The Xeva User's Guide

Arvind Mer [1,2] ***and Benjamin Haibe-Kains*** [1,2,3,4,5]

[1]Princess Margaret Cancer Centre, University Health Network, Toronto, Canada
[2]Department of Medical Biophysics, University of Toronto, Toronto, Canada
[3]Department of Computer Science, University of Toronto, Toronto, Canada
[4]Vector Institute, Toronto, Ontario, Canada
[5]Ontario Institute for Cancer Research, Toronto, Ontario, Canada

September 28, 2020

Contents

Introduction

The Xeva package provides efficient and powerful functions for patient-drived xenograft (PDX) based pharmacogenomic data analysis [1].

Installation and Settings

Xeva requires that several packages be installed. All dependencies are available from CRAN or Bioconductor:

```
if (!requireNamespace("BiocManager", quietly = TRUE))
    install.packages("BiocManager")
BiocManager::install("Xeva", version = "3.9")
```

The package can also be installed directly form GitHub using devtools:

```
#install devtools if required
install.packages("devtools")

#install Xeva as:
devtools::install_github("bhklab/Xeva")
```

Load Xeva into your current workspace:

```
library(Xeva)
```

Load the dataset you wish to analyze. For the sake of this tutorial, here we load the Novartis PDXE [2] breast cancer dataset as an example:

```
data(brca)
print(brca)
```

```
## XevaSet
## name: PDXE.BRCA
## Creation date: Fri Sep 14 11:41:33 2018
## Number of models: 849
## Number of drugs: 22
## Moleculer dataset: RNASeq, mutation, cnv
```

Definitions

Before we further dive into the analysis and visualization, it is important to understand the terminology used in the *Xeva* package. In a **Xeva** object, the **experiment** slot stores the data for each individual PDX/mouse. With the exception of tumor growth data (time vs. tumor volume), for each individual PDX/mouse, you can access metadata such as the patient's age, sex, tissue histology, and passage information. All of this metadata is stored in the **pdxModel** class, where a unique ID called `model.id` is given to each PDX/mouse model. As for the

tumor growth information, Xeva provides separate functions for retrieving and visualizing time vs. tumor volume data. We will see later how to get these data for an individual *model.id*, but first, let's define some other terms that appear in the *Xeva* package.

A PDX experiment can be one of the two categories:

- **treatment** represents experiments in which the PDX receives some kind of drug (or drug combination)
- **control** represents experiments in which the PDX receives no drug

To see the effect of a drug, several replicate experiments are done for both the control and the treatment categories. In **Xeva**, a collection of PDX *model.ids* originating from the same patient is organized in **batches** (*batch*). A *batch* has two arms: *control* and *treatment*. This is illustrated in Figure 1.

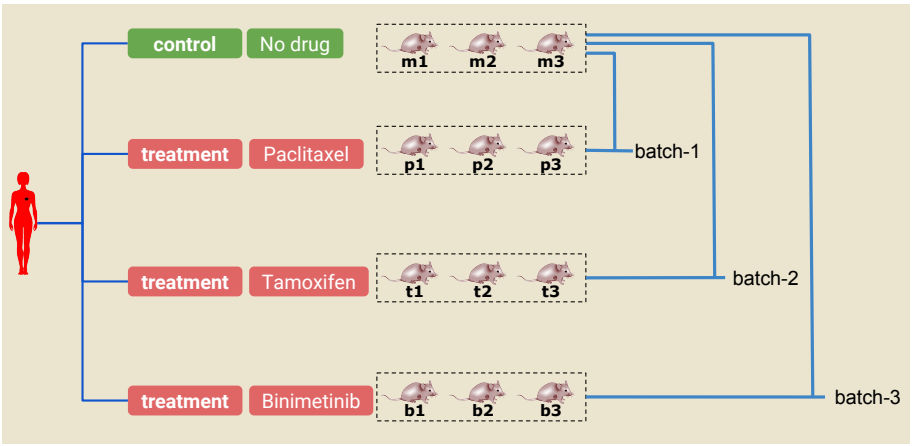

Figure 1: A PDX experiment
The text under each of the PDX/mouse (ie. m1, m2, p1, etc.) denotes the *model.id* in **Xeva**. In this example, three PDXs are delclared as control (m1, m2, and m3). Similarly, in the treatment arm, 3 PDXs are given the drug paclitaxel (p1, p2, and p3), 3 are given tamoxifen (t1, t2, and t3), and 3 are given binimetinib (b1, b2, b3). The PDXs in the control arm and one of the treatment arms together constitute a *batch*. For example, control arm models (m1, m2, and m3) and treatment arm models (t1,t2, and t3) together create a batch called batch-2.

A **Xeva** object binds together all individual experiments, batch information, and molecular data into one single class called XevaSet.

Data Access

As mentioned earlier, **Xeva** stores metadata for each individual PDX model. We can retrieve the meta-information about each PDX, such as number of models and tissue type, using:

```
brca.mod <- modelInfo(brca)
dim(brca.mod)

## [1] 849   5

brca.mod[1:4, ]

##               model.id tissue   tissue.name patient.id       drug
```

```
## X.1004.BG98 X.1004.BG98   BRCA Breast Cancer     X-1004      BGJ398
## X.1004.biib X.1004.biib   BRCA Breast Cancer     X-1004 binimetinib
## X.1004.BK20 X.1004.BK20   BRCA Breast Cancer     X-1004      BKM120
## X.1004.BY19 X.1004.BY19   BRCA Breast Cancer     X-1004      BYL719
```

