FedRefed.rmd

I - Loading

```r
#================================ Loading libraries ================================= #
#updateR()
library("DESeq2")
```

1 - Libraries

```
## Le chargement a nécessité le package : S4Vectors

## Le chargement a nécessité le package : stats4

## Le chargement a nécessité le package : BiocGenerics

## Le chargement a nécessité le package : generics

##
## Attachement du package : 'generics'

## Les objets suivants sont masqués depuis 'package:base':
##
##     as.difftime, as.factor, as.ordered, intersect, is.element, setdiff,
##     setequal, union

##
## Attachement du package : 'BiocGenerics'

## Les objets suivants sont masqués depuis 'package:stats':
##
##     IQR, mad, sd, var, xtabs

## Les objets suivants sont masqués depuis 'package:base':
##
##     anyDuplicated, aperm, append, as.data.frame, basename, cbind,
##     colnames, dirname, do.call, duplicated, eval, evalq, Filter, Find,
##     get, grep, grepl, is.unsorted, lapply, Map, mapply, match, mget,
##     order, paste, pmax, pmax.int, pmin, pmin.int, Position, rank,
##     rbind, Reduce, rownames, sapply, saveRDS, table, tapply, unique,
##     unsplit, which.max, which.min

##
## Attachement du package : 'S4Vectors'
## L'objet suivant est masqué depuis 'package:utils':
##
##     findMatches

## Les objets suivants sont masqués depuis 'package:base':
##
##     expand.grid, I, unname

## Le chargement a nécessité le package : IRanges

## Le chargement a nécessité le package : GenomicRanges

## Le chargement a nécessité le package : Seqinfo

## Le chargement a nécessité le package : SummarizedExperiment

## Le chargement a nécessité le package : MatrixGenerics

## Le chargement a nécessité le package : matrixStats

##
## Attachement du package : 'MatrixGenerics'

## Les objets suivants sont masqués depuis 'package:matrixStats':
##
##     colAlls, colAnyNAs, colAnys, colAvgsPerRowSet, colCollapse,
##     colCounts, colCummaxs, colCummins, colCumprods, colCumsums,
##     colDiffs, colIQRDiffs, colIQRs, colLogSumExps, colMadDiffs,
##     colMads, colMaxs, colMeans2, colMedians, colMins, colOrderStats,
##     colProds, colQuantiles, colRanges, colRanks, colSdDiffs, colSds,
##     colSums2, colTabulates, colVarDiffs, colVars, colWeightedMads,
##     colWeightedMeans, colWeightedMedians, colWeightedSds,
##     colWeightedVars, rowAlls, rowAnyNAs, rowAnys, rowAvgsPerColSet,
##     rowCollapse, rowCounts, rowCummaxs, rowCummins, rowCumprods,
##     rowCumsums, rowDiffs, rowIQRDiffs, rowIQRs, rowLogSumExps,
##     rowMadDiffs, rowMads, rowMaxs, rowMeans2, rowMedians, rowMins,
##     rowOrderStats, rowProds, rowQuantiles, rowRanges, rowRanks,
##     rowSdDiffs, rowSds, rowSums2, rowTabulates, rowVarDiffs, rowVars,
##     rowWeightedMads, rowWeightedMeans, rowWeightedMedians,
##     rowWeightedSds, rowWeightedVars

## Le chargement a nécessité le package : Biobase

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

##
## Attachement du package : 'Biobase'
## L'objet suivant est masqué depuis 'package:MatrixGenerics':
##
##     rowMedians

## Les objets suivants sont masqués depuis 'package:matrixStats':
##
##     anyMissing, rowMedians
```

```r
library("ggplot2")
library("ggrepel")
library("readxl")
library("tidyr")
```

```
##
## Attachement du package : 'tidyr'

## L'objet suivant est masqué depuis 'package:S4Vectors':
##
##     expand
```

```r
library("dplyr")
```

```
##
## Attachement du package : 'dplyr'

## L'objet suivant est masqué depuis 'package:Biobase':
##
##     combine

## L'objet suivant est masqué depuis 'package:matrixStats':
##
##     count

## Les objets suivants sont masqués depuis 'package:GenomicRanges':
##
##     intersect, setdiff, union

## L'objet suivant est masqué depuis 'package:Seqinfo':
##
##     intersect

## Les objets suivants sont masqués depuis 'package:IRanges':
##
##     collapse, desc, intersect, setdiff, slice, union

## Les objets suivants sont masqués depuis 'package:S4Vectors':
##
##     first, intersect, rename, setdiff, setequal, union

## Les objets suivants sont masqués depuis 'package:BiocGenerics':
##
##     combine, intersect, setdiff, setequal, union
## L'objet suivant est masqué depuis 'package:generics':
##
##     explain

## Les objets suivants sont masqués depuis 'package:stats':
##
##     filter, lag

## Les objets suivants sont masqués depuis 'package:base':
##
##     intersect, setdiff, setequal, union
```

```
library("stringr")
```

```
#================================= Spot checking================================= #
# Example for KO check expression :
fc <- read.delim("/home/kenza/data/MetID_2/featurecounts.txt", comment.char = "#", check.names = FALSE)
rownames(fc) <- fc$Geneid
head(fc)
```

2 - Loading featurecounts table

```
