Tutorial: analysing Microarray data using BioConductor

Guiyuan Lei

Centre for Integrated Systems Biology of Ageing and Nutrition (CISBAN)
School of Mathematics & Statistics
Newcastle University
http://www.mas.ncl.ac.uk/~ngl9/

4 Feb, 2008

Outline of Tutorial

Tutorial on CISBAN Internal wiki, under 'Tools & Resources' page `http://bioinf.ncl.ac.uk/cisban/doku.php?id=resources:resourceshome`

- Introduction to Bioconductor
- Fibroblast data set (Chapter 2) and Yeast time course data (Chapter 3)
  - Pre-process of data
  - Model Fitting for Identifying Differential Expression
  - Network Inference

Outline of Tutorial

Tutorial on CISBAN Internal wiki, under 'Tools & Resources' page `http://bioinf.ncl.ac.uk/cisban/doku.php?id=resources:resourceshome`

- Introduction to Bioconductor
- Fibroblast data set (Chapter 2) and Yeast time course data (Chapter 3)
  - Pre-process of data
  - Model Fitting for Identifying Differential Expression
  - Network Inference

Why Bioconductor

- Bioconductor:
  - open source software for bioinformatics
  - provide innovative methodology for analyzing genomic data
  - using R statistical computing environment
- R: Powerful grapphic feature and cut-edge statistical techniques, around 800 packages available, around 60 basic packages (like affy, limma) in Bioconductor
- Published Papers using Bioconductor `http://www.bioconductor.org/pub`
  - Google Scholar Beta, PubMed, BEPress (Berkeley Electronic Press), Biostatistics, BioMed Central Bioinformatics and Ingenta
  - For example, in Bioinformatics, 161 papers found with 'Bioconductor' in title

Why Bioconductor

- Bioconductor:
  - open source software for bioinformatics
  - provide innovative methodology for analyzing genomic data
  - using R statistical computing environment
- R: Powerful grapphic feature and cut-edge statistical techniques, around 800 packages available, around 60 basic packages (like affy, limma) in Bioconductor
- Published Papers using Bioconductor
  `http://www.bioconductor.org/pub`
  - Google Scholar Beta, PubMed, BEPress (Berkeley Electronic Press), Biostatistics, BioMed Central Bioinformatics and Ingenta
  - For example, in Bioinformatics, 161 papers found with 'Bioconductor' in title

Why Bioconductor

- Bioconductor:
  - open source software for bioinformatics
  - provide innovative methodology for analyzing genomic data
  - using R statistical computing environment
- R: Powerful grapphic feature and cut-edge statistical techniques, around 800 packages available, around 60 basic packages (like affy, limma) in Bioconductor
- Published Papers using Bioconductor `http://www.bioconductor.org/pub`
  - Google Scholar Beta, PubMed, BEPress (Berkeley Electronic Press), Biostatistics, BioMed Central Bioinformatics and Ingenta
  - For example, in Bioinformatics, 161 papers found with 'Bioconductor' in title

Getting Bioconductor and associated packages

- Install base packages, such as `affy` and `limma`

```
source("http://bioconductor.org/biocLite.R")
biocLite()
```

- Install specific packages, such as `yeast2probe`

```
source("http://bioconductor.org/biocLite.R")
biocLite("yeast2probe")
```

- Set search directories in the .Renviron file, e.g.

```
R_LIBS=/data/Rpackages/
```

- Update Bioconductor

```
source("http://bioconductor.org/biocLite.R")
update.packages(repos=biocinstallRepos(), ask=FALSE)
```

Pre-process of data

- Entering data into Bioconductor
- Extraction of Cerevisiae probesets
- Exploratory data analysis
- Normalising Microarray data
- Probeset level expression to gene level expression
- Principal Component Analysis

Entering data into Bioconductor

```
library(affy)
fns2 = list.celfiles(path="data2", full.names=TRUE)
rawdata = ReadAffy(filenames=fns2)
print(rawdata)
```