The output shows that the *brca* dataset contains 849 PDX models. We can also see the time vs. tumor volume data for a model using:

```
model.data <- getExperiment(brca, model.id = "X.1004.BG98")
head(model.data)
```

```
##      model.id drug.join.name time volume body.weight volume.normal
## 1 X.1004.BG98         BGJ398    0  199.7        28.2     0.0000000
## 2 X.1004.BG98         BGJ398    2  181.9        28.0    -0.0891337
## 3 X.1004.BG98         BGJ398    5  172.7        28.4    -0.1352028
## 4 X.1004.BG98         BGJ398    9  129.6        27.2    -0.3510265
## 5 X.1004.BG98         BGJ398   12   91.3        26.7    -0.5428142
## 6 X.1004.BG98         BGJ398   16  117.1        26.2    -0.4136204
```

Similarly, for **batch** names, we can obtain all predefined batch names using:

```
batch.name <- batchInfo(brca)
batch.name[1:4]
```

```
## [1] "X-1004.BGJ398"      "X-1004.binimetinib" "X-1004.BKM120"
## [4] "X-1004.BYL719"
```

The information about a **batch** can be shown using:

```
batchInfo(brca, batch = "X-1004.binimetinib")
```

```
## $`X-1004.binimetinib`
## name = X-1004.binimetinib
## control = X.1004.uned
## treatment = X.1004.biib
```

Here, for the batch named *X-1004.binimetinib*, we can see that the control sample is *X.1004.uned* and the treatment sample is *X.1004.biib*.

Visualizing PDX Growth Curve

Xeva provides a function to plot time vs. tumor volume data for individual models as well as for individual batches. These data can be plotted by using the name of the batch:

```
plotPDX(brca, batch = "X-4567.BKM120")
```

You can choose to see different aspects of this visualization. For example, we can plot normalized volume; we can also change the colors of the lines:

```
plotPDX(brca, batch = "X-4567.BKM120", vol.normal = TRUE, control.col = "#a6611a",
        treatment.col = "#018571", major.line.size = 1, max.time = 40)
```

Data can also be visualized at the patient level by specifying `patient.id`:

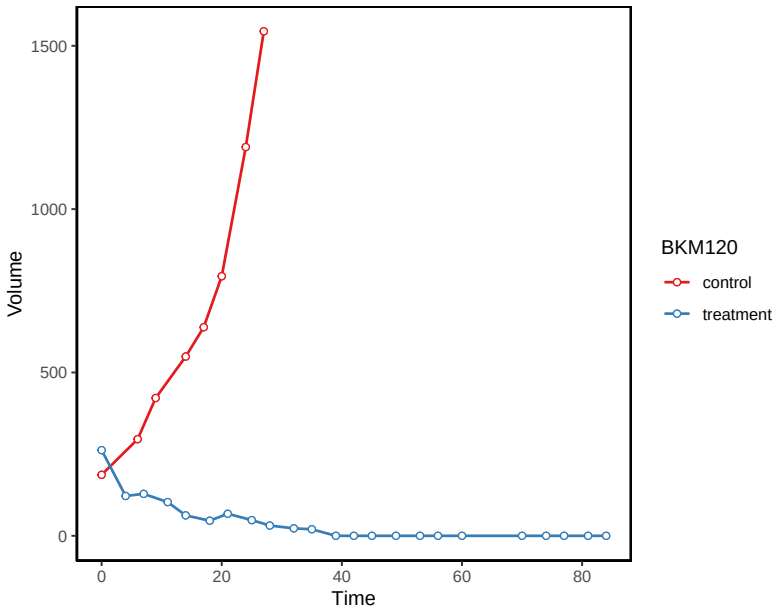

Figure 2: Tumor growth curves for a batch of control and treated PDXs

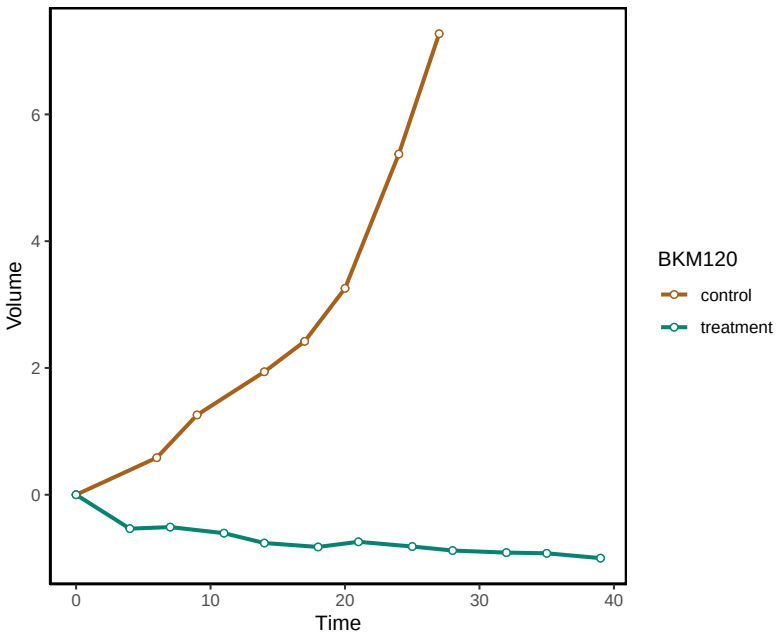

Figure 3: Tumor growth curves for a batch of control and treated PDXs
Here, the volume is normalized and plots are truncated at 40 days

```
plotPDX(brca, patient.id="X-3078", drug="paclitaxel",control.name = "untreated")
```

