch         *Print the pdx batch*

---

**Description**

Print the pdx batch

**Usage**

```
## S3 method for class 'pdxBatch'  
print(x, ...)
```

**Arguments**

x                   pdxBatch object  
...                  Other arguments

**Value**

prints pdxBatch

---

repx

*Example PDX dataset*

---

**Description**

A Xeva object containing anonymous PDX data with replicates. Each batch has 5 replicates.

**Usage**

data(repx)

**Format**

An object of class XevaSet of length 1.

---

response

*compute PDX response*

---

**Description**

response Computes the drug response of an individual PDX model or batch.

**Usage**

```
response(
  object,
  model.id = NULL,
  batch = NULL,
  res.measure = c("mRECIST", "slope", "AUC", "angle", "abc", "TGI", "lmm"),
  treatment.only = FALSE,
  max.time = NULL,
  impute.value = TRUE,
  min.time = 10,
  concurrent.time = TRUE,
  vol.normal = FALSE,
  log.volume = FALSE,
  verbose = TRUE
)
```

**Arguments**

<code>object</code>	Xeva object.
<code>model.id</code>	<code>model.id</code> for which the drug response is to be computed.
<code>batch</code>	<code>batch.id</code> or experiment design for which the drug response is to be computed.
<code>res.measure</code>	Drug response measure. See Details below
<code>treatment.only</code>	Default FALSE. If TRUE, give data for non-zero dose periods only (if dose data are available).
<code>max.time</code>	Maximum time for data.
<code>impute.value</code>	Default FALSE. If TRUE, impute the missing values.
<code>min.time</code>	Default <b>10</b> days. Used for <i>mRECIST</i> computation.
<code>concurrent.time</code>	Default FALSE. If TRUE, cut the batch data such that control and treatment will end at same time point.
<code>vol.normal</code>	If TRUE it will normalize the volume. Default FALSE.
<code>log.volume</code>	If TRUE log of the volume will be used for response calculation. Default FALSE
<code>verbose</code>	Default TRUE will print information.

**Details**

At present the following response measures are implemented

- *mRECIST* Computes *mRECIST* for individual PDX models
- *slope* Computes slope of the fitted individual PDX curves
- *AUC* Computes area under a PDX curve for individual PDX models
- *angle* Computes angle between treatment and control PDX curves
- *abc* Computes area between the treatment and control PDX curves
- *TGI* Computes tumor growth inhibition using treatment and control PDX curves
- *Imm* Computes linear mixed model (*Imm*) statistics for a PDX batch

**Value**

Returns model or batch drug response object.

**Examples**

```
data(brca)
response(brca, model.id="X.1004.BG98", res.measure="mRECIST")

response(brca, batch="X-6047.paclitaxel", res.measure="angle")

ed <- list(batch.name="myBatch", treatment=c("X.6047.LJ16", "X.6047.LJ16.trab"),
           control=c("X.6047.uned"))
response(brca, batch=ed, res.measure="angle")
```



---

selectModelIds                    *To select model IDs based on drug name and/or tissue type.*

---

### Description

To select model IDs based on drug name and/or tissue type.

### Usage

```
selectModelIds(object, drug = NULL, drug.match.exact = TRUE, tissue = NULL)
```

```
## S4 method for signature 'XevaSet'
```

```
selectModelIds(object, drug = NULL, drug.match.exact = TRUE, tissue = NULL)
```

### Arguments

object	The XevaSet.
drug	Name of the drug.
drug.match.exact	Default TRUE.
tissue	Tumor type. Default NULL.

### Value

A vector with the matched model . ids.

### Examples

```
data(brca)
df = selectModelIds(brca, drug="trastuzumab", drug.match.exact=TRUE, tissue="BRCA")
head(df)
df2 = selectModelIds(brca, drug="trastuzumab", drug.match.exact=FALSE)
head(df2)
```

---

sensitivity                    *Get sensitivity for an Xeva object*

---

### Description

Given a Xeva object, it will return a data . frame detailing sensitivity information.

### Usage

```
sensitivity(object, type = c("model", "batch"), sensitivity.measure = NULL)
```

**Arguments**

object            The Xeva dataset.  
 type             Sensitivity type (either model or batch).  
 sensitivity.measure    Name of the sensitivity.measure. Default NULL will return all sensitivity measures.

**Value**

A data.frame with model or batch ID and sensitivity values.

**Examples**

```
data(brca)
head(sensitivity(brca, type="batch"))
head(sensitivity(brca, type="model"))
```

---

setResponse	<i>set PDX response</i>
-------------	-------------------------

---

**Description**

setResponse sets response of all PDXs in an Xeva object.

**Usage**

```
setResponse(
  object,
  res.measure = c("mRECIST", "slope", "AUC", "angle", "abc", "TGI", "Imm"),
  min.time = 10,
  treatment.only = FALSE,
  max.time = NULL,
  vol.normal = FALSE,
  impute.value = TRUE,
  concurrent.time = TRUE,
  log.volume = FALSE,
  verbose = TRUE
)
```

**Arguments**

object            Xeva object.  
 res.measure      Response measure, multiple measures are allowed. See Details below  
 min.time         Minimum number of days for *mRECIST* computation. Default **10** days.  
 treatment.only   Default FALSE. If TRUE, give data for non-zero dose periods only (if dose data are available).

