Usage deseqAbstraction

Per Ludvik Brattås

09/02/2018

Contents

Tutorial on deseqAbstraction

Easy analysis and visualization of featureCount

1 Setup of workspace

```
# clean environment
rm(list=ls())
gc()
```

```
##          used (Mb) gc trigger (Mb) max used (Mb)
## Ncells 374222 20.0     592000 31.7   460000 24.6
## Vcells 573510  4.4    1308461 10.0   811794  6.2
```

```
# install and load packages

# load deseqAbstraction
library(devtools)
install_github("perllb/deseqAbstraction")
library(deseqAbstraction)
library(DESeq2)
```

2 Create deseqAbs object

2.1 Define path to count file

```
path <- "/home/pbrattaas//Dropbox (MN)/Per/PhD/Projects/DNAmeth/hNES DNMT1 KO/RNAseq/PairedParam/Quant/
```

2.2 Define colData (describe samples)

```
colData <- data.frame(condition=c(rep("CTR",3),rep("KO",3)),
                      samples=c(107,108,109,110,111,112))
```

2.3 Initialize deseqAbs object

```
dnmt <- deseqAbs$new(name="DNMT1KO",filename=path,colData = colData)
```

```
## >>>Creating deseqAbs object
## >>Reading featureCount file
## - ..featureCount file reading done. Access rawdata with $rawfile
## >>Fetching Positional info from file
## - ..complete! Get position of genes with $pos
## >>Getting countData matrix
## - ..complete! Access raw countData with $rawCounts
```

3 Run DESeq pipeline

- You can try to do this in one function $fullAuto().
- Full auto has four main steps:

(1) • create DEseq object (access with $deseq)

(2) • Do diffex analysis (access with $test)

(3) • Do varianceStablizingTransformation (access with $VST)

(4) • Do RPKM normalization (access with $rpkm)

- However, for this automation, it is critical that your deseqAbs object has

(1) • the raw featureCount file in the correct format

(2) • the colData data.frame with at least one column called "condition", which contain the grouping of your samples, that will later be used for diffex anaysis

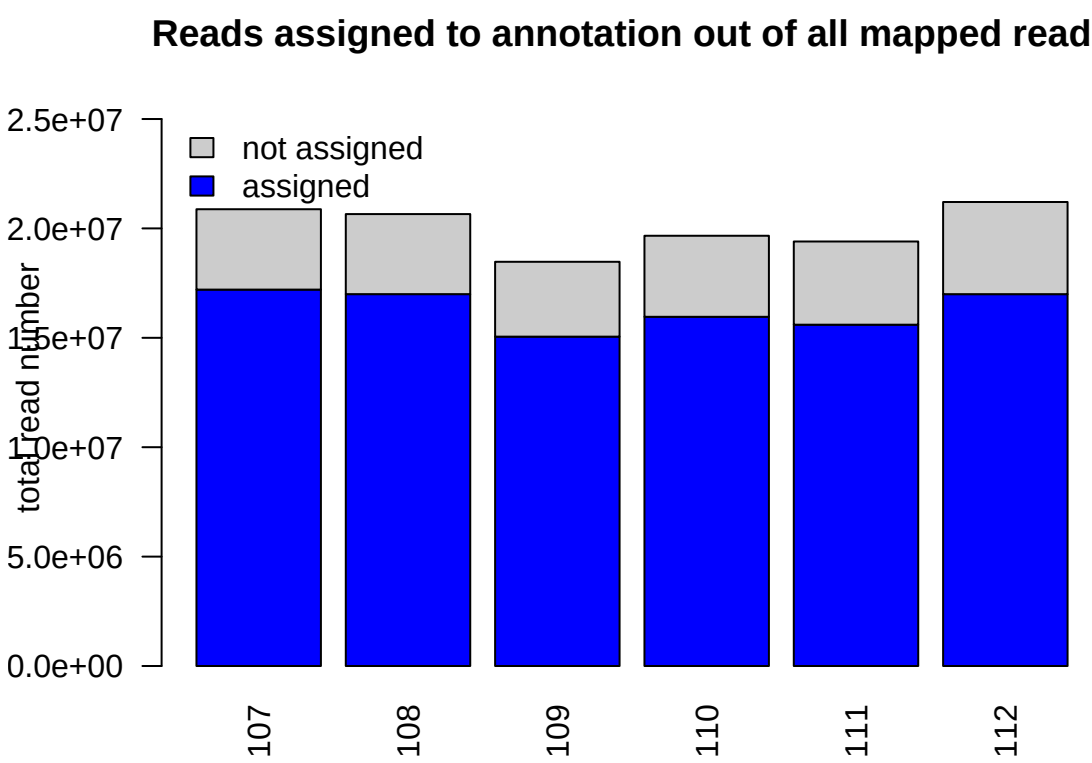
```
# run the automated pipe from creating deseq object to normalization (rpkm, vst) and diffex analysis!
dnmt$fullAuto()
```

```
## >>Running DESeq

## estimating size factors

## estimating dispersions

## gene-wise dispersion estimates

## mean-dispersion relationship

## final dispersion estimates

## fitting model and testing

## - ..complete! Access object with $deseq
## >>Testing for differential expression..
## - ..complete! Diffex completed with default values. Access with $test.
## >>Performing not-blind variance stabilizing transformation
##  - ..complete! not-blind variance stabilizing transformation
## >>Computing FPKM..
## - ..complete! FPKM computed. Access with $FPKM.
## >>Computing mean normalized counts of each condition
## - ..complete! Mean normalized counts computed for each condition. access mean with $baseMean$Mean, ar
## >>Computing mean FPKM of each condition
## - ..complete! Mean normalized expression computed for each condition. access mean with $baseMean$Mean
```

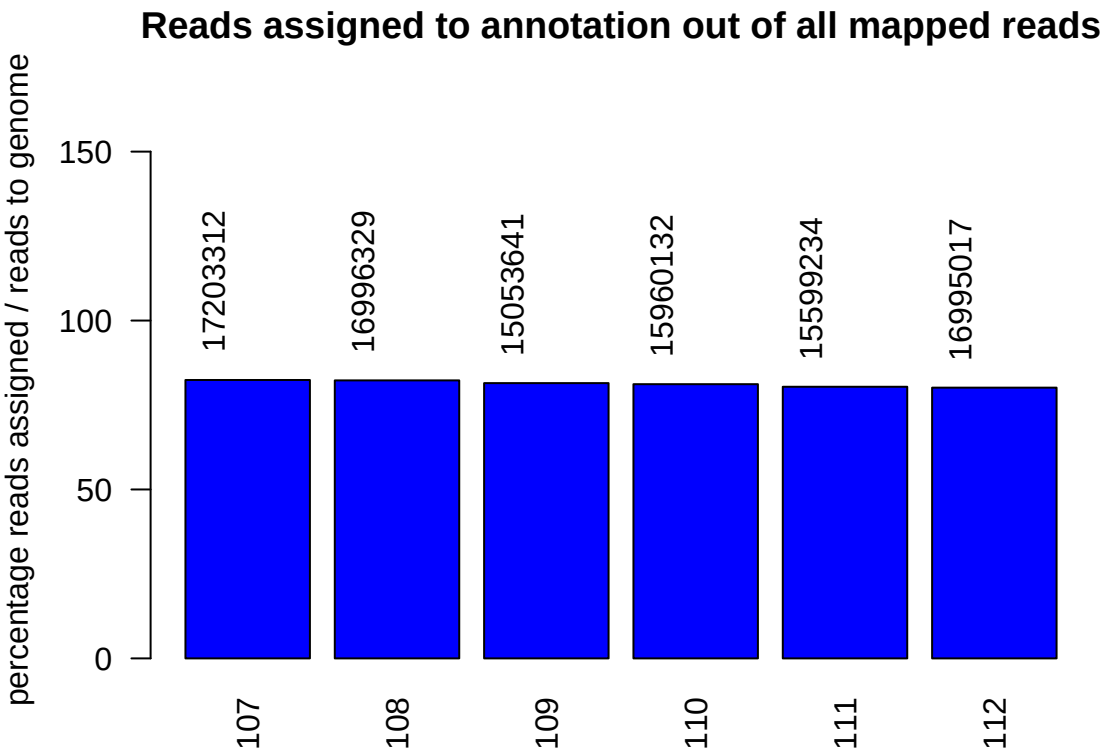
4 Basic QC

```
dnmt$sampleQC()
```

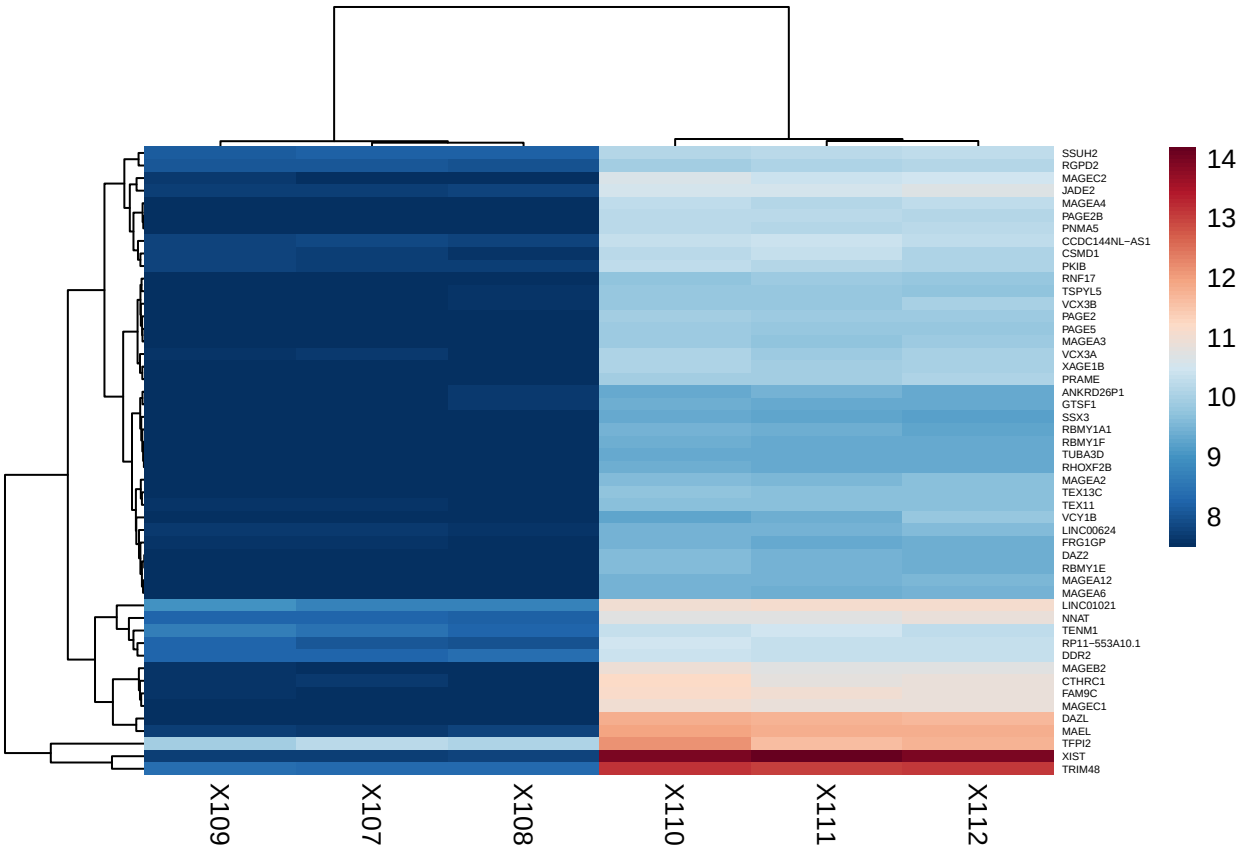
```
## >>Plotting reads assigned to database. Also showing non-mapped reads.
```