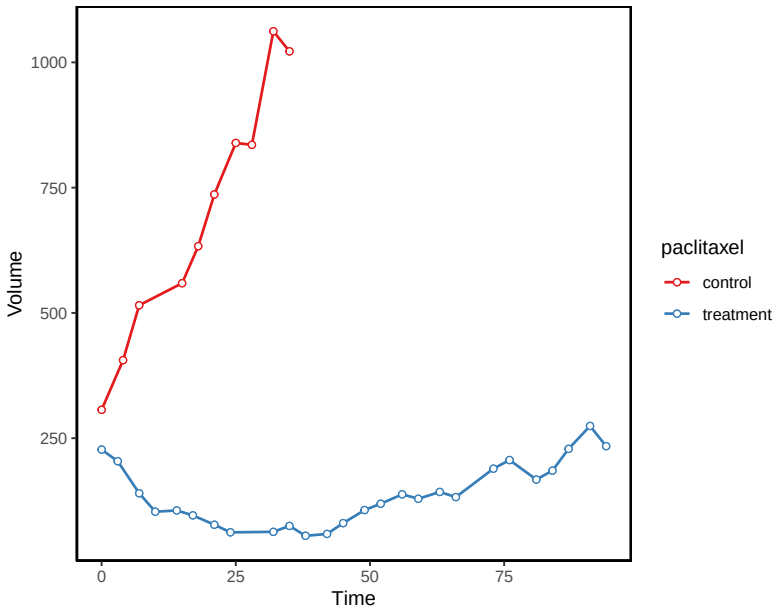

Figure 4: **Tumor growth curves for a batch of control and treated PDXs generated using patient ID and drug name**

Replicate-based PDX experiments

Xeva can also handle replicate-based experiment design. The datasets included in the package also contain replicate-based PDX experiments. To plot replicate-based data:

```
data("repdx")
plotPDX(repdx, vol.normal = TRUE, batch = "P1")
```

```
plotPDX(repdx, batch = "P3", SE.plot = "errorbar")
```

```
plotPDX(repdx, batch = "P4", vol.normal = TRUE,  SE.plot = "ribbon")
```

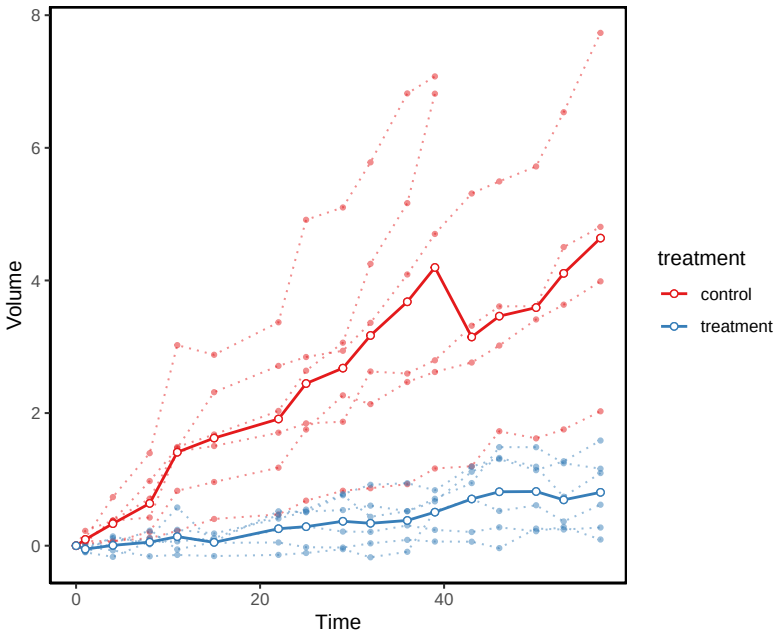

Figure 5: Tumor growth curves for a batch of control and treated PDXs with replicates

