The Xeva User’s Guide

Arvind Mer\textsuperscript{1,2} and Benjamin Haibe-Kains\textsuperscript{1,2,3,4,5}

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

May 1, 2024

Contents

1 Introduction .............................................. 2
2 Installation and Settings ............................... 2
3 Definitions .............................................. 2
4 Data Access ............................................. 3
5 Visualizing PDX Growth Curve .................... 4
6 Replicate-based PDX experiments ................. 7
7 PDX Model Drug Response ......................... 8
8 Gene-drug association ................................. 11
9 Creating new Xeva object ......................... 12
10 SessionInfo ........................................... 16
The Xeva User’s Guide

1 Introduction

The Xeva package provides efficient and powerful functions for patient-driven 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:

```r
if (!requireNamespace("BiocManager", quietly = TRUE))
  install.packages("BiocManager")
BiocManager::install("Xeva", version = "3.8")
```

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

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

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

Load Xeva into your current workspace:

```r
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:

```r
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.

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:

```r
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:

```r
code
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:

```r
code
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:

```r
code
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:

```r
code
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:

```r
code
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**:

```r
code
plotPDX(brca, patient.id="X-3078", drug="paclitaxel",control.name = "untreated")
```
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.
Figure 4: Tumor growth curves for a batch of control and treated PDXs generated using patient ID and drug name.
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:

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

![Tumor growth curves for a batch of control and treated PDXs with replicates](image)

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

```r
plotPDX(repx, batch = "P3", SE.plot = "errorbar")
plotPDX(repx, batch = "P4", vol.normal = TRUE, SE.plot = "ribbon")
```
Xeva can effectively summarize PDX drug response data. Here we summarize the mRECIST values for the models in our dataset:

```r
brca.mr <- summarizeResponse(brca, response.measure = "mRECIST")
brca.mr[1:5, 1:4]
```
Waterfall plots are also commonly used to visualize PDX drug response data. Xeva provides a function to visualize and color waterfall plots:

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

![Figure 8: Waterfall plot for binimetinib drug response in PDXs](image)

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:

```r
mut <- summarizeMolecularProfiles(brca, drug = "BYL719", mDataType="mutation")
model.type <- Biobase::exprs(mut)["CDK13", ]
model.type[grepl("Mut", model.type)] <- "mutation"
model.type[!grepl("Mut", model.type)] <- "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)
```

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 9: Waterfall plot for BYL719 drug response in PDXs

```r
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.

```r
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: log(volume) ~ time * exp.type
##     (Intercept) time exp.type:treatment
## 5.32592390 0.03091321 -0.28718018
## time:exp.type:treatment
## -0.01983870
##
## Random effects:
## Formula: ~1 | model.id
## (Intercept) Residual
## StdDev: 0.3802407 0.1975011
##
## Number of Observations: 194
```
8 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 below, we are computing the association between gene expression (RNASeq) and slope of the PDXs for the drug ‘tamoxifen’ using linear regression (lm).

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

## Running for drug tamoxifen

<table>
<thead>
<tr>
<th>feature</th>
<th>drug</th>
<th>estimate</th>
<th>se</th>
<th>n</th>
<th>tstat</th>
<th>fstat</th>
</tr>
</thead>
<tbody>
<tr>
<td>PIK3R6</td>
<td>tamoxifen</td>
<td>0.12174756</td>
<td>0.1654268</td>
<td>38</td>
<td>0.73596007</td>
<td>0.541637222</td>
</tr>
<tr>
<td>FGFR1</td>
<td>tamoxifen</td>
<td>-0.16019355</td>
<td>0.1645143</td>
<td>38</td>
<td>-0.97373653</td>
<td>0.948162830</td>
</tr>
<tr>
<td>NTRK2</td>
<td>tamoxifen</td>
<td>0.16560706</td>
<td>0.1643653</td>
<td>38</td>
<td>1.00755487</td>
<td>1.015166810</td>
</tr>
<tr>
<td>AKT1</td>
<td>tamoxifen</td>
<td>0.00996873</td>
<td>0.1666584</td>
<td>38</td>
<td>0.05981535</td>
<td>0.003577876</td>
</tr>
<tr>
<td>FGFR4</td>
<td>tamoxifen</td>
<td>0.07953256</td>
<td>0.1661387</td>
<td>38</td>
<td>0.47871178</td>
<td>0.229164971</td>
</tr>
</tbody>
</table>

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:

