Supplementary Material for Booth et al.

Derivation of the Score Statistic

Let y_{kj} denote the jth count under treatment k, where $k = 1, \dots, K$. Suppose that the counts are independent Poisson variates with means given by

$$\log \mu_{kj} = \beta_k + \log L + \log N_{kj} \,,$$

where $\log L$ and $\log N_{kj}$ are known offsets. The log-likelihood is given by

$$l(\boldsymbol{\beta}) = \sum_k \sum_j (y_{kj} \log \mu_{kj} - \mu_{kj}) \,.$$

Differentiating the log-likelihood with respect to components of $\boldsymbol{\beta}$ reveals the kth component of the score function, $S(\boldsymbol{\beta})$, to be

$$\frac{\partial l}{\partial \beta_k} = \sum_j (y_{kj} - \mu_{kj}) \,,$$

Differentiating again reveals the information matrix (the negative Hessian) to be $I(\boldsymbol{\beta}) = \text{diag}\left\{\sum_j \mu_{kj}\right\}_{k=1}^{K}$.

The score statistic for testing $\beta_1 = \cdots = \beta_K$ is given by $U = \hat{S}'\hat{I}^{-1}\hat{S}$, where $\hat{S}$ and $\hat{I}$ are the score function and information matrix evaluated at the null ML estimate.

Now, if $\beta_1 = \cdots = \beta_K = \beta$ say, then

$$\frac{\partial l}{\partial \beta} = \sum_k \sum_j (y_{kj} - \mu_{kj}) \,.$$

Setting this derivative shows that the null ML estimate satisfies, $\bar{y} = e^{\hat{\beta}} L \bar{N}$, from which it follows that the null fitted values are given by

$$\hat{\mu}_{kj} = \frac{N_{kj}}{\bar{N}} \bar{y} \,.$$

Thus, the score statistic is given by

$$U = \sum_{k=1}^{K} \frac{\left[\sum_j \left(y_{kj} - \frac{N_{kj}}{\bar{N}}\bar{y}\right)\right]^2}{\sum_j \frac{N_{kj}}{\bar{N}}\bar{y}} \,.$$

When $K = 2$ we can use the fact that $\bar{y} = (n_1\bar{y}_1 + n_2\bar{y}_2)/n$ to show that

$$\sum_j \left(y_{1j} - \frac{N_{1j}}{\bar{N}} \bar{y} \right) = \frac{n_1 n_2}{n\bar{N}} \left(\bar{N}_2 \bar{y}_1 - \bar{N}_1 \bar{y}_2 \right) = \sum_j \left(y_{2j} - \frac{N_{2j}}{\bar{N}} \bar{y} \right) ,$$

and so the score statistic can be rewritten as

$$U = \frac{n \left[\sum_j \left(y_{1j} - \frac{N_{1j}}{\bar{N}} \bar{y} \right) \right]^2}{\frac{n_1 \bar{N}_1}{\bar{N}} \frac{n_2 \bar{N}_2}{\bar{N}} \bar{y}} .$$