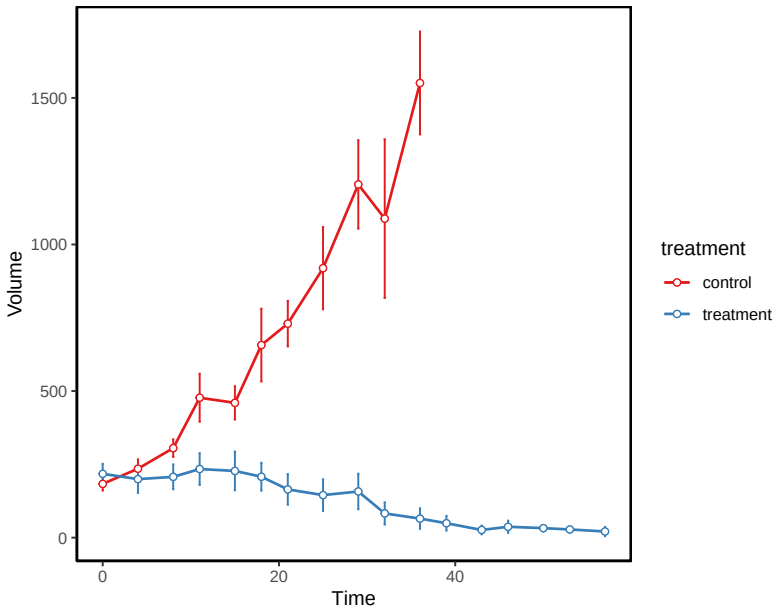

Figure 6: Errorbar visualization for tumor growth curves of a PDX batch

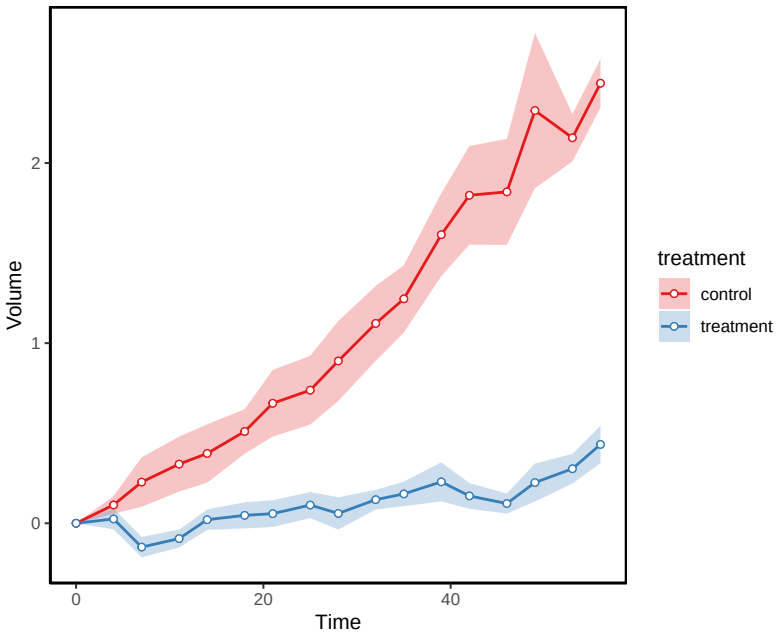

Figure 7: Ribbon visualization for tumor growth curves of a PDX batch

PDX Model Drug Response

PDX response calculation

In Xeva, we have implemented several response metrics. There are two kinds of response metrics available:

1. Model response: This type of response is computed for any individual PDX model. Examples include how fast a PDX model is growing or how long the PDX survives. Such response does not take into account the control PDX as it is computed for an individual PDX model. In this category the following metrics can be computed:
   (a) mRECIST and associated metrics best.response (BR), best.average.response (BAR). mRECIST divides response into 4 categories [1].
      - complete response (CR): BR $<$ -95% and BAR $<$ -40%
      - partial response (PR): BR $<$ -50% and BAR $<$ -20%
      - stable disease (SD): BR $<$ 35% and BAR $<$ 30%
      - progressive disease (PD): not otherwise categorized

   (b) slope is defined as angle of PDX volume curve.
   (c) AUC (area under the curve) represents area covered by PDX curve per day (AUC divided by time).
2. Batch response: This type of response is computed by taking into account models in control and treatment group. In this category the following metrics can be computed:
   (a) Angle between treatment and control PDXs
   (b) abc (area between the curves) is computed as difference in AUC of control and treatment.
   (c) TGI (tumor growth inhibition)
   (d) lmm (linear mix model)
   (e) bmRECIST (batch level mRECIST, computed separately for control and treatment arm)

In Xeva we have provided a unified function **response** to compute all different response metrics. Functions for individual response metrics are also available for use (see documention of the package for details). To compute mRECIST of a model:

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

```
## computing mRECIST for X.1004.BG98
## mRECIST = PR
## Best average response = -28.0754
## Best response = -54.2814
```

Similarly, TGI (a batch level response) can be computed as:

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

```
## computing TGI for batch X-6047.paclitaxel
## TGI = 57.448529
```

Setting response for XevaSet

In a newly created Xevaset, one has to compute the response for all models and batches before using it for analysis. This can be done easily using the **setResponse** function. In this example we are computing mRECIST for all models and getting an updated XevaSet (which contains mRECIST values):

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

Next we update the object by computing the TGI as:

```
brca  <- setResponse(brca, res.measure = "TGI", verbose=FALSE)
```

Visualizing PDX response

Xeva can effectively summarize PDX drug response data. Here we summarize the **mRECIST** values for the models in our dataset:

```
brca.mr <- summarizeResponse(brca, response.measure = "mRECIST")
brca.mr[1:5, 1:4]
```

```
##                 X-1004 X-1008 X-1286 X-1298
## BGJ398              PR     SD     PD     SD
## binimetinib         PD     SD     SD     PD
## BKM120              SD     SD     SD     PR
## BYL719              SD     PR     SD     PD
## BYL719 + LEE011     PD     SD     SD     PD
```

These **mRECIST** values can be visualized using:

```
plotmRECIST(brca.mr, control.name="untreated", row_fontsize=13, col_fontsize=12)
```

Waterfall plots are also commonly used to visualize PDX drug response data. Xeva provides a function to visualize and color waterfall plots:

```
waterfall(brca, drug="binimetinib", res.measure="best.average.response")
```

It is useful to color the bars of your waterfall plot by genomic properties. Here we create a waterfall plot for drug BYL719 and color it based on the mutation status of the CDK13 gene. First, we extract the genomic data for the models. Then, we can plot the waterfall plots:

```
mut <- summarizeData(brca,drug = "BYL719", mDataType="mutation")
model.type <- Biobase::exprs(mut)["CDK13", ]
model.type[grepl("Mut", model.type)] <- "mutation"
model.type[model.type!="mutation"] <- "wild type"
model.color <- list("mutation"="#b2182b", "wild type"="#878787")
waterfall(brca, drug="BYL719", res.measure="best.average.response",
```

