Gene Set Enrichment Analysis Linear Model
(GSEAlm)

Michael Steelman

Why use gene sets?

- After controlling for multiple hypotheses few if any single genes will be significant
- You might have a long list of statistically significant genes with no understanding of how they act together, making interpretation very difficult
- May miss pathway effects
- Different studies looking at the same process could have little overlap in significant genes

GSEA

- Study the difference between two phenotypes
- Gene sets are considered significant when genes are correlated with the top or bottom of a list of DE genes
- Is more or less a data reduction technique
  - We get a matrix of average expression values of each gene for each sample
  - Filter the genes and find DE
  - Use those statistics to find significant gene sets

Issues

- What would you foresee as the biggest problem in GSEA?
  - Only one dichotomous variable of interest

GSEAlm

- Oron and Gentleman (2013) made an R package that uses GSEA in a linear model
- What would be an advantage of this?
  - Multiple variables
  - Helps check for problems in the data

GSEAlm (cont.)

- The regression equation is:
  - $Y_{gi} = \beta_{g0} + \sum^{p}_{j=1} X_{ij}\beta_{gj} + \varepsilon_{gi}$
- Where
  - Y_{gi} = the gene expression value of gene g in sample I
  - P = the number of explanatory variables
  - X_{ij} = the value of the j-th variable in the i-th sample
  - β_{gj} = the true effect of variable j on expression of g
  - ε_{gi} = random error assumed ~ $N(0,\sigma^2)$

Filtering

- We are only looking for samples that show one of the two phenotypes we care about
- Commonly assumed that only 40% of genes are expressed in tissues
- We go from 12,625 genes in 128 samples to 3,462 genes in 79 samples

R Code: Filtering

```
> library("GSEAlm")
> data(ALL)

> ### Some filtering; > bcellIdx <- grep("^B",
    as.character(ALL$BT))
> bcrOrNegIdx <- which(as.character(ALL$mol.biol)
+ %in% c("NEG", "BCR/ABL"))
> esetA <- ALL[ , intersect(bcellIdx, bcrOrNegIdx)]
> esetA$mol.biol = factor(esetA$mol.biol) # recode factor
> ### Non-specific filtering
> esetASub <- nsFilter(esetA,var.cutoff=0.6,var.func=sd)$eset
```

Locating Gene Sets

- Locate gene sets based on chromosomal location
- Especially important in datasets (such as ALL) known to have chromosomal irregularities
- Set the minimum gene set size to 5 genes
- Create an incidence matrix
  - Gene set x genes
  - Genes are 1 if they are in the gene set, else 0

Locating Gene Sets (cont.)

- The “MAPAmat” function can give us the matrix directly
- Alternatively, we can make a hierarchal chromosome tree then make the matrix based off the tree
  - The advantage of this is we have the tree in our memory which will be needed for later
- The base of the tree is the node indicating a human
- We have 509 gene sets with 3459 genes

R Code: Locating Gene Sets

```
minBandSize = 5
> haveMAP = sapply(mget(featureNames(esetASub), hgu95av2MAP), function(x) !all(is.na(x)))
> workingEset = esetASub[haveMAP, ]
> entrezUniv = unlist(mget(featureNames(workingEset), hgu95av2ENTREZID))
> ### Creating incidence matrix and keeping the graph structure

> AgraphChr=makeChrBandGraph("hgu95av2.db",univ=entrezUniv)
> AmatChr = makeChrBandInciMat(AgraphChr)
> AmatChr3 = AmatChr[rowSums(AmatChr)>=minBandSize,]
> # ##Re-ordering incidence matrix columns

> egIds = sapply(featureNames(workingEset), function(x) hgu95av2ENTREZID[[x]])
> idx = match(egIds, colnames(AmatChr))
> AmatChr3 = AmatChr3[, idx]
> colnames(AmatChr3)=featureNames(workingEset)
> # Updating our graph to include only the bands that actually
> # appear in the matrix (doing it a bit carefully though...)

> AgraphChr3 = subGraph(c("ORGANISM:Homo sapiens",rownames(AmatChr3)),AgraphChr)
```

