Xeva Tutorial

Arvind Mer

2017-01-16

Load Xeva

Load Xeva library and data.

```
library(Xeva)
data(lpdx)
head(modelInfo(lpdx))
```

```
##                                  model.id                      donor
## PHLC1106_P5.501.A1.1 PHLC1106_P5.501.A1.1 11101S-213RC-312S(F)-412S-
## PHLC1106_P5.504.A4.1 PHLC1106_P5.504.A4.1 11101S-213RC-312S(F)-412S-
## PHLC1106_P5.506.B1.1 PHLC1106_P5.506.B1.1 11101S-213RC-312S(F)-412S-
## PHLC1106_P5.507.B2.1 PHLC1106_P5.507.B2.1 11101S-213RC-312S(F)-412S-
## PHLC1106_P5.508.B3.1 PHLC1106_P5.508.B3.1 11101S-213RC-312S(F)-412S-
## PHLC1106_P5.511.C1.1 PHLC1106_P5.511.C1.1 11101S-213RC-312S(F)-412S-
##                           dob sex     PHLC biobase.id patient.id
## PHLC1106_P5.501.A1.1 Aug31.14   F PHLC1106   PHLC1106   PHLC1106
## PHLC1106_P5.504.A4.1 Aug31.14   F PHLC1106   PHLC1106   PHLC1106
## PHLC1106_P5.506.B1.1 Aug31.14   F PHLC1106   PHLC1106   PHLC1106
## PHLC1106_P5.507.B2.1 Aug31.14   F PHLC1106   PHLC1106   PHLC1106
## PHLC1106_P5.508.B3.1 Aug31.14   F PHLC1106   PHLC1106   PHLC1106
## PHLC1106_P5.511.C1.1 Sep14.14   F PHLC1106   PHLC1106   PHLC1106
```

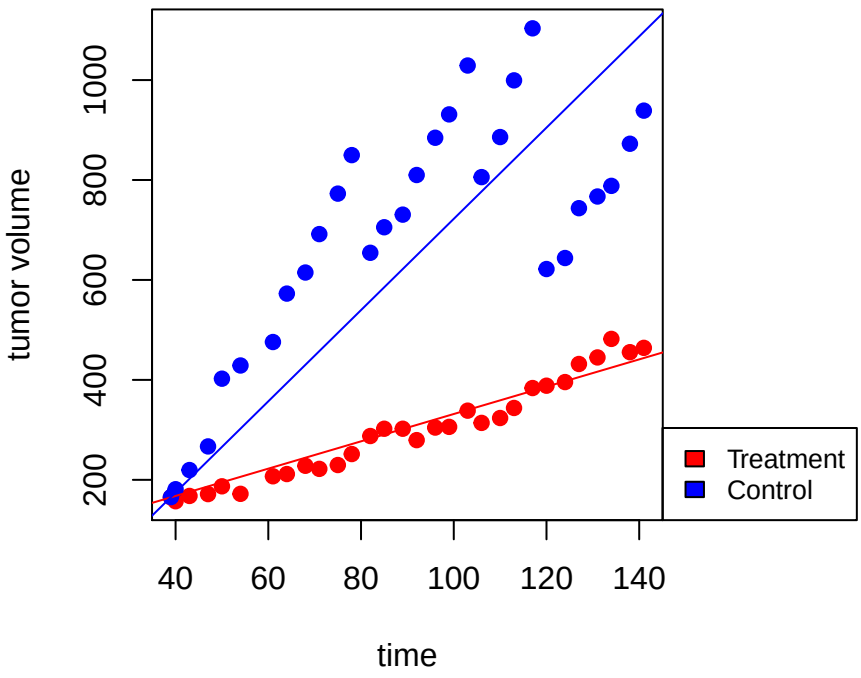
Models which belongs to same batch are in one list which is stored in expDesign slot. For example

```
print(batchNames(lpdx))
```

```
##   PHLC1106_P5    PHLC111_P7    PHLC119_P5    PHLC153_P6    PHLC181_P7
## "PHLC1106_P5"  "PHLC111_P7"  "PHLC119_P5"  "PHLC153_P6"  "PHLC181_P7"
##    PHLC189_P5    PHLC191_P5    PHLC191_P7    PHLC196_P5    PHLC215_P5
##  "PHLC189_P5"  "PHLC191_P5"  "PHLC191_P7"  "PHLC196_P5"  "PHLC215_P5"
##    PHLC229_P6    PHLC235_P4    PHLC655_P7     PHLC82_P5
##  "PHLC229_P6"  "PHLC235_P4"  "PHLC655_P7"   "PHLC82_P5"
```

