ShrinkBayes: Bayesian analysis of high-dimensional omics data

Mark A. van de Wiel

November 23, 2012

Department of Epidemiology & Biostatistics
VU University Medical Center
Amsterdam, The Netherlands

`mark.vdwiel@vumc.nl`

Contents

1 Overview

ShrinkBayes is a package for Bayesian (differential) expression analysis of high-dimensional -omics data. It applies Emprical Bayes-type *multi*-parameter shrinkage to improve parameter estimation and inference. You should use it because it

- Is very flexible in terms of study designs
- Allows for random effects in a GLM setting
- Applies to many different data types, including Gaussian (e.g. miRNA, mRNA arrays, HT RNAi), counts (e.g. label-free proteomics) and zero-inflated counts (e.g. AGE, RNAseq)
- Was demonstrated to be more reproducible and powerful than other methods in small-sample settings (Van de Wiel et al., 2012, 2013)
- Is much more computationally efficient than MCMC, because it makes use of INLA
- Provides (Bayesian) False Discovery Rates.

This document provides an overview on the usage of the ShrinkBayes package. For more detailed information on the methodology, performance and assumptions we refer to the articles (Van de Wiel et al., 2012, 2013). As example data we attached three data sets: 1) Simulated Gaussian data with 1500 rows and 284 columns; 2) HTRNAi, a data set with 960 rows and 6 columns containing Gaussian normalized HT RNAi data; and 3) CAGEdata1000, with 10,000 rows (features) and 25 columns (samples) containing normalized sequencing-based counts.

The simulated data set is used to illustrate the following aspects of ShrinkBayes:

- Ease of use in plain 2-group setting
- Proof of concept: ShrinkBayes correctly identifies the prior effect size distribution
- How to use a parametric mixture prior with point mass at zero?

The HT RNAi data set is used to illustrate the following aspects of ShrinkBayes:

- Analysis of Gaussian data (hence this example also covers most microarray data)
- Use of an offset in your model
- How to shrink multiple fixed effects and error variance?
- Use of a non-symmetric prior

The CAGE data set is used to illustrate the following aspects of ShrinkBayes:

- Analysis of (zero-inflated) count data (example also covers other sequencing data like RNAseq)
- Dealing with a complex design, including nuisance factors and blocked individuals
- Modeling and analysing random effects
- Estimation and inference for multiple pair-wise comparisons
- Joint shrinkage of fixed effect, random effect and overdispersion
- Using a mixture distribution on overdispersion

All examples use a fairly similar same work flow:

1. Load the normalized data
2. Set up the model and study design, possibly including multiple comparisons
3. Apply joint iterative procedure to shrink multiple parameters
4. Fit models for all data rows using the shrunken priors (
5. Combine posteriors when multiple models have been applied to the same data (e.g. Poisson and Negative Binomial)
6. Update priors for crucial parameters to non-parametric or mixture forms
7. Update posteriors accordingly using the fitted non-parametric or mixture priors
8. Compute summaries like posterior means (parameter estimates) or posterior tail probabilities (local false discovery rates)
9. Compute Bayesian False Discovery Rates

For the analysis of the two real data sets we focus on illustrating the settings as used in (Van de Wiel et al., 2013, HT RNAi data) and (Van de Wiel et al., 2012, CAGE data), but will also discuss alternatives when appropriate.

2 Example 1: Gaussian simulation setting, 2-group setting

2.1 Workflow

The example data set contains 1500 rows (features) and 8 samples (divided in two groups of 4). The first 1000 represent non-differential features (so proportion non-differential equals 2/3), the last 500 represent differential features with mean group difference simulated from a $N(0,1)$ distribution. The noise is simulated from a $N(0,0.5^2)$ distribution. See Section 6.1 for the code used to simulate the data.

```
> library(ShrinkBayes)
> data(datsim)
> head(datsim)
```

```
            [,1]         [,2]       [,3]        [,4]       [,5]        [,6]
[1,] -0.04191917  0.070037754  0.8198383 -0.35443640  0.1128436 -0.19965003
[2,] -1.01134453 -0.235494343 -0.3670104 -0.53177746  0.3673819 -0.03100180
[3,]  0.19333209 -0.008211341 -0.1544538  0.29010757 -0.1937141 -0.01311311
[4,] -1.03722998 -0.391160705 -0.2452836 -0.98763216 -0.8069727 -0.12469640
[5,] -0.40153756  0.040684804  0.5219447 -0.09762872  0.1809304 -0.23406040
[6,] -0.82760690  0.706451170  1.0892873  0.55709375  0.3335727  1.01849605
           [,7]        [,8]
[1,]  0.5394400 -0.57643251
[2,] -0.1966407 -0.20431000
[3,]  0.2264545  0.01468320
[4,] -0.4004245  0.03656696
[5,] -0.5077155 -0.60289124
[6,]  0.5251242 -0.02278950
```

Loads the package and the data.

```
> ncpus2use <- 10
```

Sets the number of cpus to use for parallel computations. Combined with the `ncpus=ncpus2use` argument in the functions this fixes the number of cpus used. Note that for all functions the default is 2. If ncpus2use is larger than the actual number of available cpus, computations will still run.

```
> group <- factor(c(rep("group1", 4), c(rep("group2", 4))))
```

Defines covariate “group”, corresponding to the columns of the datsim data.

```
> form = y ~ 1 + group
```

Define the model formula. Specification should be according to the `inla` `formula` argument.

```
> shrinksimul <- ShrinkGauss(form = form, dat = datsim, shrinkfixed = "group",
+     ncpus = ncpus2use)
```

This function simultaneously shrinks the fixed effect parameter ‘group’ and the Gaussian error standard deviation (default), using standard parametric priors. The function may take considerable computing time. See Section 3 for discussion on a) some arguments of the ShrinkGauss function b) retrieving various types of information from `shrinksimul`.

```
> fitg <- FitAllShrink(form, dat = datsim, fams = "gaussian", shrinksimul,
+     ncpus = ncpus2use)
```

Applies `inla` to compute posteriors for all data rows using the priors stored in `shrinksimul`. IMPORTANT: for better performance in the next function (`MixtureUpdatePrior`), the prior for the `shrinkfixed` parameter as used in `ShrinkGauss` (here: “treatment”) is by default dispersed by a factor 10 (hence a vaguer prior is used), see Section 3.

```
> mixtpriorinit <- MixtureUpdatePrior(fitall = fitg, modus = "gauss",
+     shrinkpara = "group", ncpus = ncpus2use, p0vec = c(0.25,
+         0.4, 0.6, 0.8, 0.9, 0.95, 0.99), sdvec = c(0.2, 0.3,
+         0.4, 0.5, 0.75, 1))
```

Finds the (approximately) best mixture prior of the form

$$\pi(\beta) = p_0\delta_0 + (1 - p_0)N(\beta; 0, \tau^2), \tag{1}$$

so a mixture of zero-point mass (group effect equals zero) and a central Normal distribution. Grid search is used to maximize total log marginal likelihood. Inclusion of a point mass may be attractive for the purpose of statistical inference (Van de Wiel et al., 2012).

```
> best <- mixtpriorinit$allparam[1, ]
> best

       p0     stdev sumloglik
    0.600     1.000  1014.987
```

Displays the best values of the parameters. We use those to refine the grid search. In our experience, one refinement is usually sufficient.

```
> mixtprior <- MixtureUpdatePrior(fitall = fitg, modus = "gauss",
+     shrinkpara = "group", ncpus = ncpus2use, p0vec = seq(0.5,
+         0.7, by = 0.01), sdvec = seq(0.8, 1.5, by = 0.1))

> bestfinal <- mixtprior$allpara[1, ]
> bestfinal

       p0     stdev sumloglik
    0.600     0.900  1019.843
```

The final best values of the parameters. Note the good performance of the algorithm: the true values are `p0 = 0.67` and `= 1`

```
> mixtpostshr <- MixtureUpdatePosterior(fitg, mixtprior, ncpus = ncpus2use)
```

Update the posteriors of the `"group"` parameter for all data rows using the mixture prior.