T-tests

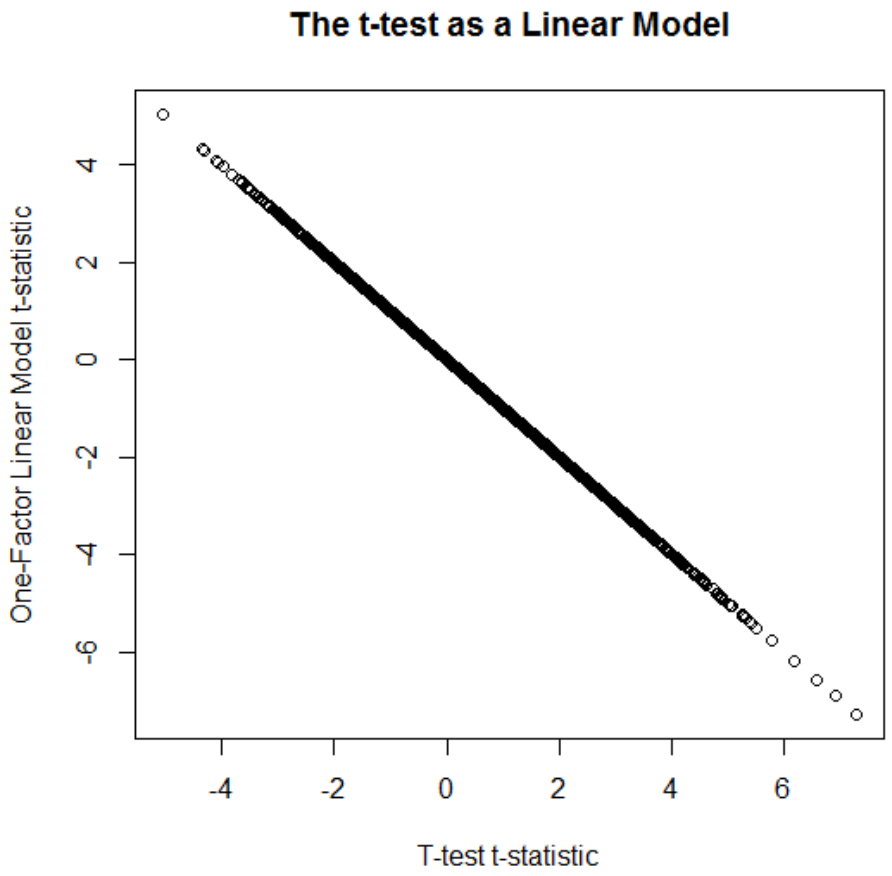
- A single, dichotomous, explanatory variable used in a t-test is essentially the same as fitting in a linear model
- Values of t-stat in “rowttests” and “lmPerGene” are opposite but equal in magnitude
- We can make them equal using the “relevel” function before we calculate t-stat

R Code: T-tests

```
> lmPhen <- lmPerGene(workingEset,~mol.biol)
> ##fit the t-tests model
> tobsChr <- rowttests(workingEset,"mol.biol")
> ## fit it via the linear-model interface
> lmEsts = lmPhen$tstat[2,]
> plot (tobsChr$stat,lmEsts,main="The t-test as a Linear Model",
+ xlab="T-test t-statistic",ylab="One-Factor Linear Model t-statistic")
> ### Re-leveling the factor

> workingEset$mol.biol<-relevel(workingEset$mol.biol,ref="NEG")
> lmPhen <- lmPerGene(workingEset,~mol.biol)
> lmEsts = lmPhen$tstat[2,]
```

T-tests (cont.)


The t-test as a Linear Model
One-Factor Linear Model t-statistic
T-test t-statistic
-6
-4
-2
0
2
4
-4
-2
0
2
4
6

T-tests (cont.)

- Pick the ordering that makes the most sense to your data
- In this case, the absence of mutation (“NEG”) is a more reasonable reference group
- What advantage does using the “lmPerGene” compared to “rowttests” offer?
  - A matrix of residuals for each of our raw data points