```r
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 drug tamoxifen

<table>
<thead>
<tr>
<th>feature</th>
<th>drug</th>
<th>estimate</th>
<th>se</th>
<th>n</th>
<th>tstat</th>
<th>fstat</th>
<th>pvalue</th>
<th>df</th>
<th>fdr</th>
</tr>
</thead>
<tbody>
<tr>
<td>PIK3R6</td>
<td>tamoxifen</td>
<td>0.10155551</td>
<td>0.1658050</td>
<td>38</td>
<td>0.6124998</td>
<td>0.3751550</td>
<td>0.5440570</td>
<td>36</td>
<td>0.6800713</td>
</tr>
<tr>
<td>FGFR1</td>
<td>tamoxifen</td>
<td>0.25347267</td>
<td>0.1612238</td>
<td>38</td>
<td>1.5721794</td>
<td>2.4717480</td>
<td>0.1246577</td>
<td>36</td>
<td>0.5882184</td>
</tr>
<tr>
<td>NTRK2</td>
<td>tamoxifen</td>
<td>0.15268488</td>
<td>0.1647125</td>
<td>38</td>
<td>0.9269782</td>
<td>0.8592885</td>
<td>0.3601112</td>
<td>36</td>
<td>0.5882184</td>
</tr>
<tr>
<td>AKT1</td>
<td>tamoxifen</td>
<td>-0.19722241</td>
<td>0.1633931</td>
<td>38</td>
<td>-1.2070422</td>
<td>1.4569510</td>
<td>0.2352874</td>
<td>36</td>
<td>0.5882184</td>
</tr>
<tr>
<td>FGFR4</td>
<td>tamoxifen</td>
<td>-0.06903067</td>
<td>0.1662691</td>
<td>38</td>
<td>-0.4151744</td>
<td>0.1723698</td>
<td>0.6804783</td>
<td>36</td>
<td>0.5882184</td>
</tr>
</tbody>
</table>
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`.

```r
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 drug tamoxifen

<table>
<thead>
<tr>
<th>feature</th>
<th>drug</th>
<th>cor</th>
<th>p.value</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>PIK3R6</td>
<td>0.10155551</td>
<td>0.5440570</td>
</tr>
<tr>
<td>2</td>
<td>FGFR1</td>
<td>0.25347267</td>
<td>0.1246577</td>
</tr>
<tr>
<td>3</td>
<td>NTRK2</td>
<td>0.15268488</td>
<td>0.3601112</td>
</tr>
<tr>
<td>4</td>
<td>AKT1</td>
<td>-0.19722241</td>
<td>0.2352874</td>
</tr>
<tr>
<td>5</td>
<td>FGFR4</td>
<td>-0.06903067</td>
<td>0.6804783</td>
</tr>
</tbody>
</table>

### 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:

  ```r
  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:

  ```r
  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
  ```
experiment A data.frame with time vs. tumor volume information. The required columns are `model.id`, `time`, `volume`, and `drug`. Other available 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:

```r
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:

```r
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:

```r
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
```
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:

```r
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:

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

```r
## Drug columns are
## drug
print(xeva.set)
## XevaSet
## name: example xevaSet
## Creation date: Wed May 1 02:36:14 2024
## Number of models: 6
## Number of drugs: 2
## Molecule dataset: RNASeq
```
References


10 SessionInfo

