2022 SISG Bayesian Statistics for Genetics R Notes: Multiple Testing

Jon Wakefield
Departments of Statistics and Biostatistics
University of Washington

2022-07-19

Prostate Cancer Example

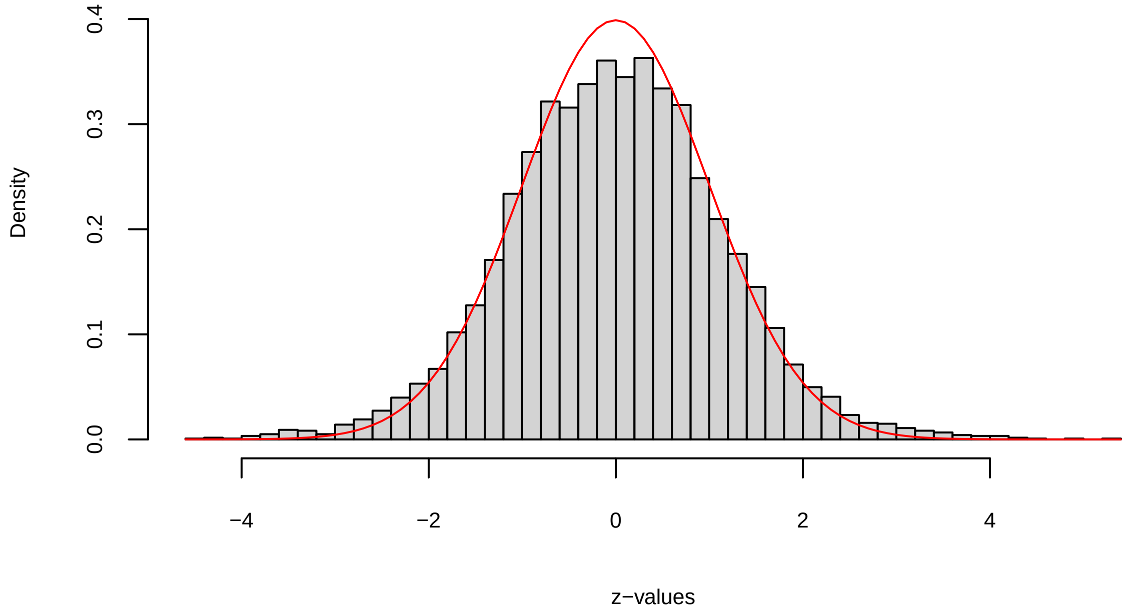
These data come from the Efron and Hastie book website:

https://web.stanford.edu/~hastie/CASI/data.html

```
if (!requireNamespace("BiocManager", quietly = TRUE)) install.packages("BiocManager")
BiocManager::install("qvalue")
library(knitr)
library(qvalue)
library(locfdr)
library(ashr)
library(xtable)
```