##      Geneid  Chr Start   End Strand Length 10.sorted.bam 11.sorted.bam
## TrnP   TrnP chrM 15356 15422      -     67           941          1410
## TrnT   TrnT chrM 15289 15355      +     67             0             0
## CYTB   CYTB chrM 14145 15288      +   1144         89698        166120
## TrnE   TrnE chrM 14071 14139      -     69            26           156
## ND6     ND6 chrM 13552 14070      -    519         56692         31474
## ND5     ND5 chrM 11742 13565      +   1824         64105         50837
##      12.sorted.bam 13.sorted.bam 14.sorted.bam 15.sorted.bam 17.sorted.bam
## TrnP          1093          1053           962          1052          1277
## TrnT             1             1             0             2             4
## CYTB        120050        101192         97450        228639        192686
## TrnE           149            69            25           262           367
## ND6          15064         23667         60696         72430         28872
## ND5          68569         43346         60978         99256         98309
##      18.sorted.bam 19.sorted.bam 20.sorted.bam 22.sorted.bam 23.sorted.bam
## TrnP           866          2195           938          1032          1720
## TrnT             2             1             0             1             0
## CYTB        169005        171643         76546         77572        286158
## TrnE           145           150            23            41           182
## ND6          64386         41129         66578         14287         70988
## ND5          63017         67128         50131         32970         94935
##      24.sorted.bam 26.sorted.bam 27.sorted.bam 28.sorted.bam 29.sorted.bam
## TrnP          1852          1315          1284           789          1374
## TrnT             0             1             0             1             1
## CYTB         90066        198543        231084         77877        101473
## TrnE            48           229           195            32            36
## ND6          27578         26339         54509         28536         22411
## ND5          40804         78147         90997         36413         43378
##      2.sorted.bam 33.sorted.bam 35.sorted.bam 36.sorted.bam 37.sorted.bam
## TrnP         1795          1736          1288          1339           938
## TrnT            1             2             0             7             0
## CYTB       169501        237610        108176        230173        110452
## TrnE          119           568            77           175            44
## ND6         38223         46206         33968         40715         19930
## ND5         88246         95994         50924         79397         40768
##      38.sorted.bam 39.sorted.bam 3.sorted.bam 41.sorted.bam 43.sorted.bam
## TrnP          1308          1279          864          1647          1785
## TrnT             1             1            3             4            10
## CYTB        217213         88877       146229        345204        258005
## TrnE           148            26          221           333           602
## ND6          41249         15180        20020         61004         53407
## ND5          82937         51682        53880        118163        108693
##      44.sorted.bam 45.sorted.bam 47.sorted.bam 49.sorted.bam 4.sorted.bam
## TrnP          1643          1719          1510          1731         1407
## TrnT             1             2             0             2            1
## CYTB        309861        269876        146144        302975       114632
## TrnE           388           316           125           271          100
## ND6          67382         33755         36513         53151        22931
## ND5         124662         77161         59715         87006        44212
##      50.sorted.bam 52.sorted.bam 53.sorted.bam 54.sorted.bam 55.sorted.bam
## TrnP          1830          1260          1362          2394          1988
## TrnT             7             1             1             0             2
## CYTB        205821        268533        176796        175145        123228
## TrnE           302           298           282           118            83
## ND6          32916         48644         24599         37401         35141
## ND5          73455         90552        103501         91097         65556
##      5.sorted.bam 9.sorted.bam
## TrnP         1414          913
## TrnT            8            2
## CYTB        83027       105852
## TrnE           41           60
## ND6         22933        11640
## ND5         43278        33120
```

```r
# Keep only the counts from featurecounts
rawcounts <- fc[, -(1:6)]
rawcounts <- as.matrix(rawcounts)
storage.mode(rawcounts) <- "numeric"
bam_names <- colnames(fc)[-(1:6)] # gives bam names
sample_names <- str_extract(string = bam_names, pattern = '\\d{1,2}')
colnames(rawcounts) <- sample_names

# Remove NA values from rawcounts table
rawcounts <- na.omit(rawcounts)