R code for Model 3

```
#Set working directory/folder. This folder should contain this R file,
#the data file in the required format, and the file "proteomics.bug"

#Load the following packages.
library(coda)
library(BRugs)
library(R2WinBUGS)

# 1.  Read in data
# 2.  Format data for OpenBUGS
# 3.  Function to generate initial values for OpenBUGS
# 4.  Set the parameters (MCMC chains) to be saved
# 5.  Call OpenBUGS
# 6.  Extract results and write to file

#########################################################################
# 1.  Read in data
# Comma separated data file with variables: Protein, Length, W1,...,Wk,
# M1,..,Mk, where W1 denotes 1st wildtype replicate count and M1 denotes
# the 1st mutant replicate
df=read.table("Syntheticdataset2fold.csv",sep=",",header = TRUE)

#########################################################################
# 2.  Format data for OpenBUGS
P=df[,1]               # P should be the Protein names (first column)
L=as.numeric(df[,2])   # L should be the length (second column)
n=dim(df)[2]-2         # n: the number of replicates (control+treatment)
Y=as.matrix(df[,3:(3+n-1)])  # the response columns
p=dim(Y)[1]            # p: the number of proteins
N=apply(Y,2,mean)      # N: the average count for each replicate
logL=log(L)
logN=as.numeric(log(N))
G=rep(c(-1,1),each=n/2)# assumes equal number of ctrl and trt reps
data=list(Y=Y,G=G,logL=logL,logN=logN,p=p,n=n)  # the data for OpenBUGS
```

```
##########################################################################
# 3.  Function to generate initial values for OpenBUGS
inits=function(){list(
  I = rep(0,p),          # Indicator for treatment effect
  b0 = rnorm(p,0,1),     # protein specific random effects
  b1 = rnorm(p,0,1),
  tau0 = 1,              # precision b0
  psi1 = 0,              # mean b1
  tau1 = 1,              # precision b1
  beta0 = -log(mean(L)), # fixed intercept
  beta1 = 0,             # fixed effect for treatment
  pi1 = 0.1              # prob for prior on nonnull status
  )}
##########################################################################
# 4.  Set the parameters (MCMC chains) to be saved
parameters=c("I", "tau0", "psi1", "tau1", "beta0", "beta1", "pi1")


##########################################################################
# 5.  Call OpenBUGS
# model = "the path where the bugs model can be found"
# n.chains = how many mcmc chains to run (3 recommended)
# n.iter = the total number of iterations to run
# n.burnin = the number of iterations to burnin
# n.thin = k;  every kth iteration will be saved
# debug = TRUE;  if TRUE, any error messages will be displayed
ms.sim=bugs(data,inits,parameters,
  model="proteomics.bug",  # file containing BUGS language
  n.chains=3,n.iter=10000,n.burnin=5000,n.thin=5,
  debug=TRUE,DIC=FALSE,program="OpenBUGS",
  codaPkg=TRUE,save.history=FALSE)

##########################################################################
# 6.  Extract results and write to file
X=ms.sim$sims.matrix
I.mean=apply(X[,1:p],2,mean)
```

```
write.table(data.frame(Protein=P,Prob=I.mean),
  "bugs_output.csv",sep=",",row.names=F)

########################################################################
```

BUGS language for Model 3 (“proteomics.bug”)

```
model
{
 for (i in 1:p)
 {
  for (j in 1:n)
  {
   mu[i,j]<-exp(beta0+beta1*G[j]+b0[i]+b1[i]*G[j]*I[i]+logL[i]+logN[j])
   Y[i,j]~dpois(mu[i,j])
  }
  b0[i]~dnorm(0,tau0)
  b1[i]~dnorm(psi1,tau1)
  I[i]~dbern(pi1)
 }
 pi1~dunif(0,1)
 beta0~dnorm(0,.01)
 beta1~dnorm(0,.01)
 tau0~dgamma(0.1,0.1)
 psi1~dnorm(0,0.1)
 tau1~dgamma(0.1,0.1)
}
```