Residuals

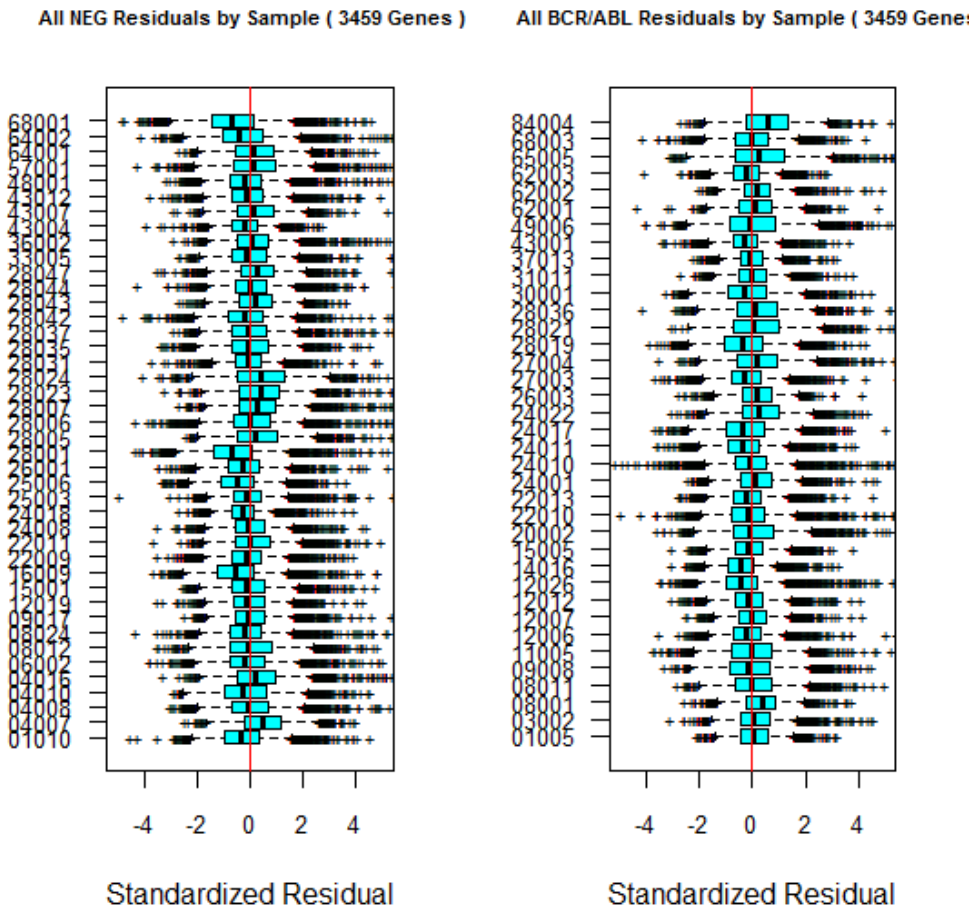
- Residuals themselves are meaningless
  - What is considered a high expression value in two different genes could be completely different values
- Solutions?
  - Studentized residuals

Studentized Residuals

- Arrange the studentized residuals by sample to find out if any samples have unusually high or low expression values

```
> lmPhenRes <- getResidPerGene(lmPhen)
>resplot(resmat=exprs(lmPhenRes),fac=working
   Eset$mol,cex.main=.7,cex.axis=.6,
 horiz=TRUE,lims=c(-5,5),xname="",col=5,cex=.3)
```

Studentized residuals



Gene Set Significance

- Several methods to test the significance, but authors favored the J-G statistic for simplicity
- $\tau_S = \sum_{g \, \varepsilon \, S} (t_g - t_{ref}) \, / \, \mathrm{sqrt}(|S|)$
  - The reference is the median of the t-scores
  - $|S|$ is the size of gene set S
- Base p-values on permutations
- Uses the “gsealmPerm” function
- Ignored in this model