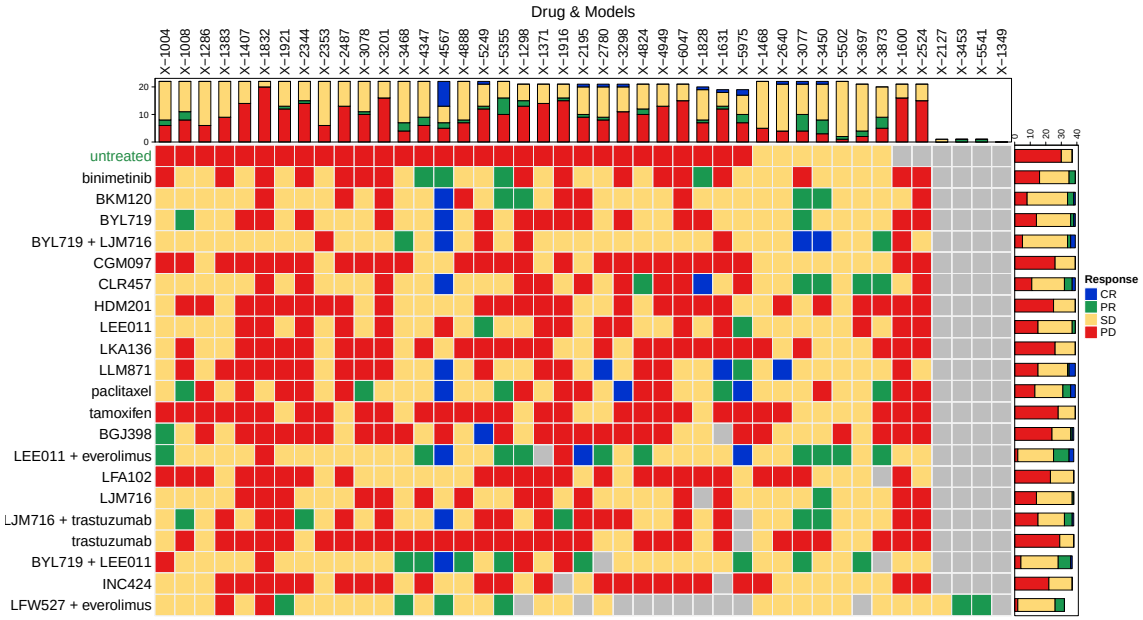

Figure 8: mRECIST plot for PDXE breast cancer data

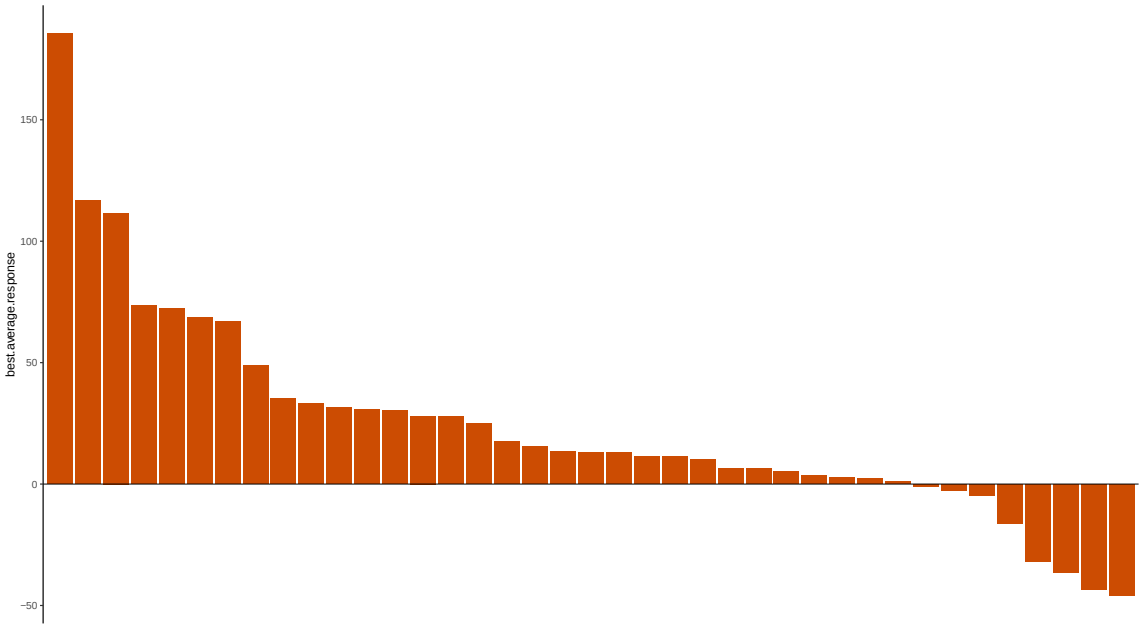

Figure 9: Waterfall plot for binimetinib drug response in PDXs

```
                model.id=names(model.type), model.type= model.type,
                type.color = model.color)
```

In Xeva we have implemented difference matrix to compute PDX response. The Xeva function **response** provides a unified interface for this purpose. In the example below we compute the angle between treatment and control PDXs