Strain	0 hours	1 hour	2 hours	3 hours	4 hours
Mutant 1	yeast01.cel	yeast02.cel	yeast03.cel	yeast04.cel	yeast05.cel
Wild type 1	yeast06.cel	yeast07.cel	yeast08.cel	yeast09.cel	yeast10.cel
Mutant 2	yeast11.cel	yeast12.cel	yeast13.cel	yeast14.cel	yeast15.cel
Wild type 2	yeast16.cel	yeast17.cel	yeast18.cel	yeast19.cel	yeast20.cel
Mutant 3	yeast21.cel	yeast22.cel	yeast23.cel	yeast24.cel	yeast25.cel
Wild type 3	yeast26.cel	yeast27.cel	yeast28.cel	yeast29.cel	yeast30.cel

Mask file for Cerevisiae probesets

- Mask file to filter out pombe probesets
  `http://www.affymetrix.com/Auth/support/downloads/mask_files/s_cerevisiae.zip`

```
s_cerevisiae<-scan("s_cerevisiae.msk", skip=2, list("", ""))
pombe_filter_out<-s_cerevisiae[[1]]
```

- RemoveProbe [1]

```
source("RemoveProbes.r")
library(yeast2probe)
cleancdf = cleancdfname("yeast2")
RemoveProbes(listOutProbes=NULL, pombe_filter_out,
                          "yeast2cdf","yeast2probe")
```

[1] W.G. Alvord et al., "A microarray analysis for differential gene expression in the soybean genome using Bioconductor and R.", *Briefings in Bioinformatics*, September 2007

Cerevisiae probesets IDs

- Yeast probeset IDs

```
library(yeast2)
genenames = as.list(yeast2GENENAME)
YeastProbeID <- names(genenames)
```

- Cerevisiae probeset IDs

```
CerevisiaeProbeID <-
 YeastProbeID[-match(pombe_filter_out,YeastProbeID)]
```

Cerevisiae gene names

- Yeast Transcript IDs from annotation file Yeast_2.na24.annot.csv.zip

```
yeast2annotation=read.csv(file="yeast2annotation.csv",
            header=TRUE,stringsAsFactors=FALSE)
YeastTranscriptID<-yeast2annotation[,3]
yeast2annotationProbesetID<-yeast2annotation[,1]
YeastTranscriptID<-YeastTranscriptID[match(YeastProbeID,
                                    yeast2annotationProbesetID)]
```

- Yeast gene names

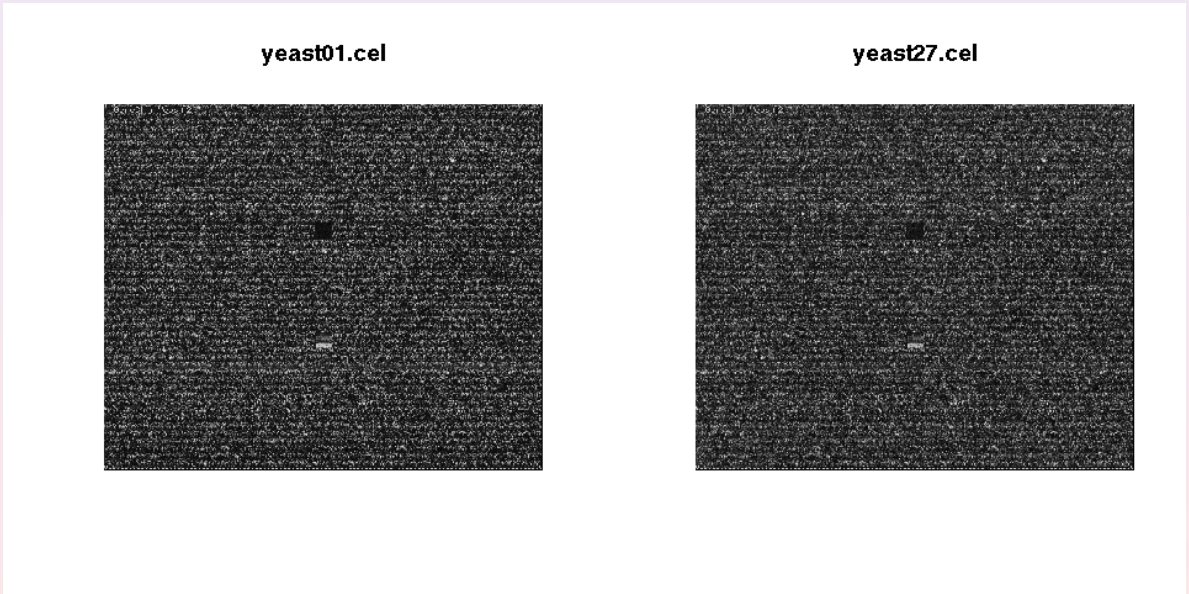
```
YeastGeneName<-character()
for(i in 1:length(YeastProbeID)){
 YeastGeneName[i]=genenames[i][[1]]
 if(is.na(YeastGeneName[i])){
  YeastGeneName[i]=YeastTranscriptID[i]
  }
}
```