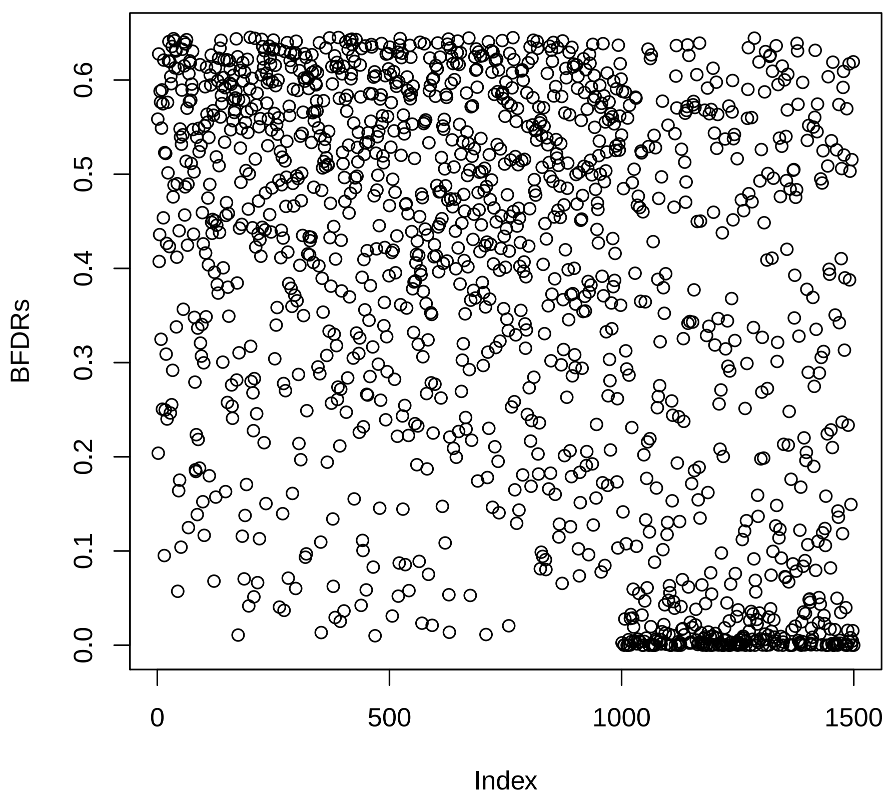
```
> lfdrless <- SummaryWrap(mixtpostshr, thr = 0, direction = "lesser")
> lfdrgreat <- SummaryWrap(mixtpostshr, thr = 0, direction = "greater")
```

Compute posterior (tail-)probabilities under shrinkage prior. Here `lfdrless = `$P(\beta_{\text{group}} \leq 0|Y)$ and `lfdrgreat = `$P(\beta_{\text{group}} \geq 0|Y)$ are computed. These can be interpreted as local false discovery rates corresponding to a one-sided interval null-hypothesis $H_0 : \beta_{\text{group}} \leq 0$ (for `lfdrless`) or $H_0 : \beta_{\text{group}} \geq 0$ (for `lfdrgreat`). See (Van de Wiel et al., 2012). Practically: if for a given feature `min(lfdrless,lfdrgreat)` is small, it means that the posterior is largely concentrated on either side of zero, wich indicates that the group effect is different from 0. This fact is used when computing our *two-sided* version of Bayesian False Dicovery Rate (Ventrucci et al., 2011; Van de Wiel et al., 2012, BFDR):

```
> BFDRs <- BFDR(lfdrless, lfdrgreat)
```

Computes two-sided BFDRs. These can be interpreted as false discovery rates (in a cumulative sense) when one would select all features with smaller or equal null-probabilities (lfdrs) than the given feature.

```
> plot(BFDRs)
```



Plots the BFDRs against the feature index. Here we know that the first 1000 are generated from the null-hypotheses, the latter 500 from the alternative (see Section 6.1).

The computations above assume that we would know the correct prior of β_{group} (since it is actually simulated from 1). Although it is fairly common to assume a fixed specific model in statistical computations, it is worthwhile to explore alternatives. `ShrinkBayes` offers two: 1) use a variety of parametric mixture models and study the robustness of the results against the parametric shape of the prior; 2) use a nonparametric prior. 1) is explored in the next Section, 2) in Sections 3 and 4.

2.2 Results when using a misspecified prior

Below we estimate a misspecified prior, namely a symmetric mixture of a zero point mass and two non-central Normals: $\pi(\beta) = (1-p_0)/2 * N(\beta; -\mu, \tau^2) + p_0\delta_0 + (1-p_0)/2 * N(\beta; \mu, \tau^2)$

```
> mixtpriorinit2gauss <- MixtureUpdatePrior(fitall = fitg, modus = "mixt",
+     shrinkpara = "group", ncpus = ncpus2use)
```

Here, we use default settings voor `p0vec`, `muvec` and `sdvec`.

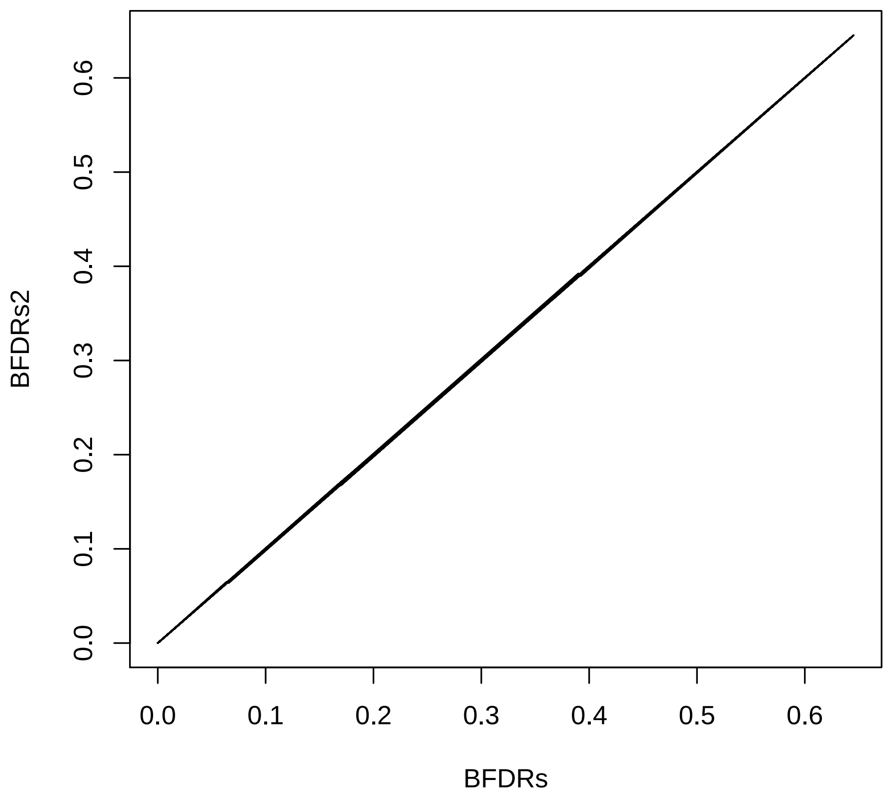
```
> best <- mixtpriorinit2gauss$allparam[1, ]
> best

       p0        mu     stdev sumloglik 
    0.600     0.500     0.750  1019.176 

> mixtprior2gauss <- MixtureUpdatePrior(fitall = fitg, modus = "mixt",
+     shrinkpara = "group", ncpus = ncpus2use, p0vec = seq(0.5,
+         0.7, by = 0.02), meanvec = seq(0.3, 0.5, by = 0.05),
+     sdvec = seq(0.8, 1.2, by = 0.1))

> bestfinal <- mixtprior2gauss$allpara[1, ]
> bestfinal

       p0        mu     stdev sumloglik 
    0.600     0.400     0.800  1019.670 
```