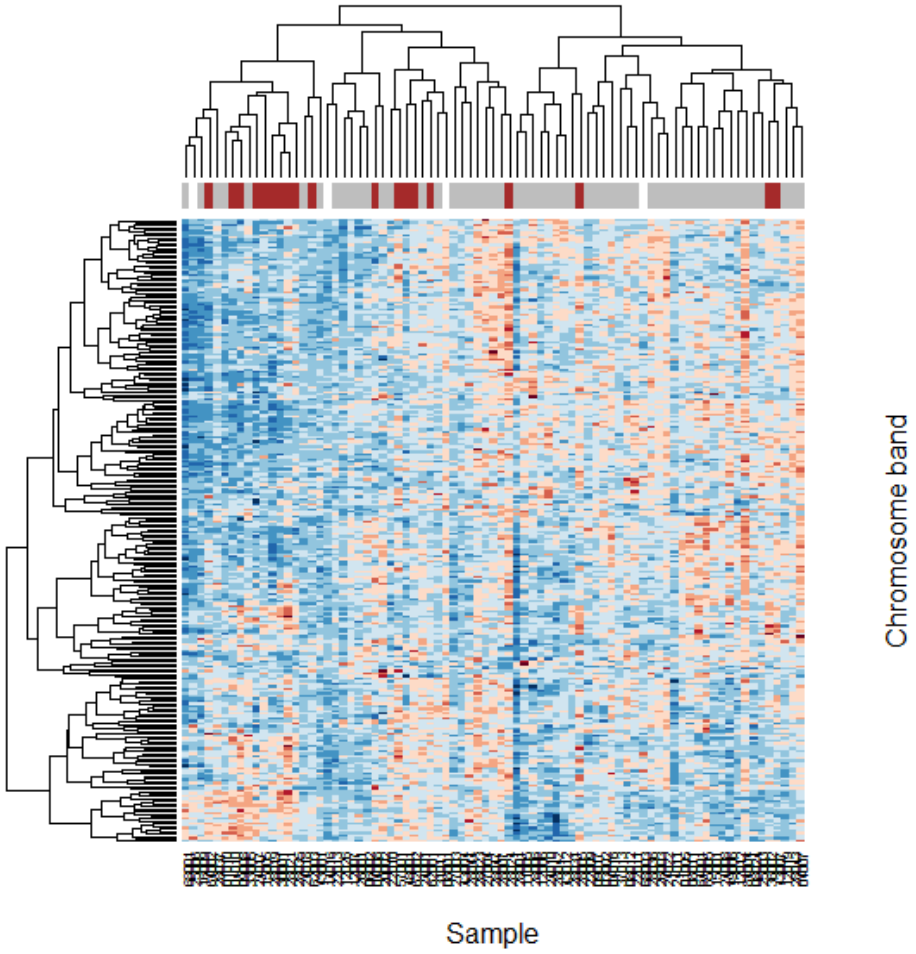
Residual Diagnostics

- $r_{Si} = \sum_{g \, \varepsilon \, S} r_{gi} \, / \, \mathrm{sqrt}(|S|)$
  - Where r_{gi} is the studentized residual of gene g in sample I
- These are the residuals of the gene sets
- Only the leaves of the hierarchal tree of gene sets are included
  - This is possible because of how we chose to get original residuals from the t-tests
- Based on correlation not Euclidean distance

R code: Residual Diagnostics

```
> ## now we are going to aggregate residuals over chromosome bands
>
> stdrAchr=GSNormalize(exprs(lmPhenRes),AmatChr3)
> kinetColors <- ifelse(workingEset$kinet=="hyperd.","brown", "grey")
> onecor=function(x) as.dist(1-cor(t(x))) # To get correlation-based heatmap
> ### In the heatmap we only use the lowest-level bands, or "leaves" of the graph
>
> ChrLeaves=leaves(AgraphChr3,"out")
> ### for safety
> ChrLeaves=ChrLeaves[ChrLeaves %in% rownames(AmatChr3)]
> LeafGenes=which(colSums(AmatChr3[ChrLeaves,])>0)
> bandHeatmap=heatmap(stdrAchr[ChrLeaves,],scale="row",col = HMcols,
+ ColSideColors=kinetColors,keep.dendro=TRUE,distfun=onecor,
+ labRow=FALSE,xlab="Sample",ylab="Chromosome band")
```

Residual Diagnostics (cont.)