- Cerevisiae gene names

```
CerevisiaeGeneName<-
 YeastGeneName[-match(pombe_filter_out,YeastProbeID)]
```

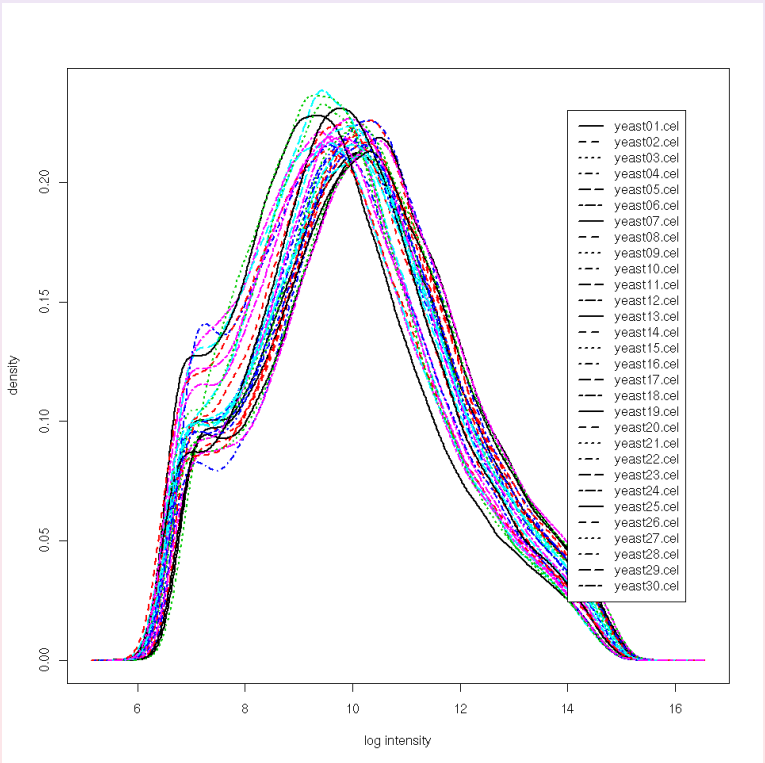
Exploratory data analysis: examining raw images

```
png(filename="cerevisiaeimage.png",width=960, height=480)
par(mfrow=c(1,2))
image(rawdata[,1])
image(rawdata[,27])
dev.off()
```