As in the previous Section we refine the search once to obtain the best parameter values. Let us now recompute the posteriors, lfdrs and BFDRs of β_{group} under this mixture prior:

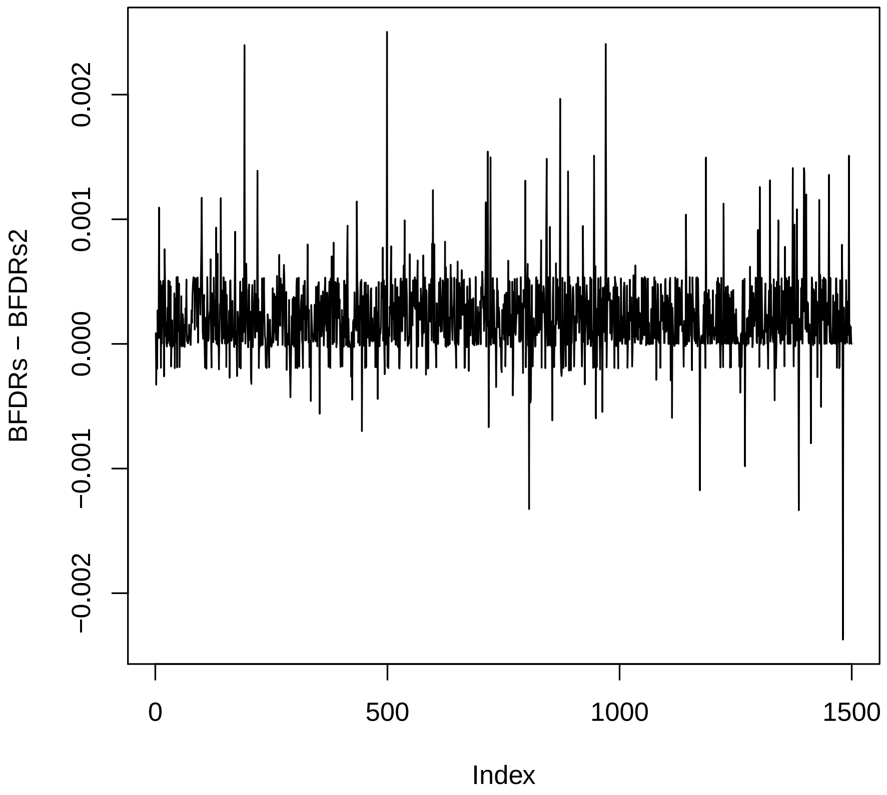
```
> mixtpostshr2 <- MixtureUpdatePosterior(fitg, mixtprior2gauss,
+     ncpus = ncpus2use)
> lfdrless2 <- SummaryWrap(mixtpostshr2, thr = 0, direction = "lesser")
> lfdrgreat2 <- SummaryWrap(mixtpostshr2, thr = 0, direction = "greater")
> BFDRs2 <- BFDR(lfdrless2, lfdrgreat2)
```

Finally we plot the BFDRS as computed under the correct prior (see previous Section; BFDRs) and under the misspecified prior (BFDRs2).

```
> plot(BFDRs, BFDRs2, type = "l")
```



```
> plot(BFDRs - BFDRs2, type = "l")
```



We observe they are very similar in this setting.

3 Example 2: Gaussian setting, HT RNAi data

```
> library(ShrinkBayes)
> data(HTRNAi)
> head(HTRNAi)
```

```
                 s1         s2         s3          s4         s5         s6
siRNA1  0.54317601  0.1953989  0.3883244 -0.03428689  0.6049207 -0.1790861
siRNA2  0.63533948  0.5057837  0.3928841  0.25338641  0.6935488  0.4185978
siRNA3 -0.91691878 -1.0838714 -0.6184123 -1.31090945 -0.2335765 -1.0377135
siRNA4 -0.74854963 -0.7619164 -0.2532474 -1.06233410 -0.5529374 -0.8189226
```

```
siRNA5  0.01110011 -0.6777245  0.3444957 -0.78924657  0.3554455 -0.8163929
siRNA6  0.59106644  0.5445666  0.4172433  0.31420140  0.6532168  0.5535300
```

Loads the package and the data

```
> ncpus2use <- 10
```

Sets the number of cpus to use for parallel computations.

```
> treatment <- factor(rep(c("untreated", "treated"), 3))
> assay <- factor(rep(1:3, each = 2))
```

Defines covariates "treatment" and "assay", corresponding to the columns of the HTRNAi data.

```
> offsetvalue <- c(0.1703984, -0.6958495, 0.3079694, -0.5582785,
+     0.225121, -0.6411269)
```

Defines an offset (often not needed). In this particular example, the offsets are computed from HT RNAi data with a positive control which is run for the same 6 screens. The offsets are posterior mean estimates from the same model as below (from many technical repeats per screen), but without shrinkage (because only one feature, the positive control, is involved). Use of an offset guarantees that the parameter estimates can be intepreted as deviation from the positive control.

```
> form = y ~ offset(offsetvalue) + 1 + treatment + assay
```

Define the model formula. Specification should be according to the `inla formula` argument.

```
> shrinksimul <- ShrinkGauss(form = form, dat = HTRNAi, shrinkfixed = "treatment",
+     shrinkaddfixed = "assay", fixedmeanzero = FALSE, ncpus = ncpus2use)
```

This function simultaneously shrinks the fixed effect parameters 'treatment' and 'assay'. Note that in this particular setting it is likely that many treatment parameters are negative (since this concerns an effect with respect to a positive control; see (Van de Wiel et al., 2013)). Therefore, we set `fixedmeanzero = FALSE`, so that we do not fix the mean of the prior to zero (which is the logical default in many microarray settings). In addition, the Gaussian error standard deviation is shrunken by default (`shrinksigma = TRUE`). This function may take considerable computing time.