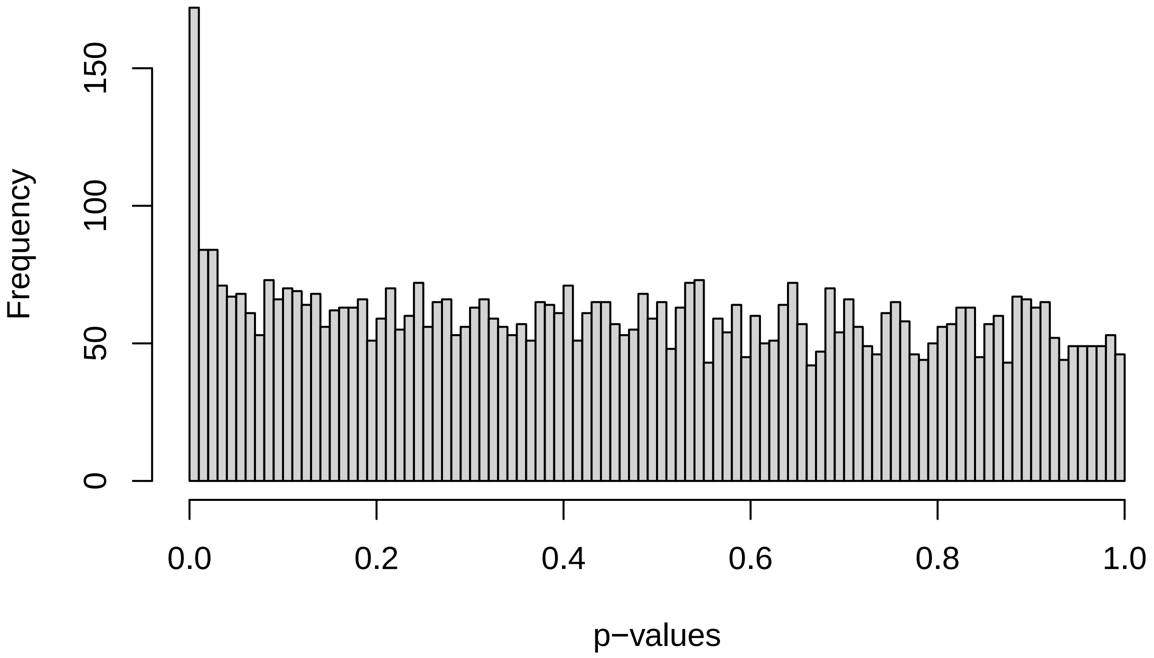
```
download.file("http://faculty.washington.edu/kenrice/sisgbayes/prostz.txt",
    destfile = "prostz.txt")
prostz <- read.table("prostz.txt", header = F, sep = ",")
zvalues <- prostz$V1
prostz = unlist(prostz)
n = length(prostz)
n
## [1] 6033
```

```
hist(prostz, nclass = 60, freq = F, ylim = c(0, 0.45),
    main = "", xlab = "z-values", cex.lab = 0.7, cex.axis = 0.7)
curve(dnorm(x), add = T, col = "red")
```



Histogram of p-values

```r
pvalues = 2 * pnorm(abs(prostz), mean = 0, sd = 1,
    lower.tail = F)
hist(pvalues, breaks = sqrt(n), main = "", xlab = "p-values")
```



Bonferroni

```r
alpha = 0.05
alpha.bon = alpha/n
```

```
alpha.bon
## [1] 8.287751e-06
which(pvalues < alpha.bon)
##  V1332  V1610 V11720
##    332    610   1720
# Just to check:
which(p.adjust(pvalues, method = "bonferroni") < alpha)
##  V1332  V1610 V11720
##    332    610   1720
```

Holm's procedure

```
# hist(pvalues, breaks = sqrt(n))
levels = alpha/(n + 1 - seq(1, n))
pvalues.order = order(pvalues)
flag.index = pvalues.order[which(pvalues[pvalues.order] <=
    levels)]
flag.index
## [1]  610 1720  332
# If we don't want to do 'by hand' use this
# command: flag.index = which(p.adjust(pvalues,
# method = 'holm') <0.05) or use p.adjust
# function directly
```

Control expected number of false discoveries (EFD)

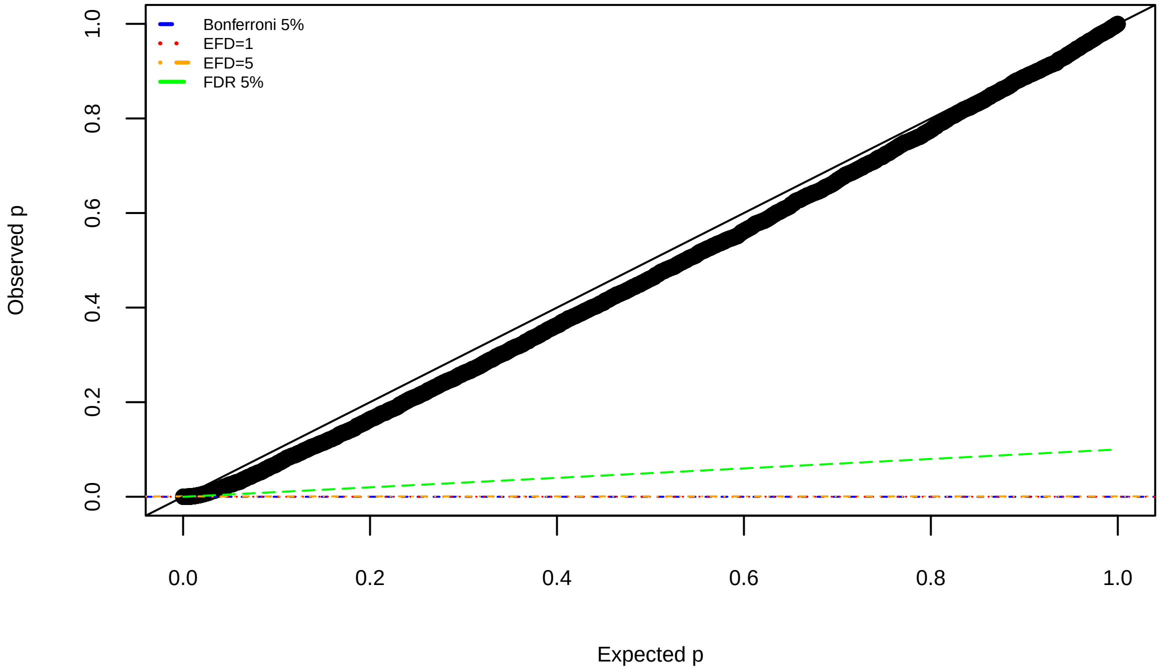
```
EFD = 1
alpha.EFD = EFD/n
flag.index.1 = which(pvalues < alpha.EFD)
# flag.index.5
EFD = 5
alpha.EFD = EFD/n
flag.index.5 = which(pvalues < alpha.EFD)
# flag.index.5
```