# How many reads do I have per sample?
colSums(rawcounts)
```

```
##       10       11       12       13       14       15       17       18
## 21062483 20601777 19733978 20994864 20975117 20254960 20857692 20716712
##       19       20       22       23       24       26       27       28
## 20483196 21182483 21021516 19767725 20967532 20676808 20632011 20759497
##       29        2       33       35       36       37       38       39
## 20729354 20305674 20246561 20847546 20468292 20743901 20311768 20990044
##        3       41       43       44       45       47       49        4
## 19464342 18746544 20055232 19516405 18816502 20121260 19521765 20423546
##       50       52       53       54       55        5        9
## 19551150 20266739 20602298 20028701 20299157 21114708 20696447
```

```
splan <- read_excel("MetID_Exp2_sample_plan.xlsx")
```

3 - Create splan

```
## New names:
## * `` -> `...2`
## * `` -> `...3`
## * `` -> `...4`
```

```
splan <- as.data.frame(splan)
colnames(splan) <- splan[4,]
splan <- splan[-(1:4),]
rownames(splan) <- splan$sample
splan
```

```
##    sample    treatment genotype   color
## 2       2   Refed 1.5h       ko #9933CC
## 3       3         Fast       wt #FF3333
## 4       4         Fast       ko #FF3333
## 5       5         Fast       ko #FF3333
## 9       9 Refed 30 min       ko #CC3366
## 10     10 Refed 30 min       ko #CC3366
## 11     11 Refed 30 min       ko #CC3366
## 12     12   Refed 1.5h       wt #9933CC
## 13     13   Refed 1.5h       ko #9933CC
## 14     14   Refed 1.5h       ko #9933CC
## 15     15   Refed 1.5h       wt #9933CC
## 17     17     Refed 3h       wt #3333FF
## 18     18     Refed 3h       wt #3333FF
## 19     19     Refed 3h       ko #3333FF
## 20     20     Refed 3h       ko #3333FF
## 22     22     Refed 3h       ko #3333FF
## 23     23          