The argument `shrinkfixed` contains the parameter of primary interest ("treatment"), while `shrinkaddfixed` contains an additional (nuisance) parameter ("assay"). Currently, maximally two fixed effect parameters can be shrunken simultaneously using Gaussian priors. `ShrinkGauss` has many additional arguments. We discuss a few crucial ones.

- `ntag`: Consecutive number of features used for shrinking. Important for computing time. Default is `ntag = c(100, 200, 500, 1000)`. You may consider using smaller values (e.g. `ntag = c(50, 100)`) for trying out, but we recommend to use at least `500` for final computations.

- `maxiter`: Maximum number of iteration per value of `ntag`. Important for computing time. Default is 10. Consider using a smaller value for trying out, e.g. `maxiter = 3`, but use at least `maxiter=10` for final computations.

- `fixedmeanzero` (`addfixedmeanzero`): Should the Gaussian mean of the `shrinkfixed` (`shrinkaddfixed`) prior be fixed to 0? Set to `FALSE` when this is undesirable/unrealistic.

```
> shrinksimul$pmlist

$mufixed
[1] -0.4220818

$precfixed
[1] 15.22325

$muaddfixed
[1] 0

$precaddfixed
[1] 88.85478

$shaperand
[1] 0.001
```

```
$raterand
[1] 0.001

$mixp
[1] 0.2 0.8

$shapeerr
[1] 14.80563

$rateerr
[1] 0.9963235
```

Shows the final parameter values. Note that the second parameter of Gaussian priors is a precision (1/variance) and the second parameter of Gamma priors (here, used for the precision of the error variance) is a rate.

```
> shrinksimul$paraall

            mufixed precfixed muaddfixed precaddfixed shaperand raterand mixp1
paraall  0.0000000  0.100000          0     0.100000     0.001    0.001   0.2
paranew -0.3989701  5.064296          0     6.797183     0.001    0.001   0.2
paranew -0.4052924  7.761454          0    13.148220     0.001    0.001   0.2
paranew -0.4076438  9.568388          0    19.454890     0.001    0.001   0.2
paranew -0.4068184 10.625845          0    25.330984     0.001    0.001   0.2
paranew -0.4057708 11.233065          0    31.305701     0.001    0.001   0.2
paranew -0.4056432 11.657412          0    36.714646     0.001    0.001   0.2
paranew -0.4039103 11.704554          0    42.434640     0.001    0.001   0.2
paranew -0.4056974 11.477593          0    47.533064     0.001    0.001   0.2
paranew -0.4020863 11.127391          0    52.697961     0.001    0.001   0.2
paranew -0.4064621 10.904471          0    57.091680     0.001    0.001   0.2
paranew -0.4314527 12.320179          0    62.504527     0.001    0.001   0.2
paranew -0.4441436 13.260152          0    66.302994     0.001    0.001   0.2
paranew -0.4469023 14.138179          0    70.433235     0.001    0.001   0.2
paranew -0.4465617 14.876808          0    74.257633     0.001    0.001   0.2
paranew -0.4225703 14.760978          0    78.835063     0.001    0.001   0.2
paranew -0.4204552 15.117292          0    82.778749     0.001    0.001   0.2
paranew -0.4220818 15.223251          0    88.854781     0.001    0.001   0.2
        mixp2  shapeerr   rateerr
paraall   0.8  0.001000 0.0010000  NA
paranew   0.8  3.257580 0.3124178 100
paranew   0.8  7.238098 0.7866031 100
paranew   0.8  9.330707 0.9598083 100
paranew   0.8 10.759581 1.0055100 100
paranew   0.8 11.647377 1.0074488 100
paranew   0.8 12.577559 1.0105678 100
paranew   0.8 13.350144 1.0119014 100
paranew   0.8 13.943717 1.0049484 100
paranew   0.8 14.487178 1.0043978 100
paranew   0.8 15.039027 1.0048728 100
paranew   0.8 14.440397 0.9817486 200
paranew   0.8 13.956728 0.9741182 200
paranew   0.8 13.712501 0.9746040 200
paranew   0.8 13.308867 0.9702096 200
paranew   0.8 13.960762 0.9920718 500
paranew   0.8 14.387572 0.9939723 500
paranew   0.8 14.805634 0.9963235 960
```

Shows the consecutive estimates of the parameters. Note that parameters not involved in the shrinkage are not updated. The last columns shows the number of features used to estimate the parameters of the priors.

```
> fitg <- FitAllShrink(form, dat = HTRNAi, fams = "gaussian", shrinksimul,
+     ncpus = ncpus2use)
```

Applies `inla` to compute posteriors for all data rows using the priors stored in `shrinksimul`. IMPORTANT: the prior for the `shrinkfixed` parameter as used in `ShrinkGauss` (here: “treatment”) is by default dispersed by a factor 10 (hence a vaguer prior is used). This is done, because we experienced that it leads to better results when updating this prior to a nonparametric one (using `NonParaUpdatePrior`) or a mixture one (using `MixtureUpdatePrior`). If you do not want to disperse the prior of the `shrinkfixed` parameter, use the argument `dispersefixed=1`. Similar function arguments are available to disperse the prior of the `shrinkaddfixed` parameter or (when available) the random effects parameter. Additional arguments you may want to consider are `showupdate` (default: `FALSE`): when set to `TRUE` progression updates are shown for each `updateby` finished features. However, this may slow down the computations somewhat.

NOTE: you may see a lot of warnings from the call to inla. These usually occur for difficult data rows; this may either lead to missing results (which the subsequent function can cope with) or very wide posteriors. In both cases, it is unlikely that this leads to false positives or false negatives.

The result of `FitAllShrink` is a 2-component list. The first component `$res` is a list of `inla`-output objects of length `nr = nrow(dat)`, the number of data rows. When `nr` is large, the result may be a large object. Hence, for memory-efficiency not all inla-output is included. See the function arguments `effoutput`, `keepmargrand` and `keepmarghyper` for other options. The second component `$priors` contains the input prior parameters.

```
> fit1 <- fitg$res[[1]]
```