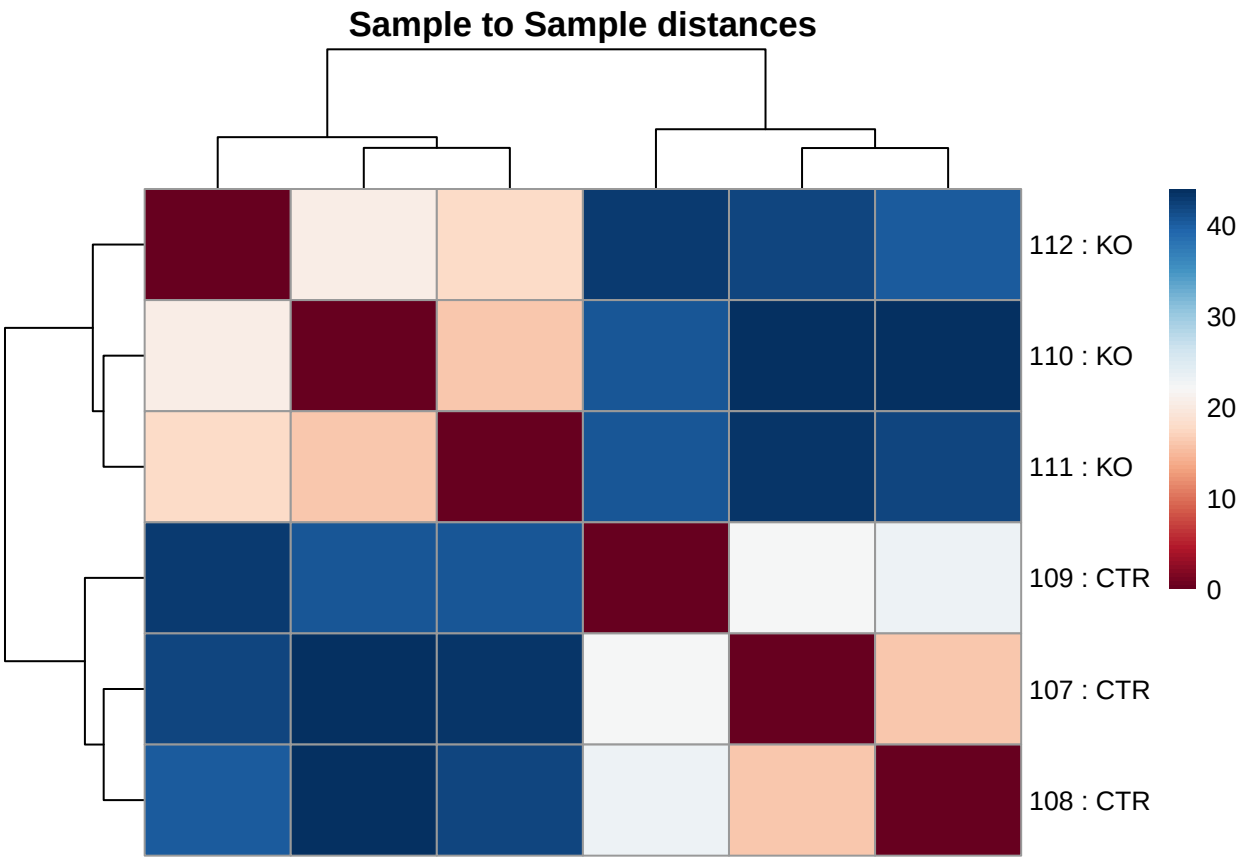

```
## >>Plotting reads assigned to database. Percentage of all reads mapping to genome.
```



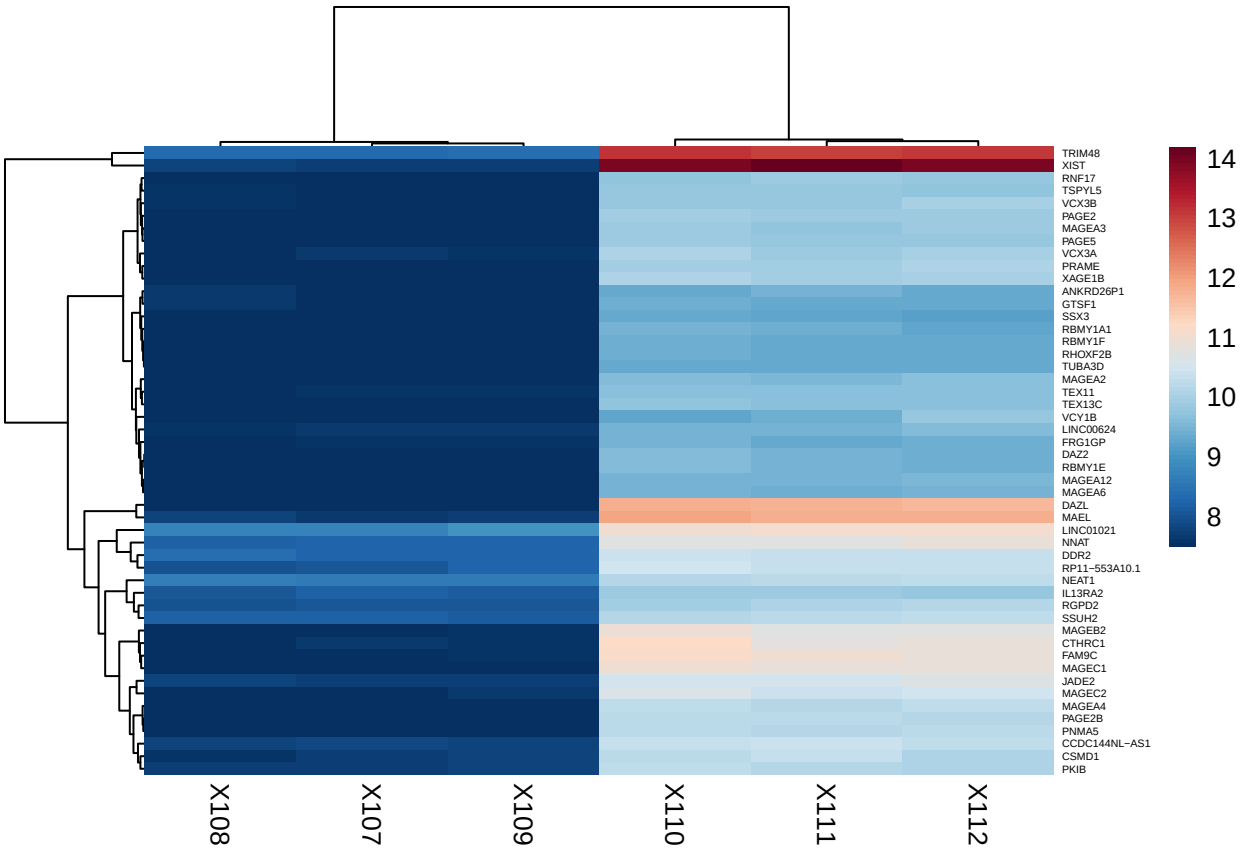

```
## >>Plotting PCA. Top 1000 most variable genes used.
## >>Plotting most variable genes.
```




```
## >>Plotting sample to sample distances.
```




```
## >>Plotting significantly changed genes.. These changes are defined by the default DESeq results() tes
```



5 Custom diffex analysis

```
## By default, DESeq2 will make diffex test between two conditions in your colData. To see which conditi
dnmt$test$Default
```

```
## log2 fold change (MAP): condition KO vs CTR
## Wald test p-value: condition KO vs CTR
## DataFrame with 56269 rows and 6 columns
##              baseMean log2FoldChange      lfcSE       stat       pvalue
##             <numeric>      <numeric>  <numeric>  <numeric>    <numeric>
## DDX11L1      9.884394     0.85141191  0.2731409  3.1171161  0.001826296
## WASH7P     431.808068     0.22924590  0.1516520  1.5116573  0.130621065
## MIR6859-1    4.141800     0.07743318  0.2400011  0.3226367  0.746970365
## MIR1302-2    2.155782     0.19684331  0.1865345  1.0552649  0.291304196
## FAM138A      0.000000             NA         NA         NA           NA
## ...               ...            ...        ...        ...          ...
## MT-ND6    1971.24608    -0.39109016 0.09798218 -3.9914419 6.567279e-05
## MT-TE       29.95648    -0.21971567 0.26448161 -0.8307408 4.061201e-01
## MT-CYB    56881.45149     0.06124375 0.07748031  0.7904427 4.292693e-01
## MT-TT       23.67375     0.23278486 0.27374576  0.8503688 3.951201e-01
## MT-TP       41.04436    -0.11222097 0.25748357 -0.4358374 6.629547e-01
##                  padj
##             <numeric>
```

```
## DDX11L1     0.01050314
## WASH7P      0.30490175
## MIR6859-1   0.87668326
## MIR1302-2   0.52104648
## FAM138A             NA
## ...                ...
## MT-ND6    0.0005863539
## MT-TE     0.6361369103
## MT-CYB    0.6569843090
## MT-TT     0.6266109357
## MT-TP     0.8264068348
```

```
## here you can see the two conditions, in this case: KO vs CTR
```

If you want to make a custom diffex test

```
## If you have more conditions, do the following to test between two other conditions
# get the names of the conditions
conds <- gsub(pattern = "condition",replacement = "",resultsNames(dnmt$deseq))
conds
```

```
## [1] "Intercept" "CTR"       "KO"
```

```
# list condition names with index
cbind(conds,1:length(conds))
```

```
##      conds
## [1,] "Intercept" "1"
## [2,] "CTR"       "2"
## [3,] "KO"        "3"
```

```
# Test KO vs. CTR
c1 <- conds[3] # since KO is the 3rd entry of the conds vector
c2 <- conds[2] # since CTR is the 2nd entry of the conds vector
# make diffex analysis , and give it a name (without space is the best) that makes it easy to access la
dnmt$makeDiffex(name = "KOvsWT",c1 = c1,c2 = c2)
```

```
## >>Testing for differential expression..
## -- Testing KO vs. CTR..
## - ..Complete! Test completed for KO vs. CTR..
```

```
## access the test data with
dnmt$test$KOvsWT
```

```
## log2 fold change (MAP): condition KO vs CTR
## Wald test p-value: condition KO vs CTR
## DataFrame with 56269 rows and 6 columns
##              baseMean log2FoldChange      lfcSE       stat       pvalue
##             <numeric>      <numeric>  <numeric>  <numeric>    <numeric>
## DDX11L1      9.884394     0.85141191  0.2731409  3.1171161  0.001826296
## WASH7P     431.808068     0.22924590  0.1516520  1.5116573  0.130621065
```

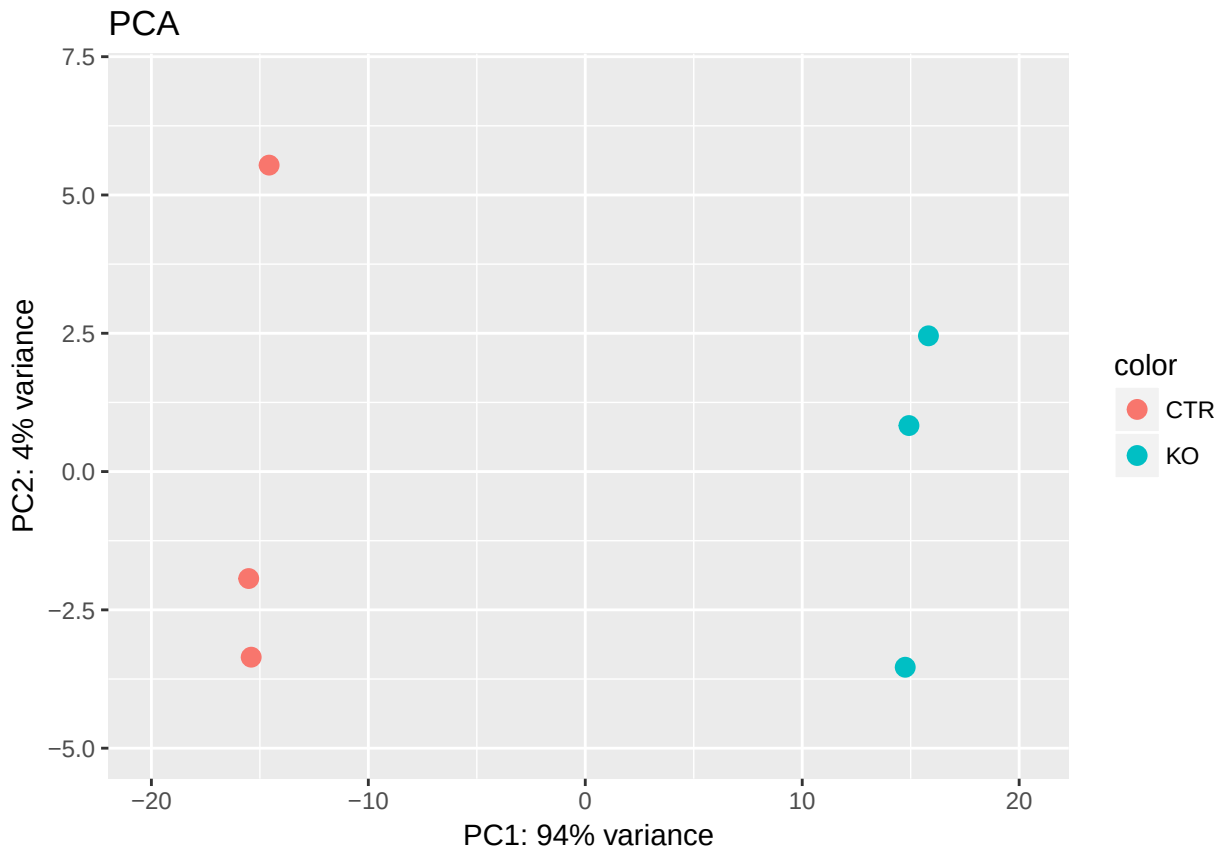
```
## MIR6859-1    4.141800     0.07743318  0.2400011  0.3226367  0.746970365
## MIR1302-2    2.155782     0.19684331  0.1865345  1.0552649  0.291304196
## FAM138A      0.000000             NA         NA         NA           NA
## ...               ...            ...        ...        ...          ...
## MT-ND6     1971.24608    -0.39109016 0.09798218 -3.9914419 6.567279e-05
## MT-TE        29.95648    -0.21971567 0.26448161 -0.8307408 4.061201e-01
## MT-CYB    56881.45149     0.06124375 0.07748031  0.7904427 4.292693e-01
## MT-TT        23.67375     0.23278486 0.27374576  0.8503688 3.951201e-01
## MT-TP        41.04436    -0.11222097 0.25748357 -0.4358374 6.629547e-01
##                   padj
##              <numeric>
## DDX11L1     0.01050314
## WASH7P      0.30490175
## MIR6859-1   0.87668326
## MIR1302-2   0.52104648
## FAM138A             NA
## ...                ...
## MT-ND6    0.0005863539
## MT-TE     0.6361369103
## MT-CYB    0.6569843090
## MT-TT     0.6266109357
## MT-TP     0.8264068348
```

```
## dnmt$test is a list of diffex objects, which can be retrieved one by one
## type dnmt$test to get a list of the diffex objects created
```

