Introduction to Analysis of Sequence Count Data using DESeq

Utah State University – Spring 2014

STAT 5570: Statistical Bioinformatics

Notes 6.4

References

- Anders & Huber (2010), "Differential Expression Analysis for Sequence Count Data", Genome Biology 11:R106

- DESeq Bioconductor package vignette, obtained in R using

```
vignette("DESeq")
```

- Kvam, Liu, and Si (2012), "A comparison of statistical methods for detecting differentially expressed genes from RNA-seq data", Am. J. of Botany 99(2):248-256.

Negative Binomial (NB) using DESeq …

- Define trt. condition of sample i: $\rho(i)$
- Define # of fragment reads in sample i for gene k:

$$R_{ki} \sim NB\left(\mu_{ki}, \sigma_{ki}^2\right)$$

- Assumptions in estimating μ_{ki} and σ_{ki}^2 :
  - $\mu_{ki} = q_{k,\rho(i)} s_i$
    - $q_{k,\rho(i)}$: per-gene abundance, prop. to true conc. of fragments
    - s_i: library size, prop. to coverage [exposure] in sample i
  - $\sigma_{ki}^2 = \mu_{ki} + s_i^2 v_{k,\rho(i)}$
    - μ_{ki}: "shot noise" – this "dominates" for low-expressed genes
    - $v_{k,\rho(i)}$: raw variance (biological variability)
  - $v_{k,\rho(i)} = v_\rho\left(q_{k,\rho(i)}\right)$
    - v_ρ: smooth function – pool information across genes to estimate variance

Estimate parameters (for NB distn.)

m = # samples; n = # genes

$$\hat{s}_i = med_k \left\{ R_{ki} \Big/ \left\{ \prod_{j=1}^{m} R_{kj} \right\}^{1/m} \right\}$$

For median calculation, skip genes where geometric mean (denom) is zero.

- denom. is geometric mean across samples
  - like a pseudo-reference sample
- $\hat{s}_i$ is essentially equivalent to $\sum_k R_{ki}$,

with robustness against very large R_{ki} for some k

Estimate parameters (for NB distn.)

$$\hat{q}_{k\rho} = \frac{1}{m_\rho} \sum_{i:\rho(i)=\rho} \frac{R_{ki}}{\hat{s}_i}$$

- m_ρ = # samples in trt. condition ρ
- this is the mean of the standardized counts from the samples in treatment condition ρ

Estimate parameters (for NB distn.)

$$\hat{w}_{k\rho} = \frac{1}{m_\rho - 1} \sum_{i:\rho(i)=\rho} \left(\frac{R_{ki}}{\hat{s}_i} - \hat{q}_{k\rho} \right)^2$$

(this is the variance of the standardized counts from the samples in trt. condition ρ)

$$z_{k\rho} = \frac{\hat{q}_{k\rho}}{m_\rho} \cdot \sum_{i:\rho(i)=\rho} \frac{1}{\hat{s}_i}$$

(an un-biasing constant)

$$\hat{v}_\rho\left(\hat{q}_{k\rho}\right) = \max\left(\hat{w}_{k\rho}, w_\rho\left(\hat{q}_{k\rho}\right)\right) - z_{k\rho}$$

- Estimate function w_ρ by plotting $\hat{w}_{k\rho}$ vs. $\hat{q}_{k\rho}$, and use parametric dispersion-mean relation:

$$w_\rho\left(\hat{q}_{k\rho}\right) = \alpha_0 + \alpha_1 / \hat{q}_{k\rho}$$

(α_0 is "asymptotic dispersion"; α_1 is "extra Poisson")

Estimating Dispersion in DESeq

1. Estimate dispersion value $\hat{w}_{k\rho}$ for each gene

2. Fit for each condition (or pooled conditions [default]) a curve through estimates (in the $\hat{w}_{k\rho}$ vs. $\hat{q}_{k\rho}$ plot)

3. Assign to each gene a dispersion value, using the maximum of the estimated [empirical] value $\hat{w}_{k\rho}$ or the fitted value $w_{\rho}\left(\hat{q}_{k\rho}\right)$
   -- this conservative approach avoids under-estimating dispersion (which would increase false positives)

Example – 3 treated vs. 4 untreated; read counts for 14,470 genes

- Published 2010 (Brooks et al., Genome Research)
- Drosophila melanogaster
- 3 samples "treated" by knock-down of "pasilla" gene (thought to be involved in regulation of splicing)

