Downstream Analysis of Transcriptomic Data

[Attribution: Modified from cluster analysis tutorial by Jennifer Bryan and Erica Acton & the GOseq tutorial by Matthew D. Young, Nadia Davidson, Alicia Oshlak, Matthew Wakefield and Gorden Smyth]

Gene Ontology enrichment analysis

GOseq

Gene ontology aalysis on your RNAseq data can be performed by GOseq and reguires a named vector with the following properties:

1. Measured genes: all genes for which RNA-seq data was gathered for your experiment. Each element of your vector should be named by a unique gene identifier.
2. Differentially expressed genes: each element of your vector should be either a 1 or a 0, where 1 indicates that the gene is differentially expressed and 0 that it is not.

```
#read in count data file. Androgen treated and untreated LNCAP cells [L
i et al., 2008].
table.summary=read.table(system.file("extdata","Li_sum.txt",package="go
seq"), sep="\t",header=TRUE,stringsAsFactors=FALSE)
counts <- table.summary[,-1]
head(counts)

##   lane1 lane2 lane3 lane4 lane5 lane6 lane8
## 1     0     0     0     0     0     0     0
## 2     0     0     0     0     0     0     0
## 3     0     0     0     0     0     0     0
## 4     0     0     0     0     0     0     0
## 5     0     0     0     0     0     0     0
## 6     0     0     0     0     0     0     0

rownames(counts) <- table.summary[,1]
grp <- factor(rep(c("Control","Treated"),times=c(4,3)))
summarized <- DGEList(counts,lib.size=colSums(counts),group=grp)
```

supported genesIDs and genomes

```
#supportedGenomes()
#supportedGeneIDs()
```

Create the named vector

DE analysis using edgeR

```
library(edgeR)
table.summary<-read.table(system.file("extdata","Li_sum.txt", package="
goseq"), sep="\t", header=TRUE, stringsAsFactors=FALSE)
counts <- table.summary[,-1]
rownames(counts)=table.summary[,1]
grp=factor(rep(c("Control", "Treated"), times= c(4,3) ) )
summarized=DGEList(counts,lib.size=colSums(counts),group=grp)

#use edgeR to estimate the biological dispersion and calculate differen
tial expression using a negative #binomial model
#using a negative binomial model
disp=estimateCommonDisp(summarized)
disp$common.dispersion

## [1] 0.05688364

#str(disp)
tested=exactTest(disp)
topTags(tested)

## Comparison of groups:  Treated-Control
##                      logFC   logCPM       PValue          FDR
## ENSG00000127954 11.557868 6.680748 2.574972e-80 1.274766e-75
## ENSG00000151503  5.398963 8.499530 1.781732e-65 4.410322e-61
## ENSG00000096060  4.897600 9.446705 7.983756e-60 1.317479e-55
## ENSG00000091879  5.737627 6.282646 1.207655e-54 1.494654e-50
## ENSG00000132437 -5.880436 7.951910 2.950042e-52 2.920896e-48
## ENSG00000166451  4.564246 8.458467 7.126763e-52 5.880292e-48
## ENSG00000131016  5.254737 6.607957 1.066807e-51 7.544766e-48
## ENSG00000163492  7.085400 5.128514 2.716461e-45 1.681014e-41
## ENSG00000113594  4.051053 8.603264 9.272066e-44 5.100255e-40
## ENSG00000116285  4.108522 7.864773 6.422468e-43 3.179507e-39
```

Format into a vector

```
genes=as.integer(p.adjust(tested$table$PValue[tested$table$logFC!=0],me
thod="BH")<.05)
names(genes)=row.names(tested$table[tested$table$logFC!=0,])
table(genes)

## genes
##     0     1
## 19535  3208

head(supportedGenomes())[,1:5]
```

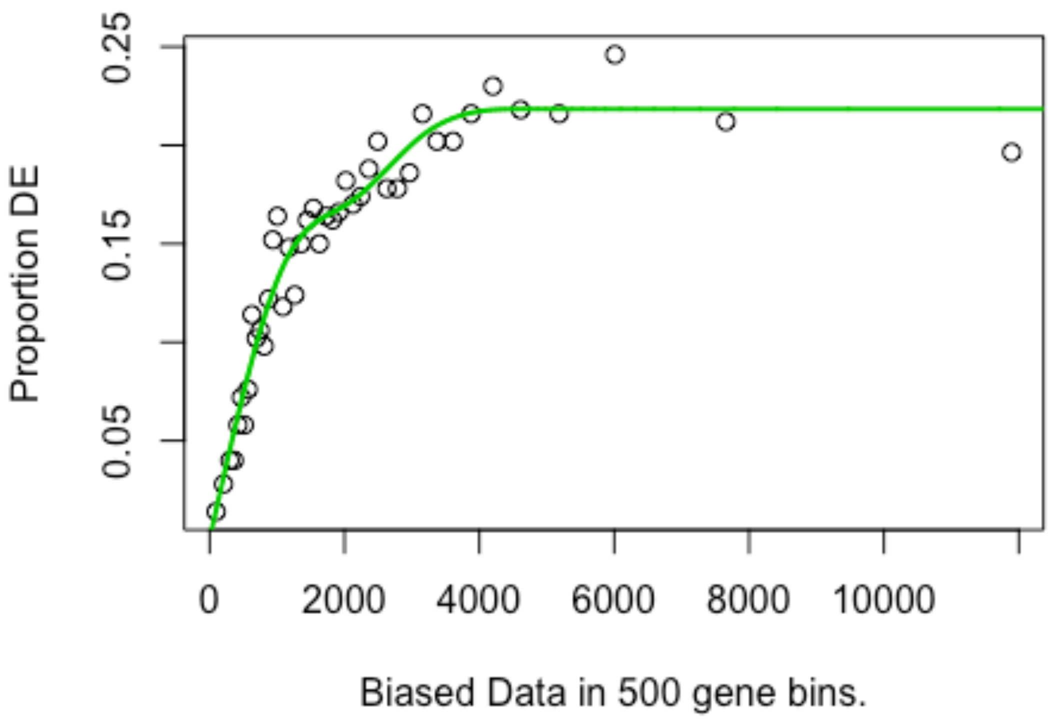
```
##         db species      date                                name
## 1     hg38   Human Dec. 2013  Genome Reference Consortium GRCh38
## 2     hg19   Human Feb. 2009  Genome Reference Consortium GRCh37
## 3     hg18   Human Mar. 2006                      NCBI Build 36.1
## 4     hg17   Human  May 2004                        NCBI Build 35
## 5     hg16   Human Jul. 2003                        NCBI Build 34
## 6 vicPac2  Alpaca Mar. 2013 Broad Institute Vicugna_pacos-2.0.1
##
AvailableGeneIDs
## 1
## 2                                                          ccdsGene,ensG
ene,exoniphy,geneSymbol,knownGene,nscanGene,refGene,xenoRefGene
## 3   acembly,acescan,ccdsGene,ensGene,exoniphy,geneSymbol,geneid,gens
can,knownGene,knownGeneOld3,refGene,sgpGene,sibGene,xenoRefGene
## 4 acembly,acescan,ccdsGene,ensGene,exoniphy,geneSymbol,geneid,gensca
n,knownGene,refGene,sgpGene,vegaGene,vegaPseudoGene,xenoRefGene
## 5                                                          acembly,ensGe
ne,exoniphy,geneSymbol,geneid,genscan,knownGene,refGene,sgpGene
## 6
```

```
head(supportedGeneIDs(),n=12)[,1:4]
```

```
##                 db          track            subtrack
## 1        knownGene      UCSC Genes                <NA>
## 2    knownGeneOld3 Old UCSC Genes                 <NA>
## 3         ccdsGene           CCDS                 <NA>
## 4          refGene   RefSeq Genes                 <NA>
## 5      xenoRefGene   Other RefSeq                 <NA>
## 6         vegaGene     Vega Genes Vega Protein Genes
## 7   vegaPseudoGene     Vega Genes   Vega Pseudogenes
## 8          ensGene  Ensembl Genes                 <NA>
## 9          acembly  AceView Genes                 <NA>
## 10         sibGene      SIB Genes                 <NA>
## 11  nscanPasaGene          N-SCAN     N-SCAN PASA-EST
## 12       nscanGene          N-SCAN              N-SCAN
##                   GeneID
## 1         Entrez Gene ID
## 2
## 3
## 4         Entrez Gene ID
## 5
## 6   HAVANA Pseudogene ID
## 7   HAVANA Pseudogene ID
## 8        Ensembl gene ID
## 9
## 10
## 11
## 12
```

