MOSClip Tutorial with TGCA Ovarian Cancer Data

Contents

Multi Omics Survival Analysis with MOSClip

This tutorial will walk you to perform a compleate analysis with MOSClip.

MOSClip is a new method that combines survival analysis and graphical model theory to test the survival association of pathways or of their connected components in a multi-omic framework. Multi-omic gene measurements are tested as covariates of a Cox proportional hazard model after dimensionality reduction of data. The final goal is to find the biological processes impacting the patient's survival.

MOSClip has a modular structure, allowing the use of one or multiple different omics, as well as different data reduction strategies and tests.

In this tutorial we will focus on the integration of three omics data regarding methylome, transcriptome and genome mutations, testing if these omics can be sinergically involved in pathways with survival prognostication power.

Furthermore, in MOSClip multiple efforts have been dedicated to the implementation of specific graphical tools to browse, manage and provide help in the interpretation of results. In this tutorial will also exploit these tools to visually represent analysis results.

Multi-omics TCGA ovarian cancer data analyses

We used some of the *MOSClip* functions to identify pathways (pathway analysis) and modules (module analysis) describing the pathway associated with survival

First of all load the necessary libraries.

```
library(org.Hs.eg.db)
library(pheatmap)
library(RColorBrewer)
library(EDASeq)
library(graphite)
library(MOSClip)
```

Prepare the multi-omic dataset and pathways

Data retrieving and data analysis will depend on your computational resources and need time.

To speed up this tutorial, we preprocessed the data for you. Details on data preparation are available in Material and Method session of the paper.

Thus, download "TCGA-OV-hg38-preprocessed-externalSurv.RData" available at

github.com/cavei/example-datasets/

The provided dataset includes matrices genes per patients of methylation status, somatic mutations, and transcript expression levels of the TCGA ovarian cancer samples.

Than, move the downloaded file inside a directory called downloadTCGA in your working directory. The TCGA dataset came along with the survival annotation from the table from Liu et al. Cell 2018.

Now we can load the preprocessed data.

```
genome = "hg38"
tumor = "OV"
project = paste0("TCGA-", tumor)
load("downloadTCGA/TCGA-OV-hg38-preprocessed-externalSurv.RData")
```

We create a directory to store the data results.

```
dirname = "MOSresults"
if (!file.exists(dirname)) {
    dir.create(dirname)
}
```

Among the data loaded we can find the object 'fup' (short of 'followup') that represents our survival annotations. We extract the slot for the 'progression free survival' (pfs).

Next, we modifiy all the multi-omics matrices assigning the type of gene identifier. Here, we will work with Entrez Gene ID, indicated with the prefix tag "ENTREZID:" compliant to Bioconductor org.dbi as used in the latest graphite R package version.

```
survAnnotations <- fup$pfs

expression <- rnaSeq
row.names(expression) <- paste0("ENTREZID:", row.names(expression))
row.names(mutation) <- paste0("ENTREZID:", row.names(mutation))
names(metClustValues$eMap) <- paste0("ENTREZID:", row.names(metClustValues$eMap))
```

Now, we are about to intersect different datasets and annotations to select those patient sample having all the 3 omics profiled.

We also filter genes in expression matrix keeping only those genes with at least 100 counts in at least one patients, to avoid data sparsity.

After the selection of patients and genes, we perform expression normalization and log2 of the *counts+1* transformation. This will ensure us to work with expression data aproximately close to a normal distribution, the most suitable distribution for the subsequent MOSClip tests.

```
survAnnot <- na.omit(survAnnotations)
patients <- row.names(survAnnot)
patients <- intersect(patients, colnames(expression))
patients <- intersect(patients, colnames(metClustValues$MET_Cancer_Clustered))
patients <- intersect(patients, colnames(mutation))
```

```
survAnnot <- survAnnot[patients, , drop = F]

if (!file.exists("survAnnot.RData")) {
    file = paste0(dirname, "/survAnnot-", as.character(Sys.Date()), ".RData")
    link = "survAnnot.RData"
    save(survAnnot, file = file)
    file.symlink(file, link)
}

expression <- expression[, patients, drop = F]
keep = apply(expression >= 100, 1, any)
expNorm <- betweenLaneNormalization(expression[keep, , drop = F], which = "upper")
pseudoExpNorm <- log2(expNorm + 1)

methylation <- metClustValues
methylation$MET_Cancer_Clustered <- methylation$MET_Cancer_Clustered[, patients,
    drop = F]

mutation <- mutation[, patients, drop = F]
```

At this point, we need the pathway knowledge-base. We will use the reactome database available at graphite R package. Please note that the identifiers convertion of all the Reactome database could take time so we create a procedure to store the converted pathways. Thus if the file exists, you can load a local version of the data.

```
library(graphite)
if (file.exists("reactome-entrez.RData")) {
    load("reactome-entrez.RData")
} else {
    reactome <- pathways("hsapiens", "reactome")
    reactome <- convertIdentifiers(reactome, "entrez")
    file = paste0(dirname, "/reactome-entrez-", as.character(Sys.Date()), ".RData")
    link = "reactome-entrez.RData"
    save(reactome, file = file)
    file.symlink(file, link)
}
```

Since Reactome database is not flat but it has an hierarchical structure, in which pathways are hierarchically divided, we are going to analyze only a subsets of them cutting the hierarchy on the basis of the pathway size.