Fed       wt #00CC33
## 24     24          Fed       ko #00CC33
## 26     26         Fast       wt #FF3333
## 27     27         Fast       wt #FF3333
## 28     28         Fast       ko #FF3333
## 29     29 Refed 30 min       ko #CC3366
## 33     33   Refed 1.5h       wt #9933CC
## 35     35   Refed 1.5h       ko #9933CC
## 36     36     Refed 3h       wt #3333FF
## 37     37     Refed 3h       ko #3333FF
## 38     38     Refed 3h       wt #3333FF
## 39     39         Fast       ko #FF3333
## 41     41          Fed       wt #00CC33
## 43     43 Refed 30 min       wt #CC3366
## 44     44 Refed 30 min       wt #CC3366
## 45     45          Fed       wt #00CC33
## 47     47          Fed       ko #00CC33
## 49     49         Fast       wt #FF3333
## 50     50          Fed       wt #00CC33
## 52     52 Refed 30 min       wt #CC3366
## 53     53 Refed 30 min       wt #CC3366
## 54     54          Fed       ko #00CC33
## 55     55          Fed       ko #00CC33
```

```
# Control Splan !
# Make sure that rownames in colData are matching with column names in rawcounts
all((colnames(rawcounts)) %in% rownames(splan))
```

```
## [1] TRUE
```

```
# Are they in the same order ?
splan <- splan[(colnames(rawcounts)),]
all((colnames(rawcounts)) == rownames(splan))
```

```
## [1] TRUE
```

```
colours <- splan$color
```

II - Visualize

```


l2_rawcounts <-data.frame(log2(1+rawcounts))

# convert wide to long
plotDat <- gather(l2_rawcounts, "x", "y")

# then plot, and rotate labels 90 degrees.
boxplot_l2raw <- ggplot(plotDat, aes(x, y)) +
  geom_boxplot() +
  scale_fill_manual(values=c) +
  xlab("") +
  ylab("log2 rawcounts")+
  theme(axis.text.x = element_text(angle = 90, vjust = 0.5))

ggsave("images/boxplot_l2rawcounts.png",width = 10, height = 4)
```

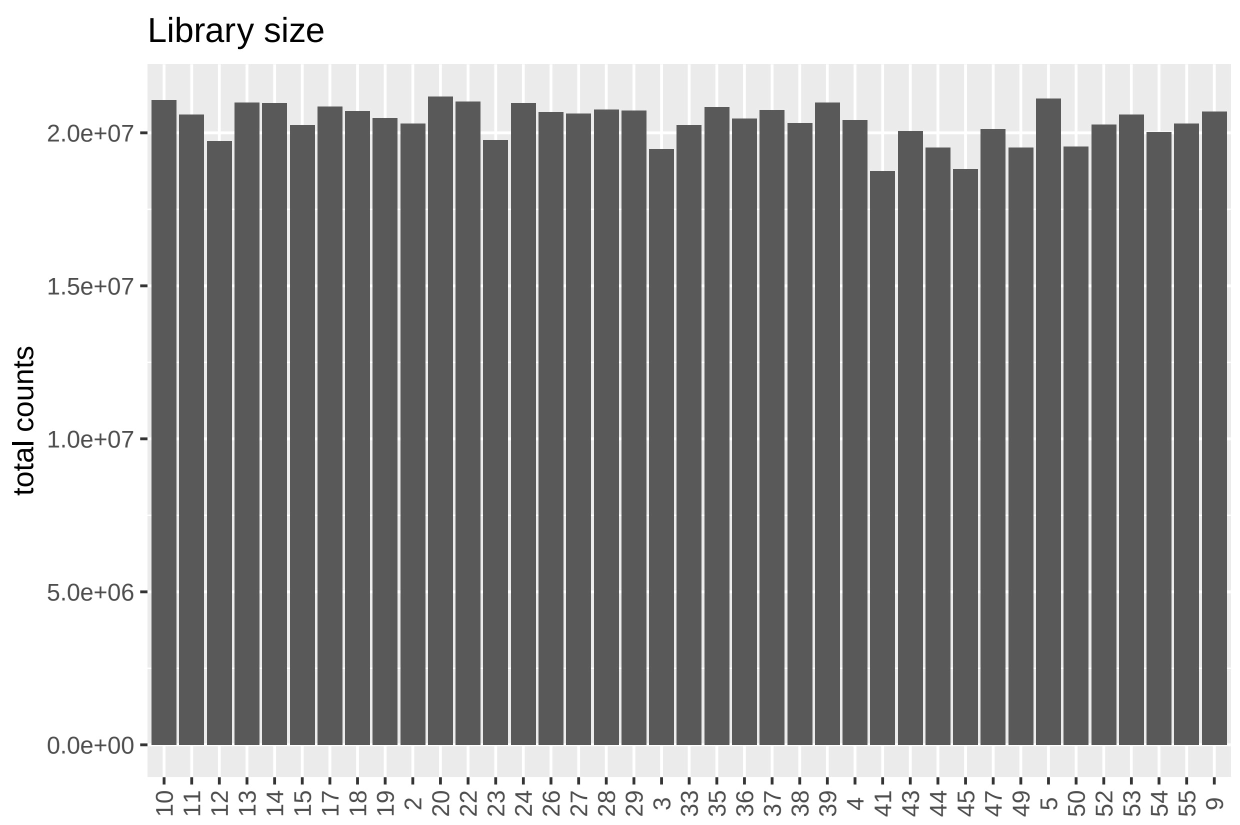
Raw counts

```r