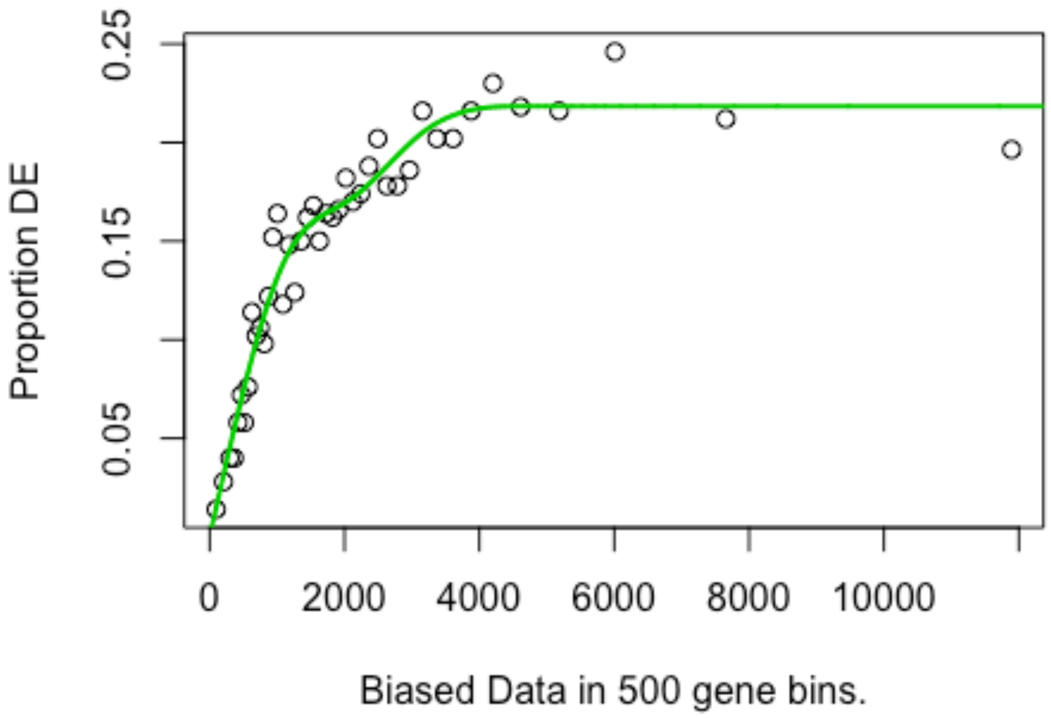
```
pwf=nullp(genes,"hg19","ensGene")
```

```
## Loading hg19 length data...

## Warning in pcls(G): initial point very close to some inequality
## constraints
```



```
head(pwf)

##                 DEgenes bias.data        pwf
## ENSG00000230758       0       247 0.03757470
## ENSG00000182463       0      3133 0.20436865
## ENSG00000124208       0      1978 0.16881769
## ENSG00000230753       0       466 0.06927243
## ENSG00000224628       0      1510 0.15903532
## ENSG00000125835       0       954 0.12711992

#GO category over representation amongst DE genes
GO.wall=goseq(pwf,"hg19","ensGene")
head(GO.wall)

#use random sampling to generate the null distribution for category mem
bership.
GO.samp=goseq(pwf,"hg19","ensGene",method="Sampling",repcnt=1000)
head(GO.samp)
```

```
#Limiting analysis to a single GO category
GO.MF=goseq(pwf,"hg19","ensGene",test.cats=c("GO:MF"))
head(GO.MF)

#FDR correction
enriched.GO=GO.wall$category[p.adjust(GO.wall$over_represented_pvalue,m
ethod="BH")<.05]
head(enriched.GO)

#Get information about each enriched term can be obtained from the GO.d
b

for(go in enriched.GO[1:5]){
  print(GOTERM[[go]])
  cat("--------------------------------------\n")
  }

#KEGG pathway analysis
pwf=nullp(genes,"hg19","ensGene")

## Warning in pcls(G): initial point very close to some inequality
## constraints
```



```
KEGG=goseq(pwf,"hg19","ensGene",test.cats="KEGG")
head(KEGG)
```

Cluster Analysis

Load photoRec dataset.

The aim of the study was to "generate gene expression profiles of purified photoreceptors at distinct developmental stages and from different genetic backgrounds". The experimental units were mice and the microarray platform was Affymetrix mouse genomic expression array 430 2.0.

publication: [http://www.ncbi.nlm.nih.gov/pubmed/16505381] GEO Acession: GSE4051

```
# original normalised photo receptor gene expression data
# file contains expression values of 29949 probes from photoreceptor ce
lls in 39 mice samples.
prDat <- read.table("~/Course_Materials?Day4/RNAseq/GSE4051_data.tsv",
header=TRUE, row.names=1)

#str(prDat, max.level=0)
# metadata
# describes the experimental condition for each sample. Gene expression
was studied at 5 different  developmental stages: day 16 of embryonic d
evelopment (E16), postnatal days 2,6 and 10 (P2, P6 and p10) as well as
4 weeks (4_weeks). Each of these 5 experimental conditions was studied
in wild type mice and Nrl knockout mice.

prDes <- readRDS("~/Course_Materials?Day4/RNAseq/GSE4051_design.rds")
#str(prDes)
sort(unique(prDes$sidNum))

##  [1]  1  2  3  4  5  6  7  8  9 10 11 12 13 14 15 16 17 18 19 20
## [21] 21 22 23 24 25 26 27 28 29 30 31 32 33 34 35 36 37 38 39

unique(prDes$devStage)

## [1] E16     P2      P6      P10     4_weeks
## Levels: E16 P2 P6 P10 4_weeks

unique(prDes$gType)

## [1] wt    NrlKO
## Levels: wt NrlKO
```

Rescale the rows.

```
#scale and transpose rows
sprDat <- (t(scale(t(prDat))))
```

```
#str(sprDat, max.level=0, give.attr=FALSE)

round(data.frame(avgBefore=rowMeans(head(prDat)),
                 avgAfter=rowMeans(head(sprDat)),
                 varBefore=apply(head(prDat),1,var),
                 varAfter=apply(head(sprDat),1,var)),2)

##                avgBefore avgAfter varBefore varAfter
## 1415670_at          7.22        0      0.02        1
## 1415671_at          9.37        0      0.35        1
## 1415672_at          9.70        0      0.15        1
## 1415673_at          8.42        0      0.03        1
## 1415674_a_at        8.47        0      0.02        1
## 1415675_at          9.67        0      0.03        1
```

Clustering

Hierarchical Clustering

Compute pairwise distances.

```
#distance metric used is "Euclidian"
pr.dis <- dist(t(sprDat), method="euclidian")
```

Create a new factor representing the interation of gType (genotype) and devStage (development stage).

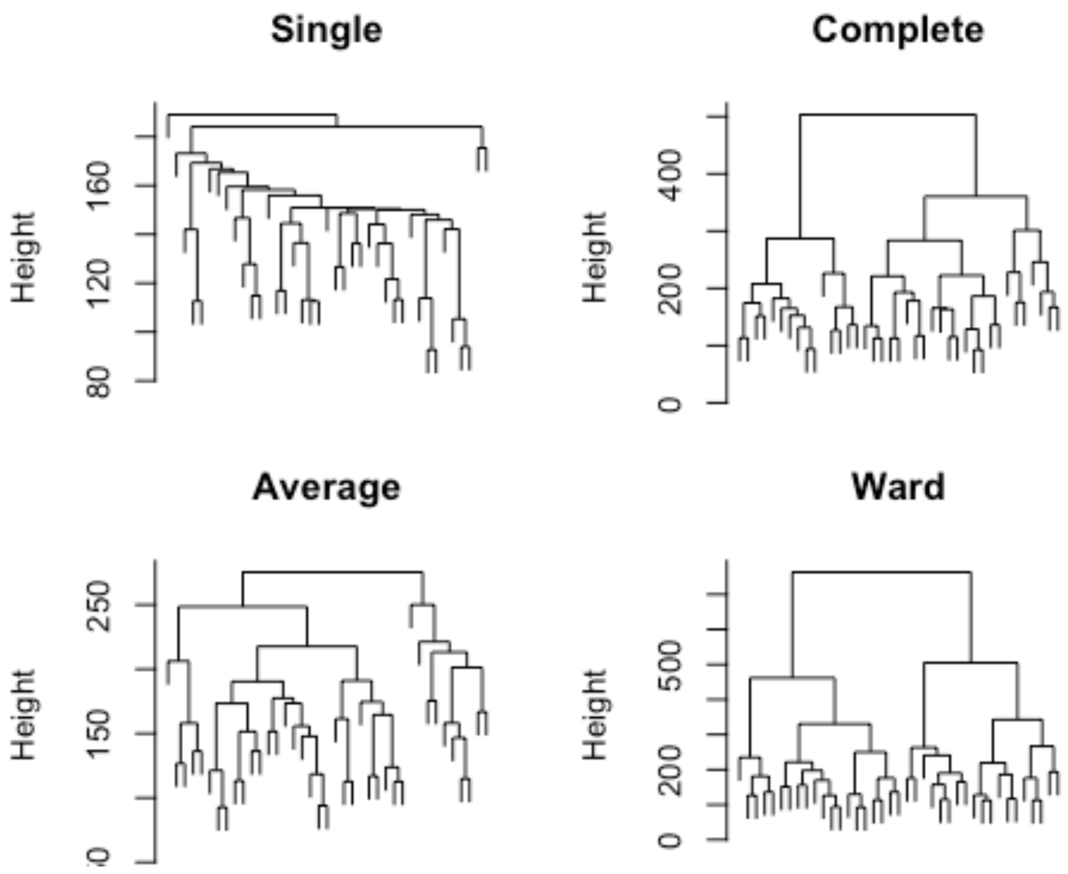
```
prDes$grp <- with(prDes, interaction(gType, devStage))
summary(prDes$grp)

##        wt.E16     NrlKO.E16         wt.P2      NrlKO.P2 
##             4             3             4             4 
##         wt.P6      NrlKO.P6        wt.P10     NrlKO.P10 
##             4             4             4             4 
##    wt.4_weeks NrlKO.4_weeks 
##             4             4
```

Compute hierarchical clustering using different linkage types.

```
pr.hc.s <- hclust(pr.dis, method='single')
pr.hc.c <- hclust(pr.dis, method='complete')
pr.hc.a <- hclust(pr.dis, method='average')
pr.hc.w <- hclust(pr.dis, method='ward.D2')
```

Plot the different hierarchical clustering types.