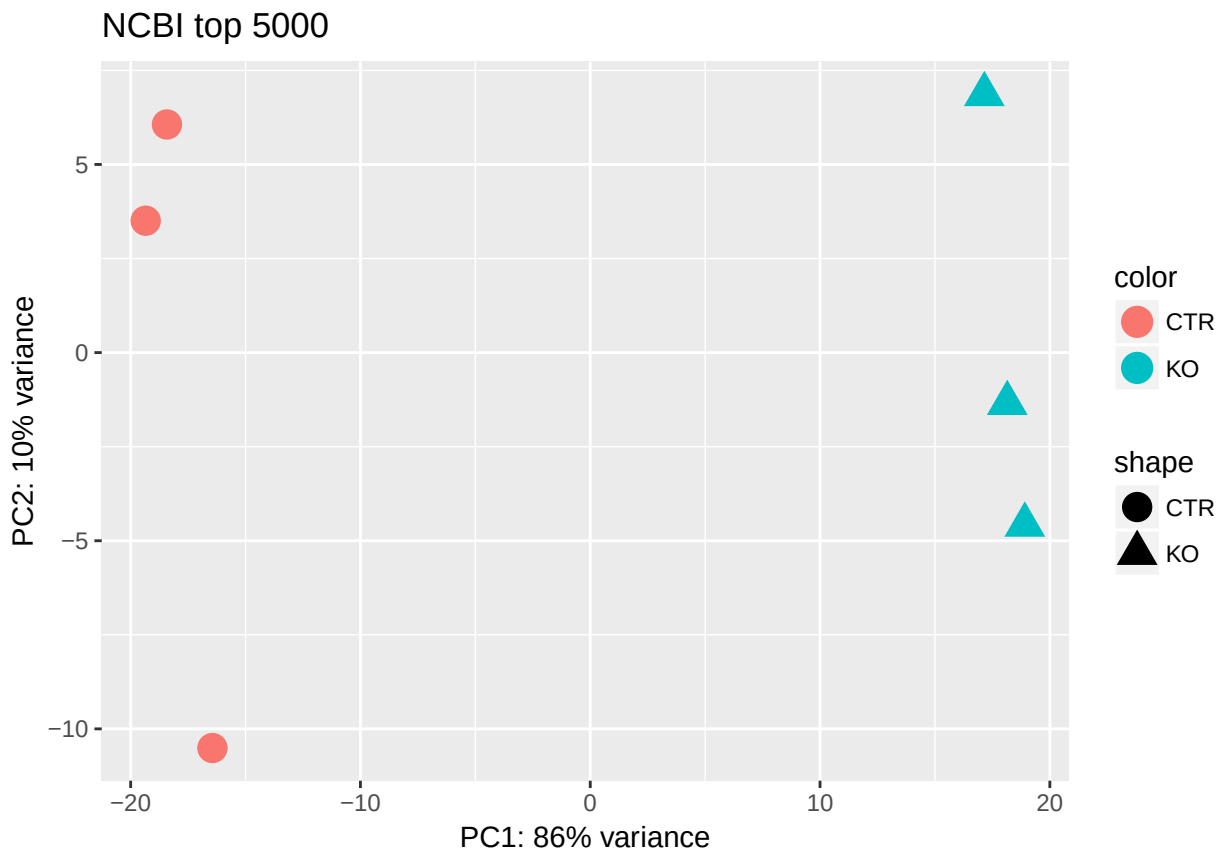
6 Visualize data

6.1 PCA

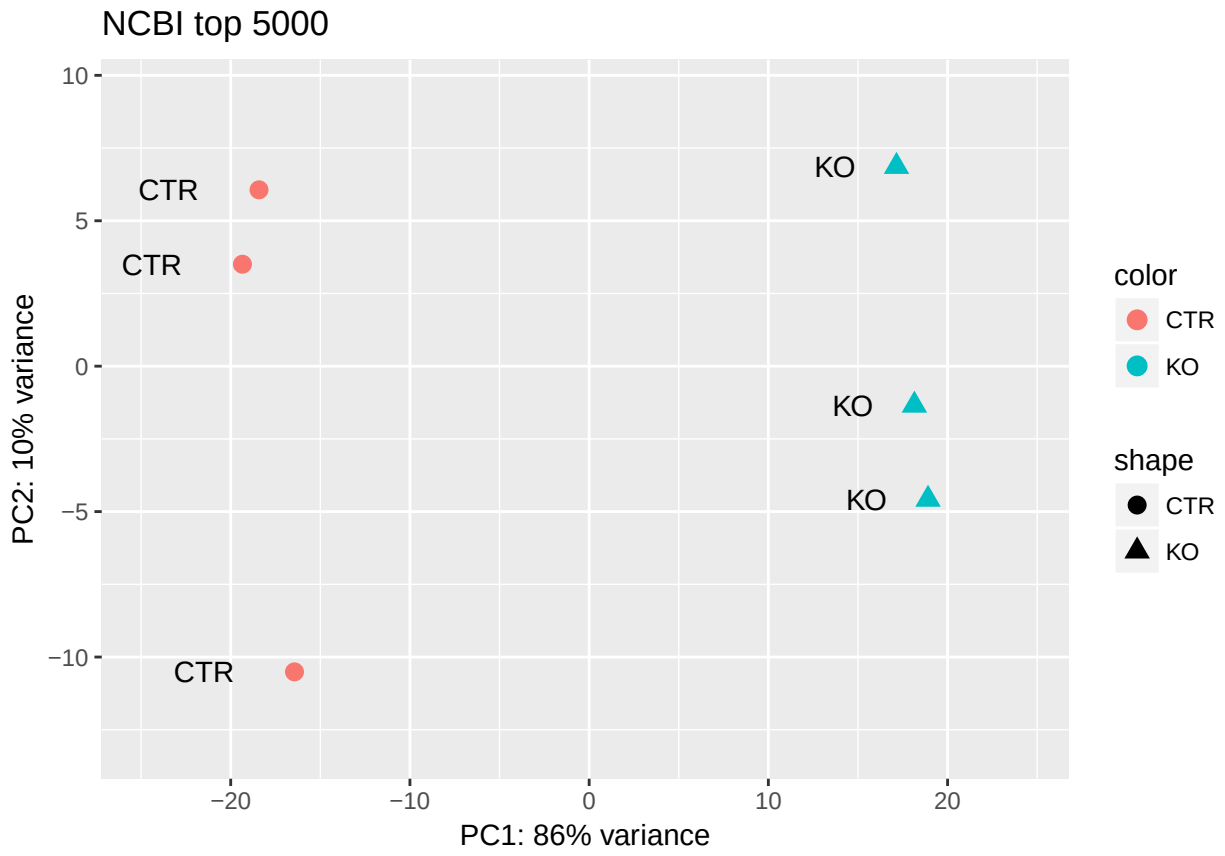
```
# Default PCA plot
dnmt$pca()
```




```
# plot PCA without labels on points
PCAplotter(dat = dnmt$VST,title = "NCBI top 5000",ntop = 5000,color = dnmt$colData$condition,shape=dnmt$
```

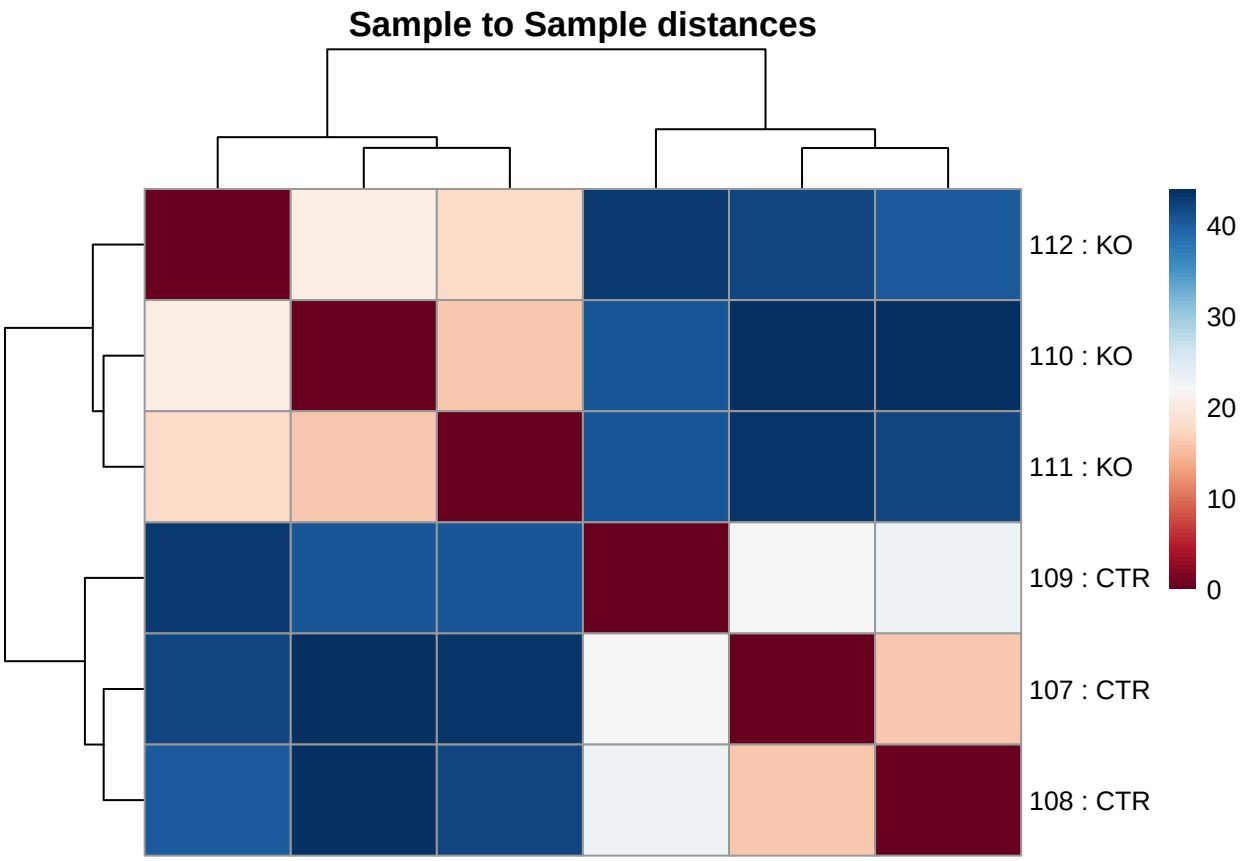



```
# plot PCA with labels
PCAplotter(dat = dnmt$VST,title = "NCBI top 5000",ntop = 5000,color = dnmt$colData$condition,shape=dnmt$
```