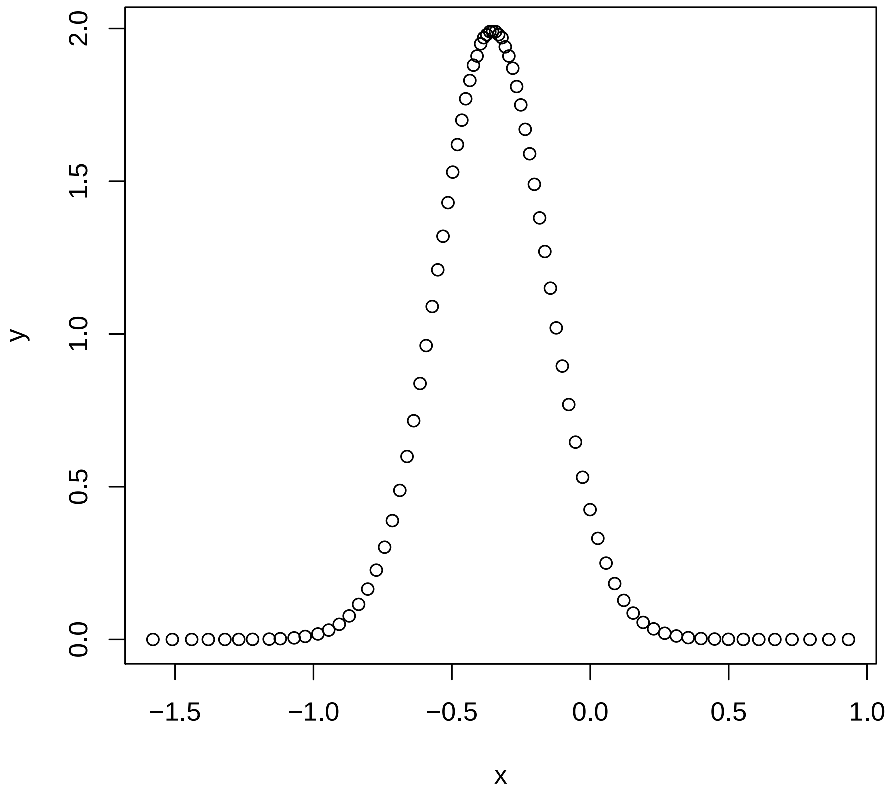
Retrieves the `inla`-output for the first data row. This contains a lot of information, including marginal posteriors of all relevant parameters in numeric format; summaries of the posteriors; model fit summaries; input-argument used by `inla`. Below we illustrate a few useful outputs.

```
> fit1$summary.fixed
```

```
                     mean    sd 0.025quant 0.5quant 0.975quant      kld
(Intercept)         0.758 0.192      0.378    0.758     1.1300 7.40e-32
treatmentuntreated -0.353 0.203     -0.752   -0.353     0.0463 0.00e+00
assay2             -0.247 0.223     -0.683   -0.248     0.1950 1.23e-32
assay3             -0.142 0.223     -0.578   -0.143     0.2990 0.00e+00
```

Shows the summaries of the fixed effect parameters.

```
> marginal <- fit1$marginals.fixed$treatmentuntreated
> plot(marginal)
```



Plots the marginal posterior (Use `type="l"` to obtain a curve instead of points). Note that this is obtained under a prior the variance of which is dispersed with a factor 10.

```
> fit1$summary.hyper
```

```
                                        mean   sd 0.025quant 0.5quant
Precision for the Gaussian observations 15.6 3.88       9.12     15.2
                                        0.975quant
Precision for the Gaussian observations       24.3
```

Shows the summaries of the hyper-parameters (parameters not involved in the regression formula). In this case only the error variance is a hyper-parameter.

```
> fit1$mlik

                                        [,1]
log marginal-likelihood (integration) -6.7
log marginal-likelihood (Gaussian)    -6.7
```

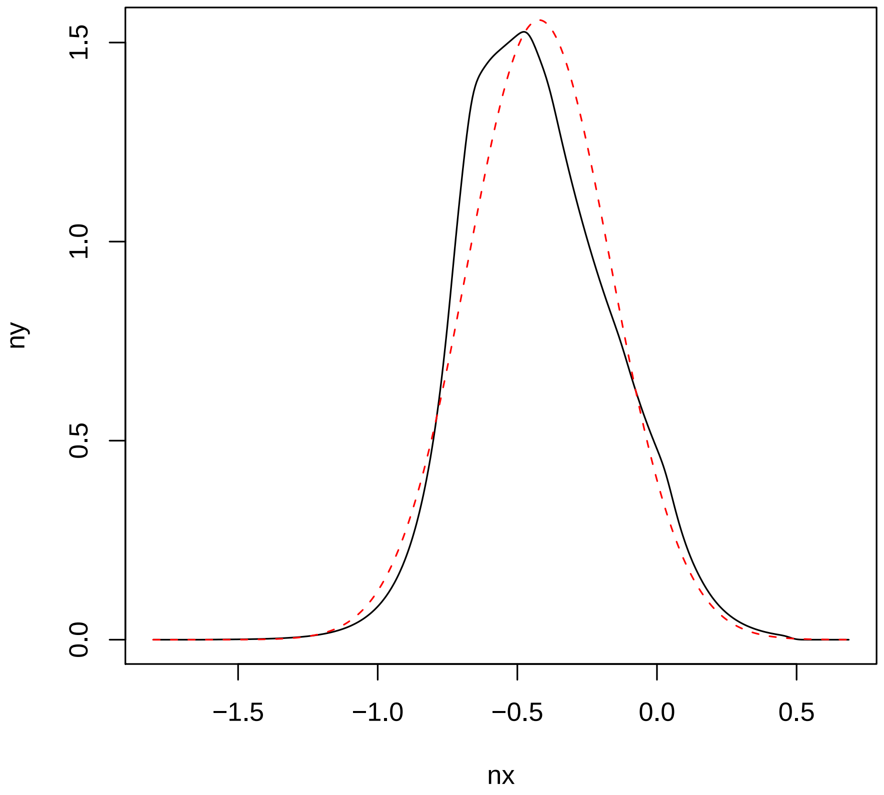
Estimation of the marginal likelihood. Can be used to compare models.

```
> npprior <- NonParaUpdatePrior(fitall = fitg, modus = "fixed",
+     shrinkpara = "treatment", ncpus = ncpus2use, symmetric = FALSE,
+     logconcave = TRUE)
```

Updates the (dispersed) Gaussian prior for `"treatment"` to a non-parametric one. Note here we specifically allow an asymmetric prior (default is `symmet-ric=TRUE`). This is context specific. See Van de Wiel et al. (2013) for explanation. Dropping the symmetry requirement may cause the non-parametric prior to become more unstable (its tails may become too dependent on specific data). Adding `logconcave=TRUE`, which forces a log-concave prior (Lutz and Rufibach, 2011), aids in increasing the stability.

Let us now compare the resulting non-parametric prior with the original Gaussian one.

```
> plot(npprior$priornew, type = "l")
> supp <- npprior$priornew[, 1]
> points(supp, dnorm(supp, mean = shrinksimul$pmlist$mufixed, sd = 1/sqrt(shrinksimul
+     type = "l", col = "red", lty = 2)
```



We observe a clear asymmetry and a difference in the tails.

```
> nppostshr <- NonParaUpdatePosterior(fitg, npprior, ncpus = ncpus2use)
```

Updates the posteriors of the `"treatment"` parameter using the new, non-parametric prior.

```
> lfdr <- SummaryWrap(nppostshr, thr = 0, direction = "lesser")
```

Here, we compute one-sided posterior null-probabilities of the kind `lfdr` $= P(\beta_{\text{treat}} \leq 0|Y)$, which can be interpreted as a local false discovery rate. One-sided, because, when comparing with a positive control, we are mainly interested in siRNAs with larger (hence positive) effects w.r.t. the positive control, so `lfdr` should be small.

```
> BFDRs <- BFDR(lfdr)
```

Computes Bayesian False Discovery Rates (Ventrucci et al., 2011) from the lfdrs.

