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

January 8, 2019

Contents

1 Introduction

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

2 Installation and Settings

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

```
source('http://bioconductor.org/biocLite.R')
biocLite('Xeva')
```

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
```

3 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.

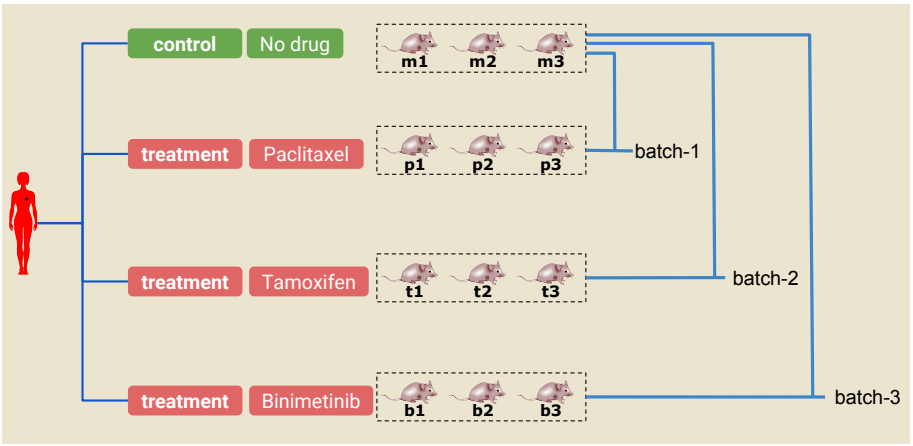

Figure 1: A PDX experiment
The text under each of the PDX/mouse (ie. m1, m2, p1, etc.) denotes the *model.id* in **Xeva**. In this example, three PDXs are delclared as control (m1, m2, and m3). Similarly, in the treatment arm, 3 PDXs are given the drug paclitaxel (p1, p2, and p3), 3 are given tamoxifen (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.

4 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*.