libsize <- data.frame(name = colnames(rawcounts),value = (colSums(rawcounts)))
libsize
```

Library size

```
##    name    value
## 10   10 21062483
## 11   11 20601777
## 12   12 19733978
## 13   13 20994864
## 14   14 20975117
## 15   15 20254960
## 17   17 20857692
## 18   18 20716712
## 19   19 20483196
## 20   20 21182483
## 22   22 21021516
## 23   23 19767725
## 24   24 20967532
## 26   26 20676808
## 27   27 20632011
## 28   28 20759497
## 29   29 20729354
## 2     2 20305674
## 33   33 20246561
## 35   35 20847546
## 36   36 20468292
## 37   37 20743901
## 38   38 20311768
## 39   39 20990044
## 3     3 19464342
## 41   41 18746544
## 43   43 20055232
## 44   44 19516405
## 45   45 18816502
## 47   47 20121260
## 49   49 19521765
## 4     4 20423546
## 50   50 19551150
## 52   52 20266739
## 53   53 20602298
## 54   54 20028701
## 55   55 20299157
## 5     5 21114708
## 9     9 20696447
```

```r
ggplot(libsize, aes(name,value)) +
  geom_bar(stat = "identity")+
  xlab("") +
  ylab("total counts")+
  theme(axis.text.x = element_text(angle = 90, vjust = 0.5)) +
  ggtitle("Library size")
```



```
ggsave("images/barplot_libsize.png",width = 10, height = 4)
```

```
dds <- DESeqDataSetFromMatrix(countData = rawcounts, colData = splan, design = ~ treatment)
```

```
## converting counts to integer mode

## Warning in DESeqDataSet(se, design = design, ignoreRank): some variables in
## design formula are characters, converting to factors

##   Note: levels of factors in the design contain characters other than
##   letters, numbers, '_' and '.'. It is recommended (but not required) to use
##   only letters, numbers, and delimiters '_' or '.', as these are safe characters
##   for column names in R. [This is a message, not a warning or an error]
```

```
dds <- estimateSizeFactors(dds)
```

```