To calculate angle between the treatment and control samples of this batch

```
batchNames <- batchNames(lpdx)
expDesign  <- expDesign(lpdx, batchNames[1])
ang <- calculateAngle(lpdx, expDesign, treatment.only = TRUE, plot=TRUE)
```




```
print(ang)
```

```
## $PHLC1106_P5
## [1] 13.84247
```

```
#par(mfrow=c(5,3))
#for(I in batchNames)
#{
#  expDesign  <- expDesign(lpdx, I)
#  ang <- calculateAngle(lpdx, expDesign, treatment.only = TRUE, #plot=TRUE)
#  print(ang)
#}
```

Summarize Response of PDXs Get slop of each model and combine summarize all model slop which belongs to same patient by "mean"

```
lpdx_slop <- summarizeResponse(lpdx, response.measure = "slop",
                               group.by="patient.id", summary.stat = "mean")
```

Get angle between treatment and control model ids. For each batch it will give one angle value

```
lpdx_angle <- summarizeResponse(lpdx, response.measure = "angle")
```

Get mutation expression profile

```
ldxe_mut <- getMolecularProfiles(lpdx, data.type="mutation")
print(ldxe_mut)
```

```
## ExpressionSet (storageMode: lockedEnvironment)
## assayData: 2666 features, 11 samples
##   element names: exprs
## protocolData: none
## phenoData
##   sampleNames: PHLC1106 PHLC111 ... PHLC82 (11 total)
##   varLabels: PHLC.ID X.ID
##   varMetadata: labelDescription
## featureData
##   featureNames: RERE SPATA21 ... MTMR1 (2666 total)
##   fvarLabels: probe.Id
##   fvarMetadata: labelDescription
## experimentData: use 'experimentData(object)'
## Annotation: MUT
```

The sample names in expression set are called biobase.id in model slot. Sample names from the expression set canb be be mapped to individual PDX model.ids as

```
# get sample names
library(Biobase)
```

```
## Loading required package: BiocGenerics
```

```
## Loading required package: parallel
```

```
##
## Attaching package: 'BiocGenerics'
```

```
## The following objects are masked from 'package:parallel':
##
##     clusterApply, clusterApplyLB, clusterCall, clusterEvalQ,
##     clusterExport, clusterMap, parApply, parCapply, parLapply,
##     parLapplyLB, parRapply, parSapply, parSapplyLB
```

```
## The following objects are masked from 'package:stats':
##
##     IQR, mad, xtabs
```

```
## The following objects are masked from 'package:base':
##
##     Filter, Find, Map, Position, Reduce, anyDuplicated, append,
##     as.data.frame, cbind, colnames, do.call, duplicated, eval,
##     evalq, get, grep, grepl, intersect, is.unsorted, lapply,
##     lengths, mapply, match, mget, order, paste, pmax, pmax.int,
##     pmin, pmin.int, rank, rbind, rownames, sapply, setdiff, sort,
##     table, tapply, union, unique, unsplit, which, which.max,
##     which.min
```