5 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 = T, 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`:

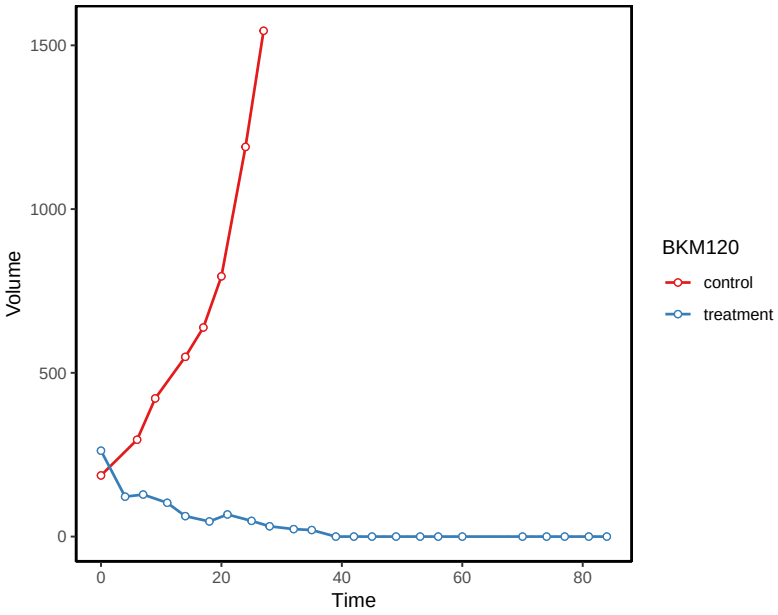

Figure 2: Tumor growth curves for a batch of control and treated PDXs

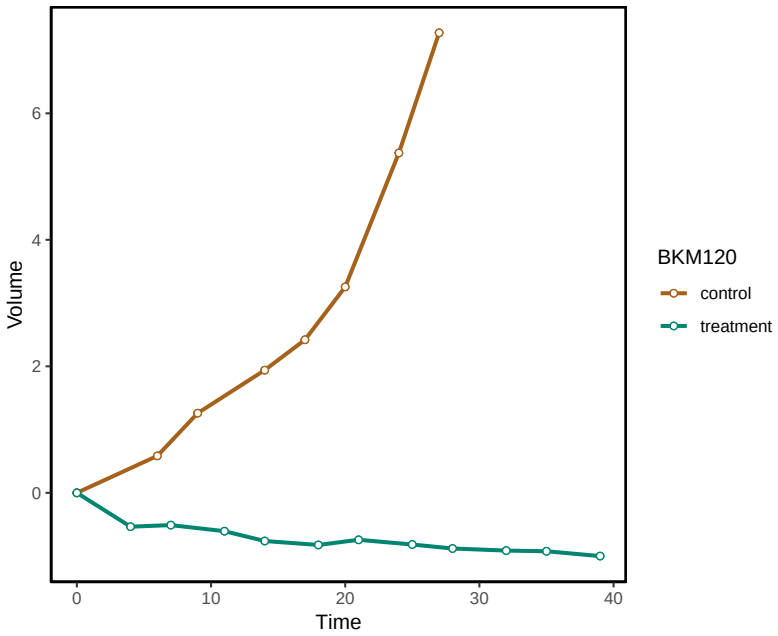

Figure 3: Tumor growth curves for a batch of control and treated PDXs
Here, the volume is normalized and plots are truncated at 40 days

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

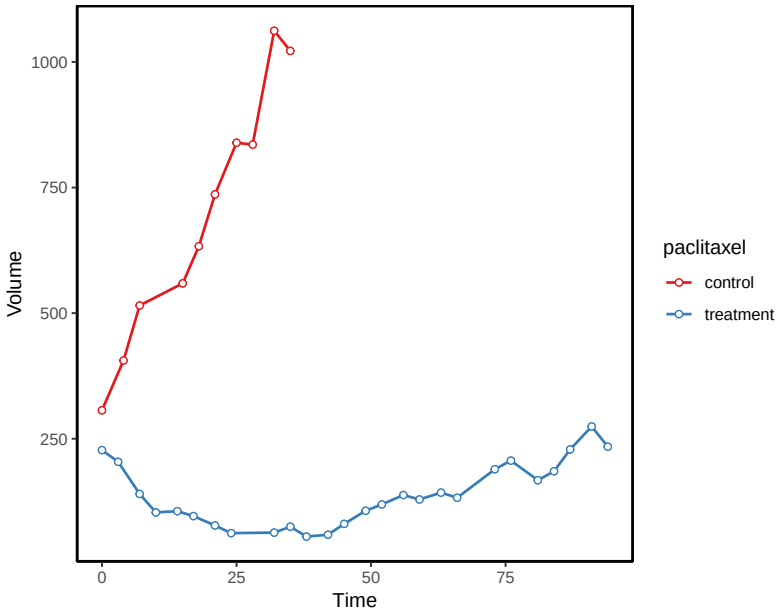

Figure 4: Tumor growth curves for a batch of control and treated PDXs generated using patient ID and drug name

6 PDX Model Drug 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:

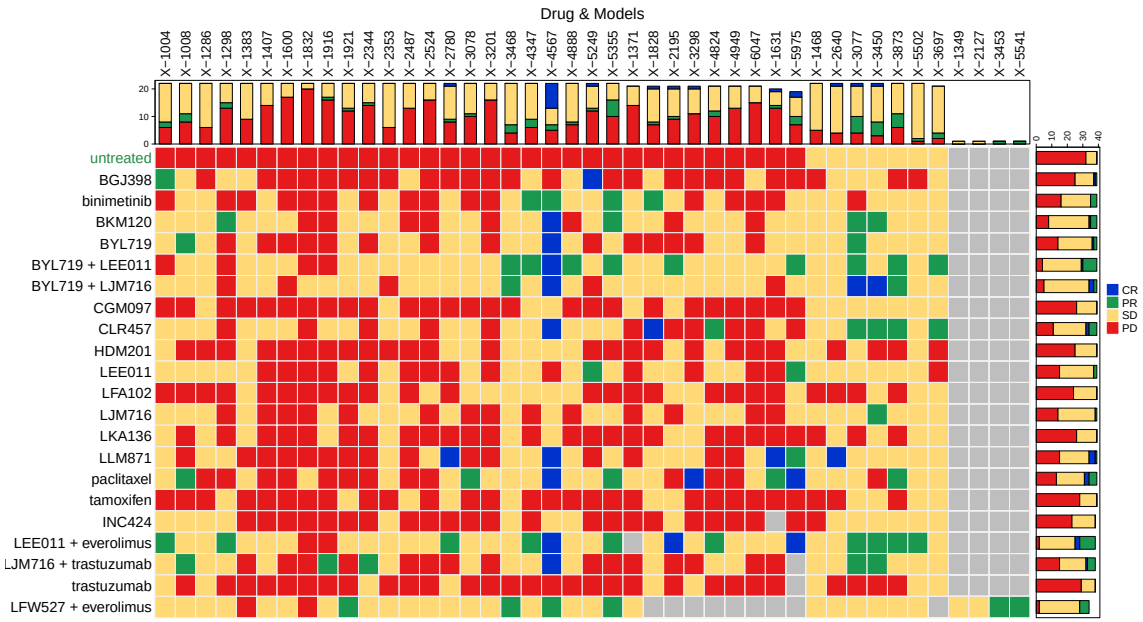

Figure 5: mRECIST plot for PDXE breast cancer data

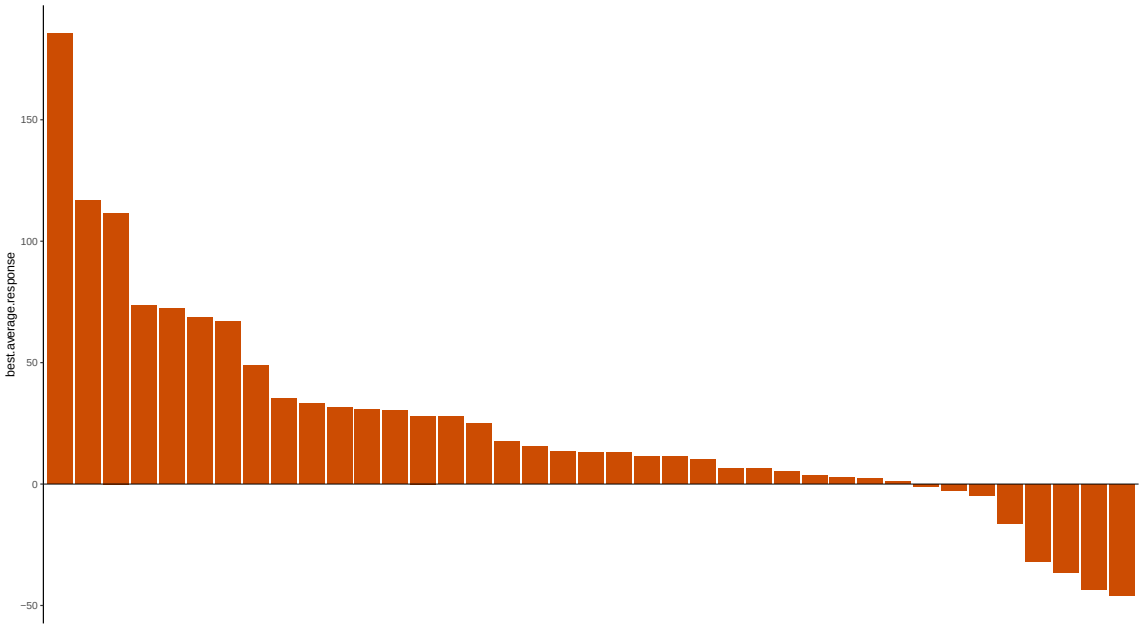

Figure 6: Waterfall plot for binimetinib drug response in PDXs