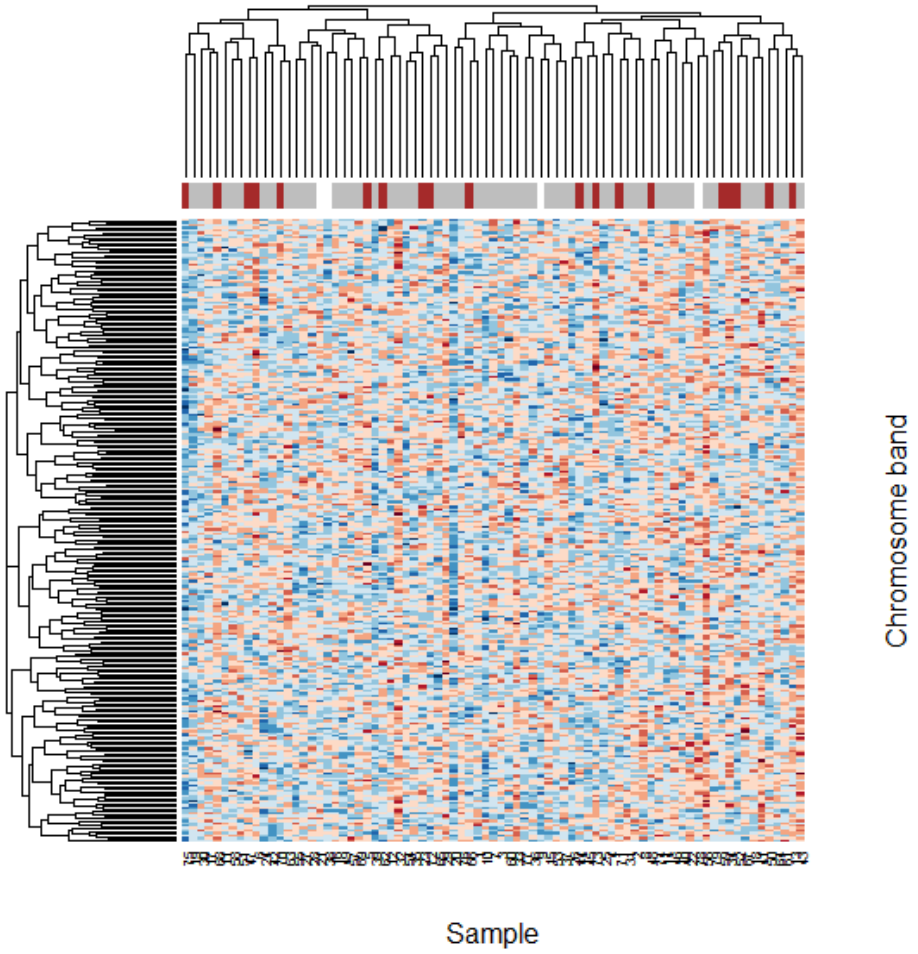


Residual Diagnostics (cont.)

- How do we know that the clustering isn't a result of the reordering that is done?
  - Let's make a new random dataset based on the leaves

```
>colbase=rexp(79,rate=3)*sample(c(-1,1),size=79,replace=T)
>randres=sweep(matrix(rnorm(length(ChrLeaves)*79),ncol=7
  9,2,colbase)
>heatmap(randres, scale="row",col = HMcols,
  ColSideColors=kinetColors,keep.dendro=TRUE,distfun=onec
  or ,labRow=FALSE,xlab="Sample",ylab="Chromosome
  band")
```

Residual Diagnostics (cont.)



More Residual Diagnostics

- The “kinet” variable that measures hyperdiploidity indicates that it has an effect on the sample.
  - Would be associated with increased expression
- Sex might also have an important role to play

3 Variable Model

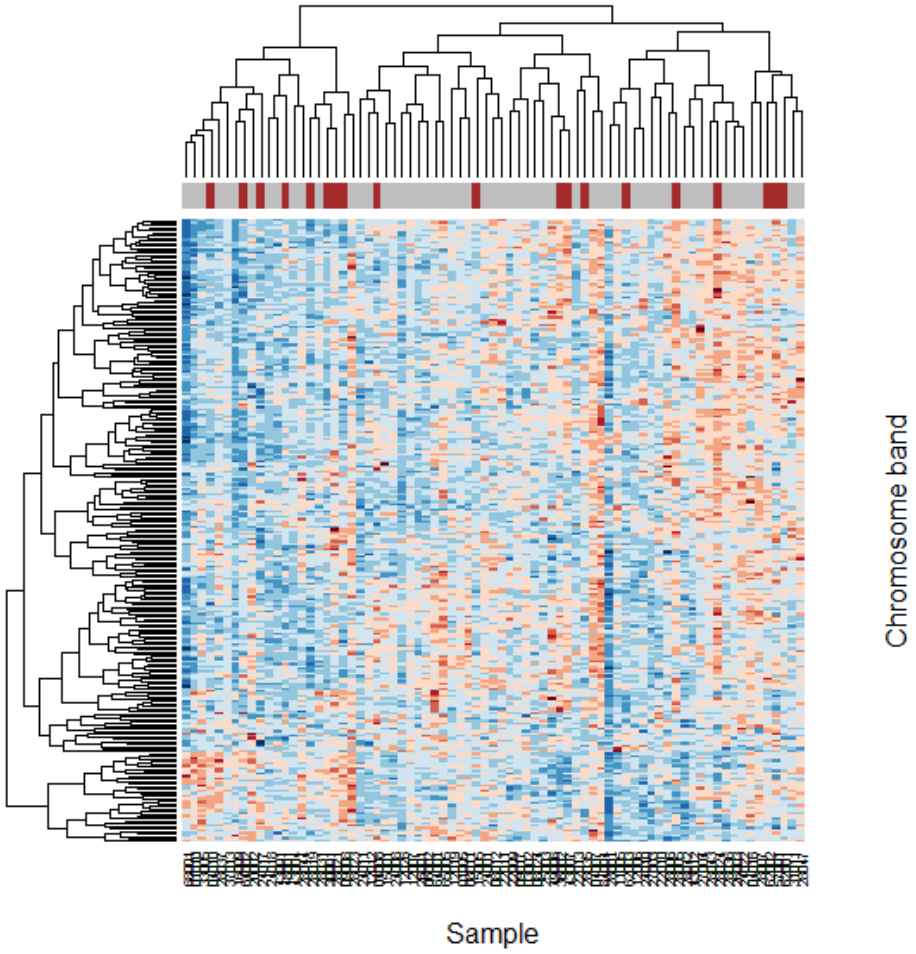
- We now have 3 variables
  - Phenotype, kinet, and sex
- What do we do with missing information?
  - Get rid of it
  - Add our own data by guessing
- We could guess on some parts, but not on others (i.e. sample 25006)

