General Sample Size and Power Analysis for high-dimensional genomic data

Maarten van Iterson and Renée de Menezes
Center for Human and Clinical Genetics,
Leiden University Medical Center, The Netherlands
Package *SSPA*, version 2.14.0

Modified: 22 October, 2013. Compiled: October 17, 2016

Contents

1 Introduction

Power and sample size analysis or sample size determination is concerned with the question of determining the minimum number of samples necessary to demonstrate the existence (or absence) of a difference between two or more populations of interest. The number of samples should be sufficient in that the statistical test will reject the null hypothesis, when there really exists a difference, with high probability or power.

Sample size determination for experiments involving high-dimensional data has several challenges as a multiple testing problem is involved, furthermore the occurrence of differentially and nondifferentialy expressed genes and that each gene individually has an effect size and a standard deviation leading to a distribution of effect sizes and standard deviations complicates the problem even further.

Power and sample size analysis for high-dimensional data was initiated by [**?**]. The multiple testing problem was controlled using the easy to apply *family-wise error rate*. Since controlling the family-wise error rate is too conservative for microarray data, methods were proposed that control the multiple testing problem using the false discovery rate [**?**, **?**, **?**]. Recently, those methods were adapted for using pilot-data in order to obtain more realistic estimates of the power [**?**, **?**, **?**].

This vignette describes *SSPA*, a package implementing a general pilot data-based method for power and sample size determination for high-dimensional genomic data, such as those obtained from microarray or next-generation sequencing experiments. The method is based on the work of Ferreira and Zwinderman [**?**] and generalized as described by van Iterson *et al.* [**?**, **?**].

By means of two simple commands (`pilotData()`, `sampleSize()`), a researcher can read in their data –a vector of test statistics– and compute the desired estimates. Other functions are provided that facilitate interpretation of results. Simulation data is used to show the general functionality and flexibility of the package and two interesting biological case study are shown.

2 Simulated data

This simulation study represents a two group microarray experiment with $m = 5000$ features and $N = 10$ samples per group. Let R_{ij} and S_{ij}, $i = 1, \ldots, m$, $j = 1, \ldots, N$ be random variables where $S_{ij} \sim N(^{-}\mu_i/_2, 1)$ and $R_{ij} \sim N(\mu_i/_2, 1)$

with the first $\mu_i = 0$ for $i = 1, \ldots, m_0$ and the remaining μ_i for $i = m_0 + 1, \ldots, m$ were drawn from a symmetric bi-triangular distribution as described by [**?**]. A total of 25 data sets were simulated with the proportion of non-differentially expressed features fixed at $\pi_0 = 0.8$. Additional technical noise is added to the model using $X \sim \log(e^R + e^\epsilon)$ and $Y \sim \log(e^S + e^\epsilon)$ with $\epsilon \sim N(0, 0.1^2)$, so that the noise is positive and additive.

```
> library(SSPA)
> library(genefilter)
> set.seed(12345)
> m <- 5000
> J <- 10
> pi0 <- 0.8
> m0 <- as.integer(m*pi0)
> mu <- rbitri(m - m0, a = log2(1.2), b = log2(4), m = log2(2))
> data <- simdat(mu, m=m, pi0=pi0, J=J, noise=0.01)
> statistics <- rowttests(data, factor(rep(c(0, 1), each=J)))$statistic
```

2.1 Deconvolution

The first step in doing the sample size and power analysis is creating a object of class *PilotData* which will contain all the necessary information needed for the following power and sample size analysis; a vector of test-statistics and the sample sizes used to compute them. A user-friendly interface for creating an object of *PilotData* is available as `pilotData()`.

Several ways of viewing the content of the PilotData object are possible either graphically or using a show-method by just typing the name of the created object of *PilotData*:

```
> pdD <- pilotData(statistics = statistics,
+                  samplesize = sqrt(1/(1/J +1/J)),
+                  distribution="norm")
> pdD
```

```
Formal class 'PilotData' [package "SSPA"] with 5 slots
  ..@ statistics  : num [1:5000] -1.98854 -0.00589 -1.26628 0.74421 -0.76088 ...
  ..@ samplesize  : num 2.24
  ..@ pvalues     : num [1:5000] 0.0468 0.9953 0.2054 0.4567 0.4467 ...
  ..@ distribution: chr "norm"
  ..@ args        : list()
```

```
> plot(pdD)
```

Now we can create an object of class *SampleSize* which will perform the estimation of the proportion of non-differentially expressed genes and the density of effect sizes. Several options are available see `?sampleSize`.