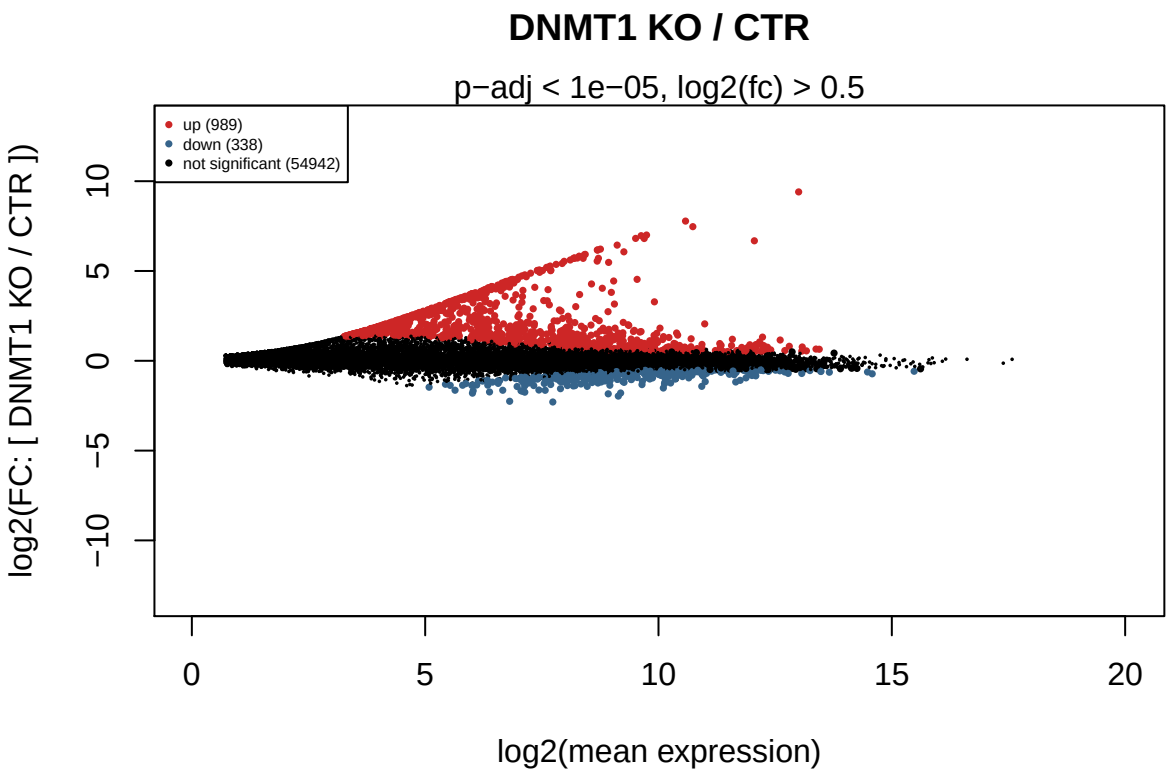
6.2 sample-to-sample distance

```
dnmt$sampleToSample()
```



6.3 maPlot

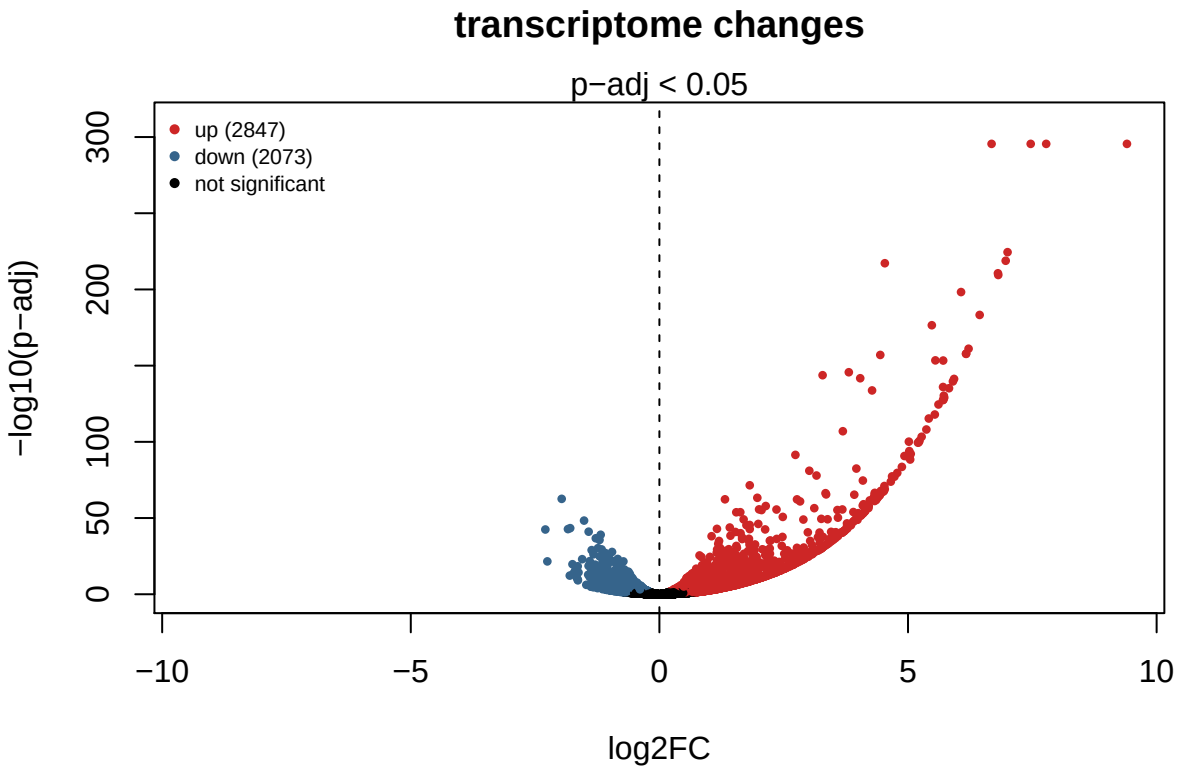
```
# make maPlot: color genes if they have p-adj and log2fc below/above given values (p and l)
maPlot(test = dnmt$test$KOvsWT,"DNMT1 KO","CTR",l = .5,p = 1e-5)

# set id = T to label points
maPlot(test = dnmt$test$KOvsWT,"DNMT1 KO","CTR",l = .5,p = 1e-5,id=T)
```



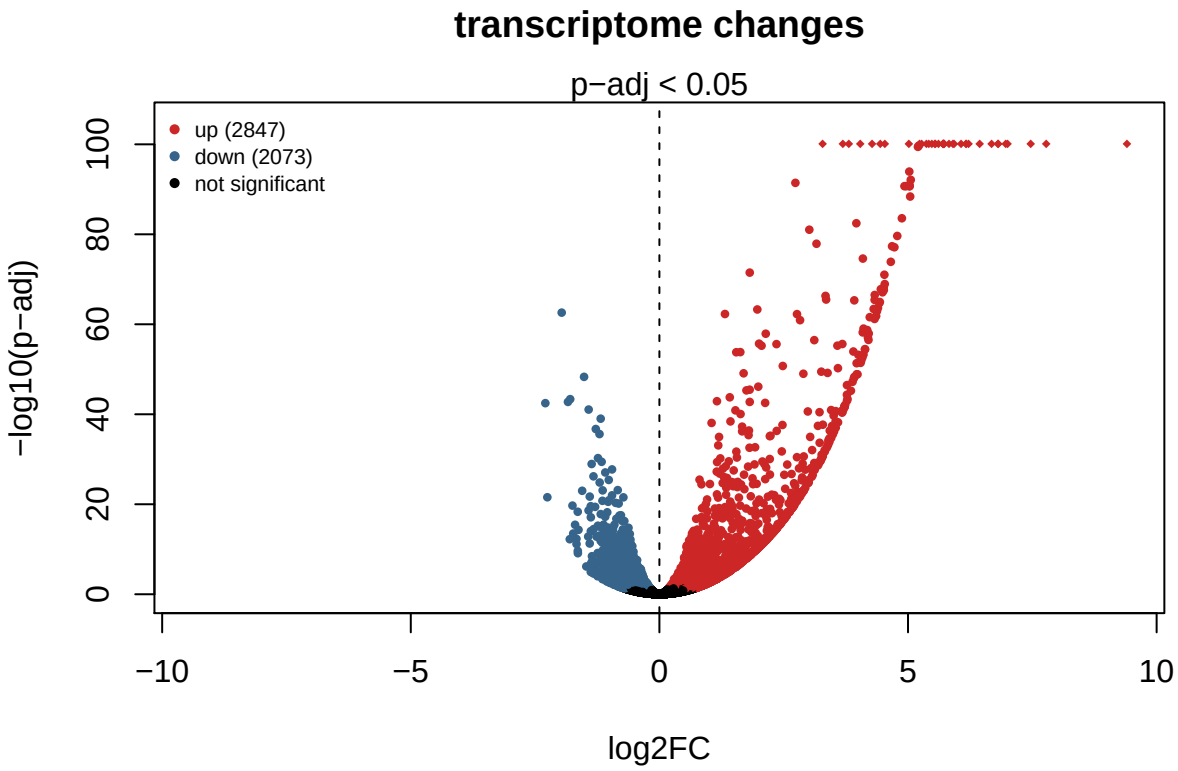

```
# now click on the points you want, and press ESC when done
```

6.4 Volcano plot

```
# generate volcanoplot.
volcanoPlot(test = dnmt$test$KOvsWT)
```




```
# cut y-axis
volcanoPlot(test = dnmt$test$KOvsWT,max = 100)
# set id = T to label points
volcanoPlot(test = dnmt$test$KOvsWT,max = 100,id = T)
```



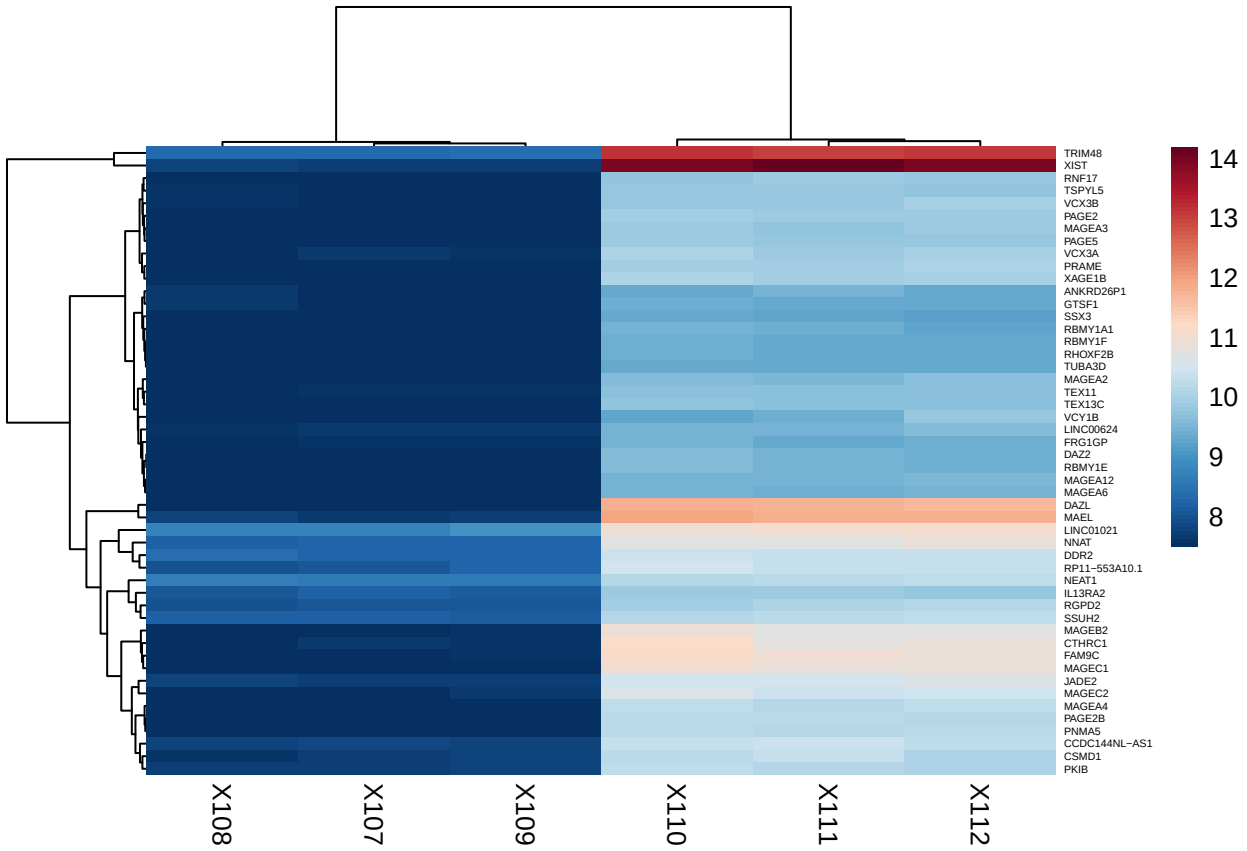

```
## integer(0)
```

```
# now click on the points you want, and press ESC when done
```