```r
sessionInfo()
```

```
# R version 4.4.0 beta (2024-04-15 r86425)
# Platform: x86_64-pc-linux-gnu
# Running under: Ubuntu 22.04.4 LTS
#
# Matrix products: default
# BLAS: /home/biocbuild/bbs-3.19-bioc/R/lib/libRblas.so
# LAPACK: /usr/lib/x86_64-linux-gnu/lapack/liblapack.so.3.10.0
#
# locale:
# [1] LC_TYPE=en_US.UTF-8 LC_NUMERIC=C
# [3] LC_TIME=en_US.UTF-8 LC_COLLATE=en_US.UTF-8
# [5] LC_MONETARY=en_US.UTF-8 LC_MESSAGES=en_US.UTF-8
# [7] LC_PAPER=en_US.UTF-8 LC_NAME=C
# [9] LC_ADDRESS=C LC_TELEPHONE=C
#[11] LC_MEASUREMENT=en_US.UTF-8 LC_IDENTIFICATION=C
#
# time zone: America/New_York
# tzcode source: system (glibc)
#
# attached base packages:
# [1] stats graphics grDevices utils datasets methods base
#
# other attached packages:
# [1] Biobase_2.64.0 BiocGenerics_0.50.0 Xeva_1.20.0
#
# loaded via a namespace (and not attached):
# [1] bitops_1.0-7 rlang_1.1.3
# [3] magrittr_2.0.3 shinydashboard_0.7.2
# [5] clue_0.3-65 GetoptLong_1.0.5
# [7] matrixStats_1.3.0 compiler_4.4.0
# [9] reshape2_1.4.4 png_0.1-8
# [11] vctrs_0.6.5 relations_0.6-13
# [13] stringr_1.5.1 pkgconfig_2.0.3
# [15] shape_1.4.6.1 crayon_1.5.2
# [17] fastmap_1.1.1 backports_1.4.1
# [19] XVector_0.44.0 labeling_0.4.3
# [21] caTools_1.18.2 utf8_1.2.4
# [23] promises_1.3.0 rmarkdown_2.26
# [25] UCSCUtils_1.0.0 tinytex_0.50
# [27] coop_0.6.3 xfun_0.43
# [29] CoreGx_2.8.0 MultiAssayExperiment_1.30.0
# [31] zlibbioc_1.50.0 GenomeInfoDb_1.40.0
# [33] jsonlite_1.8.8 highr_0.10
# [35] SnowballC_0.7.1 later_1.3.2
# [37] DelayedArray_0.30.0 BiocParallel_1.38.0
# [39] parallel_4.4.0 sets_1.0-25
# [41] cluster_2.1.6 R6_2.5.1
```
## 
<table>
<thead>
<tr>
<th>Package Name</th>
<th>Version</th>
</tr>
</thead>
<tbody>
<tr>
<td>stringi</td>
<td>1.8.3</td>
</tr>
<tr>
<td>RColorBrewer</td>
<td>1.1-3</td>
</tr>
<tr>
<td>limma</td>
<td>3.60.0</td>
</tr>
<tr>
<td>boot</td>
<td>1.3-30</td>
</tr>
<tr>
<td>GenomicRanges</td>
<td>1.56.0</td>
</tr>
<tr>
<td>iterators</td>
<td>1.0.14</td>
</tr>
<tr>
<td>knitr</td>
<td>1.46</td>
</tr>
<tr>
<td>downloadr</td>
<td>0.4</td>
</tr>
<tr>
<td>IRRanges</td>
<td>2.38.0</td>
</tr>
<tr>
<td>httpuv</td>
<td>1.6.15</td>
</tr>
<tr>
<td>Matrix</td>
<td>1.7-0</td>
</tr>
<tr>
<td>igraph</td>
<td>2.0.3</td>
</tr>
<tr>
<td>tidyselect</td>
<td>1.2.1</td>
</tr>
<tr>
<td>abind</td>
<td>1.4-5</td>
</tr>
<tr>
<td>yaml</td>
<td>0.7.3</td>
</tr>
<tr>
<td>doParallel</td>
<td>1.0.17</td>
</tr>
<tr>
<td>gplots</td>
<td>3.1.3.1</td>
</tr>
<tr>
<td>codetools</td>
<td>0.2-20</td>
</tr>
<tr>
<td>plyr</td>
<td>1.8.9</td>
</tr>
<tr>
<td>lattice</td>
<td>0.22-6</td>