The density of effect size can be shown by a call to `plot()`. When there are both up- and down-regulated genes a bimodal density is observed.

```
> ssD <- sampleSize(pdD, control=list(from=-6, to=6))
> ssD
```

```
Formal class 'SampleSize' [package "SSPA"] with 10 slots
  ..@ pi0         : num 0.781
  ..@ lambda      : num [1:512] 0 0 0 0 0 0 0 0 0 0 ...
  ..@ theta       : num [1:512] -6 -5.98 -5.95 -5.93 -5.91 ...
  ..@ control     :List of 11
  .. ..$ method    : chr "deconv"
  .. ..$ pi0Method : chr "Langaas"
  .. ..$ pi0       : num [1:90] 0.1 0.11 0.12 0.13 0.14 0.15 0.16 0.17 0.18 0.19 ...
  .. ..$ adjust    : logi TRUE
  .. ..$ a         : num 0.5
```

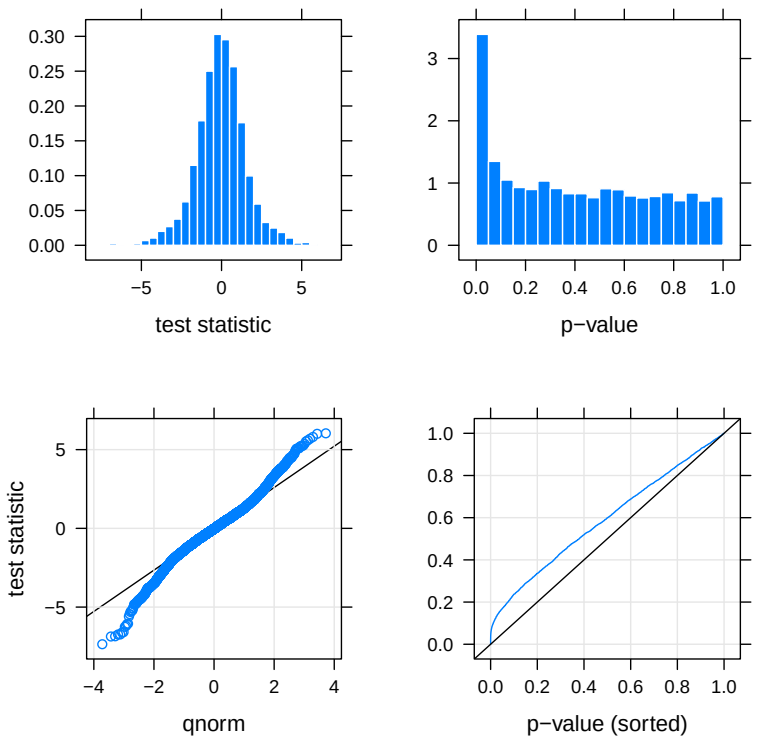

Figure 1: **Exploratory plots of the pilot-data:** Upper left panel shows a histogram of the test statistics, upper right panel a histogram of the p-values. Lower left panel shows a qq-plot for the test statistics using the user-defined null distributions. Lower right panel shows the sorted p-values against their ranks.

```
  .. ..$ bandwith  : NULL
  .. ..$ kernel    : chr "fan"
  .. ..$ from      : num -6
  .. ..$ to        : num 6
  .. ..$ resolution: num 512
  .. ..$ verbose   : logi FALSE
  ..@ info        :List of 1
  .. ..$ pi0: num 0.767
  ..@ statistics  : num [1:5000] -1.98854 -0.00589 -1.26628 0.74421 -0.76088 ...
  ..@ samplesize  : num 2.24
  ..@ pvalues     : num [1:5000] 0.0468 0.9953 0.2054 0.4567 0.4467 ...
  ..@ distribution: chr "norm"
  ..@ args        : list()
```

```
> plot(ssD, panel = function(x, y, ...)
+      {
+        panel.xyplot(x, y)
+        panel.curve(dbitri(x), lwd=2, lty=2, n=500)
+      }, ylim=c(0, 0.6))
```