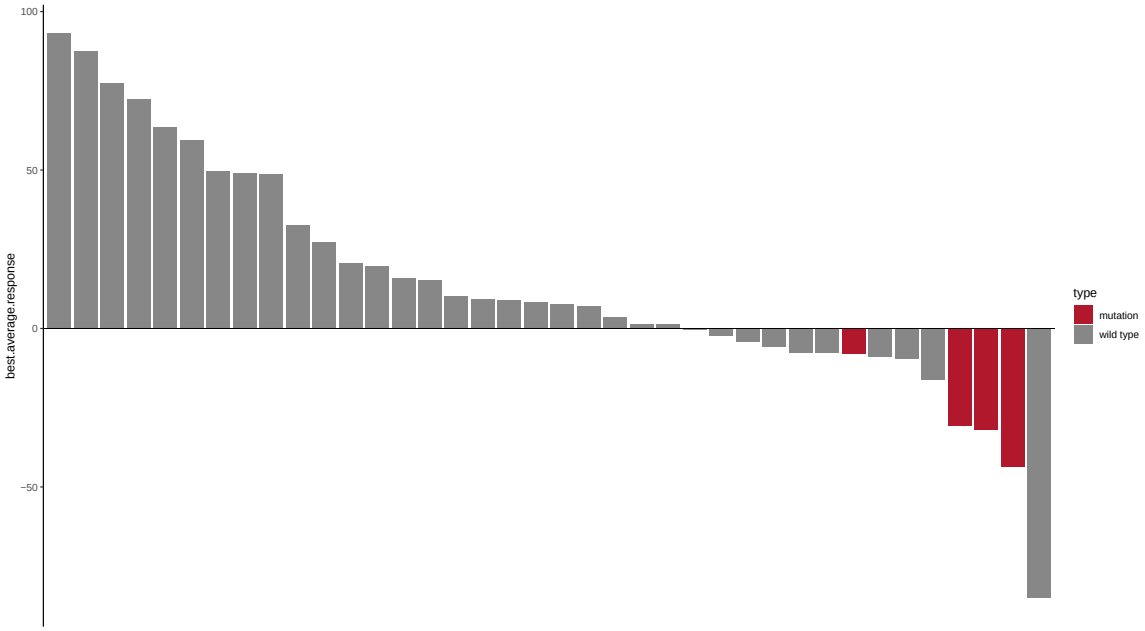

Figure 10: Waterfall plot for BYL719 drug response in PDXs

```
data("repdx")
response(repdx, batch="P1", res.measure="angle")
```

```
## computing angle for batch P1
## angle = 18.665650
## control = 85.751806
## treatment = 67.086156
```

A function for linear mixed-effects model (lmm) has also been implemented.

```
data("repdx")
response(repdx, batch="P1", res.measure="lmm")
```

```
## computing lmm for batch P1
## Linear mixed-effects model fit by REML
##   Data: data
##   Log-restricted-likelihood: 3.184927
##   Fixed: volume ~ time * exp.type
##           (Intercept)                  time      exp.typetreatment
##            5.32592390            0.03091321            -0.28718018
## time:exp.typetreatment
##           -0.01983870
##
## Random effects:
##  Formula: ~1 | model.id
##         (Intercept)  Residual
## StdDev:   0.3802407 0.1975011
##
## Number of Observations: 194
```

```
## Number of Groups: 12
```

Gene-drug association

The main aim of the pharmacogenomic experiments is to find biomarkers for drug response prediction. The Xeva package provides the **drugSensitivitySig** function to compute the univariate association between PDX's molecular data (such as gene expression) and response to a drug (gene-drug association). In the example bellow, we are computing the association between gene expression (RNASeq) and slope of the PDXs for the drug 'tamoxifen' using linear regression (lm).

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

```
## Running for 38 samples with non NA sensitivity.measure
##   feature      drug    estimate        se  n       tstat       fstat
## 1  PIK3R6 tamoxifen  0.12174756 0.1654268 38  0.73596007 0.541637222
## 2   FGFR1 tamoxifen -0.16019355 0.1645143 38 -0.97373653 0.948162830
## 3   NTRK2 tamoxifen  0.16560706 0.1643653 38  1.00755487 1.015166810
## 4    AKT1 tamoxifen  0.00996873 0.1666584 38  0.05981535 0.003577876
## 5   FGFR4 tamoxifen  0.07953256 0.1661387 38  0.47871178 0.229164971
##      pvalue df       fdr
## 1 0.4665236 36 0.7775393
## 2 0.3366852 36 0.7775393
## 3 0.3203927 36 0.7775393
## 4 0.9526335 36 0.9526335
## 5 0.6350385 36 0.7937981
```

In this above example we took only 5 features (genes), however this can be extended for any number of genes. For larger analyses, this function also provides out of box parallel computation. Users can set the number of cores using the parameter **nthread** in the function.

Users can choose different sensitivity measures of the PDX response for the association analysis by setting the parameter **sensitivity.measure**. For example, below we use *best.average.response* as the PDX's response matrix in the association analysis:

```
data(brca)
drugSensitivitySig(object=brca, drug="tamoxifen", mDataType="RNASeq",
                   features=c(1,2,3,4,5),
                   sensitivity.measure="best.average.response", fit = "lm")
```

```
## Running for 38 samples with non NA sensitivity.measure
##   feature      drug    estimate        se  n      tstat     fstat    pvalue
## 1  PIK3R6 tamoxifen  0.10155551 0.1658050 38  0.6124998 0.3751559 0.5440570
## 2   FGFR1 tamoxifen  0.25347267 0.1612238 38  1.5721794 2.4717480 0.1246577
## 3   NTRK2 tamoxifen  0.15268488 0.1647125 38  0.9269782 0.8592885 0.3601112
## 4    AKT1 tamoxifen -0.19722241 0.1633931 38 -1.2070422 1.4569510 0.2352874
## 5   FGFR4 tamoxifen -0.06903067 0.1662691 38 -0.4151744 0.1723698 0.6804783
##   df       fdr
## 1 36 0.6800713
```

```
## 2 36 0.5882184
## 3 36 0.6001853
## 4 36 0.5882184
## 5 36 0.6804783
```

For the drug-gene association analysis, users can also choose a different method of association calculation (such as concordance index, Pearson or Spearman correlation) by setting the parameter *fit*.

```
data(brca)
drugSensitivitySig(object=brca, drug="tamoxifen", mDataType="RNASeq",
                   features=c(1,2,3,4,5),
                   sensitivity.measure="best.average.response", fit="pearson")
```

```
## Running for 38 samples with non NA sensitivity.measure
##   feature      drug         cor   p.value
## 1  PIK3R6 tamoxifen  0.10155551 0.5440570
## 2   FGFR1 tamoxifen  0.25347267 0.1246577
## 3   NTRK2 tamoxifen  0.15268488 0.3601112
## 4    AKT1 tamoxifen -0.19722241 0.2352874
## 5   FGFR4 tamoxifen -0.06903067 0.6804783
```