```
mut <- summarizeMolecularProfiles(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",
```

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

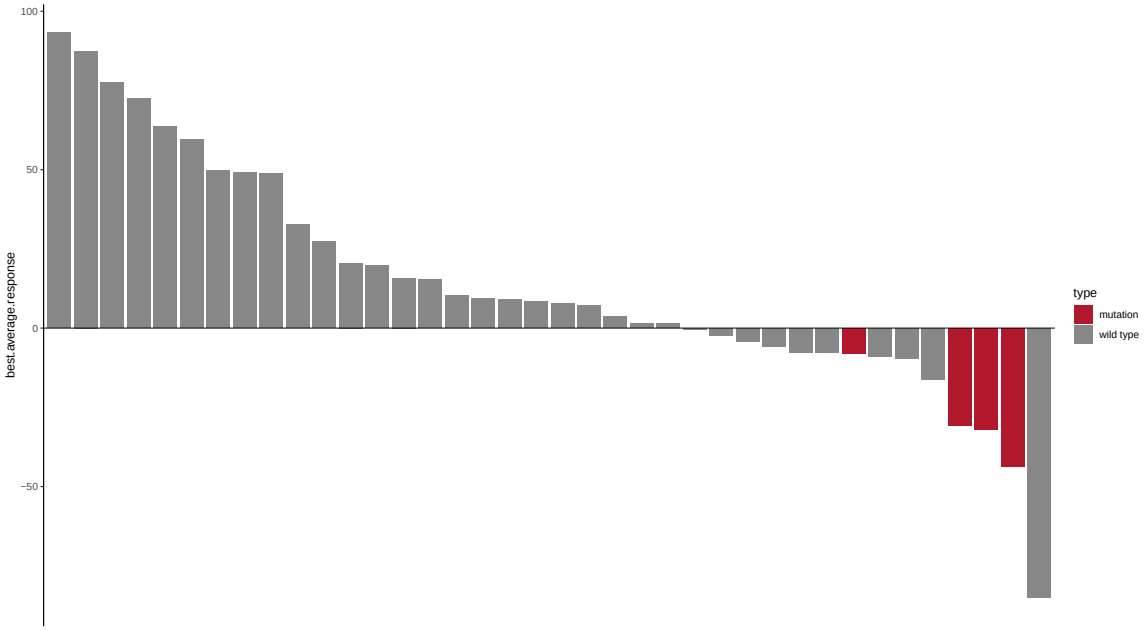

Figure 7: Waterfall plot for BYL719 drug response in PDXs

References

[1] Arvind Singh Mer, Wail Ba-alawi, Petr Smirnov, Yi Xiao Wang, Ben Brew, Janosch Ortmann, Ming-Sound Tsao, David Cescon, Anna Goldenberg, and Benjamin Haibe-Kains. Integrative pharmacogenomics analysis of patient derived xenografts. *bioRxiv*, 2018. URL: https://www.biorxiv.org/content/early/2018/11/16/471227, arXiv:https://www.biorxiv.org/content/early/2018/11/16/471227.full.pdf, doi:10.1101/471227.

[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.

7 SessionInfo