Estimating the average power for other sample sizes then that of the pilot-data can be performed with the `predictpower()`-function. The user can also give the desired false discovery rate level.

```
> Jpred <- seq(10, 20, by=2)
> N <- sqrt(Jpred/2)
```

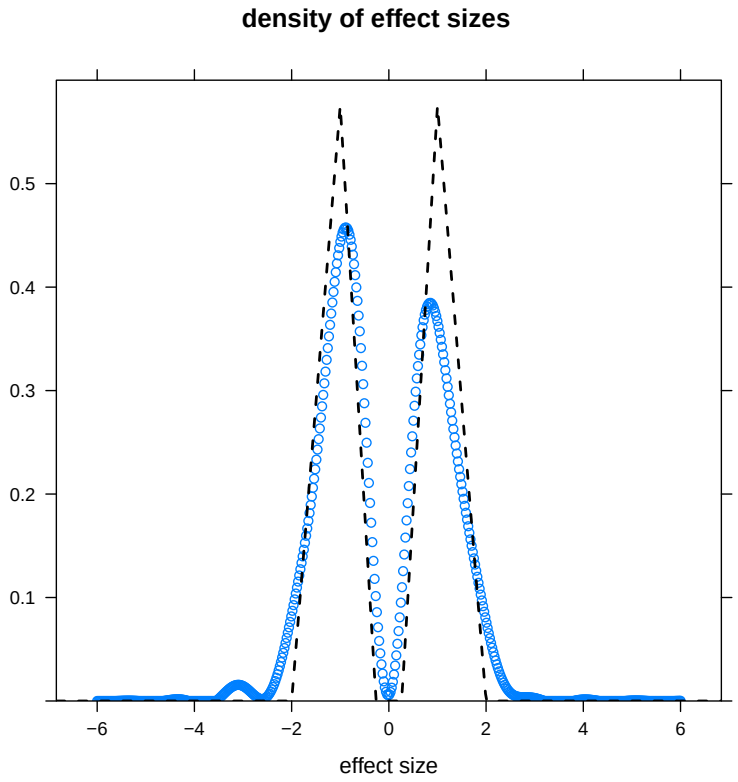

Figure 2: **Estimated density of effect sizes:** True density of effect sizes, the bitriangular distribution, and estimated density of effect sizes in blue.

```
> pwrD <- predictpower(ssD, samplesizes=N, alpha=0.05)
> matplot(Jpred, pwrD, type="b", pch=16, ylim=c(0, 1),
+         ylab="predicted power", xlab="sample size (per group)")
> grid()
```

The deconvolution estimator of the density of effect sizes is only applicable for two-group comparisons when the test statistics is (approximate) normally distributed. Note that in this situation we use the effective sample size. In all other cases e.g. two group comparison using the t statistics with the Conjugate Gradient estimator for the density of effect sizes the total sample size is used.

2.2 Conjugate Gradient

```
> pdC <- pilotData(statistics = statistics,
+                  samplesize = sqrt(2*J),
+                  distribution="t",
+                  df=2*J-2)
> ssC <- sampleSize(pdC,
+                   method="congrad",
+                   control=list(from=-6, to=6, resolution=250))
> plot(ssC, panel = function(x, y, ...)
+      {
+        panel.xyplot(x, y)
+        panel.curve(2*dbitri(2*x), lwd=2, lty=2, n=500)
+      }, xlim=c(-2,2), ylim=c(0, 1.2))
```