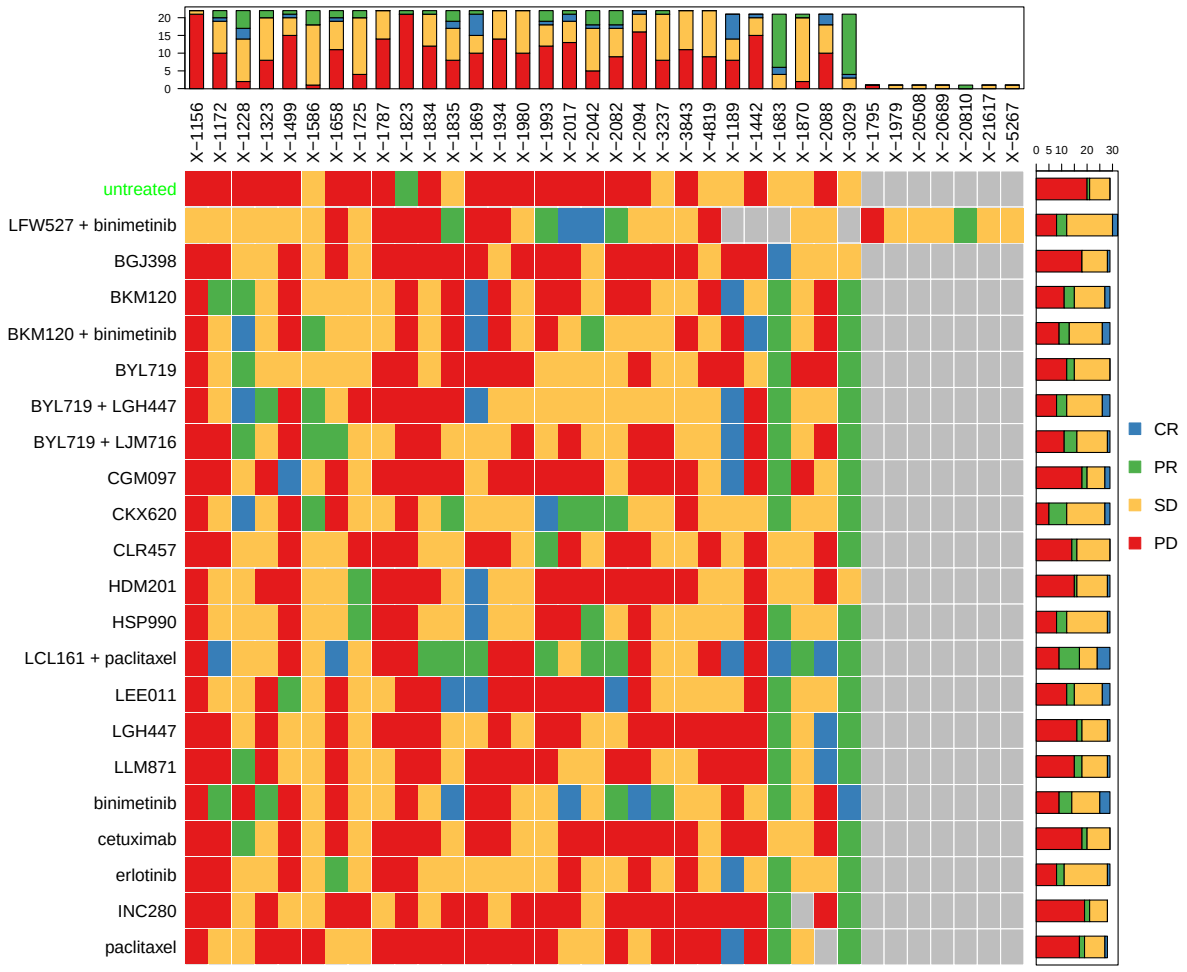
```
## Welcome to Bioconductor
##
##     Vignettes contain introductory material; view with
##     'browseVignettes()'. To cite Bioconductor, see
##     'citation("Biobase")', and for packages 'citation("pkgname")'.
```

```
sn <- Biobase::sampleNames(ldxe_mut)
smap <- mapModelSlotIds(lpdx, id=sn, id.name = "biobase.id", map.to = "model.id")
head(smap)
```