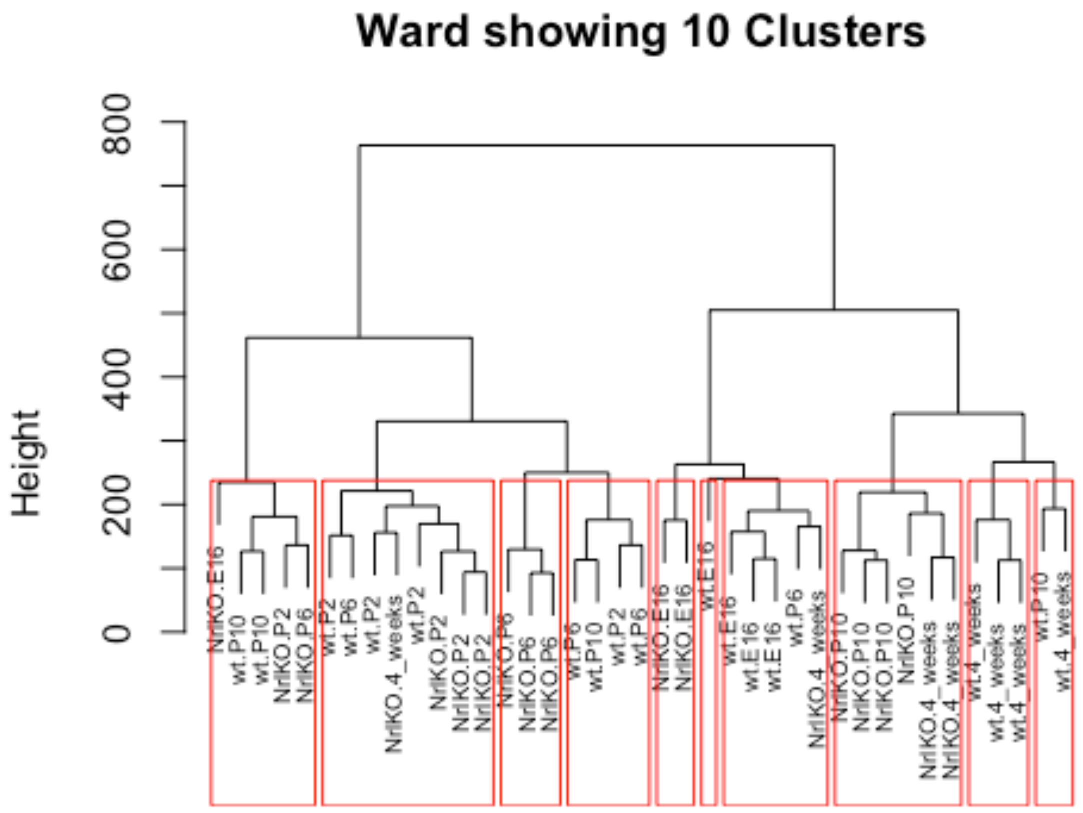
```
op <- par(mar=c(0,4,4,2),mfrow=c(2,2))
plot(pr.hc.s, labels=FALSE, main="Single", xlab="")
plot(pr.hc.c, labels=FALSE, main="Complete", xlab="")
plot(pr.hc.a, labels=FALSE, main="Average", xlab="")
plot(pr.hc.w, labels=FALSE, main="Ward", xlab="")
```



```
par(op)
```

K-Mean clustering

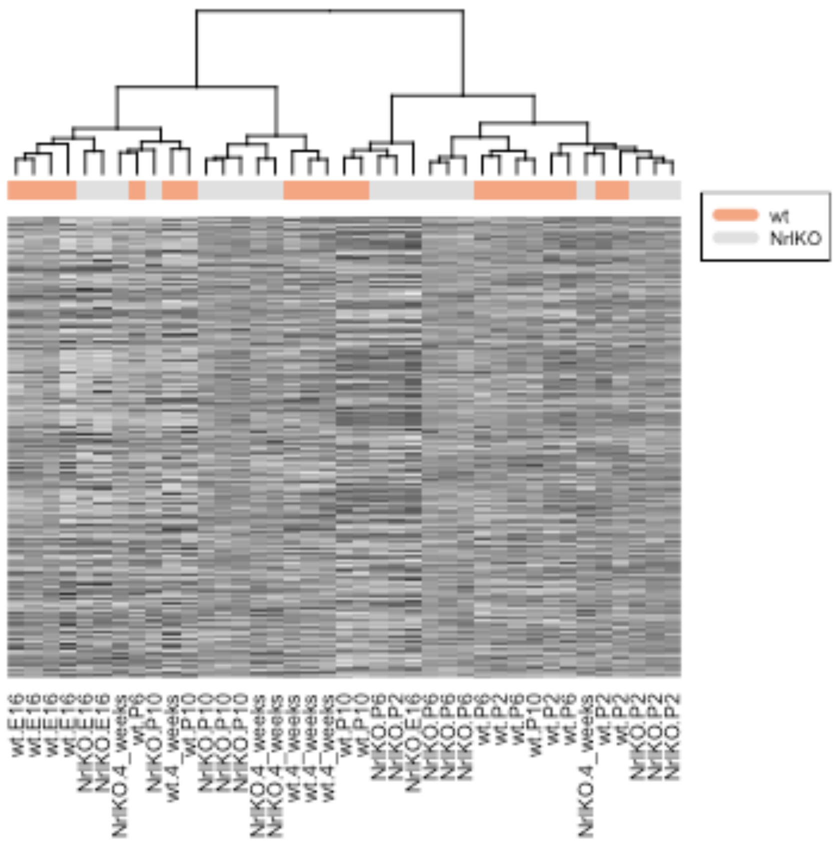
```
#Identify 10 clusters.
op <- par(mar=c(1,4,4,1))
#Ward's minimum variance method aims at finding compact, spherical clus
ters.
plot(pr.hc.w, labels=prDes$grp, cex=0.6, main="Ward showing 10 Clusters
")
rect.hclust(pr.hc.w, k=10)
```



```
par(op)
```

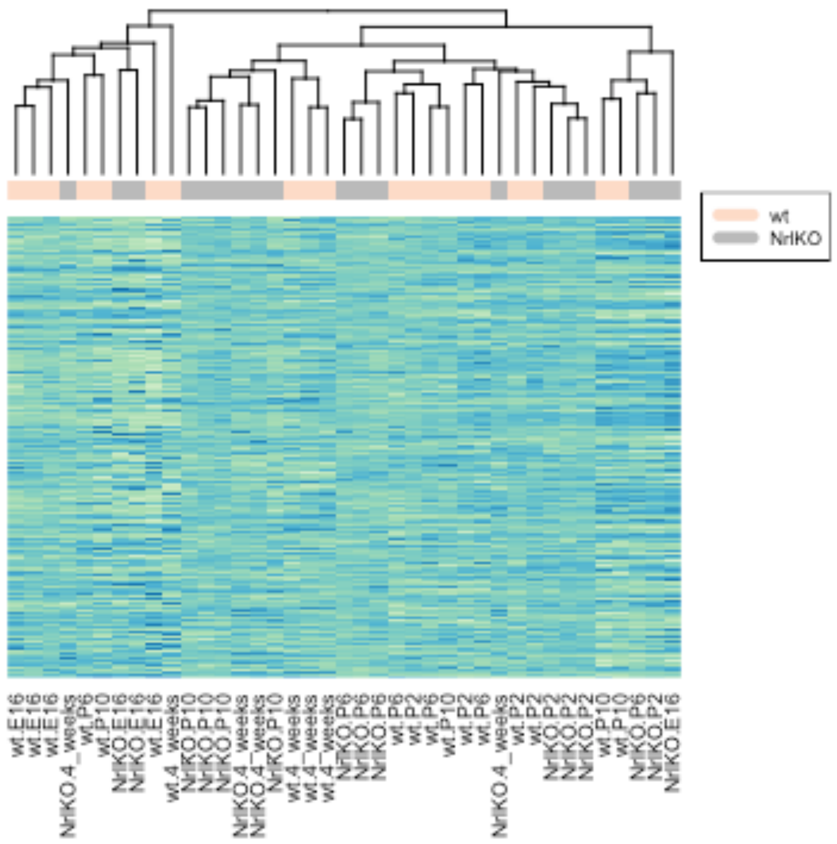
Heatmap example

```
GreyFun <- colorRampPalette(brewer.pal(n=9, "Greys"))
gTypeCols <- brewer.pal(n=11, "RdGy")[c(4,7)]


heatmap(as.matrix(sprDat), Rowv=NA, col=GreyFun (256), hclustfun= funct
ion (x) hclust(x, method='ward.D'),
        scale="none", labCol=prDes$grp, labRow=NA, margins=c(8,1), ColS
ideColor=gTypeCols[unclass(prDes$gType)])
legend("topright", legend=levels(prDes$gType), col=gTypeCols, lty=1, lw
d=5, cex=0.5)
```