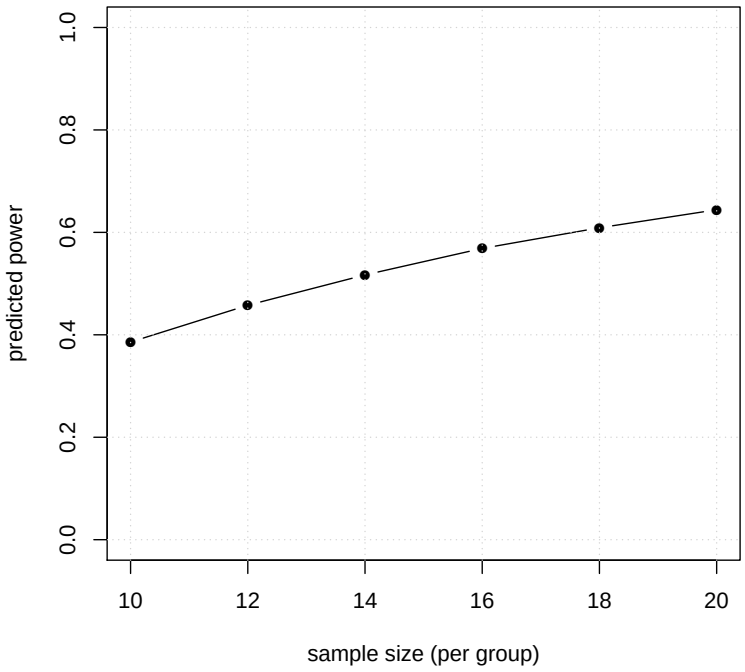

Figure 3: **Predicted power curve:** For sample sizes from 10 to 20 the power is predicted based on the information obtained from the pilot-data.

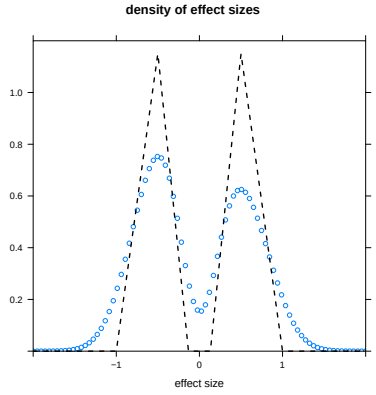

Figure 4: **Conjugate gradient method:** Estimated density of effect sizes using the conjugate gradient method.

2.3 Tikhonov regularization

```
> ssT <- sampleSize(pdC,
+                   method="tikhonov",
+                   control=list(resolution=250,
+                     scale="pdfstat",
+                     lambda = 10^seq(-10, 10, length=50),
+                     verbose=FALSE,
+                     modelselection="lcurve"))
```

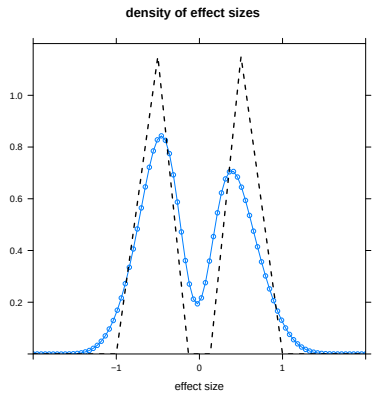

Figure 5: **Tikohonov regularization:** Estimated density of effect sizes using the Tikohonov regularization.

```
> plot(ssT, panel = function(x, y, ...)
+      {
+        panel.xyplot(x, y, type="b")
+        panel.curve(2*dbitri(2*x), lwd=2, lty=2, n=500)
+      }, xlim=c(-2,2), ylim=c(0, 1.2))
```

3 Case Study: Nutrigenomics microarray data

In this first example a set consisted of Affymetrix array data derived from a nutrigenomics experiment in which weak, intermediate and strong PPARα agonists were administered to wild-type and PPARα-null mice is used. The power and sample size analysis confirms the hierarchy of PPARα-activating compounds previously reported and the general idea that larger effect sizes positively contribute to the average power of the experiment.

PPARα is a transcription factor that is activated upon binding by a variety of agonists, both of synthetic and natural origin. In this experiment the outcome of specific PPARα activation on murine small intestinal gene expression was examined using Affymetrix GeneChip Mouse 430 2.0 arrays. PPARα was activated by several PPARα-agonists that differed in activating potency. Data of three agonists were used, namely Wy14,643, fenofibrate and trilinolenin (C18:3). The first two compounds belong to the fibrate class of drugs that are widely prescribed to treat dyslipidemia, whereas trilinolenin is an agonist frequently found in the human diet. For intestinal PPARα, Wy14,643 is the most potent agonist followed by C18:3 and fenofibrate. Since time of exposure also affects the effect size, intestines were collected 6 hrs (all three agonists) or 5 days (Wy14,643 and fenofibrate only) after exposure. Expression estimates of probesets were obtained by GC-robust multi-array (GCRMA) analysis, employing the empirical Bayes approach for background adjustment, followed by quantile normalization and summarization. For each compound and exposure time, lists of moderated t-test statistics were computed, using the empirical Bayes linear regression model as implemented in *limma*, for each contrast representing the effect of compound in PPARα-null mice compared to wild-type mice. For more details see [**?**].

```
> library(lattice)
> data(Nutrigenomics)
> str(Nutrigenomics)
```

```
'data.frame':        16540 obs. of  5 variables:
 $ wy5d  : num  2 1.22 1.19 -1.14 1 0.86 4 -1.83 -2.64 3.04 ...
 $ feno5d: num  2 -1.03 -1.11 0.33 0.22 0.22 0.24 0.36 1.73 0.54 ...
 $ X1836h: num  2.5 -2.3 -0.56 0.06 -0.89 -0.09 2 -2.12 -1.47 -1.69 ...
 $ wy6h  : num  2 1.12 0.21 -0.68 -1.06 -0.4 1.82 -1.71 1.2 1.03 ...
 $ feno6h: num  2.22 0.76 -0.37 -0.07 -1.9 ...
```