For the pathway analysis, we are going to use *reactHuge*, a subset that have all the pathways with more (or equal) than 10 nodes;

For module analysis, we are going to use *reactSmall*, so the pathways that are bigger than 20 nodes but smaller that 100 nodes. This will ensure a sufficient level of specificity of the pathway/modules and will avoid unusefull reaction redundancies.

```
nodesLength <- sapply(reactome, function(g) {
    length(intersect(graphite::nodes(g), row.names(pseudoExpNorm)))
})
reactSmall <- reactome[nodesLength >= 20 & nodesLength <= 100]
reactHuge <- reactome[nodesLength >= 10]
```

Create the Omics object

At this step we are almost ready to run MOSClip analysis, we just need to help MOSClip to organize the data in input. We do that creating a multiOmics object to wrap together expression, methylation and mutation matrices.

Given the nature of the methylation data, we expect to have more than a CpG cluster associated to a gene. For this reason, MOSCLip provide the possibility to include a dictionary to associate the methylation level of multiple CpG clusters to a single gene. Thus, the methylation object is a list composed by the methylation matrix and the dictionary to convert cluster names into genes.

For each omic matrix we need to indicate also the data reduction strategy we want to apply. In this tutorial we chosed to use PCA for expression data, cluster analysis for methylation data and a binary summary (i.e. presence/absence) of mutations. This data transformations are easily applied calling MOSClip functions, thus here we need only to provide the name of the needed function.

```
methylationObj = list(met = methylation$MET_Cancer_Clustered, dict = methylation$eMap)
# creation of the Omics object
multiOmics <- Omics(data = list(expr = pseudoExpNorm, met = methylationObj,
    mut = mutation), methods = c("summarizeWithPca", "summarizeInCluster", "summarizeToBinaryEvents"),
    specificArgs = list(pcaArgs = list(name = "exp", shrink = "FALSE", method = "sparse",
        maxPCs = 3), clusterArgs = list(name = "met"), binaryArgs = list(name = "mut",
        binaryClassMin = 10)))
multiOmics
```

Data “expr” to be process with “summarizeWithPca”. Arguments: name :exp shrink :FALSE method :sparse maxPCs :3 Data “met” to be process with “summarizeInCluster”. Arguments: name :met Data “mut” to be process with “summarizeToBinaryEvents”. Arguments: name :mut binaryClassMin :10

As you can see, in this object we specified the 3 omic data, we chosed 3 methods for data reduction, one for each omic, and three lists of additional parameters (as described in the help of each reduction function).

MOSClip module analysis

We are going to perform the multi-omics survival analysis on pathway modules.

The analysis is quite long, so to speed up feature analyses we have saved a file with all the data. *Note. We perform an a priori filter on those genes that are expressed, but it’s not mandatory.*

```
genesToConsider <- row.names(pseudoExpNorm)

if (file.exists("multiOmicsReactome.RData")) {
    load("multiOmicsReactome.RData")
} else {
    multiOmicsReactome <- lapply(reactSmall, function(g) {
        # print(g@title) # uncomment this to see the progression along pathways
        set.seed(1234)
        fcl = multiOmicsSurvivalModuleTest(multiOmics, g, survAnnot, useThisGenes = genesToConsider)
        fcl
    })
    file = paste0(dirname, "/multiOmicsReactome-", as.character(Sys.Date()),
        ".RData")
    link = "multiOmicsReactome.RData"
    save(multiOmicsReactome, file = file)
    file.symlink(file, link)
}
```

The analysis is done now. *MOSCclip* has plenty of functions to visually explore the results. In the following paragraphs we will show you some examples.