##   Note: levels of factors in the design contain characters other than
##   letters, numbers, '_' and '.'. It is recommended (but not required) to use
##   only letters, numbers, and delimiters '_' or '.', as these are safe characters
##   for column names in R. [This is a message, not a warning or an error]
```

```
vsd <- vst(dds, blind = TRUE)
rld <- rlog(dds)
```

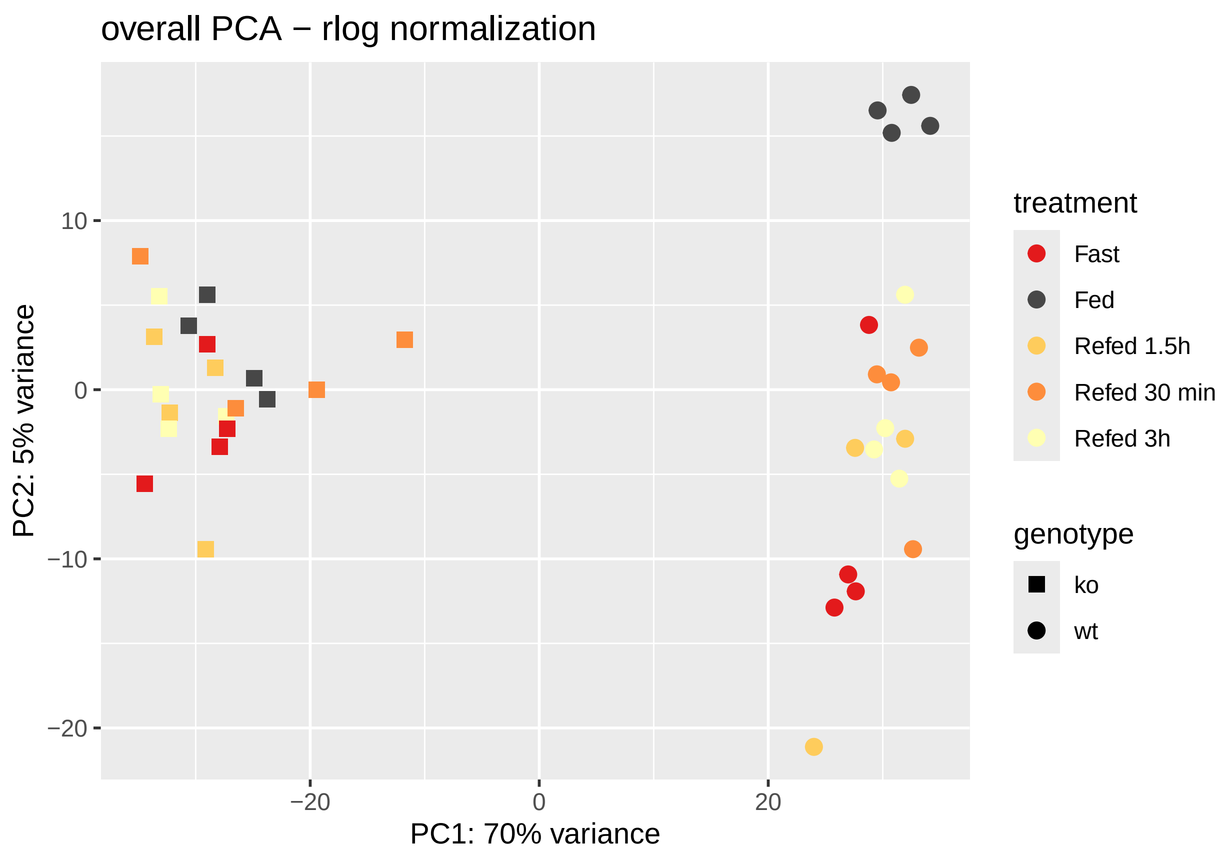
```
## rlog() may take a few minutes with 30 or more samples,
## vst() is a much faster transformation
```

```
PCArld <- plotPCA(rld, intgroup='genotype', returnData=TRUE, ntop = 1000)
```

1 - PCA rlog or vst ? - overall

```
## using ntop=1000 top features by variance
```

```
percentVar <- round(100 * attr(PCArld, "percentVar"))
pca_overall <- ggplot(PCArld, aes(x=PC1, y=PC2))+
  geom_point(aes(shape=genotype,color=treatment) ,size = 2.5)+
  scale_shape_manual(values=c(15, 19))+
  scale_color_manual(values = c("#e31a1c", "gray28", "#fecc5c","#fd8d3c","#ffffb2"))+
  xlab(paste0("PC1: ",percentVar[1],"% variance")) +
  ylab(paste0("PC2: ",percentVar[2],"% variance"))+
  ggtitle("overall PCA - rlog normalization")
pca_overall
```



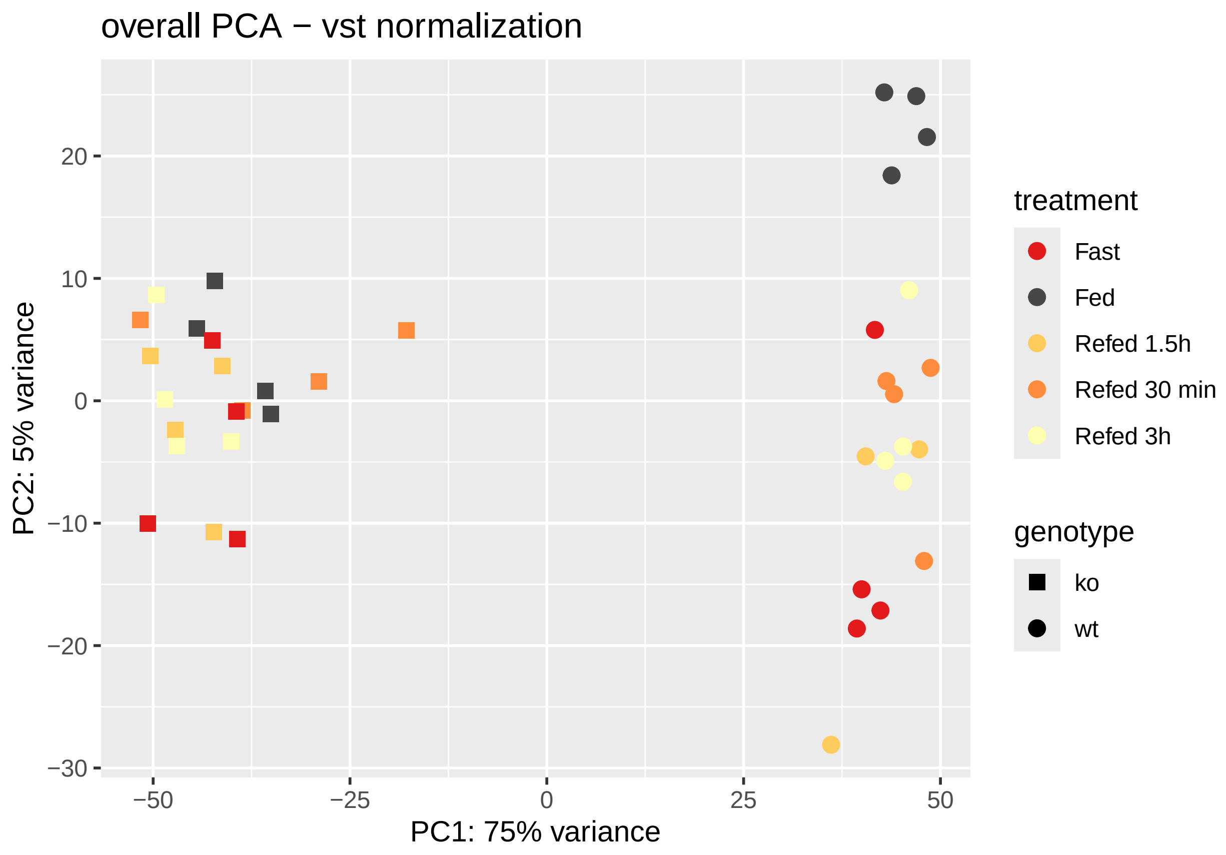
```
ggsave("images/overall_pca-rld.png")
```

```
## Saving 6.5 x 4.5 in image
```

```
PCAvst <- plotPCA(vsd, intgroup='genotype', returnData=TRUE, ntop = 1000)
```

```
## using ntop=1000 top features by variance
```

```
percentVar <- round(100 * attr(PCAvst, "percentVar"))
pca_overall <- ggplot(PCAvst, aes(x=PC1, y=PC2))+
  geom_point(aes(shape=genotype,color=treatment) ,size = 2.5)+
  scale_shape_manual(values=c(15, 19))+
  scale_color_manual(values = c("#e31a1c", "gray28", "#fecc5c","#fd8d3c","#ffffb2"))+
  xlab(paste0("PC1: ",percentVar[1],"% variance")) +
  ylab(paste0("PC2: ",percentVar[2],"% variance"))+
  ggtitle("overall PCA - vst normalization")
pca_overall