	T1	T2	T3	U1	U2	U3	U4
FBgn0000003	0	1	1	0	0	0	0
FBgn0000008	118	139	77	89	142	84	76
FBgn0000014	0	10	0	1	1	0	0
FBgn0000015	0	0	0	0	0	1	2
FBgn0000017	4852	4853	3710	4640	7754	4026	3425
FBgn0000018	572	497	322	552	663	272	321

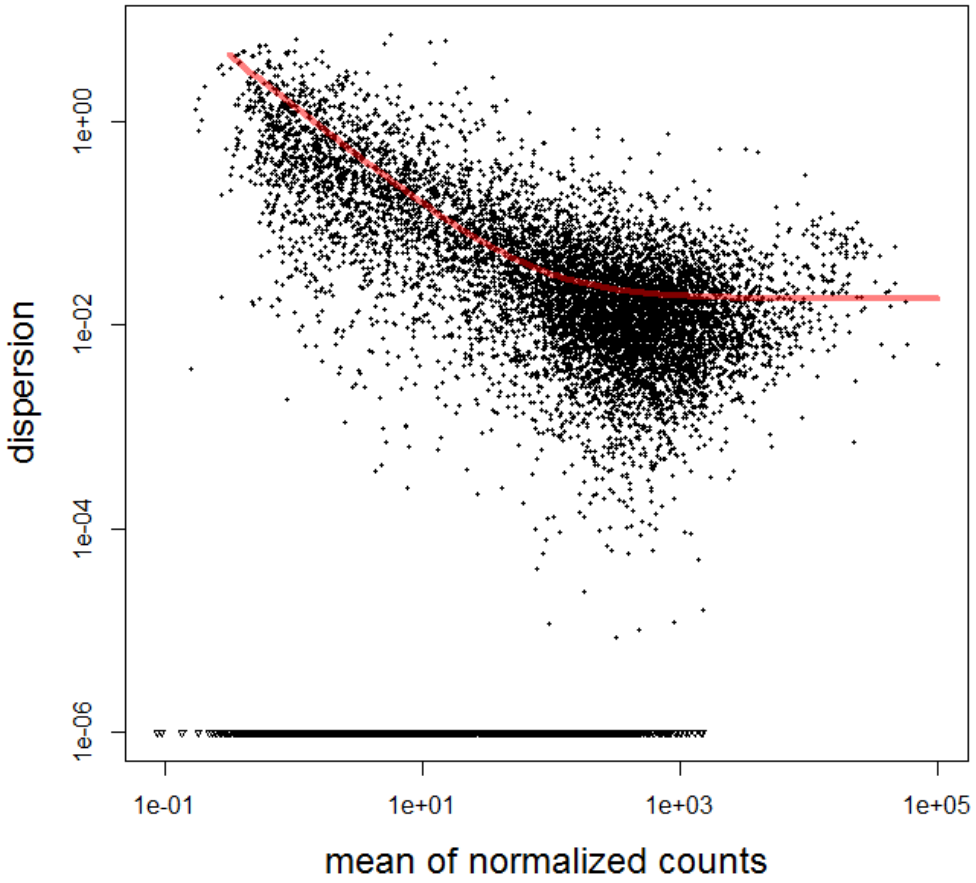
Getting started with DESeq package

- Need data in this format (previous slide)
  - Integer counts in matrix form, with columns for samples and rows for genes
  - Row names correspond to genes (or genomic regions, at least)
  - See package vignette for suggestions on how to get to this format (including from sequence alignments and annotation)
- Can use read.csv or read.table functions to read in text files
- Each column is a biological rep
  - If have technical reps, sum them together to get a single column

```
# load data
library(pasilla); data(pasillaGenes)
eset <- counts(pasillaGenes)
colnames(eset) <- c('T1','T2','T3','U1','U2','U3','U4')
head(eset)

# format data
library(DESeq)
countsTable <- eset   # counts table needs
                      # gene IDs in row names
rownames(countsTable) <- rownames(eset)
conds <- c("T","T","T","U","U","U","U")
         # 3 treated, 4 untreated
dim(countsTable) # 14470 genes, 7 samples
cds0 <- newCountDataSet(countsTable, conds)

# Estimate s (also called library size)
cds1 <- estimateSizeFactors( cds0 )
  # - Each column is divided by the geo. means of the rows

# Estimate q, w, and v
cds2 <- estimateDispersions( cds1 )

# Check quality of dispersion estimation
plotDispEsts( cds2 , cex.lab=1.5 )
```