Playing with heatmaps.

```
GnBuFun <- colorRampPalette(brewer.pal(n=9, "GnBu"))
gTypeCols <- brewer.pal(n=9, "RdGy")[c(4,7)]
heatmap(as.matrix(sprDat), Rowv=NA, col=GnBuFun (256), hclustfun= funct
ion (x) hclust(x, method='average'),
        scale="none", labCol=prDes$grp, labRow=NA, margins=c(8,1), ColS
ideColor=gTypeCols[unclass(prDes$gType)])
legend("topright", legend=levels(prDes$gType), col=gTypeCols, lty=1, lw
d=5, cex=0.5)
```



K-means Clustering

```
#Choose parameters, including k.
set.seed(31)
k <- 5
pr.km <- kmeans(t(sprDat), centers=k, nstart=50)
```

Look at the sum of squares of each cluster.

```
pr.km$withinss

## [1] 120153.14  78227.41 110209.42 100196.88 133036.47
```

Look at the composition of each cluster.

```
pr.kmTable <- data.frame(devStage=prDes$devStage, cluster=pr.km$cluster
)
prTable <- xtable(with(pr.kmTable, table(devStage, cluster)),
                  caption='Number of samples from each developmental st
age within each k-means cluster')

#align(prTable) <- "lccccc"
#print(prTable, type='html', caption.placement='top')
```

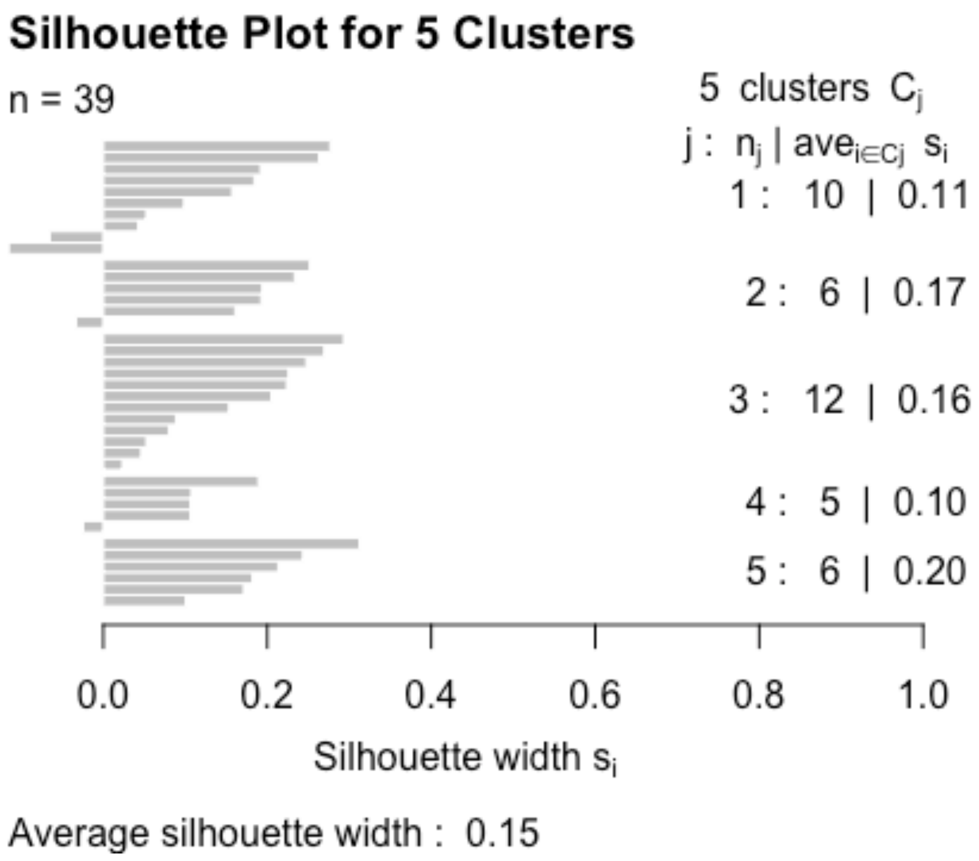
PAM Algorithm

```
pr.pam <- pam(pr.dis, k = k)
pr.pamTable <- data.frame(devStage = prDes$devStage, cluster = pr.pam$c
lustering)
pamTable <- xtable(with(pr.pamTable, table(devStage, cluster)),
    caption = "Number of samples from each developmental stage within e
ach PAM cluster")
# align(pamTable) <- 'lccccc' print(pamTable, type='html',
# caption.placement='top')
summary(pr.pam)

## Medoids:
##      ID
## [1,] "3"  "Sample_22"
## [2,] "15" "Sample_8"
## [3,] "13" "Sample_3"
## [4,] "35" "Sample_39"
## [5,] "28" "Sample_13"
## Clustering vector:
## Sample_20 Sample_21 Sample_22 Sample_23 Sample_16 Sample_17
##         1         1         1         1         2         1
##  Sample_6 Sample_24 Sample_25 Sample_26 Sample_27 Sample_14
##         1         3         3         3         3         3
##  Sample_3  Sample_5  Sample_8 Sample_28 Sample_29 Sample_30
##         3         3         2         2         1         1
## Sample_31  Sample_1 Sample_10  Sample_4  Sample_7 Sample_32
##         3         3         1         3         2         4
## Sample_33 Sample_34 Sample_35 Sample_13 Sample_15 Sample_18
##         2         2         3         5         5         5
## Sample_19 Sample_36 Sample_37 Sample_38 Sample_39 Sample_11
##         5         4         4         4         4         5
## Sample_12  Sample_2  Sample_9
##         3         5         1
## Objective function:
##    build     swap
## 136.6555 135.9811
##
## Numerical information per cluster:
##      size max_diss  av_diss diameter separation
## [1,]   10 223.8375 150.3397 284.1253   113.8168
## [2,]    6 179.4329 136.7742 226.2333   150.4049
## [3,]   12 173.7251 136.0243 221.3962   113.8168
## [4,]    5 206.8624 133.4915 254.6634   166.7790
## [5,]    6 151.4990 113.2456 209.3239   150.9273
##
## Isolated clusters:
##  L-clusters: character(0)
##  L*-clusters: character(0)
##
```

```
## Silhouette plot information:
##           cluster neighbor   sil_width
## Sample_23       1        3  0.27682006
## Sample_21       1        4  0.26316139
## Sample_6        1        5  0.19195873
## Sample_22       1        3  0.18419344
## Sample_17       1        5  0.15715159
## Sample_9        1        5  0.09849296
## Sample_29       1        4  0.05244008
## Sample_20       1        5  0.04261895
## Sample_10       1        5 -0.06534635
## Sample_30       1        3 -0.11538428
## Sample_34       2        3  0.25189437
## Sample_16       2        3  0.23344183
## Sample_7        2        3  0.19390217
## Sample_8        2        3  0.19259179
## Sample_33       2        3  0.16094180
## Sample_28       2        3 -0.03304892
## Sample_3        3        5  0.29346437
## Sample_31       3        5  0.26948006
## Sample_24       3        5  0.24757941
## Sample_35       3        5  0.22591215
## Sample_14       3        5  0.22389293
## Sample_25       3        1  0.20491307
## Sample_5        3        2  0.15337292
## Sample_27       3        2  0.08855662
## Sample_1        3        5  0.07988790
## Sample_26       3        2  0.05257790
## Sample_4        3        5  0.04628817
## Sample_12       3        5  0.02340497
## Sample_36       4        1  0.18922941
## Sample_32       4        5  0.10747139
## Sample_39       4        5  0.10632767
## Sample_38       4        5  0.10632739
## Sample_37       4        5 -0.02456609
## Sample_13       5        3  0.31248208
## Sample_2        5        4  0.24354786
## Sample_15       5        3  0.21336914
## Sample_11       5        4  0.18197474
## Sample_18       5        1  0.17144028
## Sample_19       5        3  0.10056328
## Average silhouette width per cluster:
## [1] 0.10861066 0.16662051 0.15911087 0.09695795 0.20389623
## Average silhouette width of total data set:
## [1] 0.1462392
##
## Available components:
## [1] "medoids"    "id.med"     "clustering" "objective"
## [5] "isolation"  "clusinfo"   "silinfo"    "diss"
## [9] "call"
```

Silhouette plot.

```
op <- par(mar=c(5,1,4,4))
plot(pr.pam, main="Silhouette Plot for 5 Clusters")
```