Module summary

Using the function *multiPathwayModuleReport()* We can plot the tabular summary of the top 10 modules selected by overall pvalues.

```
moduleSummary <- multiPathwayModuleReport(multiOmicsReactome)
head(moduleSummary, 10)
```

Table 1: Top 10 modules by overall pvalue

	module	pvalue	expPC1	expPC2	expPC3	met2k2	met3k2	met3k3	mutTRUE
Metabolism of polyamines	3	0	0.04	NA	NA	0	NA	NA	NA
Glycosphingolipid metabolism	22	0	0	NA	NA	0.79	NA	NA	NA
Sphingolipid metabolism	20	0	0	NA	NA	0.79	NA	NA	NA
Signaling by TGF-beta family members	15	0	0.26	0.3	0.12	NA	0.34	0	0.05
Gamma carboxylation, hypusine formation and arylsulfatase activation	3	0	0	NA	NA	0	NA	NA	NA
Interleukin-10 signaling	20	0	0	NA	NA	0.03	NA	NA	NA
Interleukin-12 signaling	10	0	0.15	0	0.03	NA	0.53	0.08	0.98
O-glycosylation of TSR domain-containing proteins	30	0	0.11	NA	NA	0.04	NA	NA	NA
Degradation of the extracellular matrix	8	0	0	0.22	0	NA	0.01	0.6	0.88
Interleukin-12 family signaling	1	0	0.02	0	0.14	NA	0.65	0.52	0.8

Resampling strategy on modules test

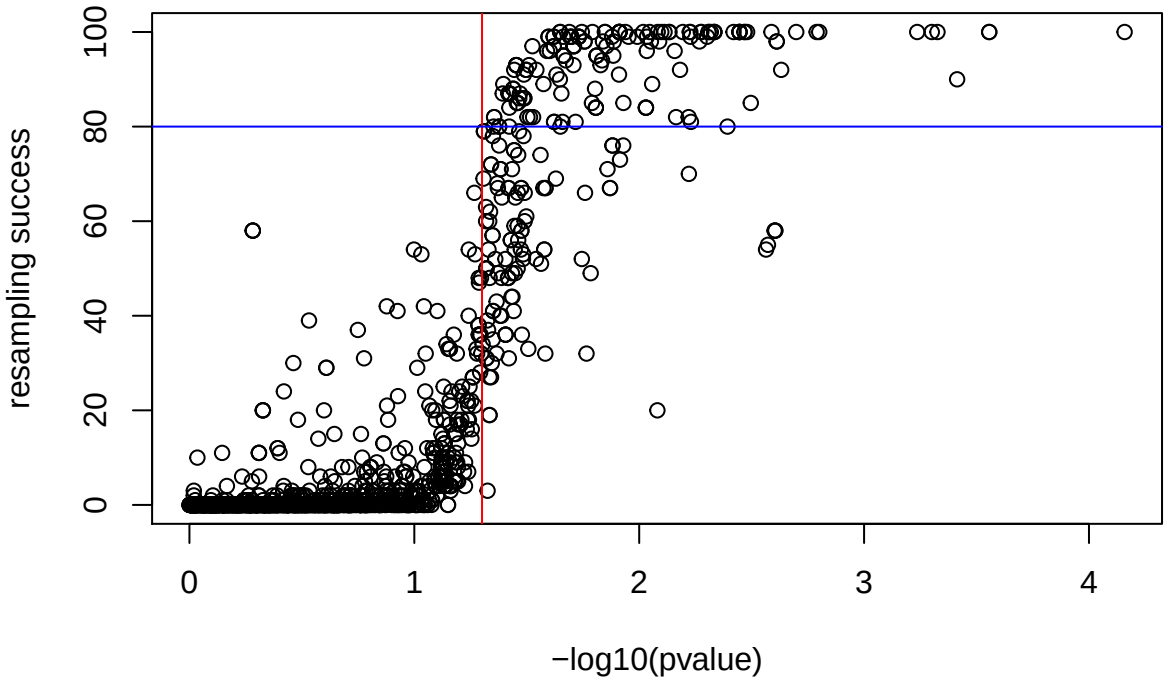
So far, we have identified some modules that are significant. We set up a series of resamplings to prioritize modules, i.e. those that remains significant even if we remove randomly the 1% of the patients. We applied the resampling procedure only to those modules with an overall pvalue $<= 0.05$.

```
useThisPathways <- unique(moduleSummary$pathway[moduleSummary$pvalue <= 0.05])
sModule <- moduleSummary[moduleSummary$pathway %in% useThisPathways, , drop = T]

if (file.exists("perms.RData")) {
    load("perms.RData")
} else {
    perms <- resampling(fullMultiOmics = multiOmics, survAnnot, reactSmall,
        nperm = 100, pathwaySubset = useThisPathways)
    file = paste0(dirname, "/perms-", as.character(Sys.Date()), ".RData")
    link = "perms.RData"
    save(perms, file = file)
    file.symlink(file, link)
}
```

Now resampling analyses has been completed and we can explore some specific modules.

```
# source('final-analysis-functions.R')
stableModulesSummary <- selectStablePathwaysModules(perms = perms, moduleSummary = sModule,
    success = 80)
```