Checking Quality of Dispersion Estimation

- Plot $\hat{w}_{k\rho}$ vs. $\hat{q}_{k\rho}$ (both axes log-scale here)
- Add fitted line for $w_\rho(\hat{q}_{k\rho})$
- Check that fitted line is roughly appropriate general trend



Test for DE between conditions A and B – similar to Fishers Exact Test

$$R_{kA} = \sum_{i:\rho(i)=A} R_{ki} \qquad R_{kB} = \sum_{i:\rho(i)=B} R_{ki} \qquad R_{kS} = R_{kA} + R_{kB}$$

- Based on NB distn. (now estimated), can calculate

$$p(a,b) = P\{R_{kA} = a, R_{kB} = b \mid a + b = R_{kS}\}$$

  for all pairs (a,b)

- P-value for gene k is prob. of a result more extreme than what was observed, just by chance, when no DE

What is "more extreme" here (for p-value calculation)?

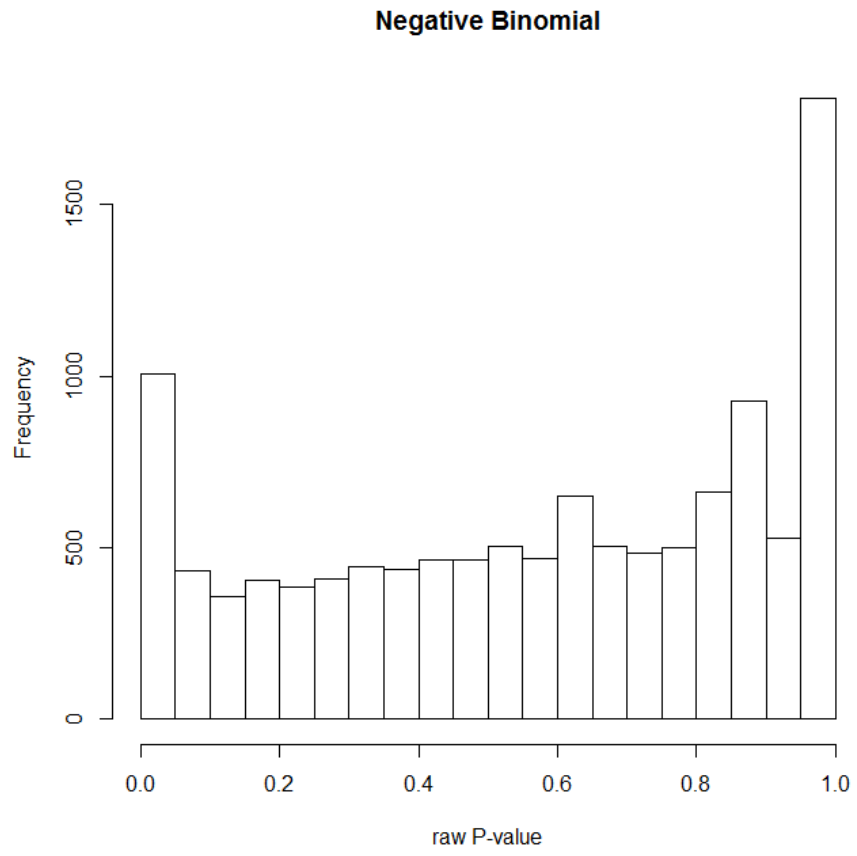
- Originally, used "less likely":

$$p_k = \sum_{p(a,b) \le p(R_{kA}, R_{kB})} p(a,b) \Big/ \sum p(a,b)$$

- But this caused problems for genes with dispersion > 1. Now instead, sum for "all value pairs that are *further out* in the sense that they cause a more extreme fold change"; "do this in a one-tailed manner and double the result":