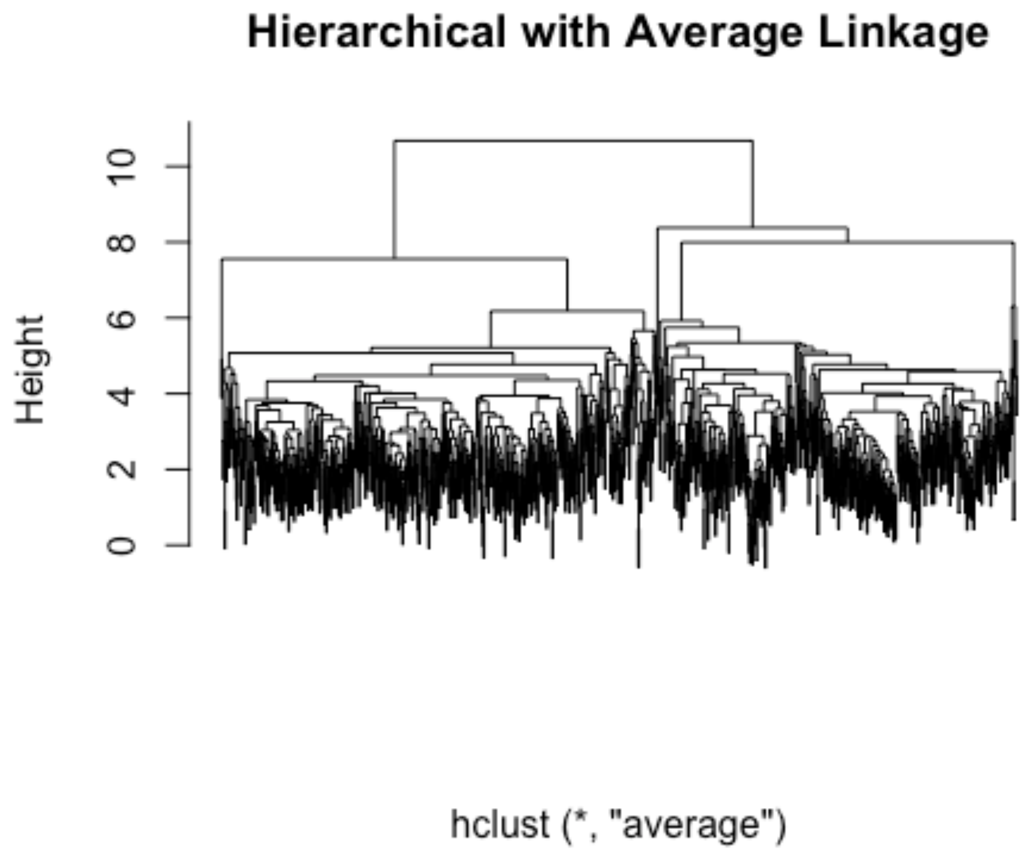
```
par(op)
```

Gene Clustering

```
#Start with the top 972 genes that showed differential expression acros
s the different developmental stage (BH adjusted p value < 10-5).
devDes <- model.matrix(~devStage, prDes)
fit <- lmFit(prDat, devDes)
ebFit <- eBayes(fit)
topDat <- topTable(ebFit, coef = grep("devStage", colnames(coef(ebFit))
), p.value=1e-05, n=972)
ttopDat <- sprDat[rownames(topDat), ]
head(ttopDat)

##              Sample_20  Sample_21  Sample_22  Sample_23
## 1440645_at -0.8052450 -0.8282436 -0.7585291 -0.8742407
## 1421084_at -1.2108650 -1.5209099 -0.9758656 -1.5028990
## 1451590_at -1.1387139 -1.1429012 -1.1308627 -1.0889898
## 1428680_at -0.7774426 -0.9124304 -0.8369714 -0.7631892
## 1450215_at -1.5844357 -1.5933804 -0.9616003 -1.3885933
## 1416041_at -0.4637899 -0.8668119 -0.5683785 -0.6406596
##             Sample_16  Sample_17   Sample_6  Sample_24
## 1440645_at -0.3668347 -0.6952831 -0.8462112 -0.5961017
## 1421084_at -0.9286942 -1.5552164 -1.4432915 -0.8845246
## 1451590_at -1.2789880 -1.1413309 -1.3653508 -1.1177774
## 1428680_at -0.3976323 -0.9551904 -0.8294255 -0.8705087
## 1450215_at -1.6178607 -1.5933804 -1.6512857 -0.6457102
## 1416041_at -0.5957577 -0.3712481 -0.6094473 -0.6302555
##             Sample_25  Sample_26  Sample_27  Sample_14
## 1440645_at -0.5551355 -0.5163254 -0.4516419 -0.6248499
## 1421084_at -1.0063126 -0.3879382 -0.9321248 -1.3275071
## 1451590_at -0.9241154 -0.7011424 -0.9963461 -0.9047492
## 1428680_at -0.6105944 -0.4110473 -0.5820877 -0.5577731
## 1450215_at -0.8490850 -0.7586962 -0.8919256 -0.5134224
## 1416041_at -0.3646771 -0.6653009 -0.5333331 -0.7271779
##              Sample_3   Sample_5   Sample_8   Sample_28
## 1440645_at -0.5738218 -0.5975391 -0.2504044 -0.62197510
## 1421084_at -1.2185840 -1.2949159 -1.0937942 -0.02257411
## 1451590_at -1.0837557 -0.9780267 -0.8518847 -0.53103387
## 1428680_at -0.5996948 -0.5661574 -0.5368123 -0.32552708
## 1450215_at -0.5713278 -0.6071067 -1.2520685 -0.41267658
## 1416041_at -0.8525747 -0.6965132 -0.7096552 -0.50923936
##              Sample_29   Sample_30  Sample_31   Sample_1
## 1440645_at -0.77362196 -0.77865290 -0.5306995 -0.4573915
## 1421084_at  0.42341022  0.31620245  0.2776077  0.3462206
## 1451590_at -0.07409619  0.04157761 -0.2394940  0.1698133
## 1428680_at -0.57034959 -0.68437651 -0.6022101 -0.6500007
## 1450215_at  0.22145741  0.03785514 -0.0859587  0.3499790
## 1416041_at -0.52238138 -0.31922760 -0.1379772 -0.7156786
##             Sample_10   Sample_4   Sample_7   Sample_32
## 1440645_at -0.6262873 -0.4624225 -0.2633411  0.07732534
## 1421084_at  0.3247791  0.3976804 -0.5041514  0.98089063
## 1451590_at  0.2928148  0.2357630 -0.4509520  1.07636088
## 1428680_at -0.6173019 -0.7589971 -0.6214940  0.75772867
## 1450215_at  0.7765012  0.5505292 -0.3712484  1.22373749
## 1416041_at -0.6554444 -0.7824839 -0.6253272 -0.10019389
##             Sample_33   Sample_34  Sample_35 Sample_13
## 1440645_at -0.2417799 -0.24968566 -0.2072820 0.0198289
## 1421084_at  0.8479530  0.89512441  1.0495036 0.8179348
## 1451590_at  0.1159020  0.06931839  1.0030834 0.7466120
## 1428680_at -0.3255271 -0.32385022  0.2379001 0.1087813
## 1450215_at  0.7341314  0.48462066  0.8894872 0.8753640
## 1416041_at -0.6554444 -0.63792167 -0.4172452 0.2130243
##              Sample_15  Sample_18   Sample_19 Sample_36
## 1440645_at -0.07647764 -0.1965015 -0.24249860  2.091857
## 1421084_at  0.94229583  0.8908361  0.85224130  1.040927
## 1451590_at  0.73090971  0.8669965  0.66286630  1.427046
## 1428680_at -0.09411951  0.3603113 -0.03291388  2.107606
## 1450215_at  0.80003995  0.8847795  0.66822293  1.313185
## 1416041_at  0.16155141  0.2738062  0.04491595  2.039766
##            Sample_37 Sample_38 Sample_39 Sample_11 Sample_12
## 1440645_at  1.996269  2.030048  2.003456 1.9919568  1.452209
## 1421084_at  1.242478  1.023774  1.216748 0.9551608  1.053792
## 1451590_at  1.547431  1.421812  1.474153 1.3747051  1.500324
## 1428680_at  1.547533  1.638084  1.358885 2.1830652  2.191450
## 1450215_at  1.294354  1.270815  1.186075 1.0118887  1.120167
## 1416041_at  2.149282  2.187613  2.286178 1.6838358  1.048638
##            Sample_2  Sample_9
## 1440645_at 2.099044 1.3070304
## 1421084_at 1.105252 0.8093582
## 1451590_at 1.385173 0.9978493
## 1428680_at 1.914767 1.4075144
## 1450215_at 0.913026 0.7435469
## 1416041_at 1.787877 1.4976571
```

Hierarchical Custering:

```
geneC.dis <- dist(ttopDat, method='euclidean')
geneC.hc.a <- hclust(geneC.dis, method='average')

plot(geneC.hc.a, labels=FALSE, main="Hierarchical with Average Linkage"
, xlab="")
```



Partitioning:

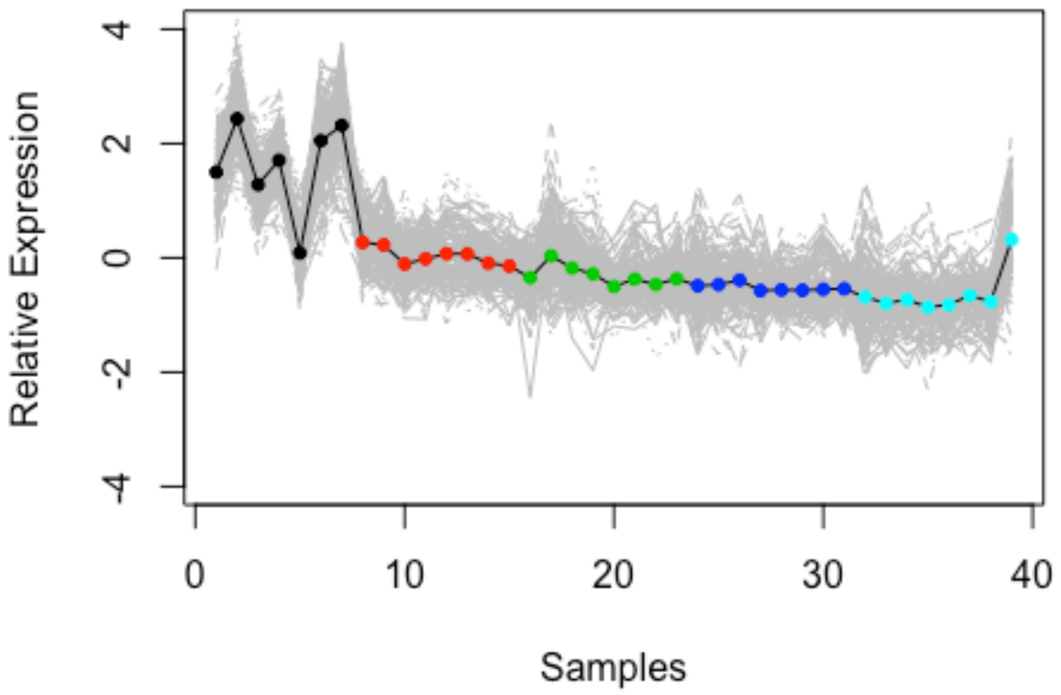
```r
set.seed(1234)
k <- 5
kmeans.genes <- kmeans(ttopDat, centers=k)
```

Choose desired cluster.

```r
clusterNum <- 1
```

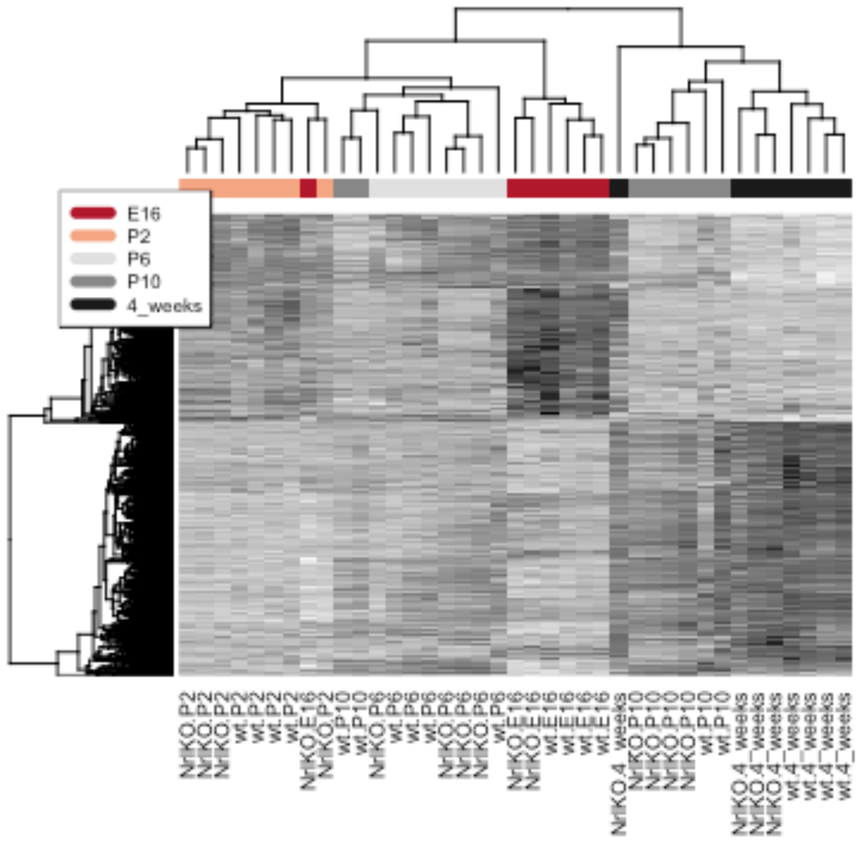
Set up axes. Plot the expression of all the genes in the selected cluster in grey. Add in the cluster center. Colour points to show the developmental stage.

```r
plot(kmeans.genes$centers[clusterNum,], ylim=c(-4,4), type='n',xlab="Sa
mples", ylab="Relative Expression")

matlines(y=t(ttopDat[kmeans.genes$cluster==clusterNum, ]), col='grey')
points(kmeans.genes$centers[clusterNum,],type='l')
points(kmeans.genes$centers[clusterNum,], col=prDes$devStage,pch=20)
```



Heatmaps (hierarchical):

```
devStageCols <- brewer.pal(n=11, "RdGy")[c(2,4,7,9,11)]
heatmap(as.matrix(ttopDat), col=GreyFun (256), hclustfun= function (x)
hclust(x, method='average'), scale="none", labCol=prDes$grp, labRow=NA,
margins=c(8,1), ColSideColor=devStageCols[unclass(prDes$devStage)])
legend("topleft", levels(prDes$devStage), col=devStageCols, lty=1, lwd=
5, cex=0.5)
```



Redefining the Attributes

Define new attributes for a gene and estimate the parameters.

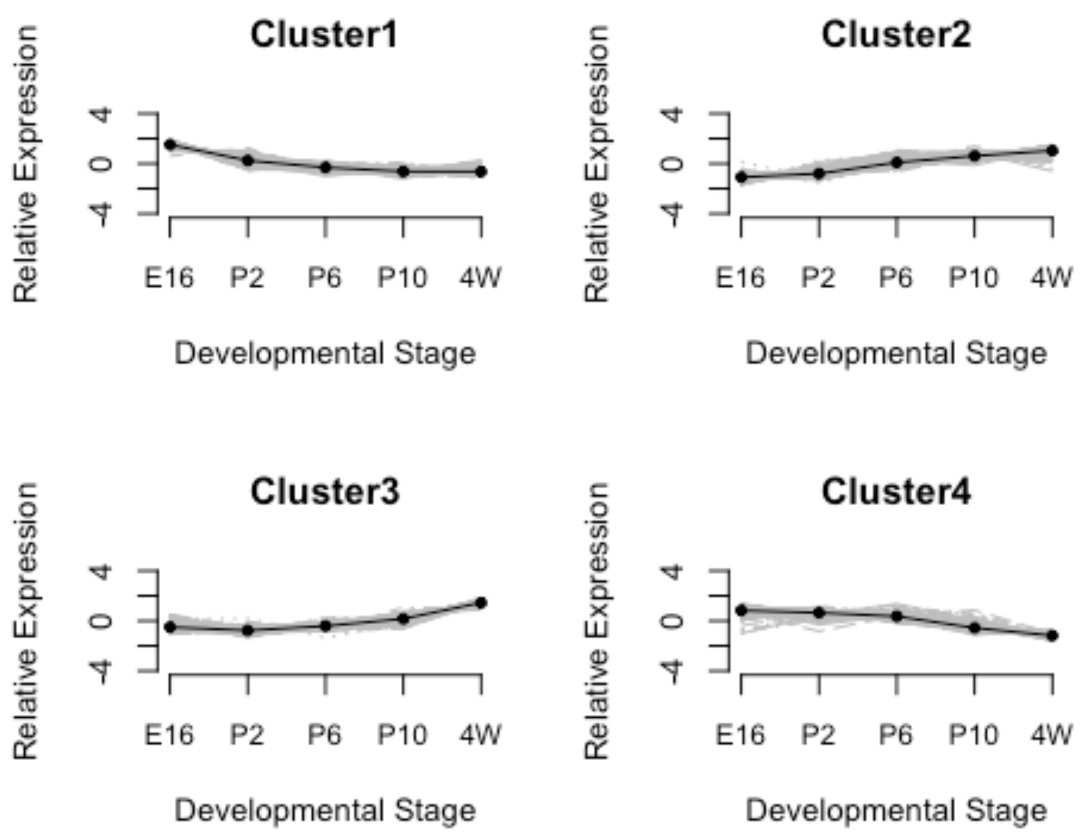
```
annoTopDat <- stack(as.data.frame(ttopDat))
annoTopDat$probeset <- rownames(ttopDat)

annoTopDat <- merge(annoTopDat,prDes,by.x="ind", by.y="sidChar")
devStageAvg <- ddply(annoTopDat, ~probeset, function(x) {
  avgbyDevStage <- aggregate(values ~ devStage,x,mean)$values
  names(avgbyDevStage) <- levels(x$devStage)
  avgbyDevStage
  })

rownames(devStageAvg) <- devStageAvg$probeset
devStageAvg$probeset <- NULL
#str(devStageAvg)
```

Look at the relative expression in all clusters with respect to developmental stage as determined by kmeans.

```
k <- 4
geneDS.km <- kmeans(devStageAvg, centers=k, nstart=50)
clust.centers <- geneDS.km$centers
```

Plot all cluster centers separately.