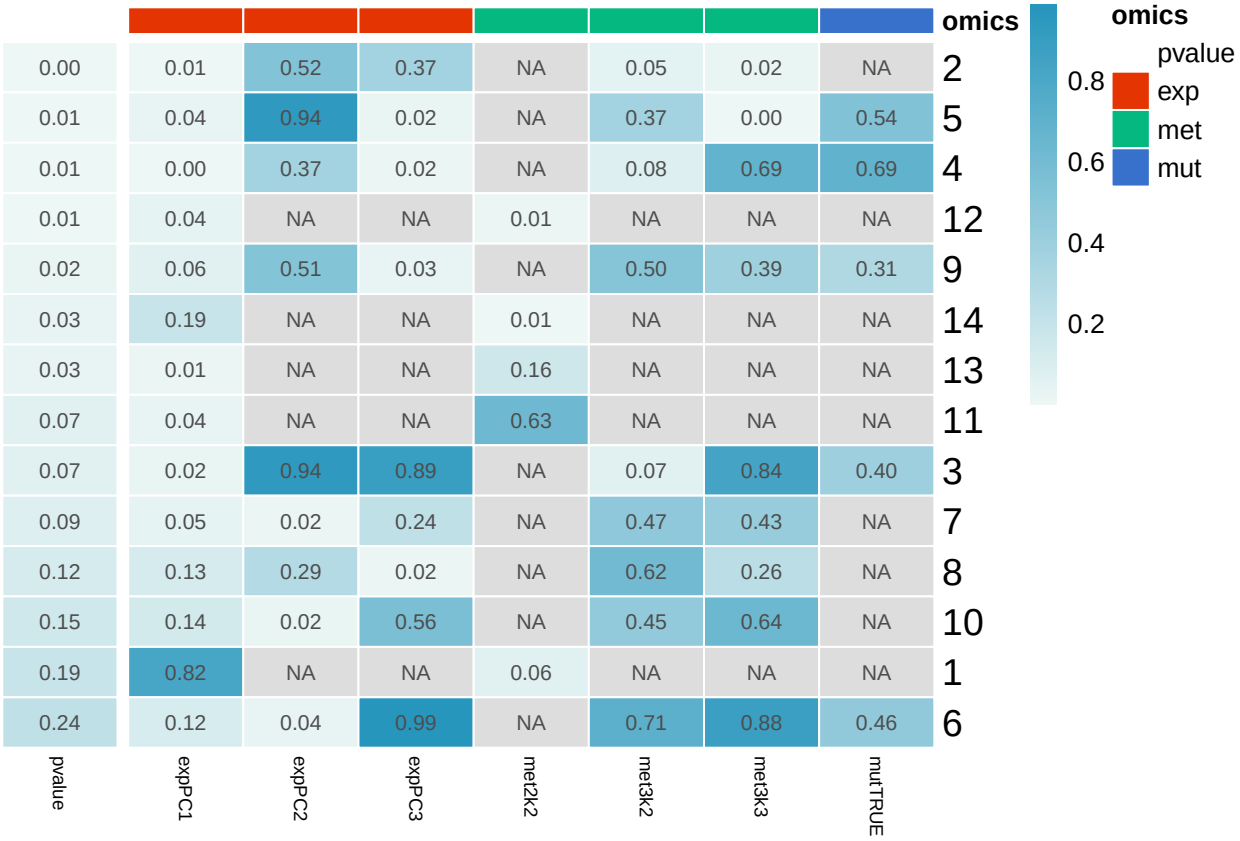



```
resampligSuccessCount <- getPathwaysModulesSuccess(perms = perms, moduleSummary = sModule)
```

Graphical exploration of MOSClip module results

Well, we are interested in the pathway called "Activation of Matrix Metalloproteinases". Let's dig further.

```
plotModuleReport(multiOmicsReactome[["Activation of Matrix Metalloproteinases"]],
    fontsize_row = 14)
```

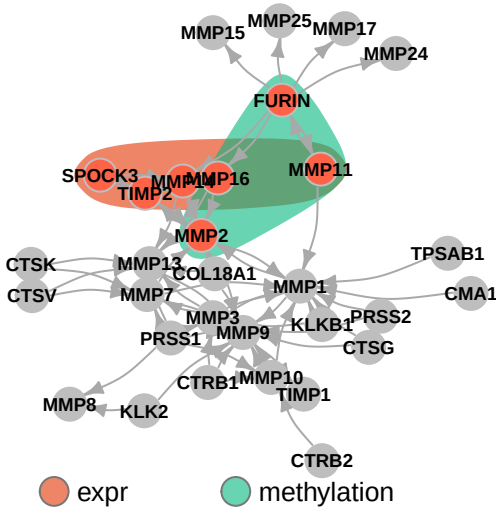


As you can see, among others, the module #2 of this pathway is significant. We can infer the omics that drives this survival behaviour from the pvalues of the model covariates: "PC1" (Expression), "Methy_3k2" and "Methy_3k3" (Methylation).