6.5 Heatmaps

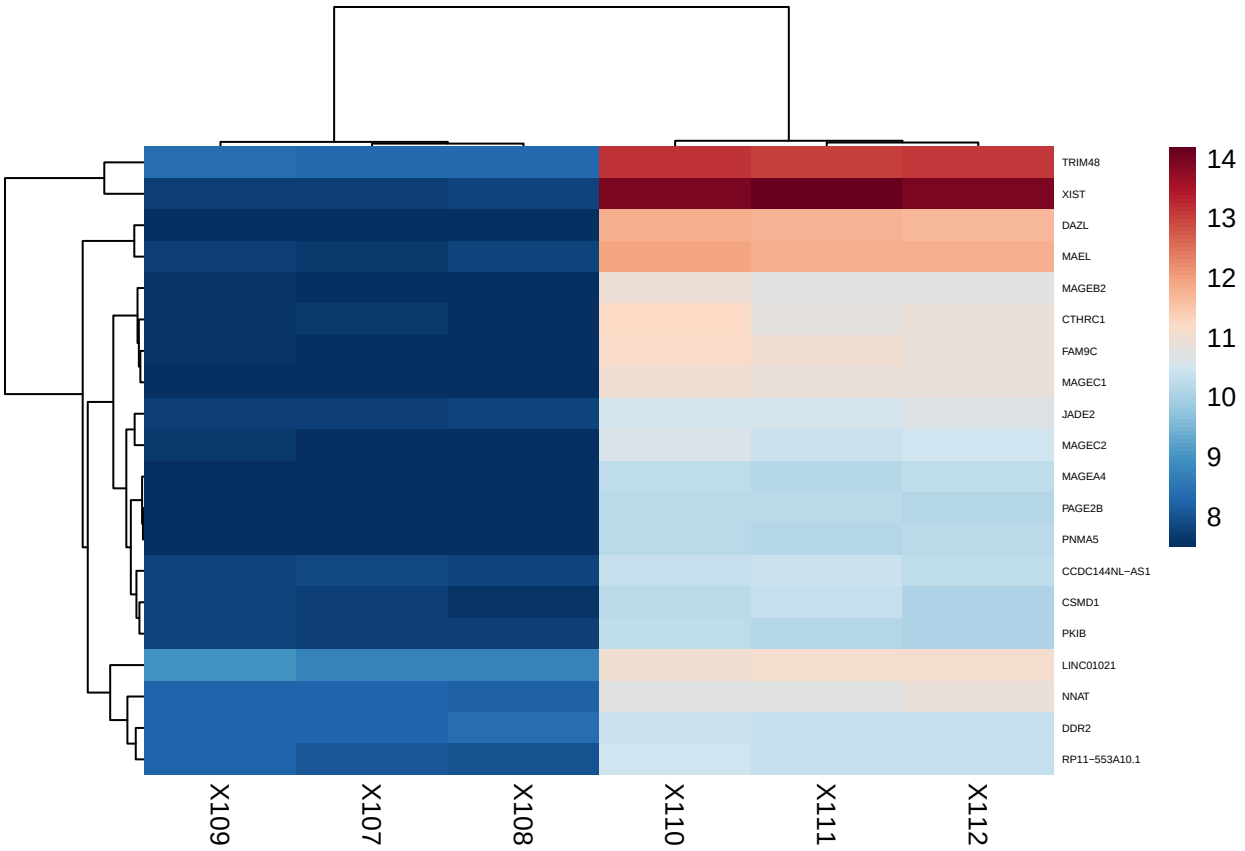
6.5.1 Most significant genes for a given test

```
# plot top significant genes
mostSignificantHeat(data = assay(dnmt$VST),test = dnmt$test$KOvsWT)
```



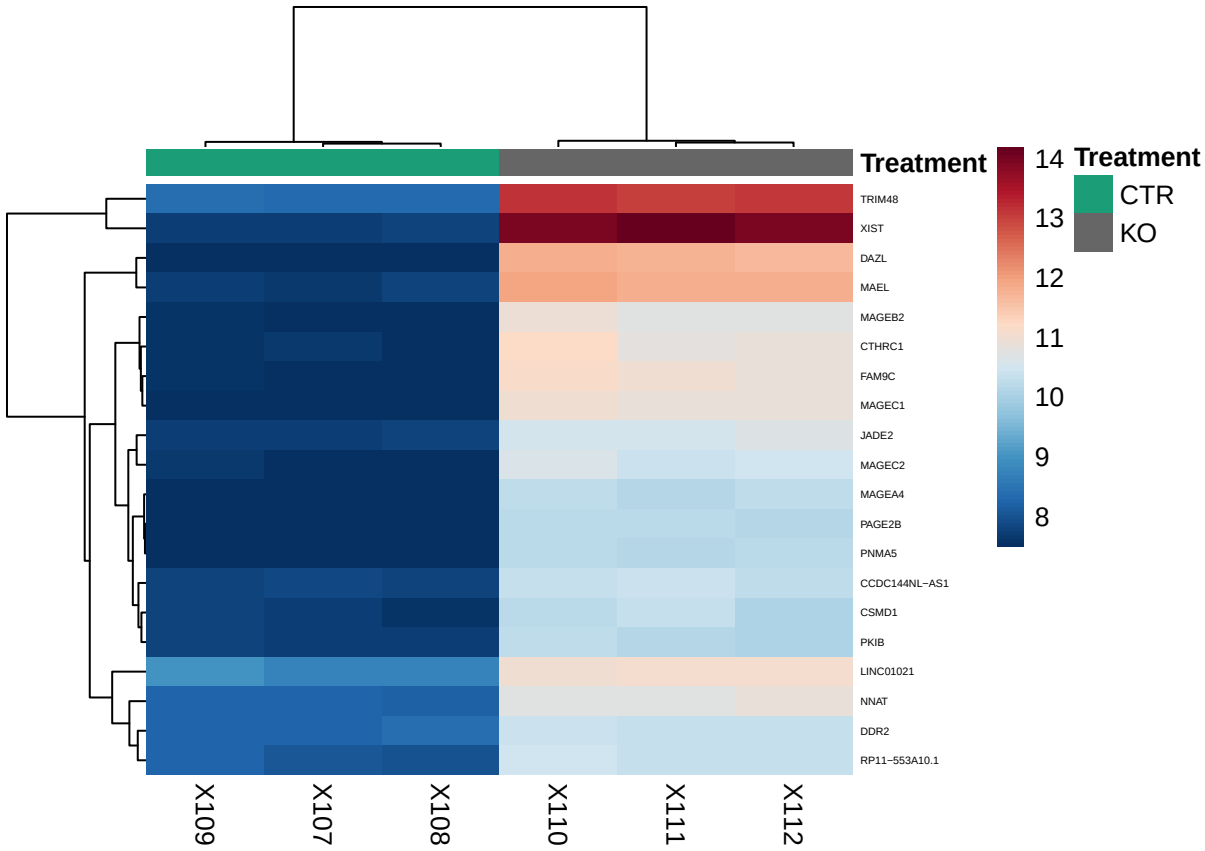

```
##  [1] "ANKRD26P1"     "CCDC144NL-AS1" "CSMD1"         "CTHRC1"
##  [5] "DAZ2"          "DAZL"          "DDR2"          "FAM9C"
##  [9] "FRG1GP"        "GTSF1"         "IL13RA2"       "JADE2"
## [13] "LINC00624"     "LINC01021"     "MAEL"          "MAGEA12"
## [17] "MAGEA2"        "MAGEA3"        "MAGEA4"        "MAGEA6"
## [21] "MAGEB2"        "MAGEC1"        "MAGEC2"        "NEAT1"
## [25] "NNAT"          "PAGE2"         "PAGE2B"        "PAGE5"
## [29] "PKIB"          "PNMA5"         "PRAME"         "RBMY1A1"
## [33] "RBMY1E"        "RBMY1F"        "RGPD2"         "RHOXF2B"
## [37] "RNF17"         "RP11-553A10.1" "SSUH2"         "SSX3"
## [41] "TEX11"         "TEX13C"        "TRIM48"        "TSPYL5"
## [45] "TUBA3D"        "VCX3A"         "VCX3B"         "VCY1B"
## [49] "XAGE1B"        "XIST"
```

```
# plot top significant genes, top 20
mostSignificantHeat(data = assay(dnmt$VST),test = dnmt$test$KOvsWT,ntop = 20)
```




```
##  [1] "CCDC144NL-AS1" "CSMD1"         "CTHRC1"       "DAZL"
##  [5] "DDR2"          "FAM9C"         "JADE2"        "LINC01021"
##  [9] "MAEL"          "MAGEA4"        "MAGEB2"       "MAGEC1"
## [13] "MAGEC2"        "NNAT"          "PAGE2B"       "PKIB"
## [17] "PNMA5"         "RP11-553A10.1" "TRIM48"       "XIST"
```

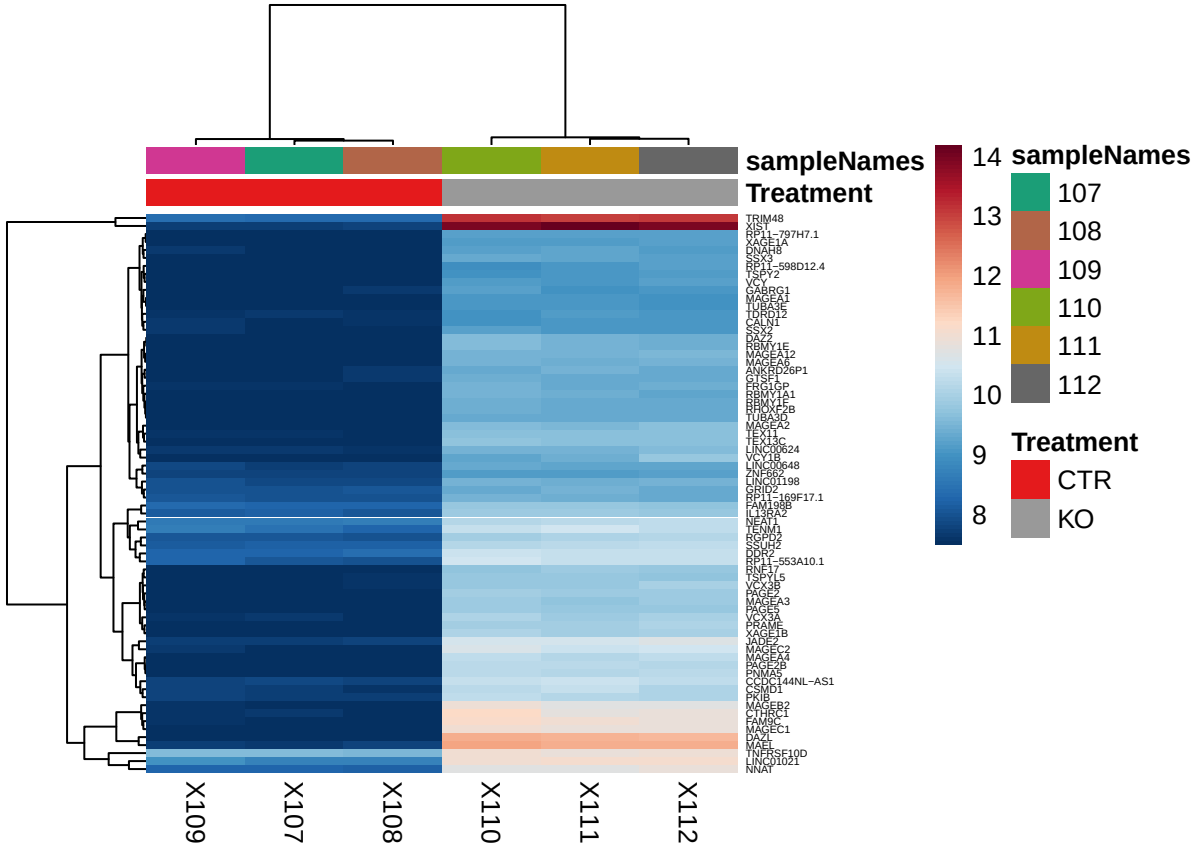
```
# plot top significant genes, top 20, with annotation of columns
mostSignificantHeat(data = assay(dnmt$VST),test = dnmt$test$KOvsWT,ntop = 20,a1 = dnmt$colData$condition
```




```
##  [1] "CCDC144NL-AS1" "CSMD1"         "CTHRC1"       "DAZL"
##  [5] "DDR2"          "FAM9C"         "JADE2"        "LINC01021"
##  [9] "MAEL"          "MAGEA4"        "MAGEB2"       "MAGEC1"
## [13] "MAGEC2"        "NNAT"          "PAGE2B"       "PKIB"
## [17] "PNMA5"         "RP11-553A10.1" "TRIM48"       "XIST"
```

```
## plot top significant genes, top 70, with two annotation of columns
mostSignificantHeat(data = assay(dnmt$VST),test = dnmt$test$KOvsWT,ntop = 70,a1 = dnmt$colData$condition
```