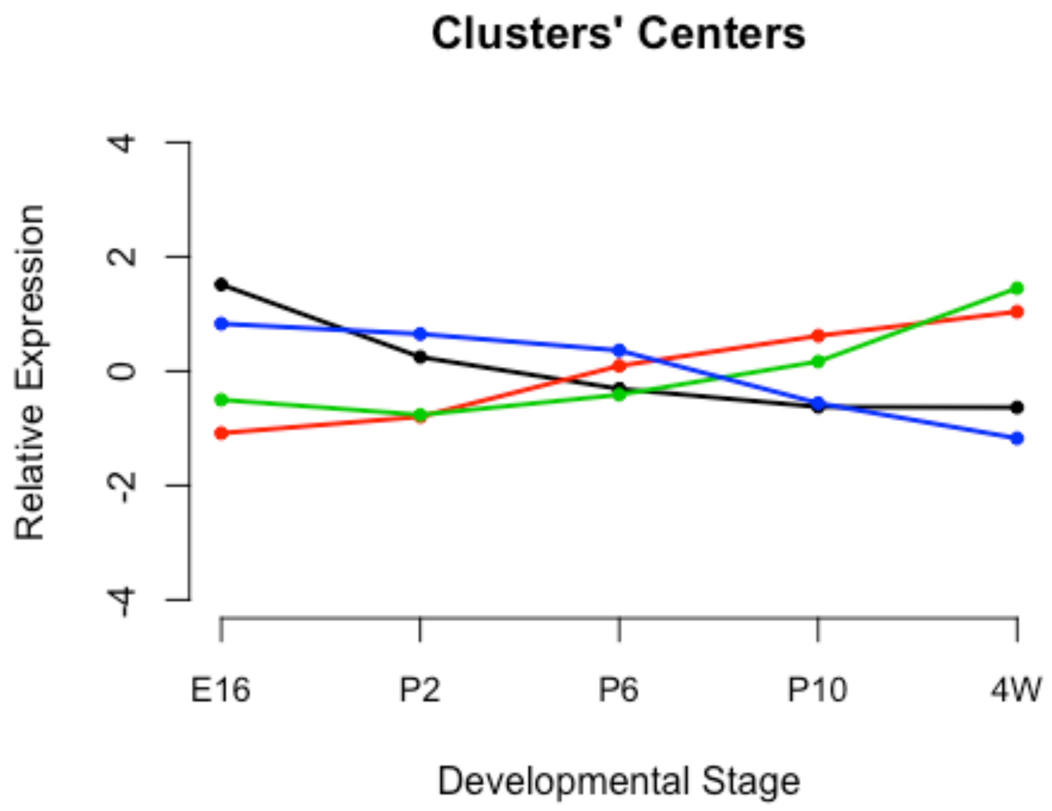
```
op <- par(mfrow=c(2,2))
for(clusterNum in 1:4) {
  plot(clust.centers[clusterNum,], ylim=c(-4,4), type='n',
       xlab="Developmental Stage", ylab="Relative Expression", axes=F,
       main=paste("Cluster", clusterNum, sep=""))
  axis(2)
  axis(1,1:5, c(colnames(clust.centers)[1:4], "4W"), cex.axis=0.9)

  matlines(y=t(devStageAvg[geneDS.km$cluster==clusterNum,]), col='grey'
)
  points(clust.centers[clusterNum,],type='l')
  points(clust.centers[clusterNum,], pch=20)
  }
```



```
par(op)
```

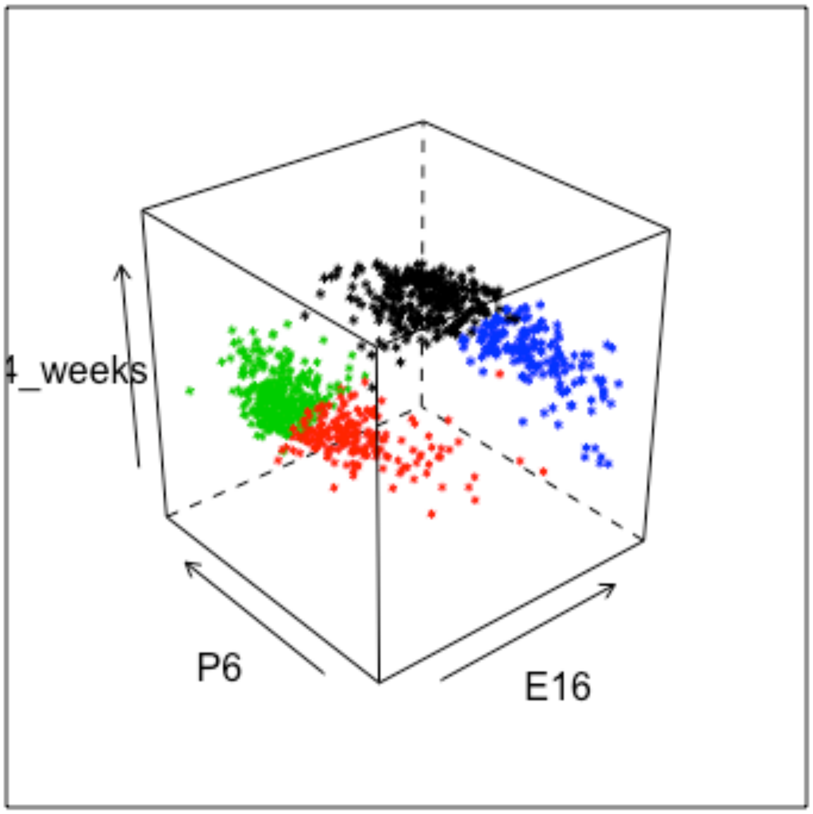
Plot to compare all clusters' centers.

```
plot(clust.centers[clusterNum,], ylim=c(-4,4), type='n',
       xlab="Developmental Stage", ylab="Relative Expression", axes=F,
       main="Clusters' Centers")
  axis(2)
  axis(1,1:5, c(colnames(clust.centers)[1:4], "4W"), cex.axis=0.9)

for(clusterNum in 1:4) {
  points(clust.centers[clusterNum,],type='l', col=clusterNum, lwd=2)
  points(clust.centers[clusterNum,], col=clusterNum, pch=20)
}
```



Plotting 3-dimensional clusters as determined by k-means.

```
cloud(devStageAvg[, "E16"] ~ devStageAvg[, "P6"] * devStageAvg[,
    "4_weeks"], col = geneDS.km$clust, xlab = "E16", ylab = "P6",
    zlab = "4_weeks")
```

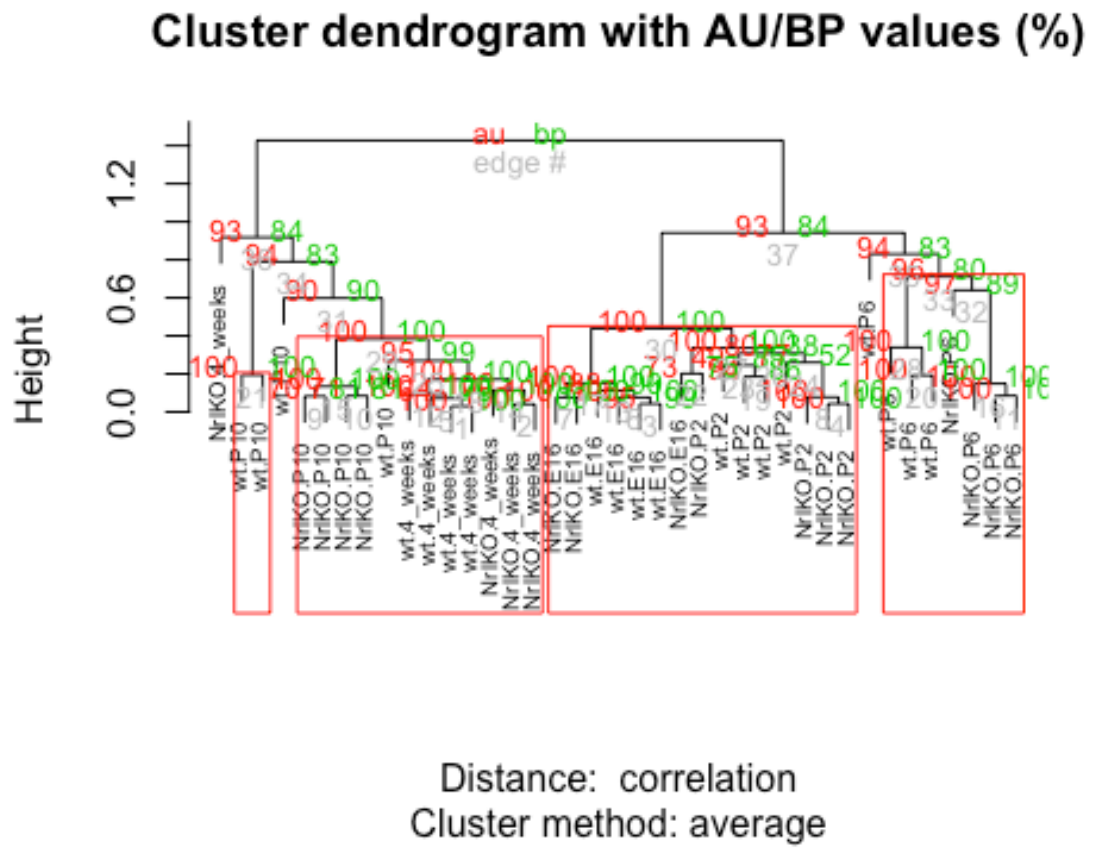


Statistical Measures to Evaluate Clusters

```
#Qualifying cluster membership.
pvc <- pvclust(ttopDat, nboot=100)
```