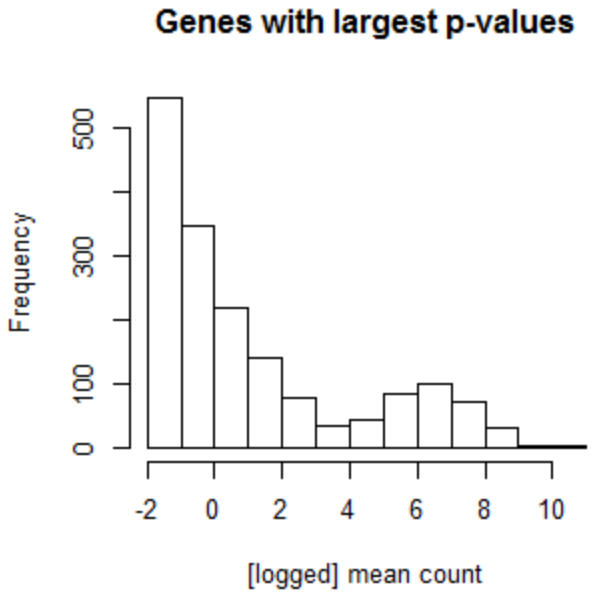
$$p_k = 2 \cdot \sum_{\frac{a}{b} > \frac{R_{kA}}{R_{kB}}} p(a,b) \Big/ \sum p(a,b)$$

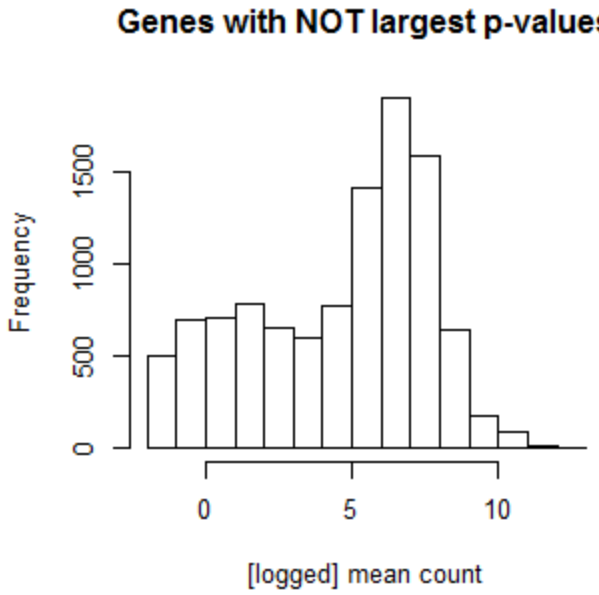
	id	baseMean	log2FoldChange	pval	padj
1	FBgn0000003	0.1594687	-Inf	0.86914737	1.0000000
2	FBgn0000008	52.2256776	-0.02523529	0.94429078	1.0000000
3	FBgn0000014	0.3897080	-0.32836850	0.84677545	1.0000000
4	FBgn0000015	0.9053584	0.94382157	1.00000000	1.0000000
5	FBgn0000017	2358.2434078	0.27560986	0.06890543	0.6982231
6	FBgn0000018	221.2415562	0.12223204	0.62488664	1.0000000



- Peak near zero:
  DE genes
- Peak near one:
  low-count genes (?)
- Default adjustment:
  BH FDR

```
# test for DE -- this takes about 1 minute
print(date())
res <- nbinomTest(cds2, "T","U")
print(date())

# see results
# (with re-ordered columns here just for convenience)
head(res)[,c(1,2,6,7,8)]
hist(res$pval,xlab='raw P-value',
     main='Negative Binomial')

# check to explain missing p-values
t <- is.na(res$pval)
sum(t) # 2634, or about 18.2% here
range(res$baseMean[t])   #  0  0
#  - only happens for undetected genes


# define sig DE genes
t <- res$padj < .05 & !is.na(res$padj)
gn.sig <- res$id[t]
length(gn.sig) # 347
```