```
sessionInfo()
```

```
## R version 3.4.4 (2018-03-15)
## 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_2.38.0      BiocGenerics_0.24.0 Xeva_1.0.0
## [4] knitr_1.20
##
## loaded via a namespace (and not attached):
##  [1] ComplexHeatmap_1.99.5 Rcpp_1.0.0            highr_0.7
##  [4] bindr_0.1.1           compiler_3.4.4        pillar_1.2.3
##  [7] RColorBrewer_1.1-2    plyr_1.8.4            Rmisc_1.5
## [10] iterators_1.0.10      tools_3.4.4           digest_0.6.18
## [13] checkmate_1.8.5       lattice_0.20-38       evaluate_0.12
## [16] tibble_1.4.2          gtable_0.2.0          pkgconfig_2.0.1
## [19] doSNOW_1.0.16         rlang_0.3.0.1         foreach_1.4.4
## [22] yaml_2.2.0            bindrcpp_0.2.2        dplyr_0.7.6
## [25] stringr_1.3.1         GlobalOptions_0.1.0   tidyselect_0.2.4
## [28] grid_3.4.4            glue_1.3.0            R6_2.2.2
## [31] GetoptLong_0.1.7      snow_0.4-2            rmarkdown_1.10.16
## [34] purrr_0.2.5           ggplot2_3.1.0         magrittr_1.5
## [37] backports_1.1.2       BBmisc_1.11           scales_1.0.0
## [40] codetools_0.2-15      htmltools_0.3.6       assertthat_0.2.0
## [43] BiocStyle_2.6.1       shape_1.4.4           circlize_0.4.6
## [46] colorspace_1.3-2      labeling_0.3          stringi_1.2.3
## [49] lazyeval_0.2.1        munsell_0.5.0         rjson_0.2.20
```