</tr>
<tr>
<td>tibble</td>
<td>3.2.1</td>
</tr>
<tr>
<td>withr</td>
<td>3.0.0</td>
</tr>
<tr>
<td>shiny</td>
<td>1.8.1.1</td>
</tr>
<tr>
<td>BumpyMatrix</td>
<td>1.12.0</td>
</tr>
<tr>
<td>evaluate</td>
<td>0.23</td>
</tr>
<tr>
<td>bench</td>
<td>1.1.3</td>
</tr>
<tr>
<td>circlize</td>
<td>0.4.16</td>
</tr>
<tr>
<td>pillar</td>
<td>1.9.0</td>
</tr>
<tr>
<td>lsa</td>
<td>0.73.3</td>
</tr>
<tr>
<td>BiocManager</td>
<td>1.30.22</td>
</tr>
<tr>
<td>MatrixGenerics</td>
<td>1.16.0</td>
</tr>
<tr>
<td>KernSmooth</td>
<td>2.23-22</td>
</tr>
<tr>
<td>DT</td>
<td>0.33</td>
</tr>
<tr>
<td>checkmate</td>
<td>2.3.1</td>
</tr>
<tr>
<td>foreach</td>
<td>1.5.2</td>
</tr>
<tr>
<td>stats4</td>
<td>4.4.0</td>
</tr>
<tr>
<td>shinyjs</td>
<td>2.1.0</td>
</tr>
<tr>
<td>piano</td>
<td>2.20.0</td>
</tr>
<tr>
<td>generics</td>
<td>0.1.3</td>
</tr>
<tr>
<td>S4Vectors</td>
<td>0.42.0</td>
</tr>
<tr>
<td>ggplot2</td>
<td>3.5.1</td>
</tr>
<tr>
<td>munsell</td>
<td>0.5.1</td>
</tr>
<tr>
<td>scales</td>
<td>1.3.0</td>
</tr>
<tr>
<td>BiocStyle</td>
<td>2.32.0</td>
</tr>
<tr>
<td>gtools</td>
<td>3.9.5</td>
</tr>
<tr>
<td>xtable</td>
<td>1.8-4</td>
</tr>
<tr>
<td>marray</td>
<td>1.82.0</td>
</tr>
<tr>
<td>glue</td>
<td>1.7.0</td>
</tr>
<tr>
<td>slam</td>
<td>0.1-50</td>
</tr>
<tr>
<td>tools</td>
<td>4.4.0</td>
</tr>
<tr>
<td>data.table</td>
<td>1.15.4</td>
</tr>
<tr>
<td>fgsea</td>
<td>1.30.0</td>
</tr>
<tr>
<td>visNetwork</td>
<td>2.1.2</td>
</tr>
<tr>
<td>fastmatch</td>
<td>1.1-4</td>
</tr>
<tr>
<td>cowplot</td>
<td>1.1.3</td>
</tr>
<tr>
<td>grid</td>
<td>4.4.0</td>
</tr>
<tr>
<td>colorspace</td>
<td>2.1-0</td>
</tr>
<tr>
<td>BBmisc</td>
<td>1.13</td>
</tr>
<tr>
<td>nlme</td>
<td>3.1-164</td>
</tr>
<tr>
<td>GenomeInfoDbData</td>
<td>1.2.12</td>
</tr>
<tr>
<td>cli</td>
<td>3.6.2</td>
</tr>
<tr>
<td>Pharmacogx</td>
<td>3.8.0</td>
</tr>
<tr>
<td>fansi</td>
<td>1.0.6</td>
</tr>
<tr>
<td>S4Arrays</td>
<td>1.4.0</td>
</tr>
<tr>
<td>ComplexHeatmap</td>
<td>2.20.0</td>
</tr>
<tr>
<td>dplyr</td>
<td>1.1.4</td>
</tr>
<tr>
<td>gtable</td>
<td>0.3.5</td>
</tr>
<tr>
<td>digest</td>
<td>0.6.35</td>
</tr>
<tr>
<td>SparseArray</td>
<td>1.4.0</td>
</tr>
<tr>
<td>farver</td>
<td>2.1.1</td>
</tr>
<tr>
<td>htmlwidgets</td>
<td>1.6.4</td>
</tr>
<tr>
<td>rjson</td>
<td>0.2.21</td>
</tr>
<tr>
<td>htmltools</td>
<td>0.5.8.1</td>
</tr>
<tr>
<td>lifecycle</td>
<td>1.0.4</td>
</tr>
<tr>
<td>httr</td>
<td>1.4.7</td>
</tr>
<tr>
<td>GlobalOptions</td>
<td>0.1.2</td>
</tr>
<tr>
<td>statmod</td>
<td>1.5.0</td>
</tr>
<tr>
<td>mime</td>
<td>0.12</td>
</tr>
</tbody>
</table>