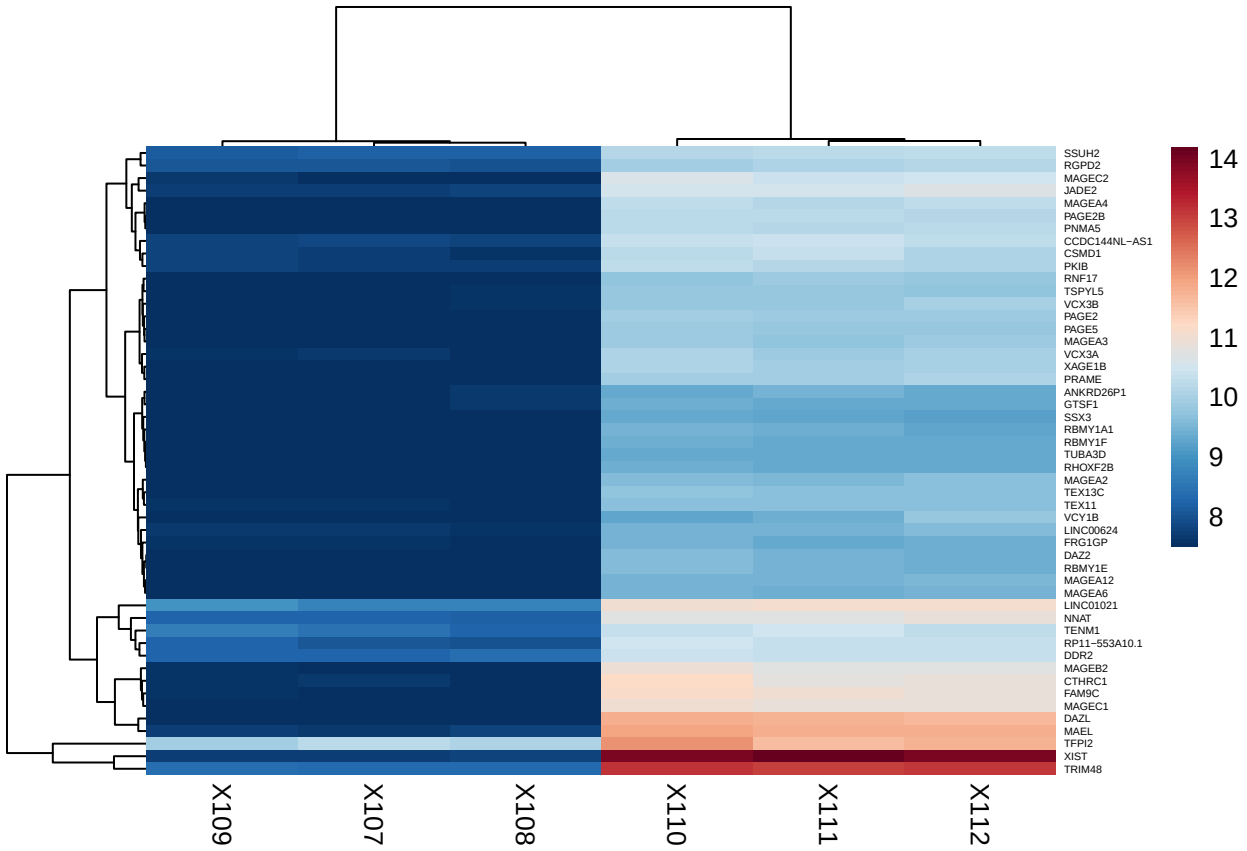
```
## Warning in brewer.pal(9, "Dark2"): n too large, allowed maximum for palette Dark2 is 8
## Returning the palette you asked for with that many colors
```




```
##  [1] "ANKRD26P1"     "CALN1"         "CCDC144NL-AS1" "CSMD1"
##  [5] "CTHRC1"        "DAZ2"          "DAZL"          "DDR2"
##  [9] "DNAH8"         "FAM198B"       "FAM9C"         "FRG1GP"
## [13] "GABRG1"        "GRID2"         "GTSF1"         "IL13RA2"
## [17] "JADE2"         "LINC00624"     "LINC00648"     "LINC01021"
## [21] "LINC01198"     "MAEL"          "MAGEA1"        "MAGEA12"
## [25] "MAGEA2"        "MAGEA3"        "MAGEA4"        "MAGEA6"
## [29] "MAGEB2"        "MAGEC1"        "MAGEC2"        "NEAT1"
## [33] "NNAT"          "PAGE2"         "PAGE2B"        "PAGE5"
## [37] "PKIB"          "PNMA5"         "PRAME"         "RBMY1A1"
## [41] "RBMY1E"        "RBMY1F"        "RGPD2"         "RHOXF2B"
## [45] "RNF17"         "RP11-169F17.1" "RP11-553A10.1" "RP11-598D12.4"
## [49] "RP11-797H7.1"  "SSUH2"         "SSX2"          "SSX3"
## [53] "TDRD12"        "TENM1"         "TEX11"         "TEX13C"
## [57] "TNFRSF10D"     "TRIM48"        "TSPY2"         "TSPYL5"
## [61] "TUBA3D"        "TUBA3E"        "VCX3A"         "VCX3B"
## [65] "VCY"           "VCY1B"         "XAGE1A"        "XAGE1B"
## [69] "XIST"          "ZNF662"
```

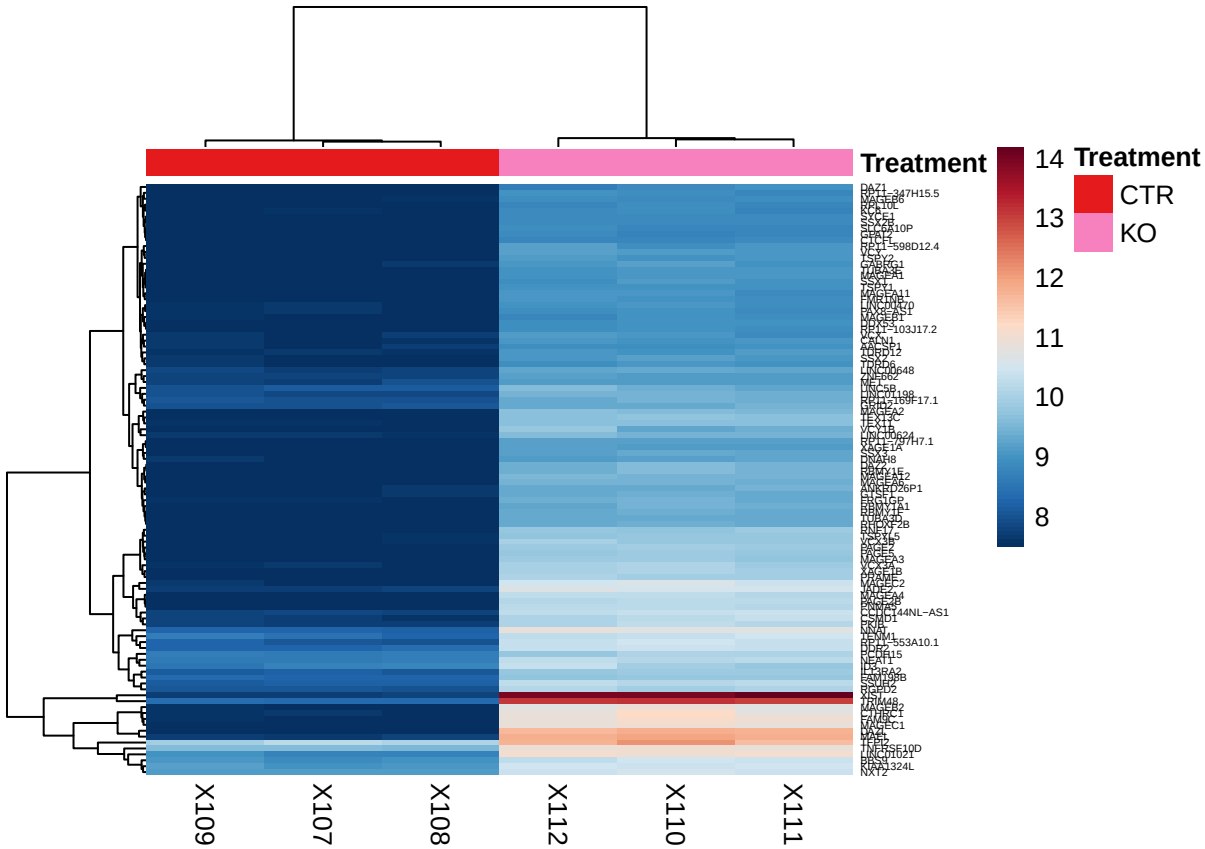
6.5.2 Most variable genes (over all conditions)

```
# plot most variable genes
mostVariableHeat(data = assay(dnmt$VST))
```




```
##  [1] "XIST"          "TRIM48"        "DAZL"          "MAEL"
##  [5] "FAM9C"         "MAGEC1"        "CTHRC1"        "MAGEB2"
##  [9] "MAGEC2"        "JADE2"         "MAGEA4"        "PAGE2B"
## [13] "PNMA5"         "NNAT"          "CCDC144NL-AS1" "XAGE1B"
## [17] "CSMD1"         "PRAME"         "PKIB"          "PAGE2"
## [21] "VCX3A"         "VCX3B"         "PAGE5"         "MAGEA3"
## [25] "RNF17"         "RP11-553A10.1" "LINC01021"     "TSPYL5"
## [29] "TEX13C"        "TEX11"         "MAGEA2"        "DDR2"
## [33] "SSUH2"         "VCY1B"         "RGPD2"         "MAGEA12"
## [37] "DAZ2"          "RBMY1E"        "MAGEA6"        "TENM1"
## [41] "RBMY1A1"       "RBMY1F"        "LINC00624"     "TUBA3D"
## [45] "RHOXF2B"       "FRG1GP"        "ANKRD26P1"     "GTSF1"
## [49] "TFPI2"         "SSX3"
```

```
# plot most variable genes
mostVariableHeat(data = assay(dnmt$VST),ntop = 100,a1 = dnmt$colData$condition,n1 = "Treatment")
```