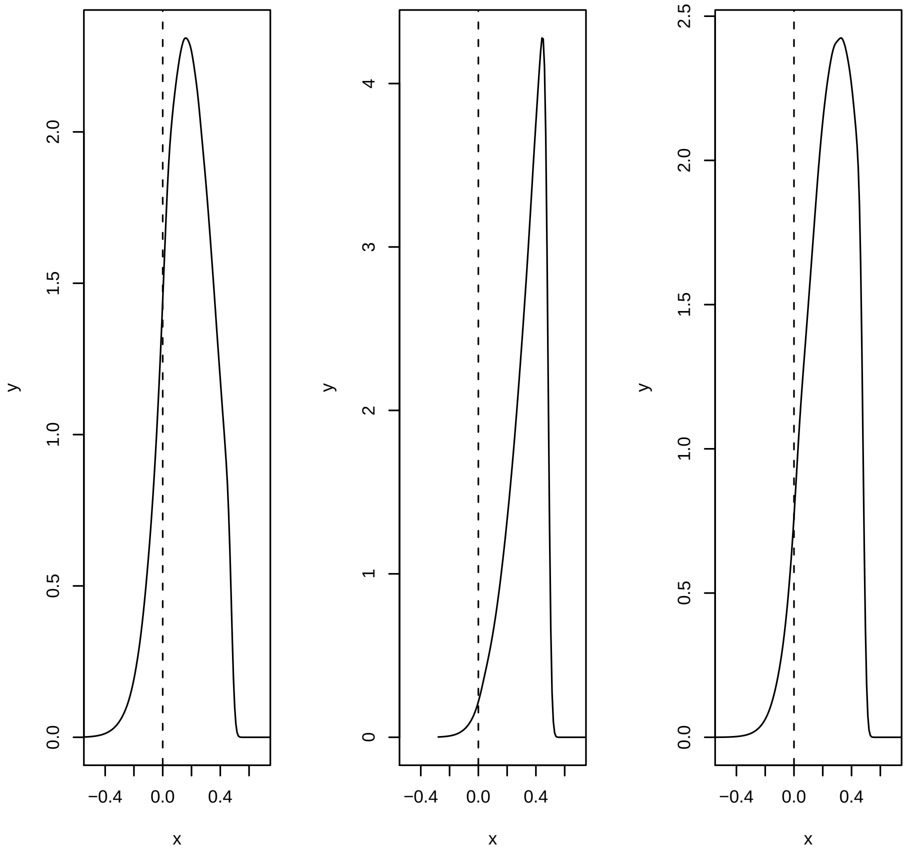
```
> whsig <- which(BFDRs <= 0.1)
> whsig
```

```
[1] 176 608 749
```

```
> BFDRs[whsig]
```

```
[1] 0.07658634 0.01426447 0.03969088
```

```
> layout(matrix(1:3, nrow = 1))
> plot(nppostshr[[whsig[1]]][[1]][[1]], xlim = c(-0.5, 0.7), type = "l")
> abline(v = 0, lty = 2)
> plot(nppostshr[[whsig[2]]][[1]][[1]], xlim = c(-0.5, 0.7), type = "l")
> abline(v = 0, lty = 2)
> plot(nppostshr[[whsig[3]]][[1]][[1]], xlim = c(-0.5, 0.7), type = "l")
> abline(v = 0, lty = 2)
```



Displays the posteriors of the significant siRNAs. Note that use of a Gaussian prior rather than a non-parametric one would have rendered only the second siRNA (id: 608) to be significant (see cite[]WielHTRNAi).

4 CAGE data

The CAGE data set below consists of normalized sequencing (count) data for 10,000 tag clusters and 25 brain samples. For illustration purposes we limit ourselves to the analysis of the first 1,000 tag clusters. Details on the analysis are available in Van de Wiel et al. (2012) which present the results on 10,000 tag clusters.

```
> library(ShrinkBayes)
> data(CAGEdata10000)
> CAGEdata <- CAGEdata10000
> CAGEdata <- CAGEdata[1:1000, ]
> CAGEdata[1:2, ]
```

```
      raw.0325_Frontal raw.0325_Hippocampus raw.0325_Putamen raw.0325_Temporal
Tag.1                0                    0                0                 0
Tag.2                0                    0                0                 0
      raw.034_Caudate raw.034_Frontal raw.034_Hippocampus raw.034_Temporal
Tag.1              17               0                   0                0
Tag.2               4               0                   0                0
      raw.05217_Caudate raw.05217_Frontal raw.05217_Hippocampus
Tag.1                76                68                    73
Tag.2                10                 7                     4
      raw.05217_Putamen raw.05217_Temporal raw.05269_Caudate raw.05269_Frontal
Tag.1                49                 57                 0                 0
Tag.2                 8                  6                 0                 0
      raw.05269_Hippocampus raw.05269_Putamen raw.05269_Temporal
Tag.1                    35                 0                  0
Tag.2                     8                 0                  0
      raw.07319_Frontal raw.07319_hippocampus raw.07319_Temporal
Tag.1                 0                     0                  0
Tag.2                 0                     0                  0
      raw.96373_Caudate raw.96373_Putamen raw.97266_Caudate raw.97266_Putamen
Tag.1                31                49                31                55
Tag.2                12                12                 7                 6
```

Load `ShrinkBayes`, the data, select the first 1,000 rows and display the first 2.

```
> data(design_brain)
> design_brain
```

```
   pers batch groupfac
1     1     0        2
2     1     0        3
3     1     0        4
4     1     0        5
5     2     1        1
6     2     0        2
```

```
7       2       0           3
8       2       0           5
9       3       1           1
10      3       1           2
11      3       1           3
12      3       1           4
13      3       1           5
14      4       0           1
15      4       0           2
16      4       1           3
17      4       0           4
18      4       0           5
19      5       0           2
20      5       0           3
21      5       0           5
22      6       1           1
23      6       1           4
24      7       1           1
25      7       1           4
```

Loads the design of the brain study.

```
> pers <- design_brain$pers
> batch <- design_brain$batch
> groupfac <- design_brain$groupfac
```

Retrieves covariates from the design matrix.

```
> ncpus2use <- 10
```

Number of cpus to use in (parallel) computations. Note that this should be specified separately for functions that allow parallel computations by the `ncpus =` argument.

```
> groupfac <- BaselineDef("groupfac", baselinegroup = "1")
```

The brain regions, coded by variable `groupfac`, are our main parameters of interest. As the design displays, we are in a multiple (>2) group setting. The function `BaselineDef` allows the user to set the baseline group, rather than let `INLA` decide. Here, the second argument is the character equivalent of the current level of the desired baseline group.

By default only comparisons with the baseline are included in the computations of posteriors. Use the following convenience function to create the other pair-wise comparisons for a given factor when this contains 3 or more levels (groups):

```
> lincombvec <- AllComp("groupfac")
```

```
> form = y ~ 1 + groupfac + batch + f(pers, model = "iid")
```

Define the model formula. Specification should be according to the `inla` formula argument. Here `batch` and `group` are fixed effects, `pers` is a random effect.

```
> shrinksimul <- ShrinkSeq(form = form, dat = CAGEdata, shrinkfixed = "groupfac",
+     shrinkrandom = "pers", mixtdisp = TRUE, ncpus = ncpus2use)
```