Benjamini Hochberg

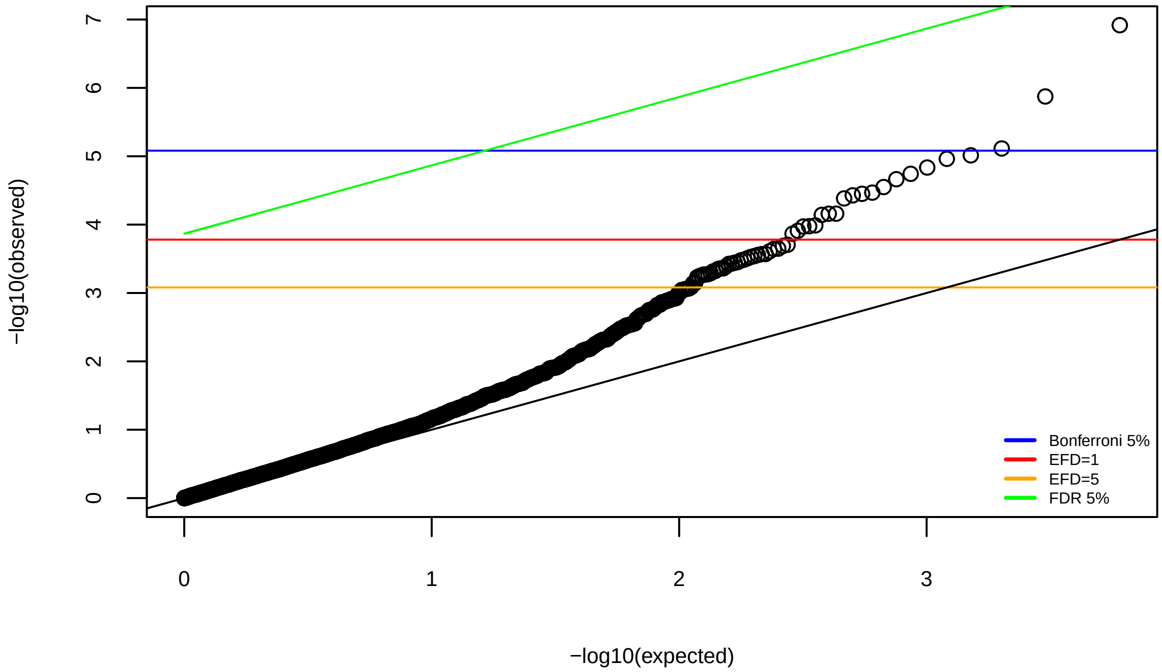
```
alphaBHset = 0.1
levels = seq(1, n) * alpha/n
pvalues.order = order(pvalues)
alpha.BH = pvalues[order(pvalues)][max(which(pvalues[pvalues.order] <=
    levels))]
# which(p.adjust(pvalues, method = 'BH') < 0.05) commented out
# because there are a lot of them

# which(pvalues < alpha.BH)

# which(p.adjust(pvalues, method = 'BH') < 0.1)
# which(p.adjust(pvalues, method = 'BH') < 0.2)
```