Creating new Xeva object

New Xeva objects can be created using *createXevaSet*. The different components that are needed by the function is as follows:

model A data.frame containing **model.id** and other relevant model information, such as tissue of origin, patient ID, and passage information. All **PDXMI** variables can be inserted into this data.frame. A required column in the data.frame is **model.id**. An example of the **model** can be found in the package as:

```
model=read.csv(system.file("extdata", "model.csv", package = "Xeva"))
head(model)
```

```
##      model.id tissue   tissue.name patient.id
## 1 X.1004.biib   BRCA Breast Cancer     X-1004
## 2 X.1008.biib   BRCA Breast Cancer     X-1008
## 3 X.1286.biib   BRCA Breast Cancer     X-1286
## 4 X.1004.uned   BRCA Breast Cancer     X-1004
## 5 X.1008.uned   BRCA Breast Cancer     X-1008
## 6 X.1286.uned   BRCA Breast Cancer     X-1286
```

drug A data.frame containing information about the drugs used in the experiments. The required column is **drug.id**. An example of the **drug** can be found in the package as:

```
drug=read.csv(system.file("extdata", "drug.csv", package = "Xeva"))
head(drug)
```

```
##       drug.id standard.name   treatment.target treatment.type
## 1 binimetinib   binimetinib MAP2K1,MAP2K2,MAPK         single
## 2   untreated     untreated  UNKNOWN,Untreated         single
```

experiment A data.frame with time vs. tumor volume information. The required columns are **model.id**, **time**, **volume**, and **drug**. Other avaliable information such as dose amount and mouse weight can be specified as columns of this data.frame . An example of the **experiment** can be found in the package as:

```
experiment=read.csv(system.file("extdata","experiments.csv",package="Xeva"))
head(experiment)
```

```
##      model.id time volume        drug
## 1 X.1004.biib    0  250.8 binimetinib
## 2 X.1004.biib    3  320.4 binimetinib
## 3 X.1004.biib    7  402.3 binimetinib
## 4 X.1004.biib   11  382.6 binimetinib
## 5 X.1004.biib   18  384.0 binimetinib
## 6 X.1004.biib   22  445.9 binimetinib
```

expDesign This is an R **list** object that contains the batch information. Each element of the list should have 3 components **batch.name**, **treatment** and **control**. The model.ids in the treatment and control must be present in **model** variable described earlier. An example of expDesign is:

```
expDesign=readRDS(system.file("extdata","batch_list.rds",package="Xeva"))
expDesign[[1]]
```

```
## $batch.name
## [1] "X-1004.binimetinib"
##
## $treatment
## [1] "X.1004.biib"
##
## $control
## [1] "X.1004.uned"
```

molecularProfiles This list contains all the molecular data such as RNAseq or mutation information. Each element of this list should contain an **ExpressionSet** object. An example of such an object is:

```
RNASeq=readRDS(system.file("extdata", "rnaseq.rds", package = "Xeva"))
print(RNASeq)
```

```
## ExpressionSet (storageMode: lockedEnvironment)
## assayData: 22 features, 2 samples
##   element names: exprs
## protocolData: none
```

```
## phenoData
##   sampleNames: X-1004 X-1008
##   varLabels: biobase.id patient.id ... tissue (5 total)
##   varMetadata: labelDescription
## featureData
##   featureNames: PIK3R6 FGFR1 ... TUBB3 (22 total)
##   fvarLabels: geneName ensembl.id
##   fvarMetadata: labelDescription
## experimentData: use 'experimentData(object)'
## Annotation:
```

modToBiobaseMap A data.frame which contains mapping between the PDX model.id and the molecularProfiles sample names. It requires 3 variables: **model.id**, **biobase.id**, and **mDataType**. An example of modToBiobaseMap is:

```
modToBiobaseMap=read.csv(system.file("extdata","modelToExpressionMap.csv",
                                     package = "Xeva"))
head(modToBiobaseMap)
```

```
##      model.id biobase.id mDataType
## 1 X.1004.biib     X-1004    RNASeq
## 2 X.1004.uned     X-1004    RNASeq
## 3 X.1008.biib     X-1008    RNASeq
## 4 X.1008.uned     X-1008    RNASeq
```

A new Xeva object can be created as:

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

```
## ##--- making experiment slot -------
## Drug columns are
## drug
## ##--- making model information -------
## ##--- making experiment design slot -------
## ##--- making sensitivity slot -------
## ##--- making drug slot -------
## ##--- making id mapping slot -------
## ##--- creating XevaSet -------
```

```
print(xeva.set)
```

```
## XevaSet
## name: example xevaSet
## Creation date: Mon Sep 28 13:44:38 2020
## Number of models: 6
## Number of drugs: 2
## Moleculer dataset: RNASeq
```

Once you have created a new XevaSet you can compute PDX response using the function 'setResponse' and update your object.

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

For more details see 7 and 7.2

References

[1] Arvind S Mer, Wail Ba-Alawi, Petr Smirnov, Yi X Wang, Ben Brew, Janosch Ortmann, Ming-Sound Tsao, David W Cescon, Anna Goldenberg, and Benjamin Haibe-Kains. Integrative pharmacogenomics analysis of patient-derived xenografts. *Cancer research*, 79(17):4539–4550, 2019.