R code: 3 Variable Model

```
> lmExpand <- lmPerGene(workingEset,~mol.biol+sex+kinet)
> lmExpandRes <- getResidPerGene(lmExpand,type="extStudent")
> lmExpandTees <- t(lmExpand$tstat[2:4,])
> lmExpandBandTees<-GSNormalize(lmExpandTees,AmatChr3)
> GSresidExpand=GSNormalize(exprs(lmExpandRes),AmatChr3)

> kinetColors2 <- ifelse(lmExpandRes$kinet=="hyperd.","brown",
    "grey")
> bandHeatmapExp=heatmap(GSresidExpand[ChrLeaves,],
    scale="row",col = HMcols,
+ ColSideColors=kinetColors2,keep.dendro=TRUE,distfun=onecor,
+ labRow=FALSE,xlab="Sample",ylab="Chromosome band")
```

3 Variable Model (cont.)



Inference

- Use permutation based p-values
  - To get p-values of each factor permute the labels of that factor in each subgroup of the remaining factors
- P-values should be used as flags to find suggested significance
  - It is not a probability due to hierarchal structure
- Again only look at leaves

R code: Inference

```
>nperm=125
>flagp=0.01
>
   pvalsExpand=gsealmPerm(workingEset[LeafG
   enes,],~mol.biol+sex+kinet,AmatChr3[ChrLeav
   es,LeafGenes],nperm=nperm,removeShift=TR
   UE)
> pvalsExpand[pvalsExpand[,1]<flagp,1]
```

Inference (cont.)

```
> pvalsExpand[pvalsExpand[,1]<flagp,1]
       7p22       11p13     11p15.1     11q13.3     18q21.1     1p36.33
0.007936508 0.007936508 0.007936508 0.007936508 0.007936508 0.007936508
   20q13.12    20q13.33    22q11.23     22q12.1     22q13.2         7p13
0.007936508 0.007936508 0.007936508 0.007936508 0.007936508 0.007936508
     7q22.1
0.007936508
> pvalsExpand[pvalsExpand[,2]<flagp,2]
    10p11.2       11q22       12q21        1q25        2q22        4q13
0.007936508 0.007936508 0.007936508 0.007936508 0.007936508 0.007936508
       5q23        6q23        7q31     14q23.1     17p13.2      1q21.1
0.007936508 0.007936508 0.007936508 0.007936508 0.007936508 0.007936508
       2p14      2p25.1        3q28        4p14      5q31.1      5q35.1
0.007936508 0.007936508 0.007936508 0.007936508 0.007936508 0.007936508
     7q21.3        8p22      9q33.3
0.007936508 0.007936508 0.007936508
```

Inference (cont.)

```
> chrNames=c(as.character(1:22),"X")
>
  pvalsHyper=gsealmPerm(workingEset,~kinet+
  mol.biol+sex,AmatChr3[chrNames,],nperm=n
  perm)
> pvalsHyper[pvalsHyper[,2]<flagp,2]
  ### over-expressed list for hyperdiploids
19                  22                  X
0.007936508 0.007936508 0.007936508
```