Multiple testing example

```
expp <- seq(1, n)/(n + 1)
psort <- sort(pvalues)
plot(expp, psort, xlim = c(0, 1), ylim = c(0, 1), ylab = "Observed p",
    xlab = "Expected p")
abline(h = 0.05/n, col = "blue")
abline(h = 1/n, col = "red")
abline(h = 5/n, col = "orange")
abline(a = 0, b = 1)
lines(expp, alphaBHset * seq(1, n)/n, col = "green")
legend("topleft", legend = c("Bonferroni 5%", "EFD=1",
    "EFD=5", "FDR 5%"), col = c("blue", "red", "orange",
    "green"), bty = "n", lwd = 2)
```

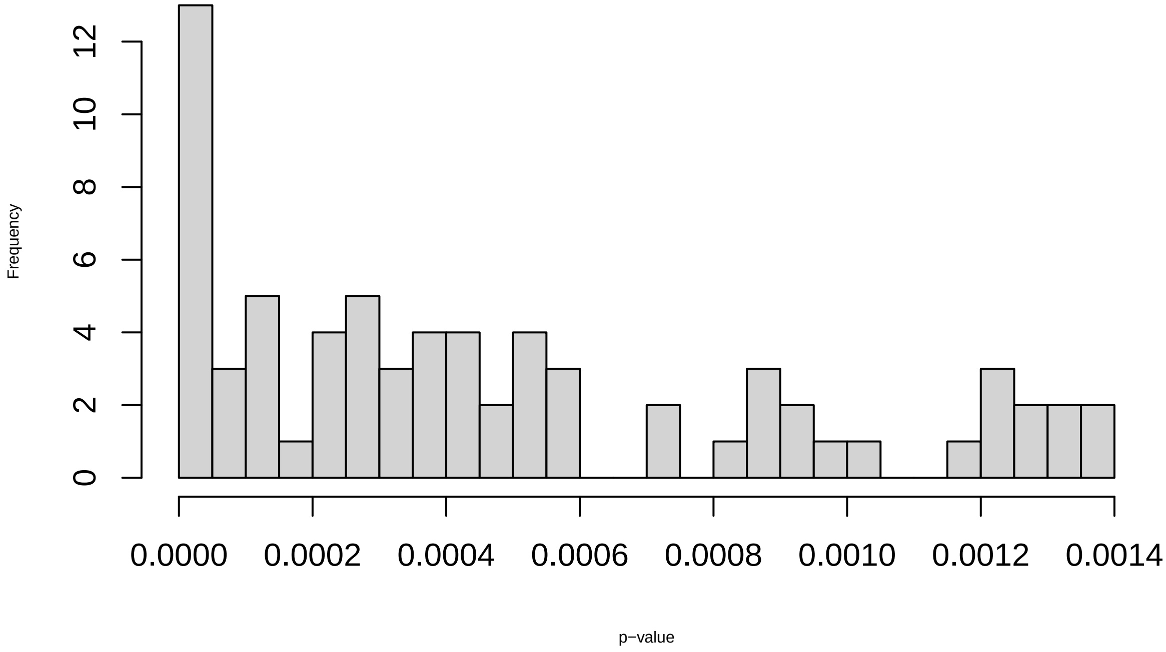


```
# Try again with focus
# pdf('multtestfig0new2.pdf')
plot(expp, psort, xlim = c(0, 0.01), ylim = c(0, 0.01),
    ylab = "Observed p", xlab = "Expected p", cex.axis = 0.7,
    cex.lab = 0.7)
abline(h = 0.05/n, col = "blue")
abline(h = 1/n, col = "red")
abline(h = 5/n, col = "orange")
abline(a = 0, b = 1)
# points(expp,alpha*seq(1,m)/m,pch=25)
lines(expp, alphaBHset * seq(1, n)/n, col = "green")
legend("bottomright", legend = c("Bonferroni 5%", "EFD=1",
    "EFD=5", "FDR 5%"), col = c("blue", "red", "orange",
    "green"), bty = "n", lwd = 2, cex = 0.5)
# dev.off()
```