[2] Hui Gao, Joshua M Korn, Stéphane Ferretti, John E Monahan, Youzhen Wang, Mallika Singh, Chao Zhang, Christian Schnell, Guizhi Yang, Yun Zhang, et al. High-throughput screening using patient-derived tumor xenografts to predict clinical trial drug response. *Nature medicine*, 21(11):1318, 2015.

SessionInfo

```
sessionInfo()
```

```
## R version 4.0.2 (2020-06-22)
## Platform: x86_64-pc-linux-gnu (64-bit)
## Running under: Linux Mint 18.3
##
## Matrix products: default
## BLAS:   /usr/lib/libblas/libblas.so.3.6.0
## LAPACK: /usr/lib/lapack/liblapack.so.3.6.0
##
## locale:
##  [1] LC_CTYPE=en_CA.UTF-8       LC_NUMERIC=C
##  [3] LC_TIME=en_CA.UTF-8        LC_COLLATE=en_CA.UTF-8
##  [5] LC_MONETARY=en_CA.UTF-8    LC_MESSAGES=en_CA.UTF-8
##  [7] LC_PAPER=en_CA.UTF-8       LC_NAME=C
##  [9] LC_ADDRESS=C               LC_TELEPHONE=C
## [11] LC_MEASUREMENT=en_CA.UTF-8 LC_IDENTIFICATION=C
##
## attached base packages:
## [1] parallel  stats     graphics  grDevices utils     datasets  methods
## [8] base
##
## other attached packages:
## [1] Biobase_2.48.0      BiocGenerics_0.34.0 Xeva_0.99.19
## [4] knitr_1.29
##
## loaded via a namespace (and not attached):
##   [1] fgsea_1.14.0                colorspace_1.4-1
##   [3] rjson_0.2.20                ellipsis_0.3.1
##   [5] circlize_0.4.10             lsa_0.73.2
##   [7] XVector_0.28.0              GenomicRanges_1.40.0
##   [9] GlobalOptions_0.1.2         clue_0.3-57
##  [11] farver_2.0.3                SnowballC_0.7.0
##  [13] DT_0.15                     codetools_0.2-16
##  [15] doParallel_1.0.15           jsonlite_1.7.0
##  [17] magicaxis_2.0.10            cluster_2.1.0
##  [19] png_0.1-7                   shinydashboard_0.7.1
##  [21] shiny_1.5.0                 BiocManager_1.30.10
##  [23] mapproj_1.2.7               compiler_4.0.2
##  [25] backports_1.1.9             Matrix_1.2-18
##  [27] fastmap_1.0.1               limma_3.44.3
##  [29] later_1.1.0.1               visNetwork_2.0.9
##  [31] htmltools_0.5.0             tools_4.0.2
##  [33] igraph_1.2.5                CoreGx_1.0.2
##  [35] gtable_0.3.0                glue_1.4.2
##  [37] GenomeInfoDbData_1.2.3      RANN_2.6.1
##  [39] reshape2_1.4.4              dplyr_1.0.2
##  [41] maps_3.3.0                  fastmatch_1.1-0
##  [43] Rcpp_1.0.5                  slam_0.1-47
```

```
##  [45] vctrs_0.3.2                 PharmacoGx_2.0.5
##  [47] nlme_3.1-149                gdata_2.18.0
##  [49] iterators_1.0.12            xfun_0.16
##  [51] stringr_1.4.0               Rmisc_1.5
##  [53] testthat_2.3.2              mime_0.9
##  [55] lifecycle_0.2.0             gtools_3.8.2
##  [57] zlibbioc_1.34.0             MASS_7.3-52
##  [59] scales_1.1.1                BiocStyle_2.16.0
##  [61] promises_1.1.1              relations_0.6-9
##  [63] SummarizedExperiment_1.18.2 RColorBrewer_1.1-2
##  [65] BBmisc_1.11                 sets_1.0-18
##  [67] ComplexHeatmap_2.4.3        yaml_2.2.1
##  [69] gridExtra_2.3               ggplot2_3.3.2
##  [71] downloader_0.4              stringi_1.4.6
##  [73] highr_0.8                   NISTunits_1.0.1
##  [75] S4Vectors_0.26.1            foreach_1.5.0
##  [77] plotrix_3.7-8               checkmate_2.0.0
##  [79] caTools_1.18.0              BiocParallel_1.22.0
##  [81] shape_1.4.4                 GenomeInfoDb_1.24.2
##  [83] rlang_0.4.7                 pkgconfig_2.0.3
##  [85] matrixStats_0.56.0          bitops_1.0-6
##  [87] pracma_2.2.9                evaluate_0.14
##  [89] lattice_0.20-41             purrr_0.3.4
##  [91] labeling_0.3                htmlwidgets_1.5.1
##  [93] tidyselect_1.1.0            plyr_1.8.6
##  [95] magrittr_1.5                R6_2.4.1
##  [97] IRanges_2.22.2              gplots_3.0.4
##  [99] generics_0.0.2              DelayedArray_0.14.1
## [101] sm_2.2-5.6                  pillar_1.4.6
## [103] RCurl_1.98-1.2              tibble_3.0.3
## [105] crayon_1.3.4                KernSmooth_2.23-17
## [107] rmarkdown_2.3               GetoptLong_1.0.2
## [109] grid_4.0.2                  data.table_1.13.0
## [111] marray_1.66.0               piano_2.4.0
## [113] digest_0.6.25               xtable_1.8-4
## [115] httpuv_1.5.4                stats4_4.0.2
## [117] munsell_0.5.0               celestial_1.4.6
## [119] tcltk_4.0.2                 shinyjs_1.1
```