<code>max.time</code>	Maximum number of days to consider for analysis. Data beyond this will be discarded. Default NULL takes full data.
<code>vol.normal</code>	If TRUE it will will normalize the volume. Default FALSE
<code>impute.value</code>	Default FALSE. If TRUE, impute the missing volume values.
<code>concurrent.time</code>	Default FALSE. If TRUE, cut the batch data such that control and treatment will end at same time point.
<code>log.volume</code>	If TRUE log of the volume will be used for response calculation. Default FALSE
<code>verbose</code>	Default TRUE will print information.

### Details

At present following response measure are implemented

- `mRECIST` Computes mRECIST for indivial PDX model
- `slope` Computes slope of the fitted indivial PDX curve
- `AUC` Computes area under a PDX curve for indivial PDX model
- `angle` Computes angle between treatment and control PDX curves
- `abc` Computes area between the treatment and control PDX curves
- `TGI` Computes tumor growth inhibition using treatment and control PDX curves
- `lmm` Computes linear mixed model (lmm) statistics for a PDX batch

### Value

Returns updated Xeva object.

### Examples

```
data(brca)
brca <- setResponse(brca, res.measure = c("mRECIST"), verbose=FALSE)
```

---

<code>slope</code>	<i>Computes slope</i>
--------------------	-----------------------

---

### Description

`slope` returns the slope for given time and volume data.

### Usage

```
slope(time, volume, degree = TRUE)
```

**Arguments**

time	A vector of time.
volume	A vector of volume.
degree	Default TRUE will give angle in degrees and FALSE will return in radians.

**Value**

Returns the slope and a fit object.

**Examples**

```
time <- c(0, 3, 7, 11, 18, 22, 26, 30, 32, 35)
volume<- c(250.8, 320.4, 402.3, 382.6, 384, 445.9, 460.2, 546.8, 554.3, 617.9)
sl <- slope(time, volume)
par(pty="s")
xylim <- range(c(time, volume))
plot(time, volume, type = "b", xlim = xylim, ylim = xylim)
abline(lm(volume~time))
```

---

subsetXeva	<i>Subset Xeva object.</i>
------------	----------------------------

---

**Description**

Subset Xeva object.

**Usage**

```
subsetXeva(object, ids, id.name, keep.batch = TRUE)
```

**Arguments**

object	The XevaSet object.
ids	IDs to be selected for.
id.name	Names of the IDs.
keep.batch	Default TRUE. If FALSE, remove all other model.ids from the experient design that do not belong to selection.

**Value**

New Xeva object.

**Examples**

```
data(brca)
print(brca)
df <- subsetXeva(brca, ids = c("X-1004", "X-1008", "X-1286"), id.name="patient.id", keep.batch=TRUE)
print(df)
```

---

`summarizeMolecularProfiles`*Summarize molecular profiles*

---

## Description

This function serves to get molecular profiles from a XevaSet object.

## Usage

```
summarizeMolecularProfiles(  
  object,  
  drug,  
  mDataType,  
  tissue = NULL,  
  sensitivity.measure = NULL,  
  unique.model = TRUE,  
  batch = NULL  
)
```

## Arguments

<code>object</code>	The XevaSet.
<code>drug</code>	Name of the drug.
<code>mDataType</code>	character, where one of the molecular data types is needed.
<code>tissue</code>	Default NULL will return all tissue types.
<code>sensitivity.measure</code>	Default NULL will return all sensitivity measures.
<code>unique.model</code>	Default TRUE will return only one sequencing ID, in the case where one model ID maps to several sequencing IDs.
<code>batch</code>	Name of the batch. Default NULL.

## Details

- If a sequencing sample belongs to multiple models, `summarizeMolecularProfiles` will create a separate column for each model.
- All models without molecular data will be removed from the output ExpressionSet.

## Value

An ExpressionSet where sample names are `model.id` and sensitivity measures will be presented in `pData`.

**Examples**

```
data(brca)
pacRNA <- summarizeMolecularProfiles(brca, drug="paclitaxel", mDataType="RNASeq",
                                     tissue= "BRCA", sensitivity.measure="mRECIST")
print(pacRNA)
```

---

summarizeResponse      *Summarize Response of PDXs*

---

**Description**

This function summarizes the drug response information of PDXs.

**Usage**

```
summarizeResponse(
  object,
  response.measure = "mRECIST",
  model.id = NULL,
  batch.id = NULL,
  group.by = "patient.id",
  summary.stat = c(";", "mean", "median"),
  tissue = NULL
)
```

**Arguments**

object	The XevaSet object.
response.measure	character indicating which response measure to use. Use the responseMeasures function to find out what measures are available for each XevaSet.
model.id	The model.id for which data is required.
batch.id	A vector of batch names. Default NULL will return all batches.
group.by	Default patient.id. Dictates how the models should be grouped together. See details below.
summary.stat	Dictates which summary method to use if multiple IDs are found.
tissue	Name of the tissue. Default NULL

**Details**

There can be two types of drug response measure.