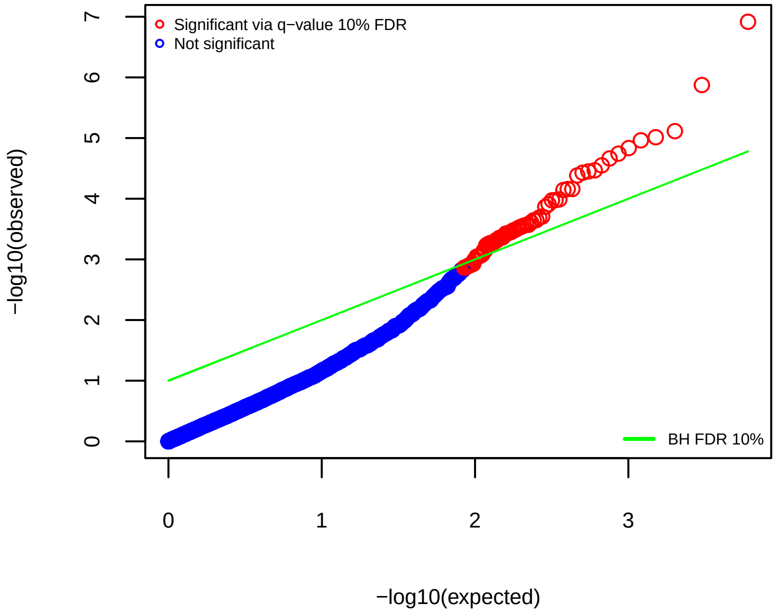
q-values: Distribution of significant p-values

```
psort2 <- sort.int(pvalues, index.return = T)
qobj10 = qvalue(p = pvalues, fdr.level = 0.1)
q10 <- which(qobj10$significant)
hist(pvalues[which(qobj10$significant)], xlab = "p-value",
    cex.lab = 0.5, nclass = 20, main = "")
```



Multiple testing example

```
alphaBHset = 0.1
plot(-log10(expp), -log10(psort), ylim = c(0, ymax),
    xlab = "-log10(expected)", ylab = "-log10(observed)",
    type = "n")
points(-log10(expp)[-c(1:length(q10))], -log10(psort)[-c(1:length(q10))],
    col = "blue")
points(-log10(expp)[c(1:length(q10))], -log10(psort)[c(1:length(q10))],
    col = "red")
lines(-log10(expp), -log10(alphaBHset * seq(1, n)/n),
    col = "green")
legend("topleft", legend = c("Significant via q-value 10% FDR",
    "Not significant"), col = c("red", "blue"), bty = "n",
    pch = 1)
legend("bottomright", legend = c("BH FDR 10%"), col = c("green"),
    bty = "n", lwd = 2)
```



q-values

```
qobj20 = qvalue(p = pvalues, fdr.level = 0.2)
# which(qobj20$significant)

# estimated proportion of null p-values
qobj20$pi0
## [1] 0.8544683
```

Local FDR

Evaluate the number passing each condition, using a natural spline to estimate f

```
out0 = locfdr(zvalues, nulltype = 0, type = 0)
which(out0$fdr < 0.1)
##  [1]  332  364  579  610  735  914 1068 1077 1089 1113 1130 1557 1720 3375 3647
## [16] 3665 3940 3991 4073 4088 4316 4331 4518 4546 4549
# summ_table_x0 <- xtable(out0$fp0, digits = 3)
# print(summ_table_x0, include.rownames=T,
# comment = FALSE)
```