We can have a look at the pathway graph and the module position in the pathway using *plotModuleInGraph()*

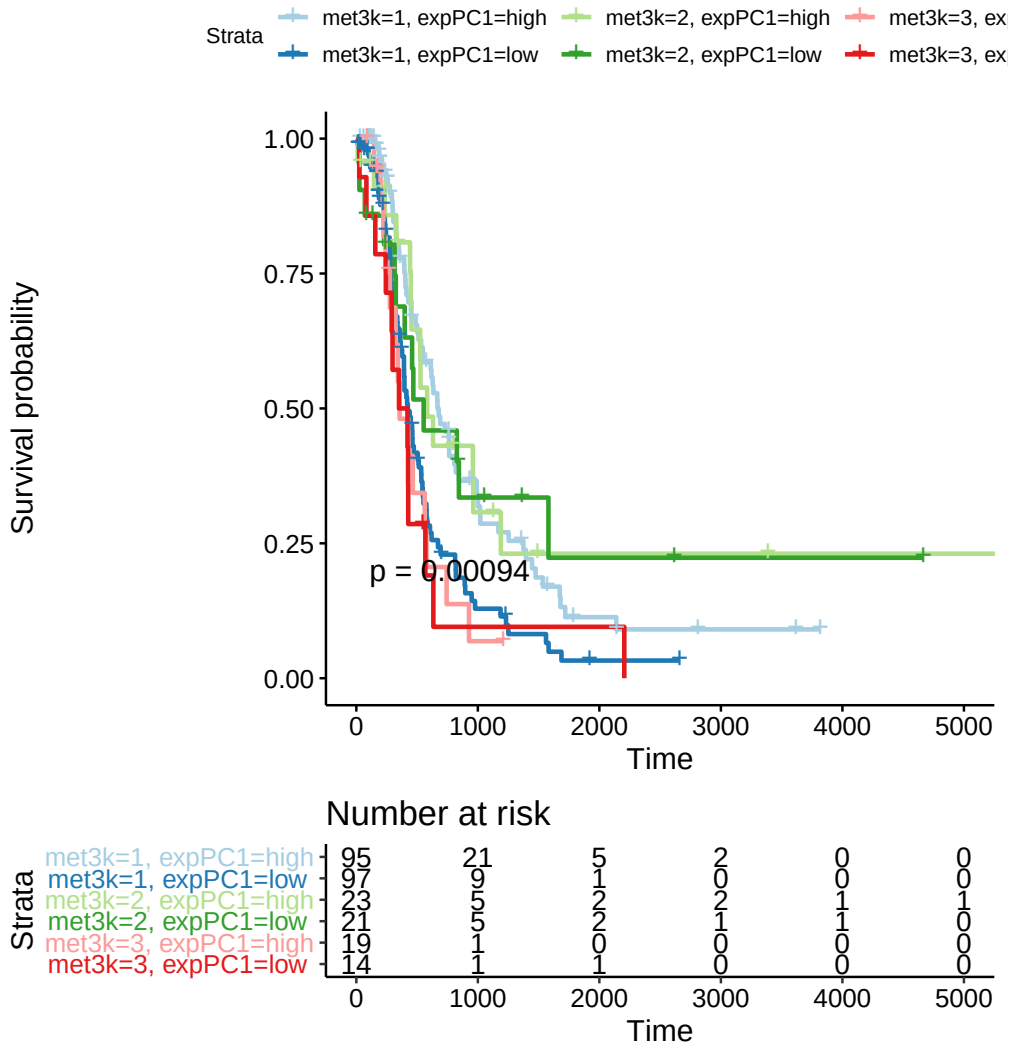
```
library(org.Hs.eg.db)
plotModuleInGraph(multiOmicsReactome[["Activation of Matrix Metalloproteinases"]],
    moduleNumber = 2, legendLabels = c("expr", "methylation"), paletteNames = c(exp = "red",
        met = "green", mut = "blue"))
```

```
## 'select()' returned 1:1 mapping between keys and columns
```



Then we can check the differences in survival using Kaplan Meier curves dividing patients in groups with different omics patterns.

```
plotModuleKM(multiOmicsReactome[["Activation of Matrix Metalloproteinases"]],
    2, formula = "Surv(days, status) ~ met3k + expPC1", paletteNames = "Paired")
```