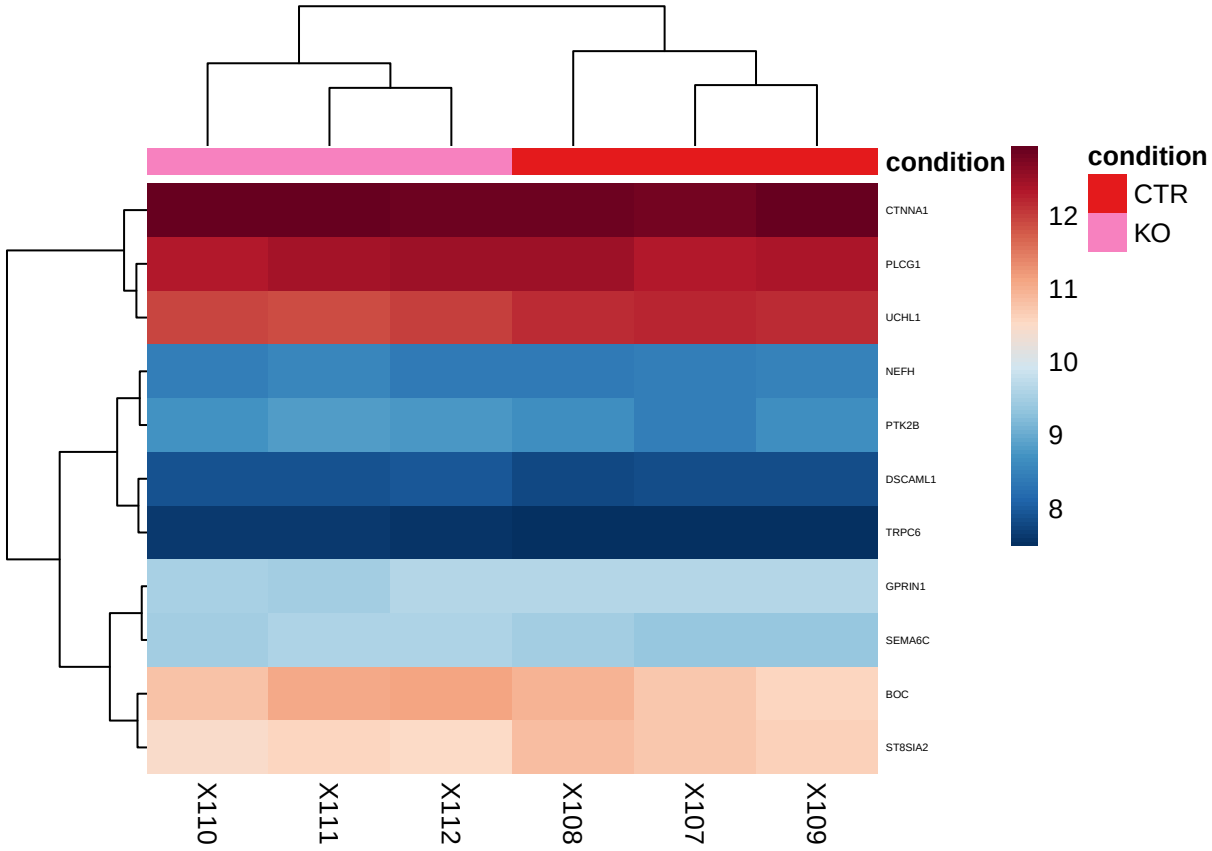



```
##   [1] "XIST"          "TRIM48"        "DAZL"          "MAEL"
##   [5] "FAM9C"         "MAGEC1"        "CTHRC1"        "MAGEB2"
##   [9] "MAGEC2"        "JADE2"         "MAGEA4"        "PAGE2B"
##  [13] "PNMA5"         "NNAT"          "CCDC144NL-AS1" "XAGE1B"
##  [17] "CSMD1"         "PRAME"         "PKIB"          "PAGE2"
##  [21] "VCX3A"         "VCX3B"         "PAGE5"         "MAGEA3"
##  [25] "RNF17"         "RP11-553A10.1" "LINC01021"     "TSPYL5"
##  [29] "TEX13C"        "TEX11"         "MAGEA2"        "DDR2"
##  [33] "SSUH2"         "VCY1B"         "RGPD2"         "MAGEA12"
##  [37] "DAZ2"          "RBMY1E"        "MAGEA6"        "TENM1"
##  [41] "RBMY1A1"       "RBMY1F"        "LINC00624"     "TUBA3D"
##  [45] "RHOXF2B"       "FRG1GP"        "ANKRD26P1"     "GTSF1"
##  [49] "TFPI2"         "SSX3"          "IL13RA2"       "RP11-797H7.1"
##  [53] "XAGE1A"        "DNAH8"         "VCY"           "NEAT1"
##  [57] "RP11-598D12.4" "SSX2"          "TSPY2"         "LINC01198"
##  [61] "TUBA3E"        "MAGEA1"        "GABRG1"        "SSX1"
##  [65] "MAGEA11"       "TSPY1"         "LINC00648"     "FMR1NB"
##  [69] "DDX53"         "FAM198B"       "TDRD6"         "CALN1"
##  [73] "RP11-103J17.2" "TDRD12"        "BBS9"          "RP11-347H15.5"
##  [77] "LINC00470"     "VCX"           "MAGEB1"        "ID3"
##  [81] "MAGEB6"        "RP11-169F17.1" "TNFRSF10D"     "PAX8-AS1"
##  [85] "ZNF662"        "KIAA1324L"     "PCDH15"        "GRID2"
##  [89] "SYCE1"         "SSX2B"         "DAZ1"          "SLC6A10P"
##  [93] "AACSP1"        "UNC5B"         "RPL10L"        "NXT2"
##  [97] "GPAT2"         "CTCFL"         "MET"           "KC6"
```

6.5.3 Heat a set of genes

```
genes <- c("LPPR4", "NEFH", "BOC", "TRPC6", "GPRIN1", "ST8SIA2", "PTK2B", "UCHL1", "CTNNA1", "PLCG1", "S

# set sd cutoff > 0 to be sure not to get error
heatGenes(data = assay(dnmt$VST),genes = genes,sd = .01,a1 =  dnmt$colData$condition,n1 = "condition")
```

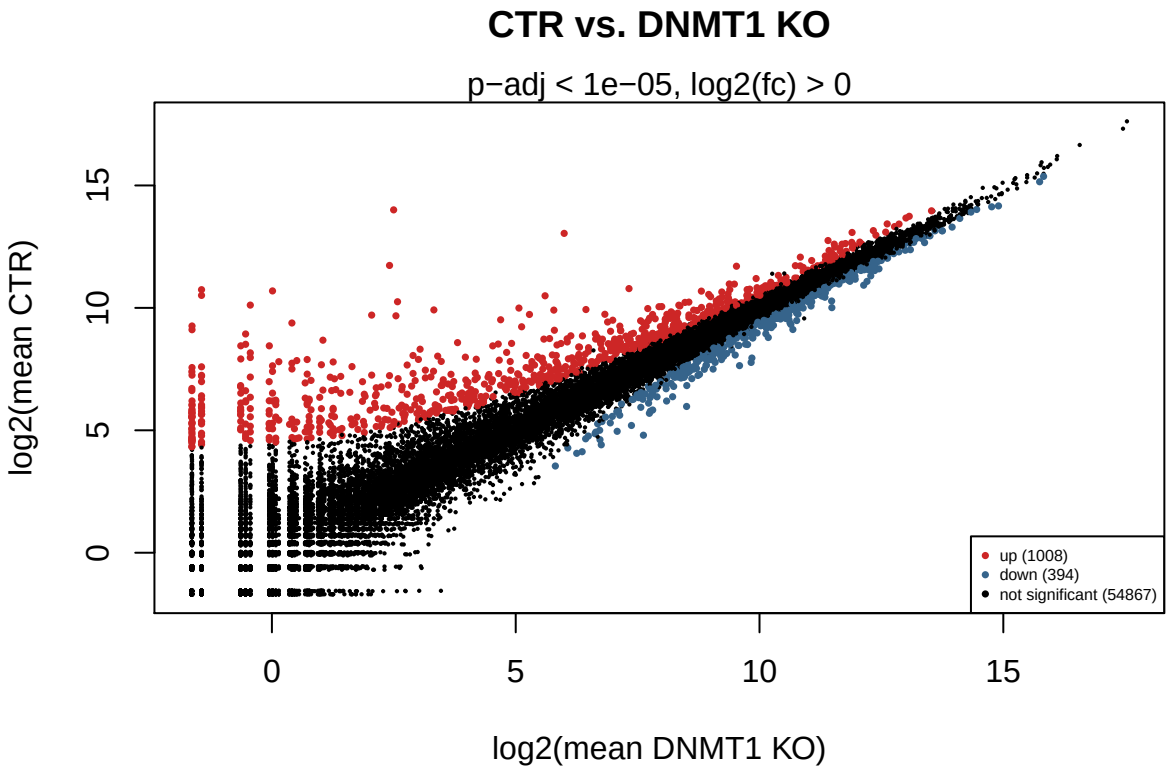
```
## [1] "There are 11 genes in you gene-set with sd > 0.01."
```



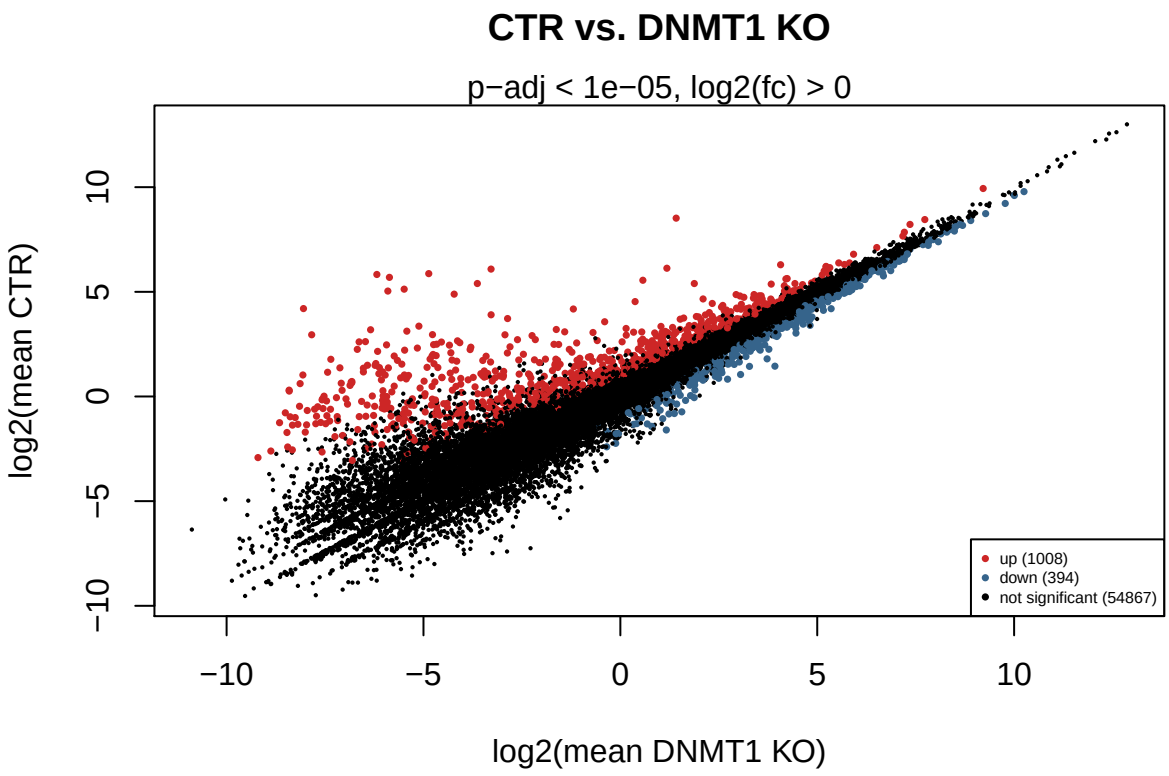

```
##  [1] "BOC"     "CTNNA1"  "DSCAML1" "GPRIN1"  "NEFH"    "PLCG1"   "PTK2B"
##  [8] "SEMA6C"  "ST8SIA2" "TRPC6"   "UCHL1"
```

6.6 Mean Plot (mean of condition A vs B)

```
## exp: mean normalized reads for each condition
## test: diffex test between the conditions
## c1 and c2: specify the name of each condition
meanPlot(exp = dnmt$baseMean$Mean,test = dnmt$test$Default,c1="DNMT1 KO",c2="CTR",p = 1e-5)
```




```
## the same, but plot in FPKM instead
meanPlot(exp = dnmt$FPKMMean$Mean,test = dnmt$test$Default,c1="DNMT1 KO",c2="CTR",p = 1e-5)
```

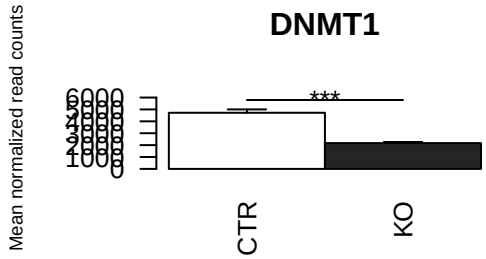


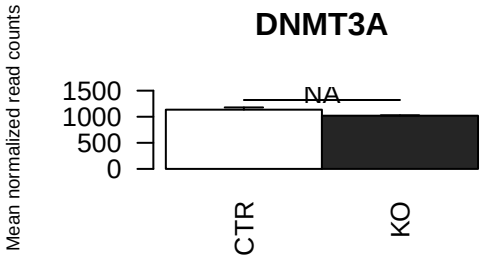
6.7 B barplots

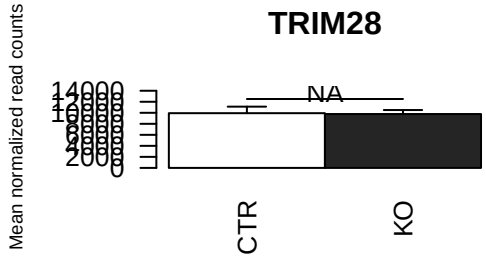
```
## give a vector of any length, with the genes you want to plot
genes <- c("DNMT1","DNMT3A","TRIM28")
dnmt$meanBars(genes = genes )
```

```
## >>> meanBar: plot your genes:
## >> DNMT1 DNMT3A TRIM28
```

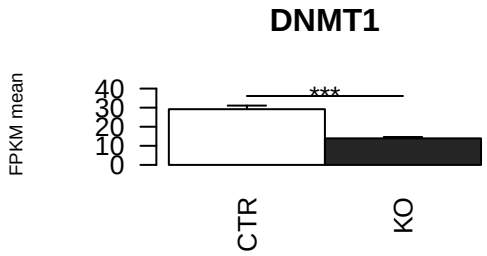
```
# to plot in FPKM instead of mean normalized reads:
dnmt$meanBars(genes = genes,FPKM = T )
```