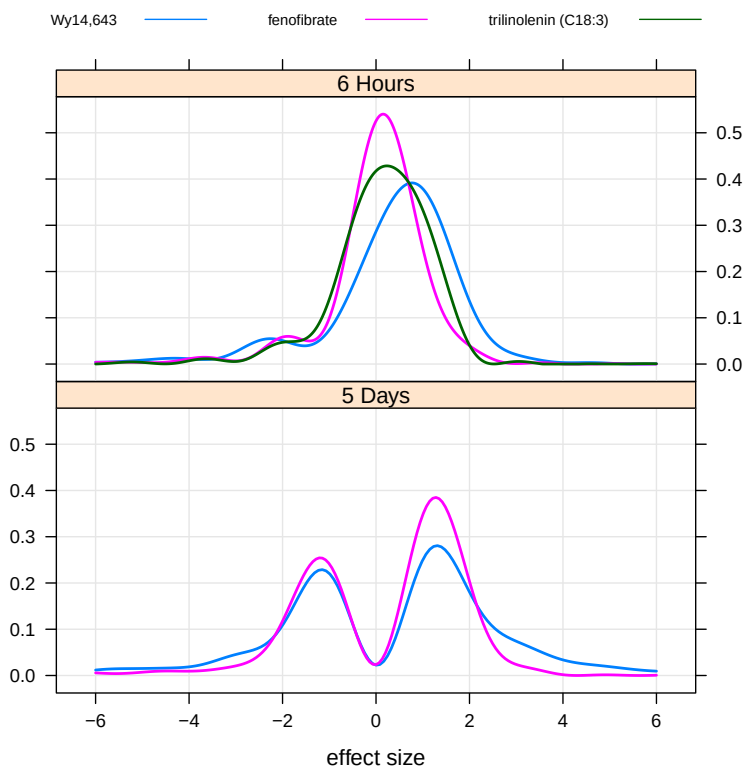

Figure 6: **Nutrigenomic example:** Estimated density of effect sizes for the five treatment exposure conditions.

```
> pd <- apply(Nutrigenomics, 2,
+             function(x) pilotData(statistics=x[-1],
+                                   samplesize=sqrt(x[1]),
+                                   distribution="norm"))
> ss <- lapply(pd, sampleSize,
+              control=list(pi0Method="Storey", a=0, resolution=2^10, verbose=FALSE))
> ##ss <- lapply(pd, sampleSize,
> ##              method = "congrad",
> ##              control=list(verbose=FALSE, resolution=2^10, from=-10, to=10))
>
> compounds <- c("Wy14,643",  "fenofibrate", "trilinolenin (C18:3)", "Wy14,643", "fenofibrate")
> exposure <- c("5 Days", "6 Hours")
> effectsize <- data.frame(exposure = factor(rep(rep(exposure, c(2, 3)), each=1024)),
+                          compound = factor(rep(compounds, each=1024)),
+                          lambda = as.vector(sapply(ss,
+                            function(x)x@lambda)),
+                          theta = as.vector(sapply(ss,
+                            function(x)x@theta)))
> print(xyplot(lambda~theta|exposure, group=compound, data=effectsize,
+              type=c("g", "l"), layout=c(1,2), lwd=2, xlab="effect size", ylab="",
+              auto.key=list(columns=3, lines=TRUE, points=FALSE, cex=0.7)))

> samplesize <- seq(2, 8)
> averagepower <- data.frame(power = as.vector(sapply(ss,
+                              function(x) as.numeric(predictpower(x, samplesize=sqrt(samplesize))))),
+                            exposure = factor(rep(rep(exposure, c(2, 3)), each=length(samplesize))),
```