Simultaneous shrinkage for `groupfac` and `pers`. In addition, the negative binomial overdispersion is shrunken by default (see `shrinkdisp` argument). Here, `batch` is not shrunken, because the effect of `batch` may not be uniform accross the range of counts. In the example we allow a mixture prior for overdispersion (`mixtdisp=TRUE`). This increases the computing time by a factor 2, because `inla` has to fit the model under zero overdispersion (leading to a (zero-inflated) Poisson instead of a zero-inflated negative binomial.) `ShrinkSeq` default uses the zero-inflated negative binomial to fit sequencing data (see (Van de Wiel et al., 2012) for argumentation) by setting `fams="zinb"`. However, other options are `fams="nb"`, `fams="poisson"`, `fams="zip"` for Negative Binomial, Poisson and zero-inflated Poisson, respectively. In case one opts for `fams="nb"` it may be wise to explicitly account for the relationship between the mean and the overdispersion when shrinking (see e.g. (Anders and Huber, 2010)). In `ShrinkSeq` this is effectuated by setting `curvedisp=TRUE`. See the discussion on the `ShrinkGauss` function in Section 3 for other important arguments of the function.

```
> shrinksimul$pmlist$mixp
```

```
[1] 0.08376754 0.91623246
```

This retruns the prior mass on the point mass of the mixture prior for overdispersion. If this would be close to zero (which we have observed in some other RNAseq applications), then it is sufficient to fit the model on all data for the (zero-inflated) Negative Binomial only. Otherwise, fits under the (zero-inflated) Poisson are also needed, like for this data set.

```
> fitzip <- FitAllShrink(form, dat = CAGEdata, fams = "zip", shrinksimul,
+     ncpus = ncpus2use, lincomb = lincombvec)
```

Fits the model on all data using the priors resulting from ShrinkSeq and Zero-Inflated-Poisson likelihood. Note that, as discussed in Section 3, by default the variance of the prior for the main parameter of interest (here `"groupfac"`) is increased by a factor 10. The `lincomb=lincombvec` argument specifies that the function should also generate posteriors for the linear combinations defined above.

```
> fitzinb <- FitAllShrink(form, dat = CAGEdata, fams = "zinb",
+     shrinksimul, ncpus = ncpus2use, lincomb = lincombvec)
```

As above, but now under the Zero-Inflated-Negative Binomial likelihood (using the shrunken Gaussian prior for log-overdispersion).

```
> cp <- CombinePosteriors(fitzip, fitzinb, shrinksimul, para = "groupfac",
+     ncpus = ncpus2use)
```

Combines the posteriors of the two fits (as outlined in the Suppl. Mat. of (Van de Wiel et al., 2012)) for the `"groupfac"` parameters and the linear combinations (included by default). Hence, `cp` contains the posteriors of all pairwise comparisons under 1) a shrunken mixture prior for overdispersion; 2) a shrunken Gamma prior for the precision of `pers`; and 3) a (deliberately too wide) Gaussian prior for `"groupfac"`. The latter will now be updated to a non-parametric prior.

```
> npprior <- NonParaUpdatePrior(fitall = cp, modus = "fixed", shrinkpara = "groupfac"
+     shrinklc = TRUE, ncpus = ncpus2use, maxiter = 3)
```

Find one common nonparametric prior for all group-wise differences, including the contrasts defined in `lincombvec`. If you do not want to include those contrasts (e.g. because you are focusing on comparison with the baseline group), set `shrinklc=FALSE` (default). Note that we advise to increase `maxiter` for obtaining final results (default = 15). Important other arguments and defaults of the `NonParaUpdatePrior` function:

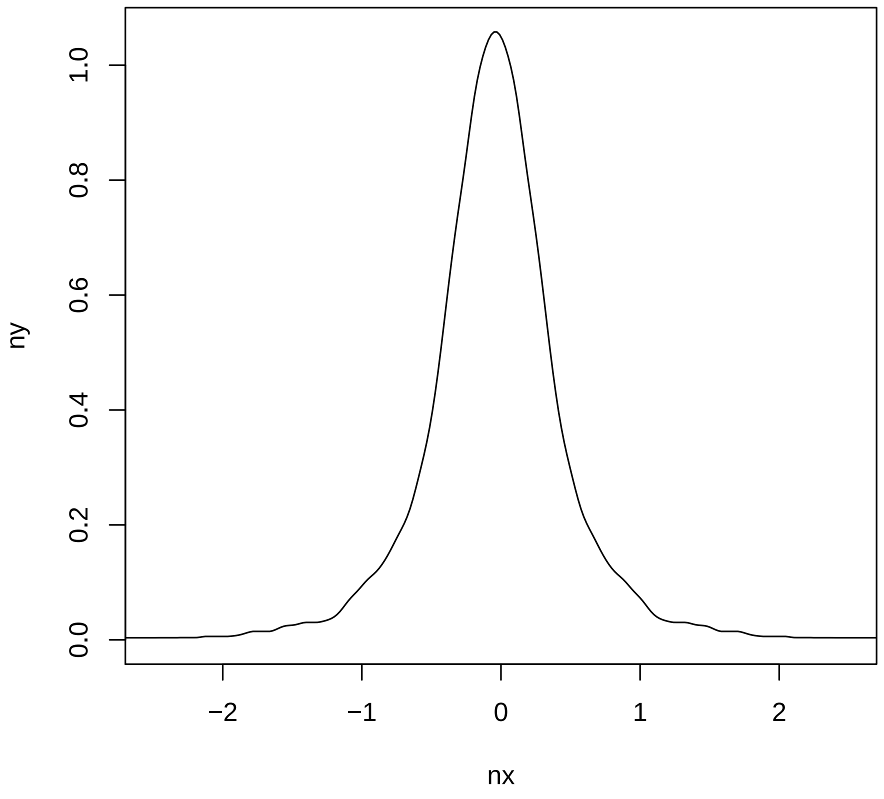
- `symmetric = TRUE`: Force the prior to be symmetric?
- `unimodal = TRUE`: Force the prior to be unimodal?
- `logconcave = FALSE`: Force the prior to be logconcave?

- `lcabscoef =c(1,1)`: Absolute values of the coefficients in the linear combinations. Currently we allow multi-contrast shrinkage only for contrasts with the same values of `lcabscoef`.

Note that here we use the default setting for the shape of prior (symmetric, unimodal, but not necessarily logconcave). See also ...

Let's have a look at the prior.

```
> theprior <- npprior$priornew
> plot(theprior, type = "l", xlim = c(-2.5, 2.5))
```



```
> quantiles <- inla.qmarginal(p = c(0.05, 0.25, 0.5, 0.75, 0.95),
+     theprior)
> quantiles
```

```
[1] -0.92262861 -0.30907530 -0.04458935  0.21989661  0.83344992
```

```
> expect <- inla.emarginal(function(x) x, theprior)
> expect
```

```
[1] -0.03955675
```

```
> sd <- sqrt(inla.emarginal(function(x) x^2, theprior) - expect^2)
> sd
```

```
[1] 0.723257
```

Computes quantiles, the mean and standard deviation of the prior. These functions can also be applied to posteriors. See ?inla.qmarginal in your R-console for other options.

```
> nppostshr <- NonParaUpdatePosterior(cp, npprior, ncpus = ncpus2use)
```

Updates the posteriors of the `"groupfac"` parameters and the contrasts involved in `lincombvec` using the new, non-parametric prior.

```
> lfdr <- SummaryWrap(nppostshr, thr = log(1.5))
```