```



```
ggsave("images/overall_pca-vst.png")
```

```
## Saving 6.5 x 4.5 in image
```

```
# Sorting dataset by genotype
splanko <- splan[splan$genotype=='ko',]
splanwt<- splan[splan$genotype=='wt',]

rawcountsko <- rawcounts[,splanko$sample]
```

```r
rawcountswt <- rawcounts[,splanwt$sample]

# WT - PCA
ddswt <- DESeqDataSetFromMatrix(countData = rawcountswt, colData = splanwt, design = ~ treatment)
```

```
## converting counts to integer mode

## Warning in DESeqDataSet(se, design = design, ignoreRank): some variables in
## design formula are characters, converting to factors

##   Note: levels of factors in the design contain characters other than
##   letters, numbers, '_' and '.'. It is recommended (but not required) to use
##   only letters, numbers, and delimiters '_' or '.', as these are safe characters
##   for column names in R. [This is a message, not a warning or an error]
```

```r
ddswt <- estimateSizeFactors(ddswt)
```

```
##   Note: levels of factors in the design contain characters other than
##   letters, numbers, '_' and '.'. It is recommended (but not required) to use
##   only letters, numbers, and delimiters '_' or '.', as these are safe characters
##   for column names in R. [This is a message, not a warning or an error]
```

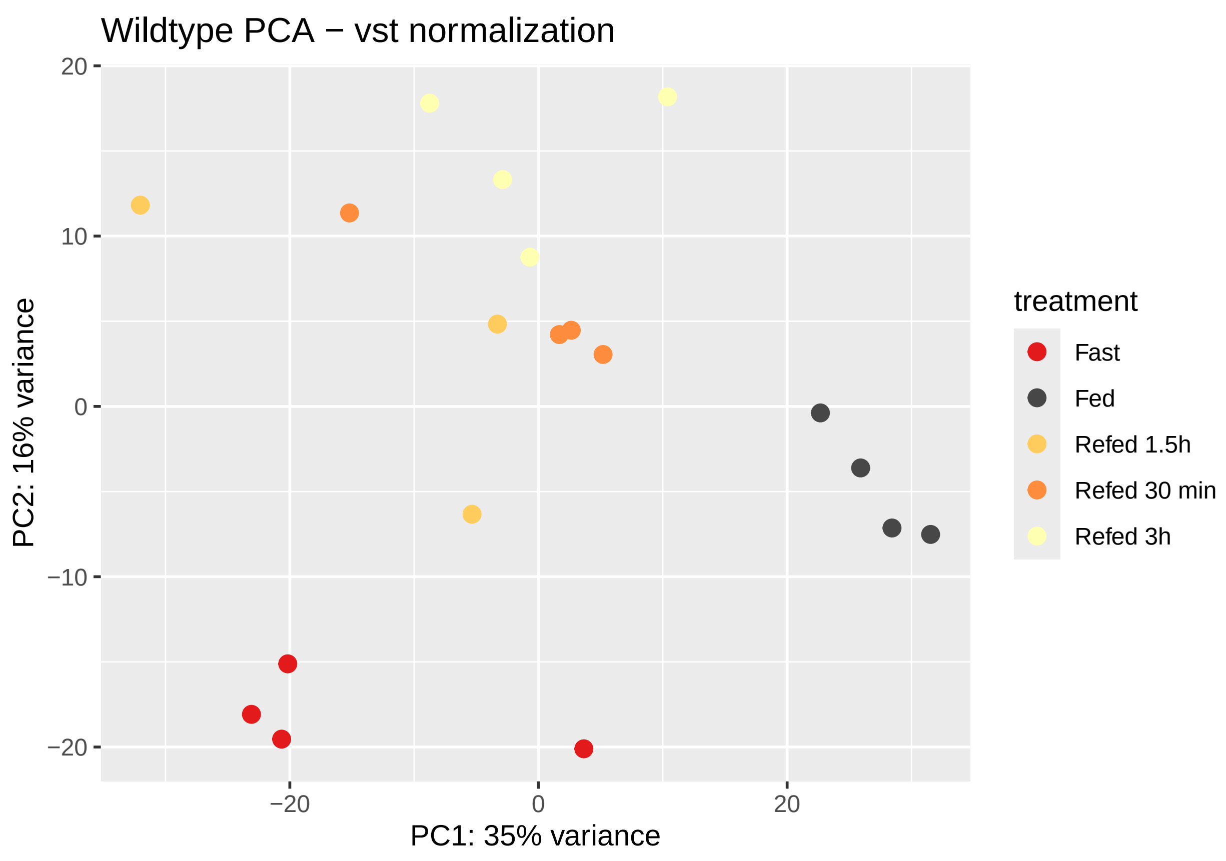
```r
vsdwt <- vst(ddswt, blind = TRUE)
rldwt <- rlog(ddswt)
```

```r
PCAwt <- plotPCA(vsdwt, intgroup='treatment', returnData=TRUE, ntop = 1000)
```

```
## using ntop=1000 top features by variance
```

```r
percentVar <- round(100 * attr(PCAwt, "percentVar"))
pca_wt <- ggplot(PCAwt, aes(x=PC1, y=PC2,color=treatment))+
  geom_point(size=2, stroke =1)+
  scale_color_manual(values = c("#e31a1c", "gray28", "#fecc5c","#fd8d3c","#ffffb2"))+
  xlab(paste0("PC1: ",percentVar[1],"% variance")) +
  ylab(paste0("PC2: ",percentVar[2],"% variance"))+