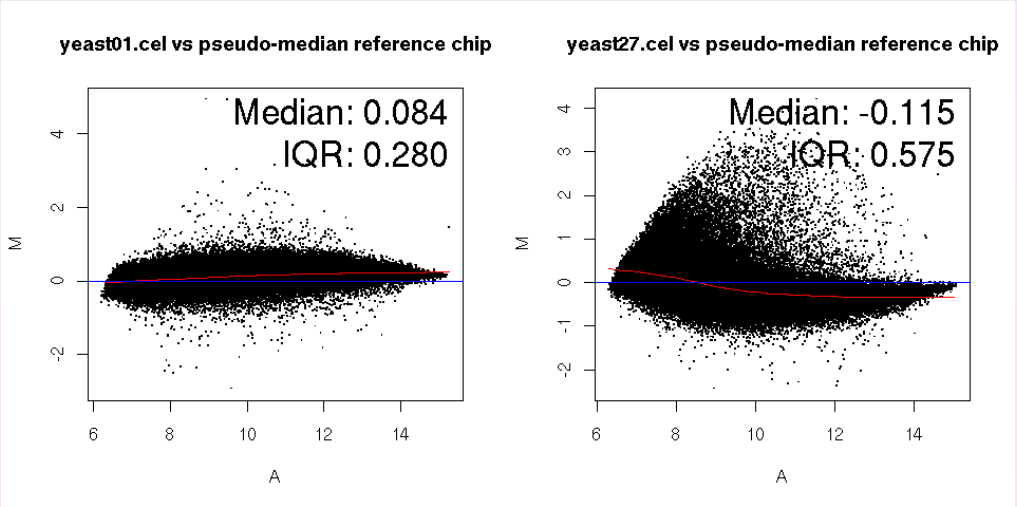
Exploratory data analysis: probe intensities

```
png(filename="yeastintensities.png",width=960, height=960)
hist(rawdata, lty=1:30, lwd=2)
legend(14, 0.60, legend=sampleNames(rawdata), lty=1:30, lwd=2)
dev.off()
```



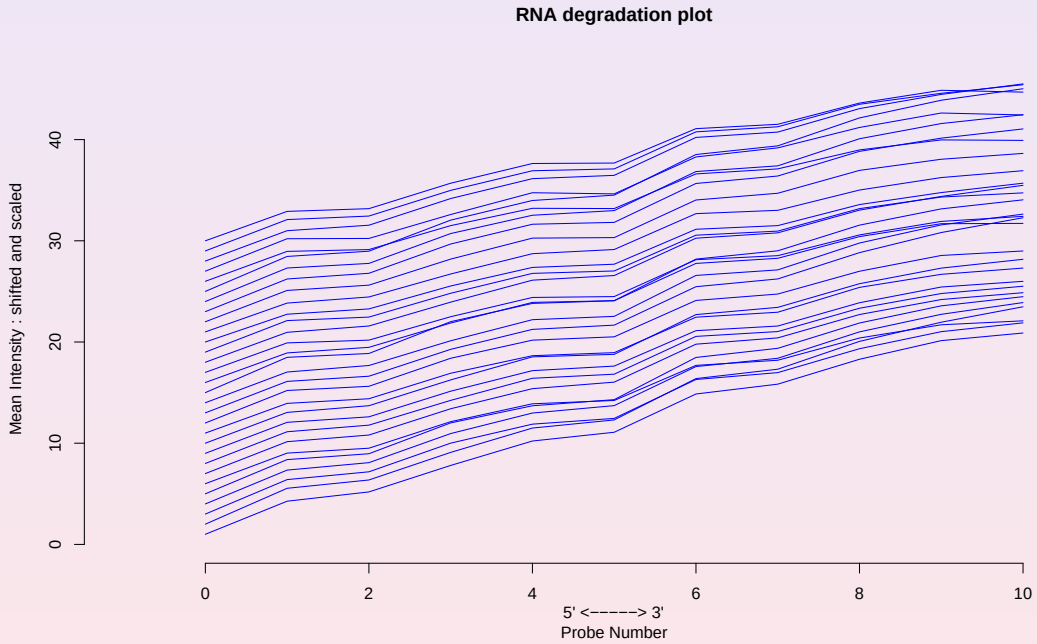
Exploratory data analysis: MA plots

```
png(filename="cerevisiaemaplot.png",width=960, height=480)
par(mfrow=c(1,2))
MAplot(rawdata,which=1)
MAplot(rawdata,which=27)
dev.off()
```