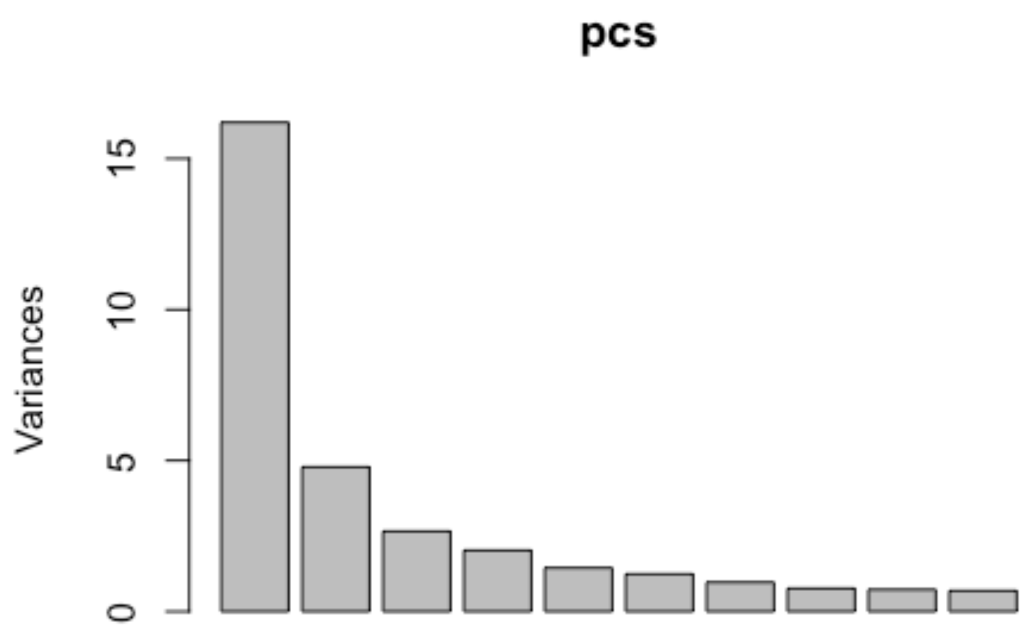
```
## Bootstrap (r = 0.5)... Done.
## Bootstrap (r = 0.6)... Done.
## Bootstrap (r = 0.7)... Done.
## Bootstrap (r = 0.8)... Done.
## Bootstrap (r = 0.9)... Done.
## Bootstrap (r = 1.0)... Done.
## Bootstrap (r = 1.1)... Done.
## Bootstrap (r = 1.2)... Done.
## Bootstrap (r = 1.3)... Done.
## Bootstrap (r = 1.4)... Done.
```

```
plot(pvc, labels=prDes$grp, cex=0.6)
pvrect(pvc, alpha=0.95)
```

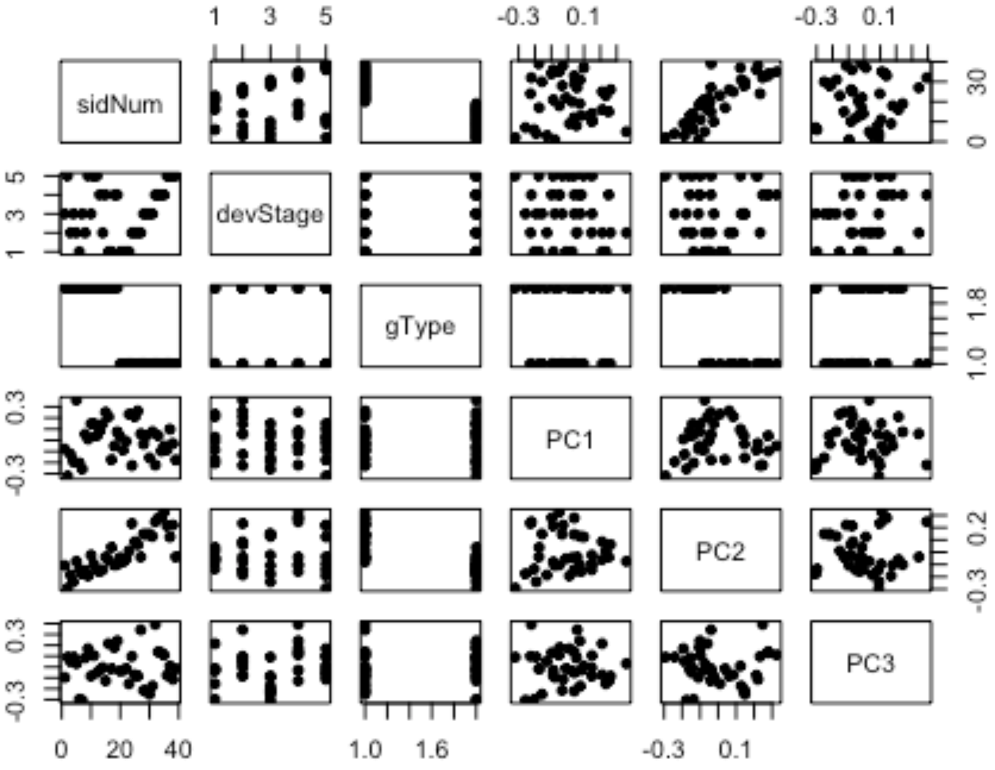


PCA (Principal Components Analysis)

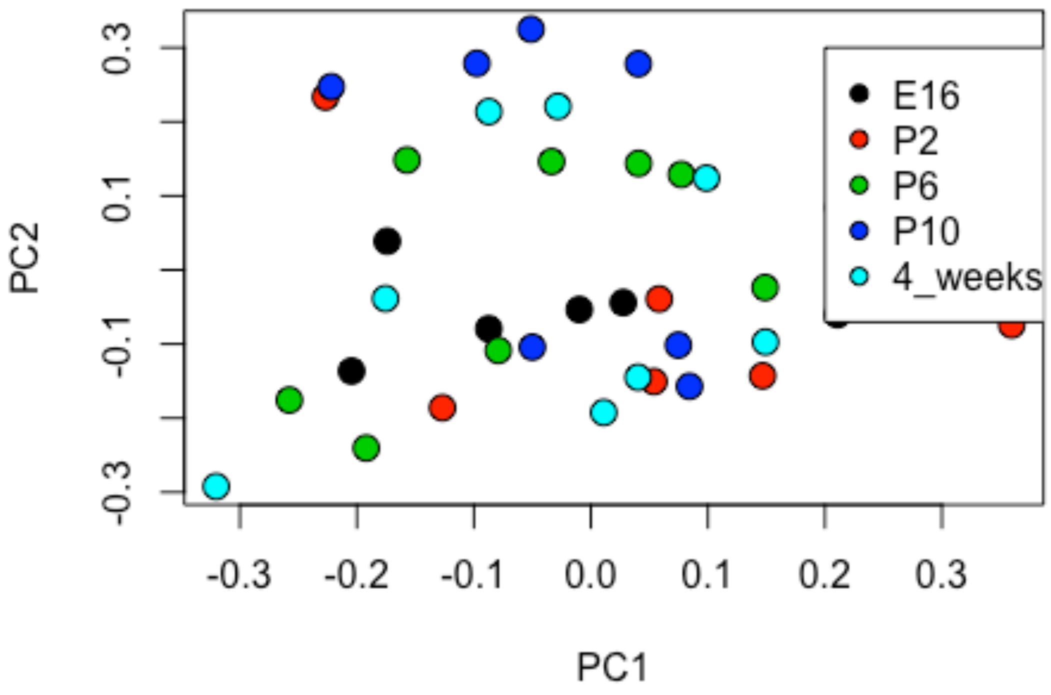
```
#plot PC
pcs <- prcomp(sprDat, center=F, scale=F)
plot(pcs)
```



```r
#scatterplot relating PCs to covariates
prinComp <- cbind(prDes, pcs$rotation[prDes$sidNum,1:10])
plot(prinComp[,c("sidNum", "devStage","gType","PC1","PC2","PC3")], pch=
19, cex=0.8)
```



```
#plot data on 2 PCs, coloured by devStage
plot(prinComp[,c("PC1","PC2")], bg=prDes$devStage, pch=21, cex=1.5)
legend(list(x=0.2, y=0.3), as.character(levels(prDes$devStage)), pch=21
, pt.bg=c(1,2,3,4,5))
```



PC2
PC1
0.3
0.1
-0.1
-0.3
-0.3
-0.2
-0.1
0.0
0.1
0.2
0.3
E16
P2
P6
P10
4_weeks