  ggtitle("Wildtype PCA - vst normalization")

pca_wt
```



```r
ggsave("images/wt_pca-vst.png")
```

```
## Saving 6.5 x 4.5 in image
```

```r
# KO -PCA
ddsko <- DESeqDataSetFromMatrix(countData = rawcountsko, colData = splanko, design = ~ treatment)
```

```
## converting counts to integer mode

## Warning in DESeqDataSet(se, design = design, ignoreRank): some variables in
## design formula are characters, converting to factors

##   Note: levels of factors in the design contain characters other than
##   letters, numbers, '_' and '.'. It is recommended (but not required) to use
##   only letters, numbers, and delimiters '_' or '.', as these are safe characters
##   for column names in R. [This is a message, not a warning or an error]
```

```r
ddsko <- estimateSizeFactors(ddsko)
```

```
##   Note: levels of factors in the design contain characters other than
##   letters, numbers, '_' and '.'. It is recommended (but not required) to use
##   only letters, numbers, and delimiters '_' or '.', as these are safe characters
##   for column names in R. [This is a message, not a warning or an error]
```

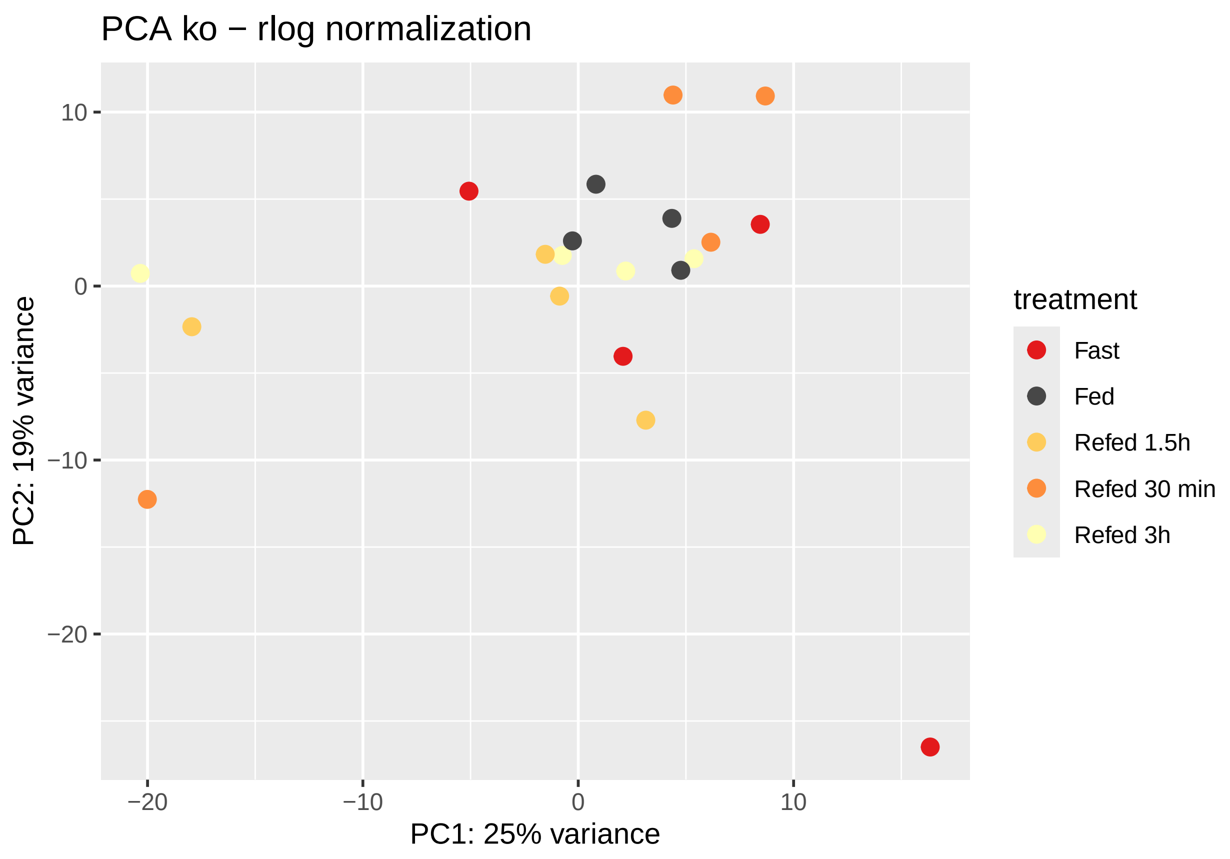
```r
vsdko <- vst(ddsko)
rldko <- rlog(ddsko)
```

```r
PCAko <- plotPCA(rldko, intgroup='treatment', returnData=TRUE, ntop = 1000)
```

```
## using ntop=1000 top features by variance
```

```r
percentVar <- round(100 * attr(PCAko, "percentVar"))
pca_ko <- ggplot(PCAko, aes(x=PC1, y=PC2,color=treatment))+
   geom_point(size=2, stroke =1)+
  scale_color_manual(values = c("#e31a1c", "gray28", "#fecc5c","#fd8d3c","#ffffb2"))+
  xlab(paste0("PC1: ",percentVar[1],"% variance")) +
  ylab(paste0("PC2: ",percentVar[2],"% variance"))+
  ggtitle("PCA ko - rlog normalization")

pca_ko
```



```r
ggsave("images/ko_pca-rlog.png")
```

```
## Saving 6.5 x 4.5 in image
```