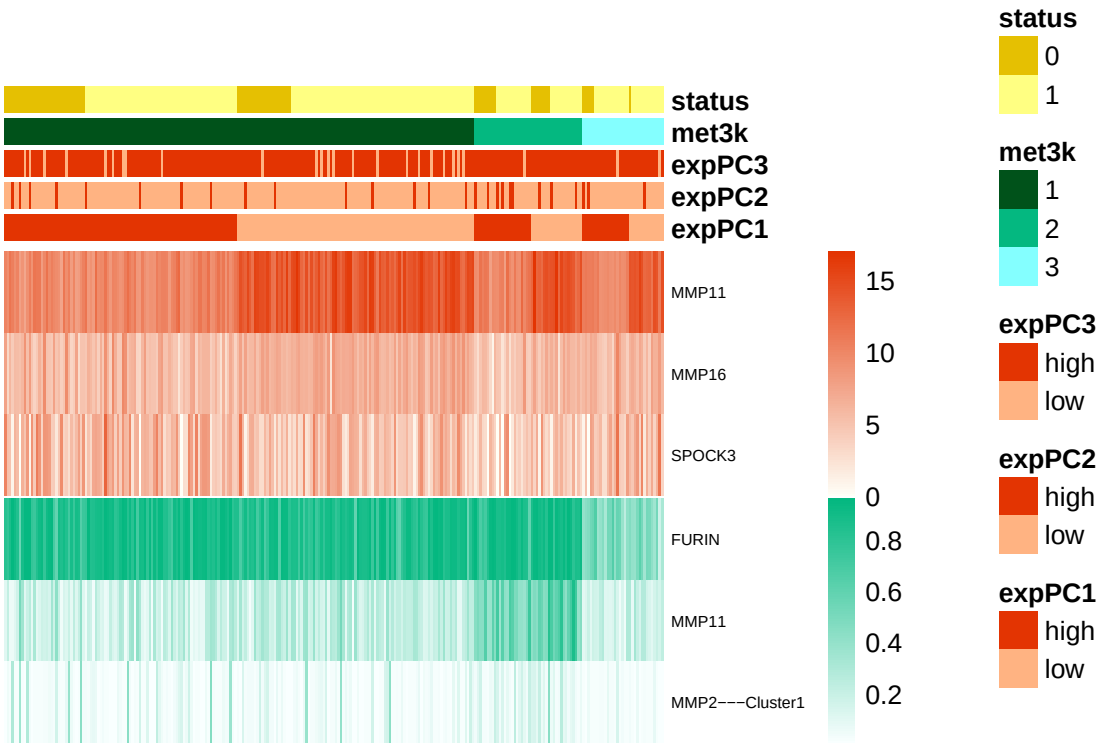


Finally, we can look at the predictive genes using heatmap and patient additional annotations.

```
additionalA <- survAnnot
additionalA$days <- NULL
additionalA$status <- as.factor(additionalA$status)

plotModuleHeat(multiOmicsReactome[["Activation of Matrix Metalloproteinases"]],
    moduleNumber = 2, paletteNames = c(exp = "red", met = "green", mut = "blue"),
    additionalAnnotations = additionalA, additionalPaletteNames = list(status = "yellow",
        letter = "green"), sortBy = c("met3k", "expPC1", "status"), nrowsHeatmaps = 2,
    withSampleNames = F, fontsize_row = 6)
```

```
## 'select()' returned 1:1 mapping between keys and columns
```



MOSClip pathway analysis

In addition to the analysis of the modules, we can perform also a general analysis on pathways. This is useful when the scope of the analysis is to have a general view of the involved processes.

In the module analysis the pathway topology is used to divide the pathway in modules and than can not be considered in the module test because all genes of a module by definition are all connected each others. In pathway test the topology can be and exploited, that's why we suggest to use the topological PCA instead of the sparse PCA previously used.

```
multiOmics@specificArgs$pcaArgs$method = "topological"
multiOmics@specificArgs$pcaArgs$shrink = TRUE
genesToConsider <- row.names(pseudoExpNorm)

if (file.exists("multiOmicsFull.RData")) {
    load("multiOmicsFull.RData")
} else {

    multiOmicsFull <- lapply(reactSmall, function(g) {
        # print(g@title)
        set.seed(1234)
        fcl = multiOmicsSurvivalPathwayTest(multiOmics, g, survAnnot, useThisGenes = genesToConsider)
        fcl
    })
    file = paste0(dirname, "/multiOmicsFull-", as.character(Sys.Date()), ".RData")
    link = "multiOmicsFull.RData"
    save(multiOmicsFull, file = file)
    file.symlink(file, link)
}
```

```
if (file.exists("multiOmicsFullHuge.RData")) {
    load("multiOmicsFullHuge.RData")
} else {

    multiOmicsFullHuge <- lapply(reactHuge, function(g) {
        print(g@title)
        set.seed(1234)
        fcl = multiOmicsSurvivalPathwayTest(multiOmics, g, survAnnot, useThisGenes = genesToConsider)
        fcl
    })
    file = paste0(dirname, "/multiOmicsFullHuge-", as.character(Sys.Date()),
        ".RData")
    link = "multiOmicsFullHuge.RData"
    save(multiOmicsFullHuge, file = file)
    file.symlink(file, link)
}
```