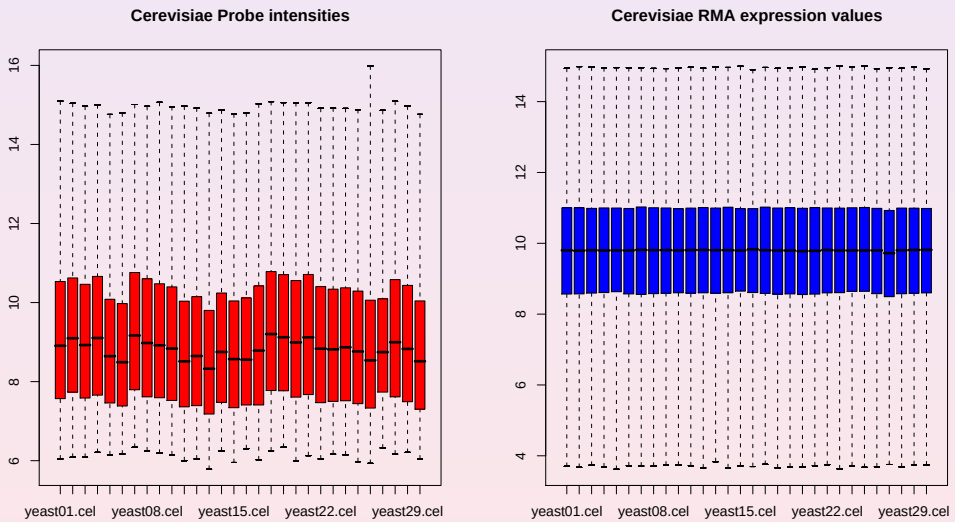
Exploratory data analysis: RNA degradation

```
RNAdeg <- AffyRNAdeg(rawdata)
plotAffyRNAdeg(RNAdeg)
```



Pre-process of data: Normalisation

```
eset.rma=rma(rawdata)
library(affyPLM)
par(mfrow=c(1,2))
boxplot(rawdata, col="red",main="Cerevisiae Probe intensities")
boxplot(eset.rma, col="blue",main="Cerevisiae RMA expression values")
```



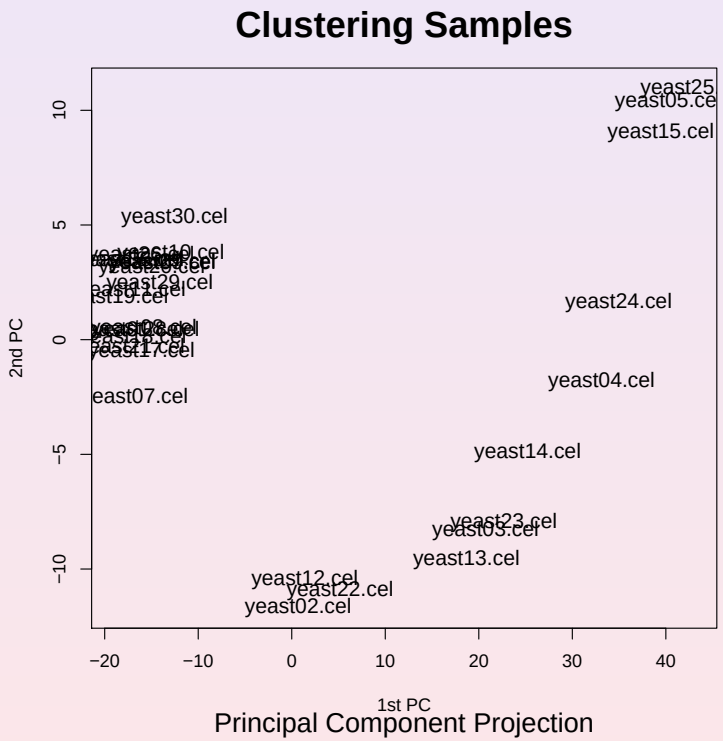
Probeset level expression to gene level expression