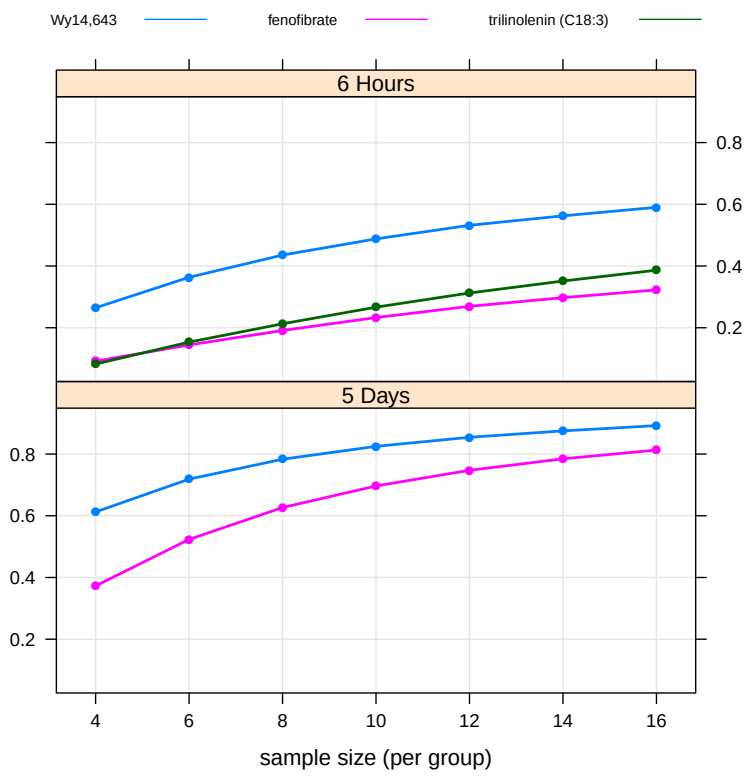

Figure 7: **Nutrigenomic example:** Predicted power curves for the five treatment exposure conditions.

```
+                          compound = factor(rep(compounds, each=length(samplesize))),
+                          samplesize = rep(2*samplesize, 5))
> print(xyplot(power~samplesize|exposure, group=compound, data=averagepower, type=c("g", "b"),
+              layout=c(1,2), lwd=2, pch=16, xlab="sample size (per group)", ylab="",
+              auto.key=list(columns=3, lines=TRUE, points=FALSE, cex=0.7)))
```

4 Case Study: deepSAGE of wild-type vs Dclk1 transgenic mice

In this example we will show how our method can be applied to count data of an RNA-seq experiment. We will use the data described by [**?**] comparing gene expression profiles in the hippocampi of transgenic δC-doublecortin-like kinase mice with that of wild-type mice. Data is available from GEO (GSE10782).

and analyzed using the generalized linear model approach implemented in *edgeR*. A tag-wise dispersion parameter was estimated using the Cox-Reid adjusted profile likelihood approach for generalized linear models as implemented in *edgeR*. Using a treatment contrast, we tested per tag if there was a genotype effect using the likelihood ratio statistic. This test statistic follow asymptotically a χ^2 distribution with one degree of freedom.

```
> ##files contains the full path and file names of each sample
> targets <- data.frame(files=files,
+                       group=rep(c("DCLK", "WT"), 4),
+                       description=rep(c("transgenic (Dclk1) mouse hippocampus",
+                         "wild-type mouse hippocampus"), 4))
> d <- readDGE(targets) ##reading the data
> ##filter out low read counts
```

```
> cpm.d <- cpm(d)
> d <- d[rowSums(cpm.d > 1) >= 4, ]
> design <- model.matrix(~group, data=d$samples)
> ##estimate dispersion
> disp <- estimateGLMCommonDisp(d, design)
> disp <- estimateGLMTagwiseDisp(disp, design)
> ##fit model
> glmfit.hoen <- glmFit(d, design, dispersion = disp$tagwise.dispersion)
> ##perform likelihood ratio test
> lrt.hoen <- glmLRT(glmfit.hoen)
> ##extract results
> tbl <- topTags(lrt.hoen, n=nrow(d))[[1]]
> statistics <- tbl$LR

> library(lattice)
> data(deepSAGE)
> str(deepSAGE)

 num [1:44882] 133.5 93.8 89.5 88.8 87.4 ...

> pd <- pilotData(statistics=deepSAGE,
+                 samplesize=8, distribution="chisq", df=1)
> ss <- sampleSize(pd, method="congrad",
+                  control=list(trim=c(0, 0.98), symmetric=FALSE, from=0, to=10))
> pwr <- predictpower(ss, samplesize=c(8, 16, 32))
> op <- par(mfcol=c(2,1), mar=c(5,4,1,1))
> plot(ss@theta, ss@lambda,
+      xlab="effect size", ylab="", type="l")
> plot(c(8, 16, 32), pwr,
+      xlab="sample size", ylab="power", type="b", ylim=c(0,1))
> grid(col=1)
> par(op)
```