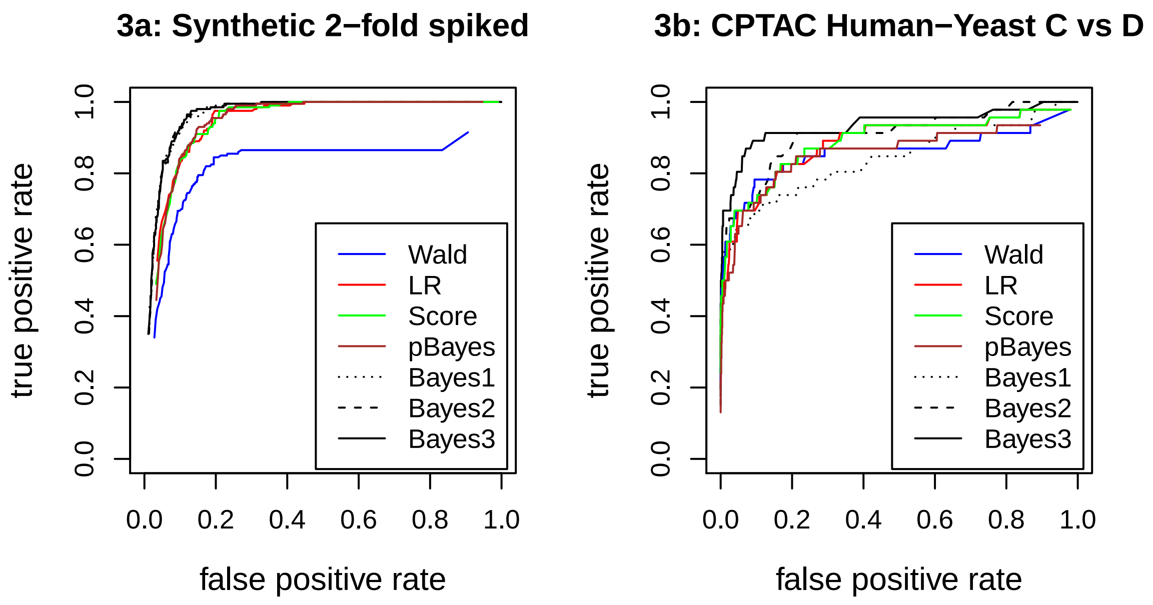

Figure 3: ROC plots comparing the performance of one-at-a-time tests with Bayesian methods 1-3 based on posterior odds of non-null status, and pseudo-Bayes factors as described in Choi et al. (2008), over the entire range of false positive rates. Figure 3a gives ROC curves for the sythetic spiked 2-fold data (Choi et. al, 2008). Figure 3b gives the corresponding curves for the CPTAC human-yeast dataset (Paulovich et al., 2010).

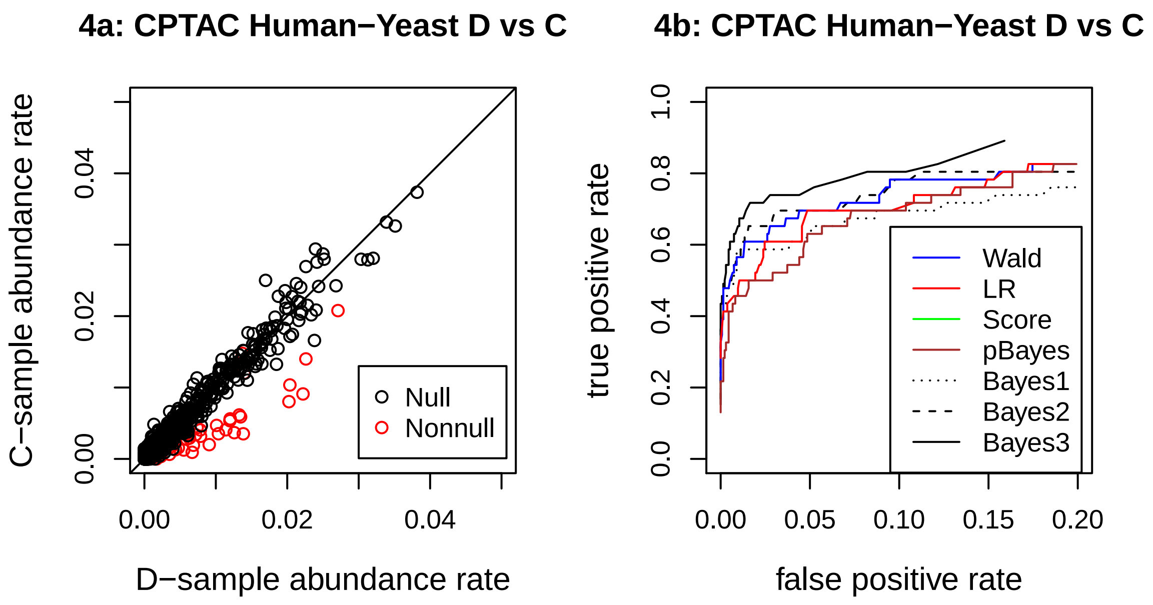

Figure 4: Relative abundance plot of the CPTAC human-yeast dataset (Paulovich et al., 2010) showing down-regulation in non-null proteins in the D-sample (Figure 4a). Figure 4b shows ROC plots comparing the performance of one-at-a-time tests with Bayesian methods 1-3 based on posterior odds of non-null status, and pseudo-Bayes factors as described in Choi et al. (2008).