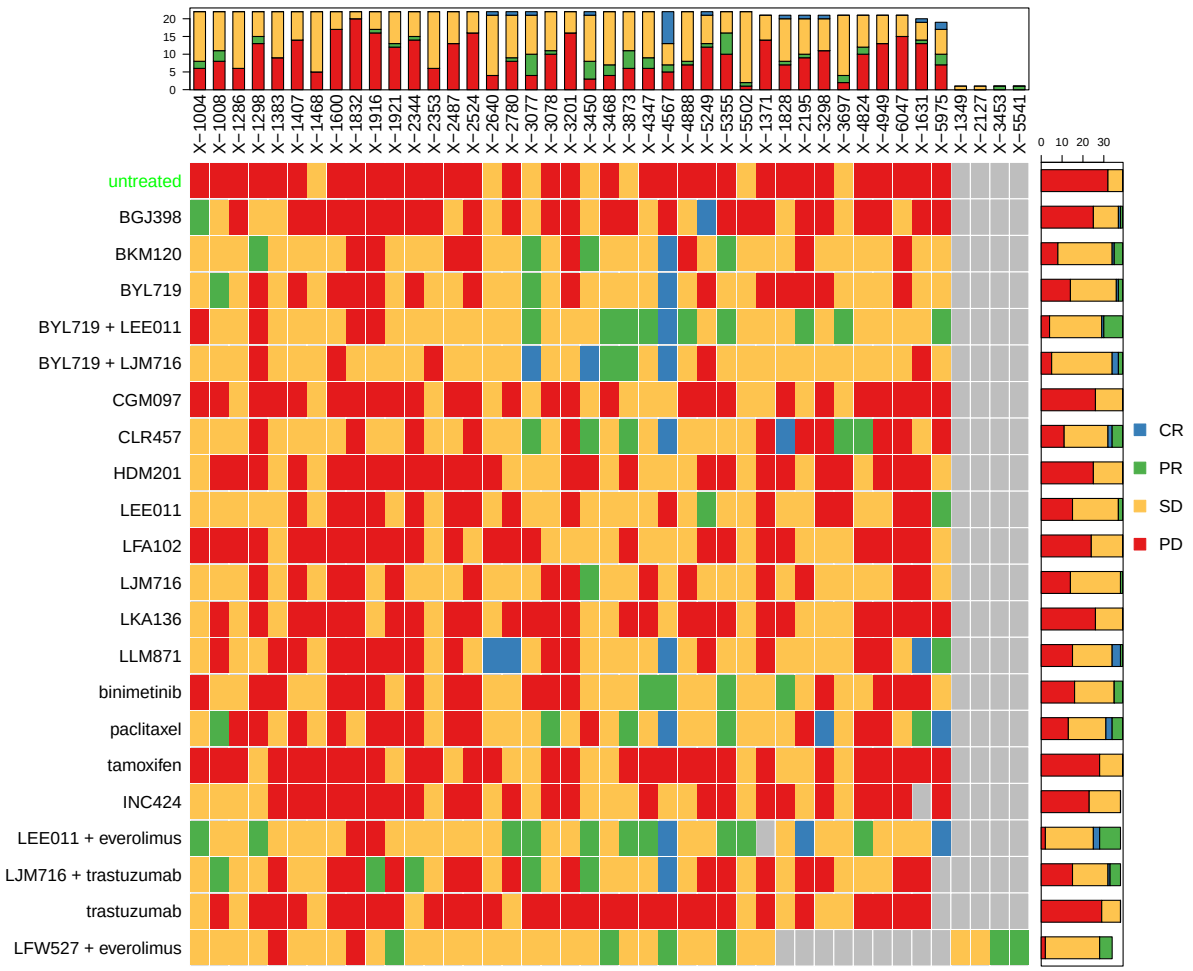
```
##                       biobase.id             model.id
## PHLC1106_P5.501.A1.1   PHLC1106 PHLC1106_P5.501.A1.1
## PHLC1106_P5.504.A4.1   PHLC1106 PHLC1106_P5.504.A4.1
## PHLC1106_P5.506.B1.1   PHLC1106 PHLC1106_P5.506.B1.1
## PHLC1106_P5.507.B2.1   PHLC1106 PHLC1106_P5.507.B2.1
## PHLC1106_P5.508.B3.1   PHLC1106 PHLC1106_P5.508.B3.1
## PHLC1106_P5.511.C1.1   PHLC1106 PHLC1106_P5.511.C1.1
```

Create mRECIST plot for PDXE Lung Cancer data

```
data(pdxe)
df <- getmRECIST(pdxe)
## add tumor.type information
dfMap <- mapModelSlotIds(object=pdxe, id=df$model.id, id.name="model.id",
                         map.to="tumor.type", unique = FALSE)
if(all(df$model.id==dfMap$model.id)) {df$tumor.type = dfMap$tumor.type}
lungDf = df[df$tumor.type=="NSCLC", ]
#pdf(file="DATA-raw/mRECIST_plot_NSCLC.pdf", width=12, height=10)
plotmRECIST(lungDf, groupBy = "biobase.id", control.name = "untreated")
```




```
#pdf(file="DATA-raw/mRECIST_plot_BRCA.pdf", width=12, height=10)
brDF = df[df$tumor.type=="BRCA", ]
plotmRECIST(brDF, groupBy = "biobase.id", control.name = "untreated")
```




```
#dev.off()
```

Creat mR vs slop bar-plot

```
data(pdxe)

pm = modelInfo(pdxe)
lungPID = unique(pm[pm$tumor.type=="NSCLC", "patient.id"])

pdxe_slop <- summarizeResponse(pdxe, response.measure = "slop",
                               group.by="patient.id", summary.stat = "mean")

lung_pdxe_slope <- pdxe_slop[, lungPID]

##-------
pdxe_mR <- summarizeResponse(pdxe, response.measure = "mRECIST_recomputed",
                               group.by="patient.id")

lung_pdxe_mR = pdxe_mR[, lungPID]
```

```
slope=c(); mR=c()
for(dn in rownames(lung_pdxe_slope))
{
  for(pi in colnames(lung_pdxe_slope))
  {
    v = c(lung_pdxe_slope[dn,pi], lung_pdxe_mR[dn,pi])
    if(!is.na(v[1]) & !is.na(v[1]))
    { slope = c(slope,v[1]); mR=c(mR,v[2]) }
  }
}

df = data.frame(mR= mR, slope= as.numeric(slope), stringsAsFactors = FALSE)
df$mR= factor(df$mR, c("CR", "PR", "SD", "PD"))

colPalette = c("#377eb8", "#4daf4a", "#fec44f", "#e41a1c")
boxplot(slope~mR, data=df, col=colPalette,
  main="mRECIST vs slope", xlab="mRECIST", ylab="slope")
```