There are usually several probesets map to one gene in Affymetrix.

```
CerevisiaeGeneNameLevels<-factor(CerevisiaeGeneName)
#Function to average the expression of probesets
#which map to same gene
probeset2genelevel<-function(onesample){
  return(tapply(onesample,CerevisiaeGeneNameLevels,mean))
}
#Do the average for each column/array
CerevisiaeGeneData<-apply(CerevisiaeProbeData,2,probeset2genelevel)
```

Principal Component Analysis

```
library(smida)
cluster.samples(t(CerevisiaeProbeData),method="pca")
```



Model Fitting for Identifying Differential Expression

- Limma model
  - Construct design matrix
  - Construct constrasts
- Plot time course for top differential expression
- Heatmap

Limma: design matrix

```
library(limma)
levels = c("m0","m1", "m2", "m3", "m4","w0","w1", "w2", "w3", "w4")
X= rep(levels, 3)
TS <- factor(X, levels= levels)
design <- model.matrix(~0+TS)
colnames(design) <- levels(TS)
```

$$E\left[\begin{pmatrix} y_{g_1} \\ y_{g_2} \\ \vdots \\ y_{g_{10}} \\ \vdots \\ y_{g_{26}} \\ y_{g_{27}} \\ y_{g_{28}} \\ y_{g_{29}} \\ y_{g_{30}} \end{pmatrix}\right] = \begin{pmatrix} 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 \\ \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots \\ 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 \end{pmatrix} \begin{pmatrix} m_0 \\ m_1 \\ m_2 \\ m_3 \\ m_4 \\ w_0 \\ w_1 \\ w_2 \\ w_3 \\ w_4 \end{pmatrix} \quad (1)$$

Limma: construct constrsts and model fitting

For identifying differential expression, combine the contrasts by comparing mutant type and wild type at time point 1,2,3 and 4.

```
#Model Fitting
fit<-lmFit(CerevisiaeProbeData, design)
mc<-makeContrasts('m1-w1','m2-w2','m3-w3','m4-w4',levels=design)
fit2<-contrasts.fit(fit, mc)
eb<-eBayes(fit2)
```

Different ways to rank the differentially expressed probesets:

```
topTable(eb,sort.by='logFC') #log-fold change

topTableF(eb) #F-statistics
```

Up and down regulated list

```
modFpvalue<-eb$F.p.value #$F-test p value
selectedgenesindx<-p.adjust(eb$F.p.value,method="bonferroni")<0.05
Sig<-modFpvalue[selectedgenesindx]
nsiggenes<-length(Sig) #number of differential expression
results1<-decideTests(eb, method="global")
modF<-eb$F #$F-test value
modFordered<-order(modF, decreasing = TRUE)
CerevisiaeRankProbe<-CerevisiaeProbeID[modFordered[1:nsiggenes]]
CerevisiaeRankGeneName<-CerevisiaeGeneName[modFordered[1:nsiggenes]]
updown<-results1[modFordered[1:nsiggenes],]
```

Probeset ID	Gene Symbol	T1	T2	T3	T4
ProbesetID 1	Gene 1	-1	-1	-1	-1
ProbesetID 2	Gene 2	1	1	1	1
ProbesetID 3	Gene 3	-1	-1	-1	-1

Table: Up and down regulated list[2]

[2] Not use real gene names here

Plot time course for top differential expression

```
#Rank the i+1'th differential expression
indx <- rank(modF) == nrow(CerevisiaeProbeData)-i
```