Resampling strategy on pathway test

As for the analysis of modules, MOSCLip gives the possibility to prioritize most important and stable pathway results running a resampling procedure as follow.

```
pathwaySummaryHuge <- multiPathwayReport(multiOmicsFullHuge)
useThisPathwaysFM <- row.names(pathwaySummaryHuge[pathwaySummaryHuge$pvalue <=
    0.05, ])
sPathwayM <- pathwaySummaryHuge[useThisPathwaysFM, , drop = T]

if (file.exists("permsPathwaysHuge.RData")) {
    load("permsPathwaysHuge.RData")
} else {
    permsPathwaysHuge <- MOSClip:::resamplingPathway(fullMultiOmics = multiOmics,
        survAnnot, reactHuge, nperm = 100, pathwaySubset = useThisPathwaysFM)
    file = paste0(dirname, "/permsPathwaysHuge-", as.character(Sys.Date()),
        ".RData")
    link = "permsPathwaysHuge.RData"
    save(permsPathwaysHuge, file = file)
    file.symlink(file, link)
}
```

Graphical exploration of MOSClip pathway results

MOSCclip has multiple functions to explore the pathway data. We provide here some examples.

Pathway summary

We can plot a report of the first 10 pathways, sorted by overall pvalue of the cox proportional hazard model.

```
plotMultiPathwayReport(multiOmicsFull, 10)
```

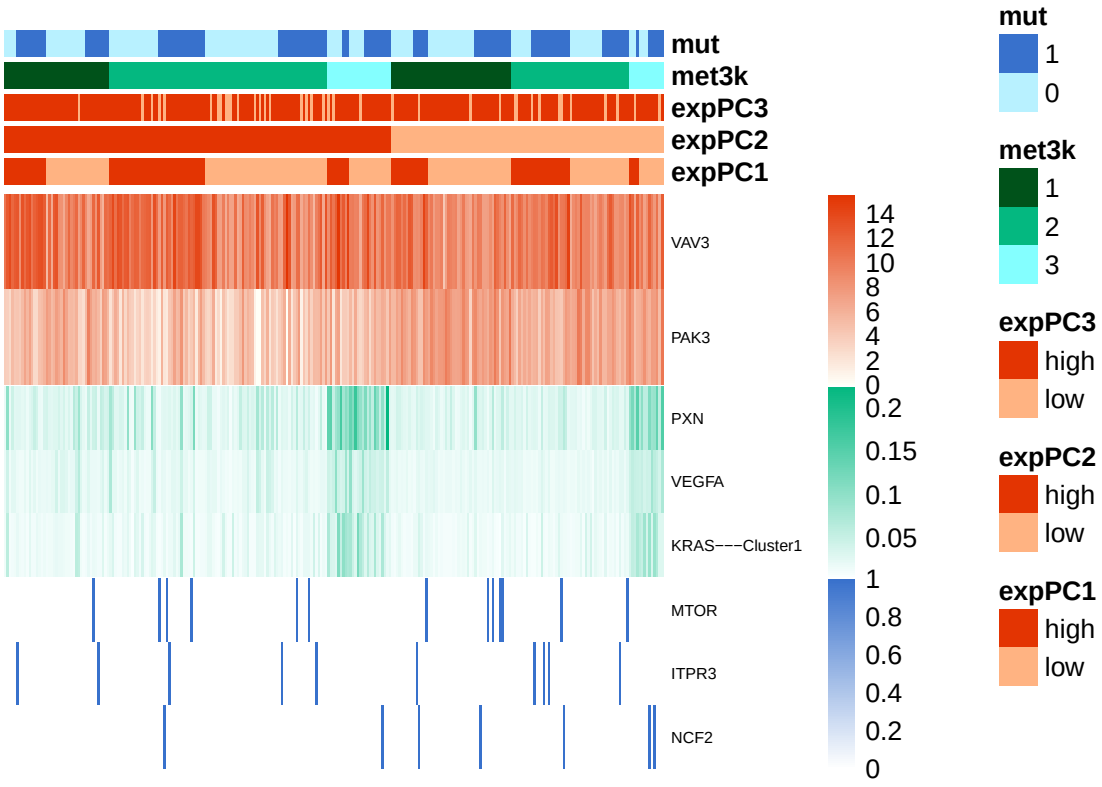
The heatmap summarized the overall pvalues of the Cox proportional hazard model using all the covariates. The first column is the p-value of the model, the others columns contain the pvalues of the single covariates. Colorscale is used to graphically visulize the $p - values$. The $p - value$ heatmap hel to figure out the involvement of different omics while in the survival prediction of the model.