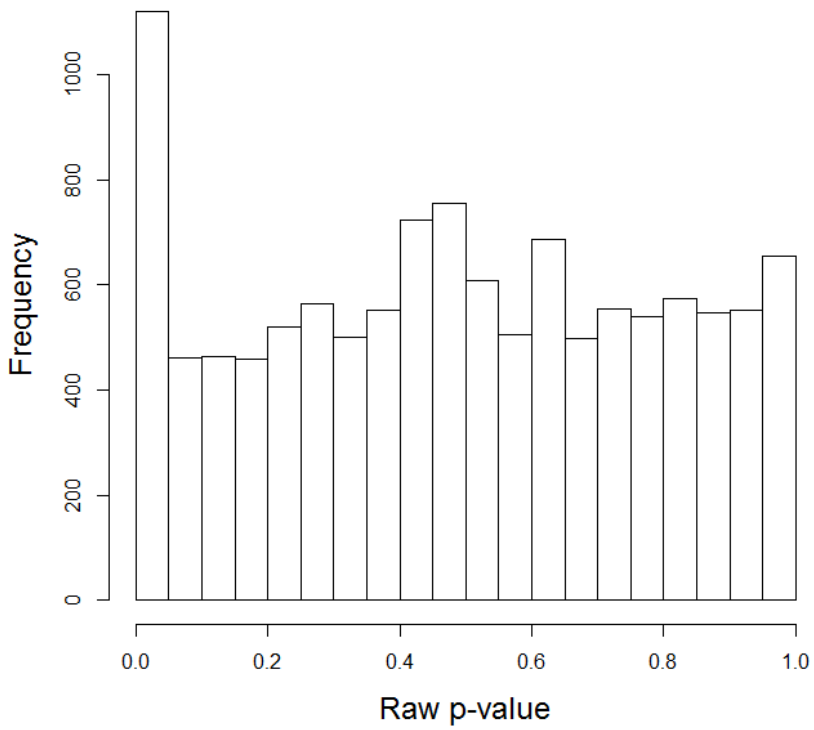

```
# check p-value peak near 1
par(mfrow=c(2,2))
counts <- rowMeans(eset)
t <- res$pval > 0.95 & !is.na(res$pval)
par(mfrow=c(2,2))
hist(log(counts[t]), xlab='[logged] mean count',
     main='Genes with largest p-values')
hist(log(counts[!t]), xlab='[logged] mean count',
     main='Genes with NOT largest p-values')
# -- tends to be genes with smaller overall counts
```

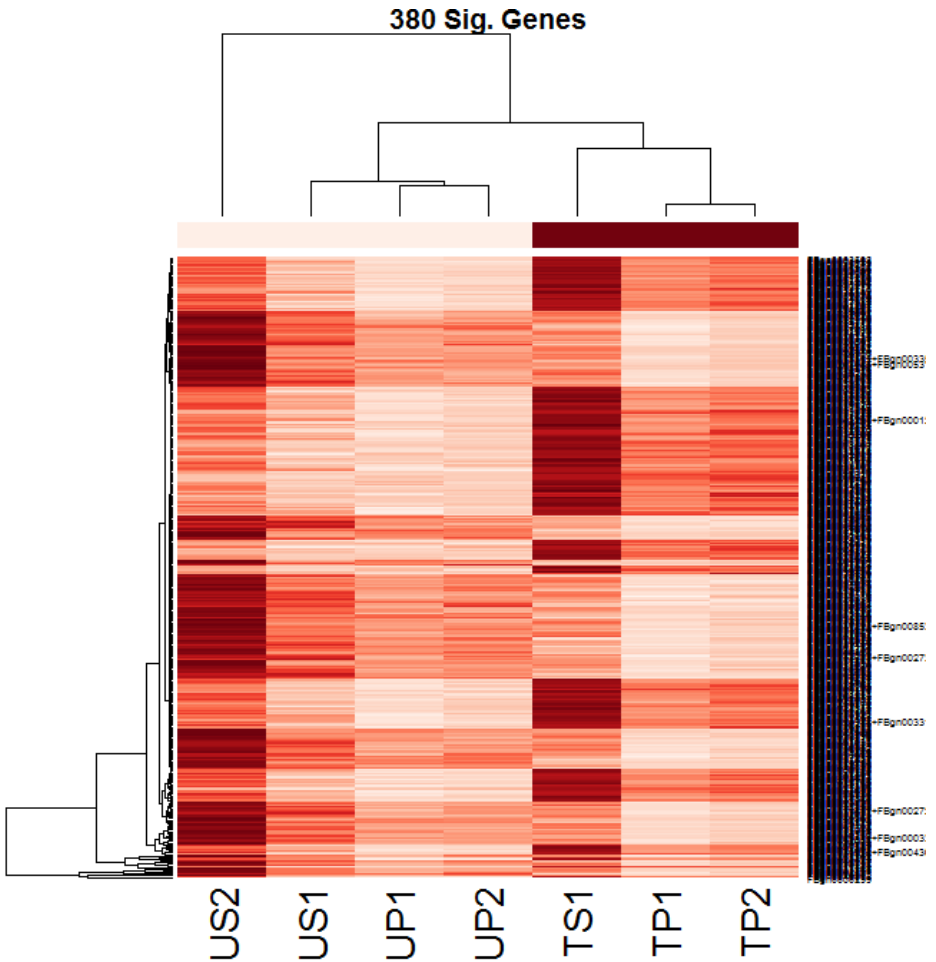
Same example, but with extra covariate

- 3 samples "treated" by knock-down of "pasilla" gene, 4 samples "untreated"
  - Of 3 "treated" samples, 1 was "single-read" and 2 were "paired-end" types
  - Of 4 "untreated" samples, 2 were "single-read" and 2 were "paired-end" types