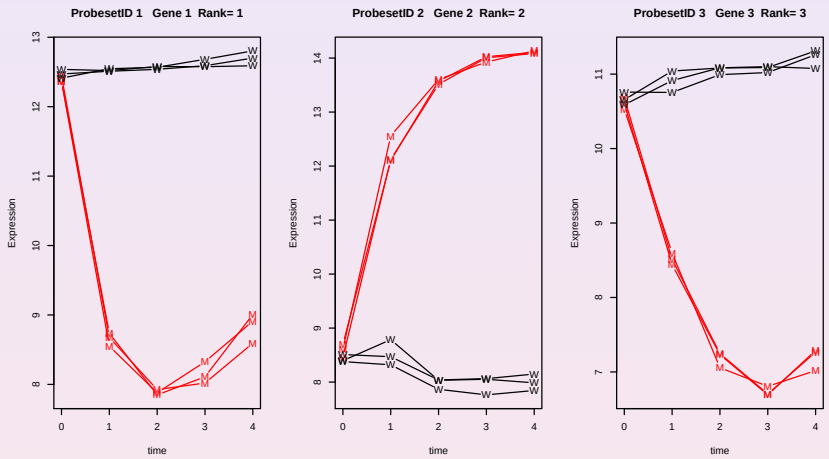


Figure: Time course expression for top 3 differentially expressed Yeast genes[3]

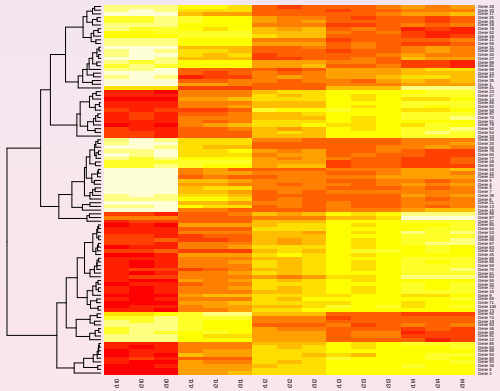
[3]Not use real gene names here

Code for plotting time course data

```
modF <- eb$F #$F is to get F-statistic
which.M <- c(seq(1,5), seq(11,15), seq(21,25))
which.W <- c(seq(6,10), seq(16,20), seq(26,30))
par(mfrow=c(1,3),ask=T,cex=0.5)#cex:font size
for(i in 0:2){
 indx <- rank(modF) == nrow(CerevisiaeProbeData)-i
 row1 = CerevisiaeProbeData[indx, which.M]
 row2 = CerevisiaeProbeData[indx, which.W]
 id=CerevisiaeProbeID[indx]
 name = CerevisiaeGeneName[indx]
 genetitle<-paste(sprintf("%.30s",id)," ",sprintf("%.30s",name)," Rank=", i+1)
 time=c(0,1,2,3,4)
 plot(time,row1[1:5],ylim=range(min(row1,row2), max(row1,row2)),
       ylab="Expression", main=genetitle,pch='M',type='b',col=2)
 lines(time, row1[6:10] , pch='M', type='b', col=2)
 lines(time, row1[11:15], pch='M', type='b', col=2)
 lines(time, row2[1:5], pch='W', type='b', col=1)
 lines(time, row2[6:10], pch='W', type='b', col=1)
 lines(time, row2[11:15], pch='W', type='b', col=1)
}
```

Heatmap

```
ngenes = 100
m = matrix(nrow=ngenes,ncol=30)
rnames = vector("list", length(1))
for(i in 0:(ngenes-1)){
 indx <- rank(modF) == nrow(CerevisiaeProbeData) - i
 m[i+1,] = CerevisiaeProbeData[indx,]
 rnames[i+1]=CerevisiaeGeneName[indx]
 }
}
heatmap(m)
```