Pathway Heatmap

We can explore also the driver genes of the pathway and their omic pattens using an heatmap plot.

```
plotPathwayHeat(multiOmicsFull[["VEGFA-VEGFR2 Pathway"]], sortBy = c("expPC2",
    "met3k"), paletteNames = c(exp = "red", met = "green", mut = "blue"), nrowsHeatmaps = 2,
    withSampleNames = F, fontsize_row = 6)
```

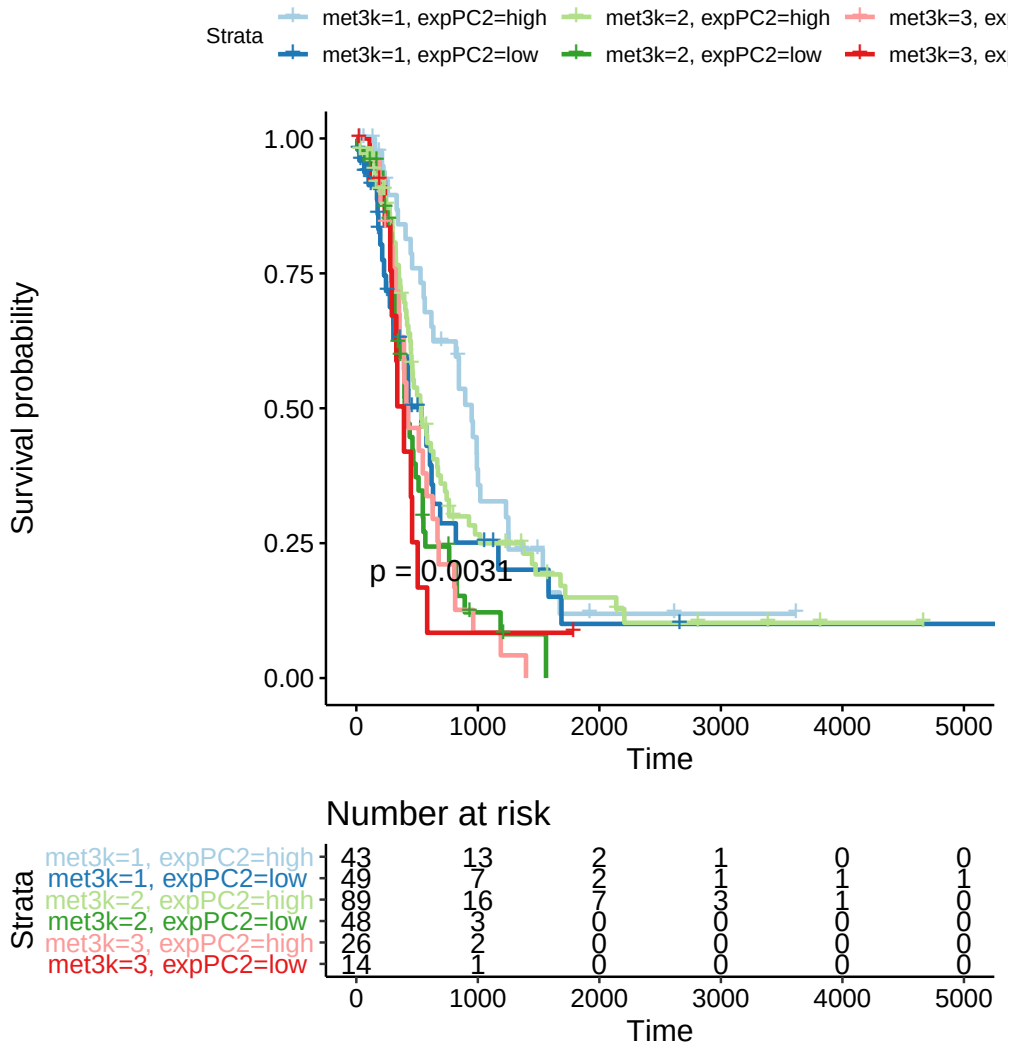
```
## 'select()' returned 1:1 mapping between keys and columns
## 'select()' returned 1:1 mapping between keys and columns
```



Pathway Kaplan Meier

Then we can also check differences in survival using Kaplan Meier curves dividing patients in groups with different omics patterns.

```
plotPathwayKM(multiOmicsFull[["VEGFA-VEGFR2 Pathway"]], formula = "Surv(days, status) ~ met3k + expPC2"
    paletteNames = "Paired")
```


Strata
met3k=1, expPC2=high
met3k=1, expPC2=low
met3k=2, expPC2=high
met3k=2, expPC2=low
met3k=3, ex
met3k=3, ex
Survival probability
1.00
0.75
0.50
0.25
0.00
p = 0.0031
0
1000
2000
3000
4000
5000
Time
Number at risk
Strata
met3k=1, expPC2=high
met3k=1, expPC2=low
met3k=2, expPC2=high
met3k=2, expPC2=low
met3k=3, expPC2=high
met3k=3, expPC2=low
43 13 2 1 0 0
49 7 2 1 1 1
89 16 7 3 1 0
48 3 0 0 0 0
26 2 0 0 0 0
14 1 0 0 0 0
0
1000
2000
3000
4000
5000
Time