Evaluate the number passing each condition using a polynomial to estimate f

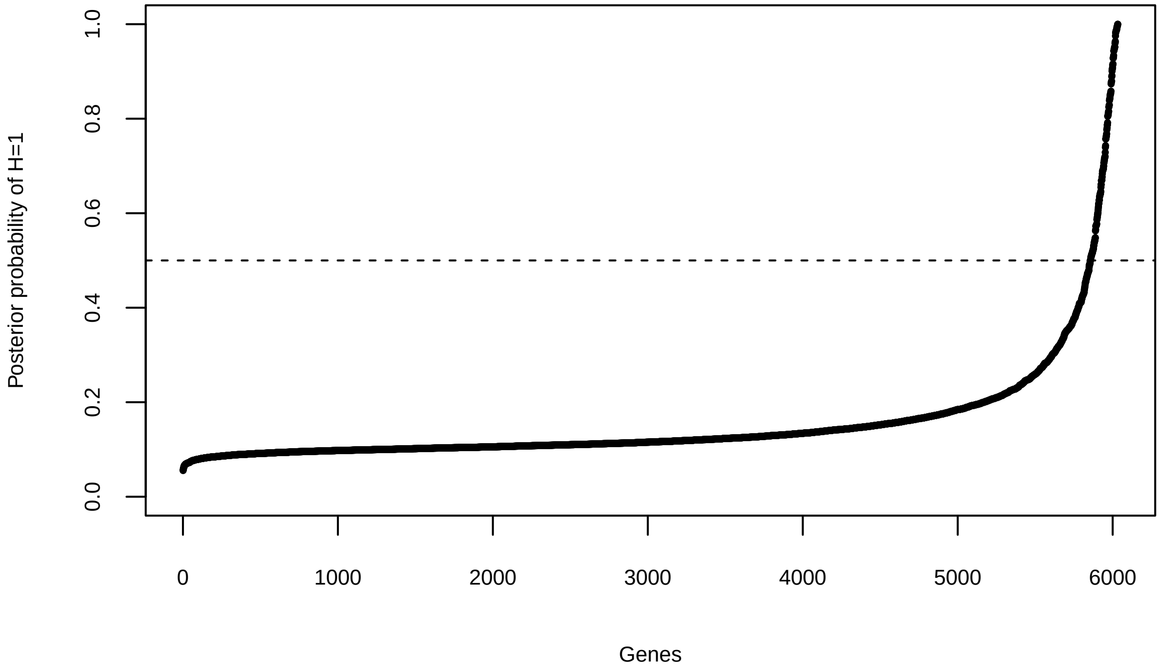
```
out1 = locfdr(zvalues, nulltype = 0, type = 1, plot = 0)
which(out1$fdr < 0.1)
##  [1]  332  364  579  610  914 1068 1077 1089 1113 1557 1720 3375 3647 3940 3991
## [16] 4073 4088 4316 4331 4518 4546
# summ_table_x1 <- xtable(out1$fp0, digits = 3)
# print(summ_table_x1, include.rownames=T,
# comment = FALSE)
```

Homemade MCMC

```
download.file("http://faculty.washington.edu/kenrice/sisgbayes/prostmat.csv",
    destfile = "prostmat.csv")
prostmat <- read.csv("prostmat.csv")
n = dim(prostmat)[1]
X.bar = apply(prostmat, 1, function(x) sum(x[1:50])/50)
Y.bar = apply(prostmat, 1, function(x) sum(x[51:102])/52)
Y = Y.bar - X.bar
s02 = apply(prostmat, 1, function(x) var(x[1:50]))
s12 = apply(prostmat, 1, function(x) var(x[51:102]))
sigma2 = s02/50 + s12/52
alpha = 0.05
```

Code is not displayed, but carries out Gibbs sampling for the mixture model described in class.

Prostate Cancer Example:



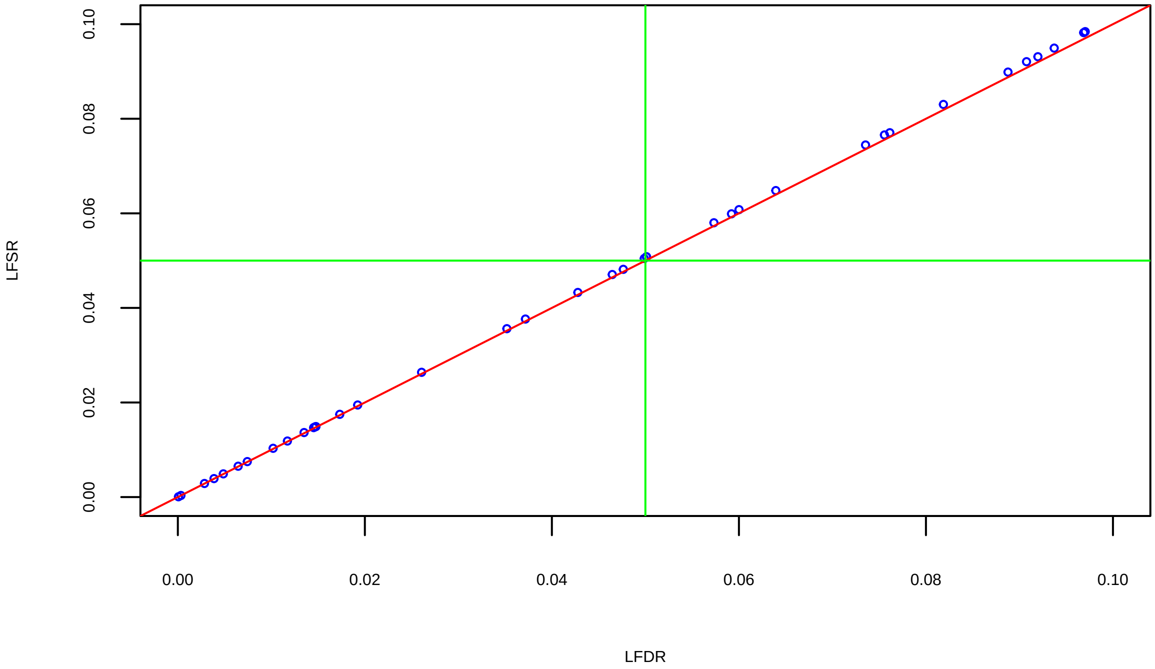
Local false sign rate approach

```
out = ash.workhorse(betahat = Y, sebetahat = sqrt(sigma2),
    mixcompdist = c("normal"), method = c("fdr"))
result = out$result
# it is a large dataframe containing
# positive probability, negative
# probability, lfsr, lfdr ect.
names(result)
##  [1] "betahat"       "sebetahat"     "NegativeProb"  "PositiveProb"
##  [5] "lfsr"          "svalue"        "lfdr"          "qvalue"
##  [9] "PosteriorMean" "PosteriorSD"
# out$fitted_g for fitted model
out$fitted_g$pi[1]  # Posterior mass on null component
## [1] 0.839573
```

```
head(result)
##        betahat sebetahat NegativeProb PositiveProb      lfsr     svalue
## 1  0.394234285 0.2656347  0.029291126   0.15911902 0.8408810 0.67579181
## 2  0.703227359 0.1900733  0.001392831   0.90157430 0.0984257 0.04354673
## 3 -0.006046081 0.2173035  0.056465210   0.05461837 0.9435348 0.86172602
## 4 -0.239860003 0.2106722  0.122247036   0.02933164 0.8777530 0.74093749
## 5 -0.033913878 0.2412444  0.063317423   0.05393859 0.9366826 0.85043901
## 6  0.215282878 0.2234540  0.033468411   0.10730389 0.8926961 0.76868139
##         lfdr     qvalue PosteriorMean PosteriorSD
## 1 0.81158985 0.65031517  0.0350449209  0.10830331
## 2 0.09703287 0.04299731  0.4186750967  0.20706512
## 3 0.88891642 0.80947053 -0.0003780271  0.05434714
## 4 0.84842132 0.71902507 -0.0211092735  0.08025763
```

```
## 5 0.88274398 0.79390938 -0.0020380021  0.05940802
## 6 0.85922770 0.74144354  0.0167199679  0.07494263
```

```
plot(result$lfsr ~ result$lfdr, xlim = c(0,
    0.1), ylim = c(0, 0.1), col = "blue",
    xlab = "LFDR", ylab = "LFSR", cex.axis = 0.5,
    cex.lab = 0.5, cex = 0.5)
abline(0, 1, col = "red")
abline(0.05, 0, col = "green")
abline(v = 0.05, col = "green")
```



```
# plot the sorted local false sign
# rate. lower the better
flag.lfsr = which(result$lfsr < 0.1)
length(flag.lfsr)
## [1] 38
# flag.lfsr
flag.lfdr = which(result$lfdr < 0.1)
length(flag.lfdr)
## [1] 38
# flag.lfdr
flag.lfsr2 = which(result$lfsr < 0.05)
length(flag.lfsr2)
## [1] 21
# flag.lfsr2
flag.lfdr2 = which(result$lfdr < 0.05)
length(flag.lfdr2)
## [1] 22
# flag.lfdr2
```