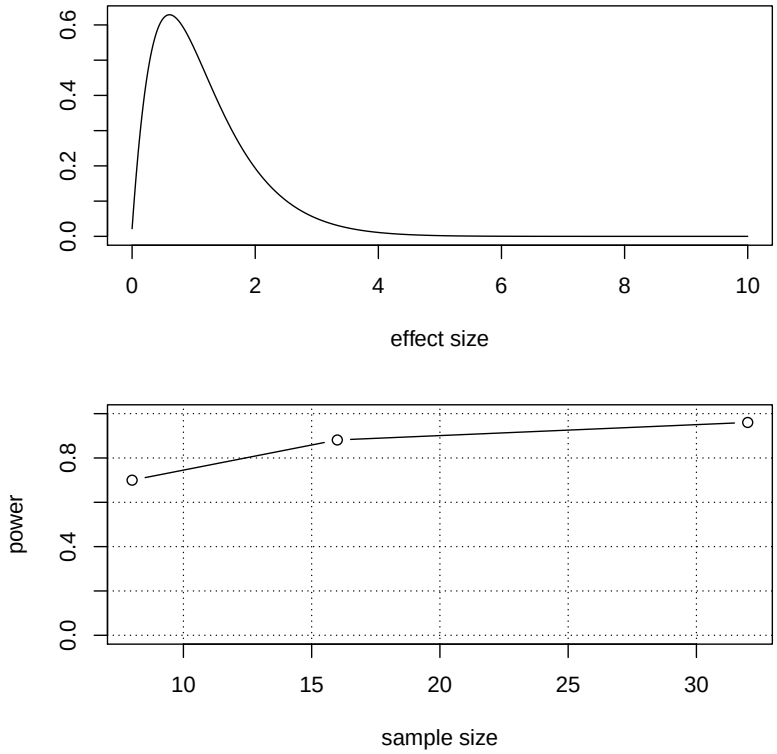

Figure 8: **Deep SAGE example:** Estimated density of effect size and predicted power curve.

5 Session info

The version number of R and packages loaded for generating the vignette were:

- R version 3.3.1 (2016-06-21), `x86_64-apple-darwin13.4.0`
- Locale: `C/en_US.UTF-8/en_US.UTF-8/C/en_US.UTF-8/en_US.UTF-8`
- Base packages: base, datasets, grDevices, graphics, methods, stats, utils
- Other packages: SSPA 2.14.0, genefilter 1.56.0, lattice 0.20-34, limma 3.30.0, qvalue 2.6.0
- Loaded via a namespace (and not attached): AnnotationDbi 1.36.0, Biobase 2.34.0, BiocGenerics 0.20.0, BiocStyle 2.2.0, DBI 0.5-1, IRanges 2.8.0, Matrix 1.2-7.1, RCurl 1.95-4.8, RSQLite 1.0.0, Rcpp 0.12.7, S4Vectors 0.12.0, XML 3.98-1.4, annotate 1.52.0, bitops 1.0-6, colorspace 1.2-7, ggplot2 2.1.0, grid 3.3.1, gtable 0.2.0, magrittr 1.5, munsell 0.4.3, parallel 3.3.1, plyr 1.8.4, reshape2 1.4.1, scales 0.4.0, splines 3.3.1, stats4 3.3.1, stringi 1.1.2, stringr 1.1.0, survival 2.39-5, tools 3.3.1, xtable 1.8-2