Network Inference

- GeneNet
- Strimmer's VAR model

GeneNet: partial correlation network

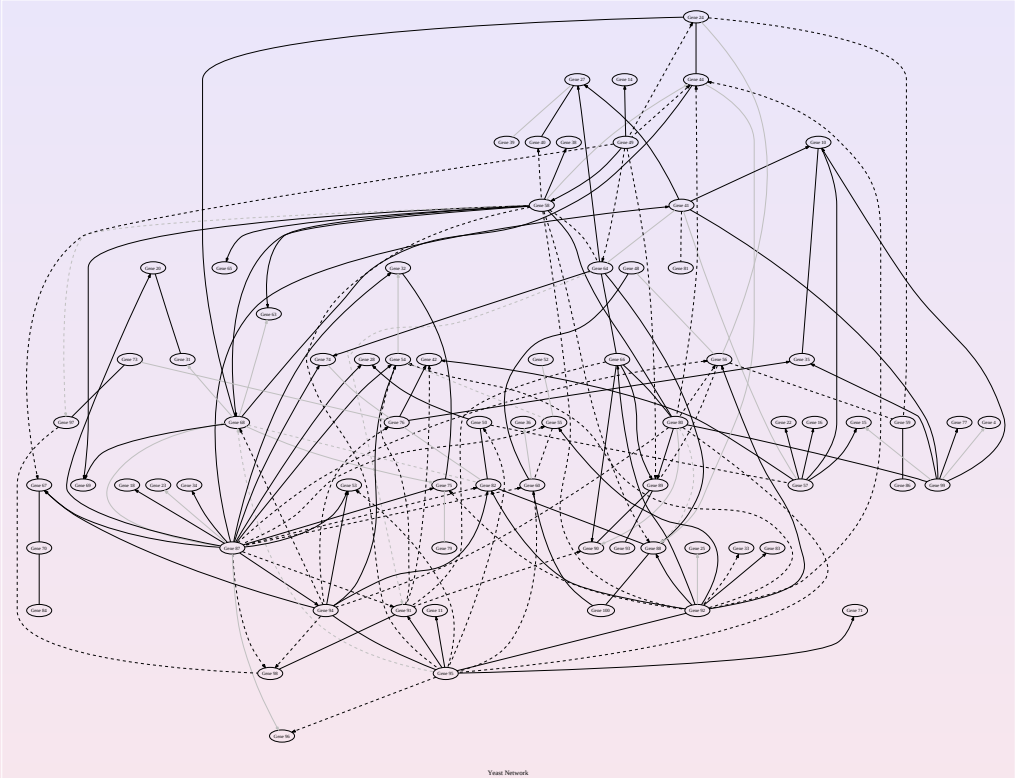


Figure: Inferred network by GeneNet package for top 100 differentially expressed Yeast genes

GeneNet step 1: build longitudial object

```
#Need to transpose data matrix, rows to be arrays
m = t(m)
#Need to rearrange the rows so that the rows are ordered by time points
#using the property of design matrix
#the entry design[i,j] with value 1 means array i is for time point j!!!
mnew = t(matrix(nrow=ngenes,ncol=30))
#Get the index of array ordered by time point
arrayindx<-numeric(0)
ntime=5
for(j in 1:ntime)
{arrayindx<-c(arrayindx,grep(1,design[,j]),grep(1,design[,j+ntime]))
}
mnew<-m[arrayindx,]
library("GeneNet")
# step 1: create longitudinal object #
mlong = as.longitudinal(mnew,repeats=c(6,6,6,6,6),time=c(0,1,2,3,4))
```

GeneNet: step 2 to step 5

```
# step 2: compute partial correlations #
pcor.dyn <- ggm.estimate.pcor(mlong, method = "dynamic")
# step 3: assign (local) fdr values to all possible edges #
m.edges <- network.test.edges(pcor.dyn,direct=TRUE)
dim(m.edges)
# step 4: construct graph containing the 150 top edges #
m.net <- extract.network(m.edges, method.ggm="number", cutoff.ggm=150)
# step 5: plot graph using graphviz #
#If rnames has no "", Need for Graphviz
for(i in 1:ngenes){
     rnames[i] = paste('"',rnames[i],'"',sep="")
    }
colnames(m) = rnames
node.labels <- colnames(m)
network.make.dot(filename="net.dot", m.net, node.labels, main="Yeast Network")
```