- Per model response: One response value for each Model, eg. mRECIST\_recomputed for each model.
- Per batch response: One response value for each Batch, eg. angle between treatment and control groups.

For the per model response output, columns will be model.id (or group.by). For the per batch response output, the group.by value can be "batch.name".

**Value**

A matrix with rows as drug names, column as group.by. Each cell contains response.measure for the pair.

**Examples**

```
data(brca)
brca.mR <- summarizeResponse(brca, response.measure = "mRECIST", group.by="patient.id")
```

---

TGI	<i>tumor growth inhibition (TGI) Computes the tumor growth inhibition (TGI) between two time-volume curves</i>
-----	--

---

**Description**

tumor growth inhibition (TGI) Computes the tumor growth inhibition (TGI) between two time-volume curves

**Usage**

```
TGI(contr.volume, treat.volume)
```

**Arguments**

contr.volume	Volume vector for control
treat.volume	Volume vector for treatment

**Value**

Returns batch response object

**Examples**

```
contr.volume <- c(1.35, 6.57, 13.94, 20.39, 32.2, 39.26, 46.9, 53.91)
treat.volume <- c(0.4, 1.26, 2.59, 3.62, 5.77, 6.67, 7.47, 8.98, 9.29, 9.44)
TGI(contr.volume, treat.volume)
```

---

waterfall	<i>waterfall plot</i> Creates waterfall plot for a given drug.
-----------	--

---

**Description**

waterfall plot Creates waterfall plot for a given drug.

**Usage**

```
waterfall(
  object,
  res.measure,
  drug = NULL,
  group.by = NULL,
  summary.stat = c(";", "mean", "median"),
  tissue = NULL,
  model.id = NULL,
  model.type = NULL,
  type.color = "#cc4c02",
  legend.name = NULL,
  yname = NULL,
  title = NULL,
  sort = TRUE
)
```

**Arguments**

object	The XevaSet object
res.measure	PDX model drug response measure
drug	Name of the drug
group.by	Group drug response data
summary.stat	How to summarize multiple values
tissue	Tissue type
model.id	Indicates which model.id to plot. Default NULL will plot all models
model.type	Type of model, such as mutated or wild type
type.color	A list with colors used for each type in the legend
legend.name	Name of the legend
yname	Name for the y-axis
title	Title of the plot
sort	Default TRUE will sort the data

**Value**

waterfall plot in ggplot2



**Examples**

```
data(brca)
waterfall(brca, drug="binimetinib", res.measure="best.avg.response_published")
## example with model.type where we color the models by TP53 mutation type
mut <- summarizeMolecularProfiles(brca, drug = "binimetinib", mDataType="mutation")
model.type <- Biobase::exprs(mut)["TP53", ]
waterfall(brca, drug="binimetinib", res.measure="best.avg.response_published",
          tissue="BRCA", model.id=names(model.type), model.type= model.type)
```

# Index

- \* **datasets**
  - brca, [7](#)
  - PDXMI, [18](#)
  - repdx, [23](#)
  
- ABC, [2](#)
- addExperimentalDesign, [3](#)
- addExperimentalDesign, XevaSet-method (addExperimentalDesign), [3](#)
- angle, [4](#)
- AUC, [5](#)
  
- batchInfo, [6](#)
- batchInfo, XevaSet-method (batchInfo), [6](#)
- brca, [7](#)
  
- createXevaSet, [8](#)
  
- dosePlot, [9](#)
- downloadXevaSet, [10](#)
- drugInform, [11](#)
- drugInform, XevaSet-method (drugInform), [11](#)
- drugSensitivitySig, [12](#)
  
- getExperiment, [13](#)
- getExperiment, XevaSet-method (getExperiment), [13](#)
- getMolecularProfiles, [15](#)
  
- Imm, [16](#)
  
- modelInfo, [16](#)
- modelInfo, XevaSet-method (modelInfo), [16](#)
- mRECIST, [17](#)
  
- PDXMI, [18](#)
- plotBatch (plotPDX), [19](#)
- plotmRECIST, [18](#)
- plotPDX, [19](#)
- print.batchResponse, [21](#)
  
- print.modelResponse, [22](#)
- print.pdxBatch, [22](#)
  
- repdx, [23](#)
- response, [23](#)
  
- selectModelIds, [25](#)
- selectModelIds, XevaSet-method (selectModelIds), [25](#)
- sensitivity, [25](#)
- setResponse, [26](#)
- slope, [27](#)
- subsetXeva, [28](#)
- summarizeMolecularProfiles, [29](#)
- summarizeResponse, [30](#)
  
- TGI, [31](#)
  
- waterfall, [32](#)