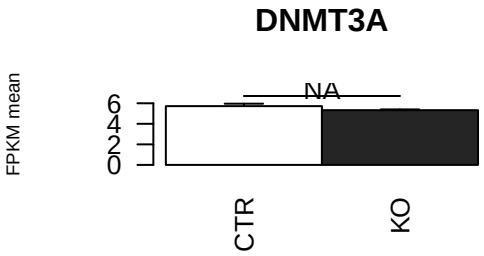


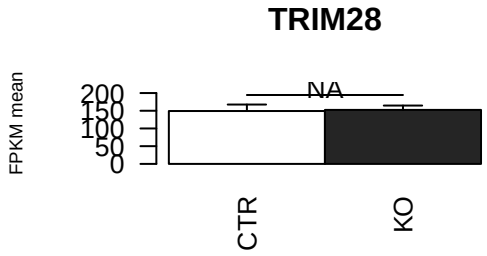





```
## >>> meanBar: plot your genes:
## >> DNMT1 DNMT3A TRIM28
```







7 Get data

7.1 Get normalized expression counts of all genes

```
head(dnmt$normCounts)
```

```
##                  X107       X108       X109       X110       X111
## DDX11L1     0.9612391   4.818654   3.309765  11.102840  22.445984
## WASH7P    397.9530080 471.264343 314.427683 413.833110 428.514249
## MIR6859-1   3.8449566   3.854923   3.309765   6.056094   4.081088
## MIR1302-2   1.9224783   0.000000   1.103255   1.009349   6.121632
## FAM138A     0.0000000   0.000000   0.000000   0.000000   0.000000
## OR4G4P      0.0000000   0.000000   0.000000   0.000000   0.000000
##                 X112
## DDX11L1    16.667882
## WASH7P    564.856017
## MIR6859-1   3.703974
## MIR1302-2   2.777980
## FAM138A     0.000000
## OR4G4P      0.000000
```

7.2 Get average normalized expression in conditions

```
# This is done automatically in $fullAuto(), but can also be done manually like this
dnmt$getAverage()
```

```
## >>Computing mean normalized counts of each condition
## - ..complete! Mean normalized counts computed for each condition. access mean with $baseMean$Mean, a
## >>Computing mean FPKM of each condition
## - ..complete! Mean normalized expression computed for each condition. access mean with $baseMean$Mea
```

```
# Retrieve :
head(dnmt$baseMean$Mean)
```

```
##                  CTR         KO
## DDX11L1     3.029886  16.738902
## WASH7P    394.548345 469.067792
## MIR6859-1   3.669882   4.613719
## MIR1302-2   1.008578   3.302987
## FAM138A     0.000000   0.000000
## OR4G4P      0.000000   0.000000
```

```
head(dnmt$baseMean$SD)
```

```
##                  CTR        KO
## DDX11L1    1.9438779  5.671906
## WASH7P    78.4737425 83.279181
## MIR6859-1  0.3119098  1.263285
## MIR1302-2  0.9647298  2.596263
## FAM138A    0.0000000  0.000000
## OR4G4P     0.0000000  0.000000
```

```
head(dnmt$baseMean$SE)
```

```
##                  CTR        KO
## DDX11L1    1.1222984  3.274676
## WASH7P    45.3068363 48.081258
## MIR6859-1  0.1800812  0.729358
## MIR1302-2  0.5569870  1.498953
## FAM138A    0.0000000  0.000000
## OR4G4P     0.0000000  0.000000
```

7.3 Get FPKM

```
# done automatically by fullAuto(), but can be done manually by:
dnmt$makeFPKM()
```

```
## >>Computing FPKM..
## - ..complete! FPKM computed. Access with $FPKM.
```

```
# retrieve:
head(dnmt$FPKM)
```

```
##                 X107      X108       X109       X110      X111       X112
## DDX11L1   0.03350337 0.1695569 0.11486302 0.39724343 0.8128678  0.6104519
## WASH7P    7.59631739 9.0817294 5.97610380 8.10890486 8.4988633 11.3298213
## MIR6859-1 3.41931422 3.4609550 2.93069611 5.52848148 3.7709242  3.4612222
## MIR1302-2 0.11386551 0.0000000 0.06506279 0.06136741 0.3767231  0.1728916
## FAM138A   0.00000000 0.0000000 0.00000000 0.00000000 0.0000000  0.0000000
## OR4G4P    0.00000000 0.0000000 0.00000000 0.00000000 0.0000000  0.0000000
```

7.4 Get average FPKM in conditions

```
# done automatically by fullAuto(), but can be done manually by:
dnmt$getAverageFPKM()
```

```
## >>Computing mean FPKM of each condition
## - ..complete! Mean normalized expression computed for each condition. access mean with $baseMean$Mean
```

```
# retrieve:
head(dnmt$FPKMMean$Mean)
```

```
##                 CTR        KO
## DDX11L1   0.10597442 0.6068544
## WASH7P    7.55138353 9.3125298
## MIR6859-1 3.27032177 4.2535426
## MIR1302-2 0.05964277 0.2036607
## FAM138A   0.00000000 0.0000000
## OR4G4P    0.00000000 0.0000000
```

```
head(dnmt$FPKMMean$SE)
```

```
##                 CTR         KO
## DDX11L1   0.03952592 0.11999391
## WASH7P    0.89679835 1.01490813
## MIR6859-1 0.17023776 0.64370818
## MIR1302-2 0.03298167 0.09232615
## FAM138A   0.00000000 0.00000000
## OR4G4P    0.00000000 0.00000000
```

```
head(dnmt$FPKMMean$SD)
```

```
##                 CTR        KO
## DDX11L1   0.06846090 0.2078355
## WASH7P    1.55330031 1.7578724
## MIR6859-1 0.29486044 1.1149353
## MIR1302-2 0.05712592 0.1599136
## FAM138A   0.00000000 0.0000000
## OR4G4P    0.00000000 0.0000000
```

7.5 Get significant genes

7.5.1 Get diffex data

```
# Get diffex-data of genes with p-adj < 0.001 and log2FC more/less than 0.2
sign <- getSign(x = dnmt$test$KOvsWT,p = .001,l = .2)
up <- sign$up
down <- sign$down

head(up)
```

```
## log2 fold change (MAP): condition KO vs CTR
## Wald test p-value: condition KO vs CTR
## DataFrame with 6 rows and 6 columns
##                baseMean log2FoldChange     lfcSE      stat       pvalue
##               <numeric>      <numeric> <numeric> <numeric>    <numeric>
## AP006222.2     42.43247      1.1190768 0.2508156  4.461752 8.129236e-06
## RP4-669L17.10 210.94489      1.7090921 0.1622386 10.534437 5.994069e-26
## RP4-669L17.8   76.11536      1.3337407 0.2564413  5.200958 1.982638e-07
## RP11-206L10.2 401.71189      0.8418732 0.1347006  6.249957 4.105670e-10
## TP73           15.71897      1.1530611 0.2810359  4.102895 4.080119e-05
## PER3          364.65156      0.6782073 0.1240911  5.465398 4.618701e-08
##                       padj
##                  <numeric>
## AP006222.2    9.295127e-05
## RP4-669L17.10 4.383433e-24
## RP4-669L17.8  3.188848e-06
## RP11-206L10.2 9.762371e-09
## TP73          3.852982e-04
## PER3          8.321931e-07
```

```
head(down)
```

```
## log2 fold change (MAP): condition KO vs CTR
## Wald test p-value: condition KO vs CTR
## DataFrame with 6 rows and 6 columns
##            baseMean log2FoldChange      lfcSE      stat       pvalue
##           <numeric>      <numeric>  <numeric> <numeric>    <numeric>
## B3GALT6   383.88861     -1.1225994 0.13243989 -8.476294 2.324812e-17
## SCNN1D     24.32859     -1.3799254 0.27885568 -4.948529 7.477643e-07
## MRPL20    890.40074     -0.4340588 0.10729578 -4.045441 5.222466e-05
## SSU72    1257.88818     -0.3781715 0.09498929 -3.981201 6.856781e-05
## FNDC10     38.44308     -1.1498191 0.26229991 -4.383605 1.167314e-05
## TP73-AS1  935.02458     -0.6994438 0.10088263 -6.933243 4.113016e-12
##                 padj
##            <numeric>
## B3GALT6  1.066895e-15
## SCNN1D   1.078925e-05
## MRPL20   4.797683e-04
## SSU72    6.084608e-04
## FNDC10   1.279326e-04
## TP73-AS1 1.203131e-10
```

7.5.2 Get names only

```
# Get names of genes with p-adj < 0.01 and log2FC more/less than .1
signID <- getSignName(x = dnmt$test$KOvsWT,p = .01,l = .2)
upID <- signID$up
downID <- signID$down

head(upID)
```

```
## [1] "FO538757.2"    "AP006222.2"    "RP4-669L17.10" "RP4-669L17.8"
## [5] "RP11-206L10.2" "TP73"
```

```
head(downID)
```

```
## [1] "B3GALT6" "SCNN1D"  "MRPL20"  "VWA1"    "SSU72"   "FNDC10"
```

7.6 Get most variable genes

```
var.50 <- dnmt$getVariable(ntop = 50)
var.50
```

```
##  [1] "XIST"          "TRIM48"        "DAZL"          "MAEL"
##  [5] "FAM9C"         "MAGEC1"        "CTHRC1"        "MAGEB2"
##  [9] "MAGEC2"        "JADE2"         "MAGEA4"        "PAGE2B"
## [13] "PNMA5"         "NNAT"          "CCDC144NL-AS1" "XAGE1B"
## [17] "CSMD1"         "PRAME"         "PKIB"          "PAGE2"
## [21] "VCX3A"         "VCX3B"         "PAGE5"         "MAGEA3"
## [25] "RNF17"         "RP11-553A10.1" "LINC01021"     "TSPYL5"
## [29] "TEX13C"        "TEX11"         "MAGEA2"        "DDR2"
## [33] "SSUH2"         "VCY1B"         "RGPD2"         "MAGEA12"
## [37] "DAZ2"          "RBMY1E"        "MAGEA6"        "TENM1"
## [41] "RBMY1A1"       "RBMY1F"        "LINC00624"     "TUBA3D"
## [45] "RHOXF2B"       "FRG1GP"        "ANKRD26P1"     "GTSF1"
## [49] "TFPI2"         "SSX3"
```

7.7 Get genes close to features (in bed file)

```
## takes as input a bed table
## E.g. a SVA-F elements
## NB: Must have col.names as defined here col.names = c("Chr","Start","End","ID",".","Strand")
svaf <- read.delim(file = "~/genomicData/Repeats/hg38/SVA/SVA_F.hg38.bed",col.names = c("Chr","Start","E
head(svaf)
```

```
##    Chr    Start      End    ID . Strand
## 1 chr1  8079152  8079209 SVA_F .      -
## 2 chr1 13013447 13013533 SVA_F .      -
## 3 chr1 13215427 13215513 SVA_F .      +
```

```
## 4 chr1 13361171 13361257 SVA_F .      +
## 5 chr1 21023240 21024912 SVA_F .      +
## 6 chr1 21387077 21388776 SVA_F .      +
```

```
# calculate genes within 15000bp from SVA-F!
# By default, this function search in Gencode v27 (hg38) annotation. This can be changed with setting a
close.svaf <- closeGenes(b=svaf,d = 15000)
```

```
## > Downloading the gencode v25 annotation of protein-coding transcripts..
```

```
## Loading required package: bitops
```

```
## > Gencode v25 protein-coding transcripts .bed downloaded!
## > (By default, this bed file is used, as it provides coordinates for each protein-coding transcript
## >> Getting genes with TSS within 15000 bps from each given feature. This might take a couple of minu
## - ..complete! Genes A with TSS within 15000 bps from each features B TSS is computed.
```

```
head(close.svaf)
```

```
##       A_ID A_chr  A_Start    A_End    A_TSS A_Strand  B_ID    B_TSS
## 1 PRAMEF18  chr1 13222695 13226133 13226133        - SVA_F 13215427
## 2 PRAMEF14  chr1 13342366 13347134 13347134        - SVA_F 13361171
## 3 PRAMEF19  chr1 13369067 13371693 13371693        - SVA_F 13361171
## 4   HNRNPR  chr1 23304688 23344336 23344336        - SVA_F 23346394
## 5   HNRNPR  chr1 23309783 23344310 23344310        - SVA_F 23346394
## 6   HNRNPR  chr1 23309786 23344322 23344322        - SVA_F 23346394
##   B_Strand Distance
## 1        +    10706
## 2        +   -14037
## 3        +    10522
## 4        +    -2058
## 5        +    -2084
## 6        +    -2072
```