	TS1	TP1	TP2	US1	US2	UP1	UP2
FBgn0000003	0	1	1	0	0	0	0
FBgn0000008	118	139	77	89	142	84	76
FBgn0000014	0	10	0	1	1	0	0
FBgn0000015	0	0	0	0	0	1	2
FBgn0000017	4852	4853	3710	4640	7754	4026	3425
FBgn0000018	572	497	322	552	663	272	321

Test trt effect while accounting for type





Negative Binomial (NB) Regression, using a Likelihood Ratio Test (LRT)

- Define # of fragment reads in sample i for gene k:

$$R_{ki} \sim NB\left(\mu_{ki}, \sigma^2_{ki}\right)$$

- After estimating parameters for NB distribution, fit (for each gene k) two models for what contributes to [log-scale] expected value:
  - "Full" model $\log\left(\mu_{ki}\right) = \beta_0 + \beta_1 trt + \beta_2 type$
  - "Reduced" model – assuming some null H_0 is true, like H_0 : "no trt effect" $\log\left(\mu_{ki}\right) = \beta_0 + \beta_2 type$

- Then test H_0 using a LRT (likelihood *L* based on NB):

$$-2\log\frac{L_{red}}{L_{full}} \sim \chi^2_{DF}$$

(DF = difference in # parameters, full – reduced)

```
# load data; recall eset object from slide 10
colnames(eset) <- c('TS1','TP1','TP2','US1','US2','UP1','UP2')
head(eset)

# format data
countsTable <- eset
rownames(countsTable) <- rownames(eset)
trt <- c("T","T","T","U","U","U","U")
type <- c("S","P","P","S","S","P","P")
cond <- cbind(trt,type)
cds0 <- newCountDataSet(countsTable, cond)

# Estimate NB distribution and check quality (as on slide 10)
cds1 <- estimateSizeFactors( cds0 )
cds2 <- estimateDispersions( cds1 )
plotDispEsts( cds2 )

# Fit full and reduced models (takes a little less than 1
min.)
fit.full <- fitNbinomGLMs( cds2, count ~ trt + type )
fit.red  <- fitNbinomGLMs( cds2, count ~ type )

# Get p-values from LRT (same order as eset rows)
pvals <- nbinomGLMTest( fit.full, fit.red)
```

```
# Visualize sig. results
hist(pvals, xlab='Raw p-value', cex.lab=1.5, cex.main=2,
     main='Test trt effect while accounting for type')

# Get sig. genes
adj.pvals <- p.adjust(pvals, "BH")
t <- adj.pvals < .05 & !is.na(adj.pvals)
sum(t) # 380
sig.gn <- rownames(eset)[t]

# Visualize sig. genes
library(RColorBrewer)
small.eset <- eset[t,]
hmcol <- colorRampPalette(brewer.pal(9,"Reds"))(256)
csc <- rep(hmcol[250],ncol(small.eset))
csc[trt=="U"] <- hmcol[10]
heatmap(small.eset,scale="row",col=hmcol,
   ColSideColors=csc, cexCol=2.5,
   main=paste(sum(t),'Sig. Genes'))
```

What else / next?
Similar to before with microarrays...

- Adjust for multiple testing
- Filtering (to increase statistical power)
  - zero-count genes?
- Visualization: Volcano plots / heatmaps / clustering (including bootplot) / PCA biplot
- Random Forests
- Characterize significant genes
  - Gene Set Testing – global test
  - GO / KEGG annotation not as convenient to access as for microarrays (yet)

Current and Future Work

- Extending (and computationally optimizing) Poisson / Negative Binomial models to specifically allow for classical experimental designs (factorial, repeated measures, etc.)
- Extending (and automating) gene set testing methods
- Incorporating other annotation information
- Making full use of preliminary alignment quality information (currently only a single-step filter: pass / fail)