Compute *two-sided* local fdrs which equals $\text{lfdr} = \min(P(\beta_{\text{contrast}} \leq \text{thr}|Y), P(\beta_{\text{contrast}} \geq -\text{thr}|Y))$. The above lfdr introduces a desirable conservativeness with respect to an alternative posterior null-probability: $\text{lfdr'} = P(-\text{thr} \leq \beta_{\text{contrast}} \leq \text{thr}|Y)$. The latter, lfdr', can not properly deal with wide posteriors that have a lot of probability mass outside $(-\text{thr}, \text{thr})$. See Van de Wiel et al. (2012) for further discussion. The threshold `thr = log(1.5)` implies that we only wish to detect effects larger than 1.5 fold.

```
> BFDRs <- BFDR(lfdr)
```

```
> head(BFDRs)
```

```
     groupfac2 groupfac3 groupfac4 groupfac5 groupfac3mingroupfac2
[1,] 0.7594013 0.7955304 0.8160210 0.8068468             0.8221437
[2,] 0.8278324 0.8494276 0.8345920 0.8368664             0.8288337
[3,] 0.8425993 0.1897167 0.7670844 0.8221101             0.1704118
[4,] 0.4884483 0.7963692 0.6165695 0.7589679             0.8493518
[5,] 0.8526093 0.8588785 0.8460374 0.8623392             0.8268282
[6,] 0.8008360 0.7607720 0.6496556 0.7627002             0.7307867
```

```
     groupfac4mingroupfac2 groupfac4mingroupfac3 groupfac5mingroupfac2
[1,]             0.8219757             0.8028797             0.8285386
[2,]             0.7963121             0.6930053             0.8182273
[3,]             0.5758759             0.8536606             0.7021078
[4,]             0.8409730             0.7933942             0.8438888
[5,]             0.7577407             0.6675664             0.8408789
[6,]             0.6133084             0.6748121             0.7305368
     groupfac5mingroupfac3 groupfac5mingroupfac4
[1,]             0.8275530             0.8296039
[2,]             0.7975236             0.8355238
[3,]             0.8588213             0.8292926
[4,]             0.8152289             0.8396536
[5,]             0.8300127             0.8504250
[6,]             0.7452639             0.7984073
```

Compute (two-sided) Bayesian FDRs for all comparisons. Hence, a matrix is returned which features as rows and comparisons as columns.

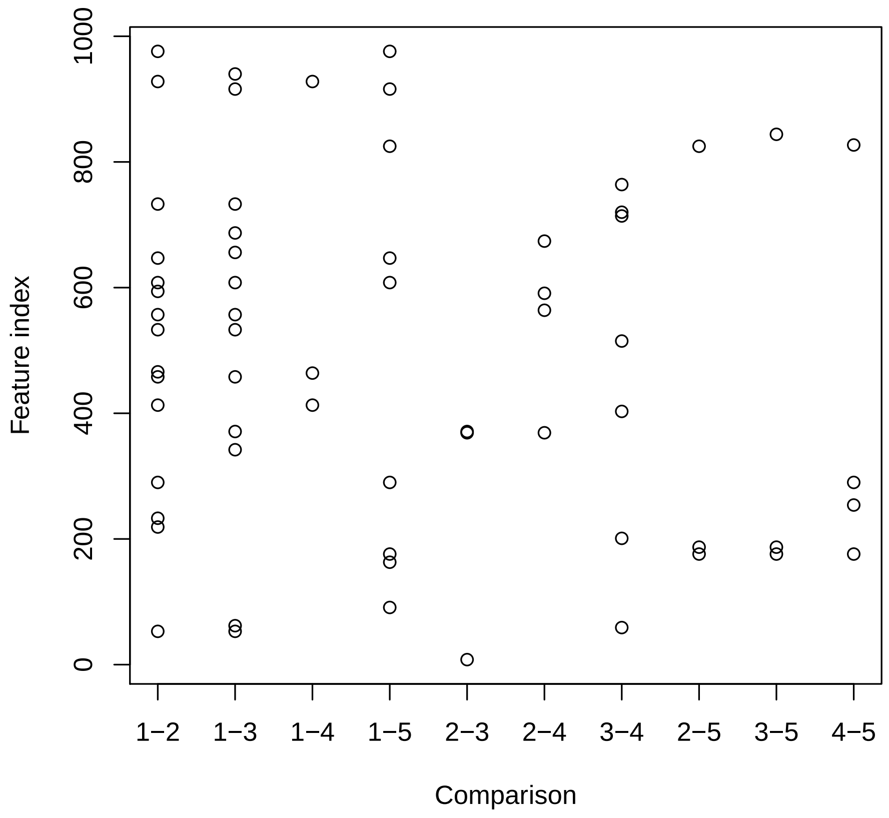
```
> BFDRmult <- BFDR(lfdr, multcomp = TRUE)
```

Compute (two-sided) Bayesian FDRs for testing the hypothesis: all comparisons parameters belong to the null domain. This allows one to perform K-group inference (like in ANOVA or an F-test) using a multiple comparison set-up. Hence, `BFDRmult` may be used to discover features showing at least one difference between groups (but not where: for that `BFDRs` is needed.)

```
> wh <- which(BFDRmult <= 0.1)
> length(wh)
```

```
[1] 43
```

```
> whcomp <- which(BFDRs[wh, ] <= 0.1, arr.ind = TRUE)
> plot(cbind(whcomp[, 2], wh[whcomp[, 1]]), type = "p", xlab = "Comparison",
+     ylab = "Feature index", xaxt = "n")
> axis(1, at = 1:10, labels = c("1-2", "1-3", "1-4", "1-5", "2-3",
+     "2-4", "3-4", "2-5", "3-5", "4-5"))
```



A plot that visualizes which features are significant for at least one comparison (hence according to `BFDRmult`) and which comparison is significant for those features (according to `BFDRs`).

5 Future topics

We plan to extent `ShrinkBayes` in many directions.

- Other data types, in particular fractional data (on 0-1 scale), e.g. methylation
- Longitudinal settings (multiple measurements over time)
- Modelling and reporting joint posteriors rather than only marginal ones

- K-sample inference (other than through multiple comparisons, which was illustrated in the Section 4)
- "Small" multivariate GLM settings. Where "small" means: non-high-dimensional (at the level of the model)
- "Large" multivariate penalized regression settings
- Inference with point mass on zero and *nonparametric* continuous component
- Using Bayes' Factors for inference in a multiplicity setting
- ...

If you are interested in any of these topics, or when you feel these are essential for your application, please let us know. We might have progressed and be able to share (preliminary) code with you.

6 Appendix

6.1 Code used for generating the simulated data set 'datsim'

```
#1500 rows (siRNAs), 8 samples, pi0 = 2/3
datsim1 <- matrix(rnorm(8000,mean=0,sd=0.5),nrow=1000)
meanvec <- matrix(rep(rnorm(500,0,1),4),nrow=500)
datsim2 <- cbind(matrix(rnorm(2000,mean=0,sd=0.5),nrow=500),matrix(rnorm(2000,mean=0,
datsim <- rbind(datsim1,datsim2)
```

References

Anders, S. and Huber, W. (2010). Differential expression analysis for sequence count data. *Genome Biol*, 11:R106.

Lutz, D. and Rufibach, K. (2011). logcondens: Computations related to univariate log-concave density estimation. *J. Statist. Software*, 39:1–28.

Van de Wiel, M., de Menezes, R., Siebring-van Olst, E., and van Beusechem, V. (2013). Analysis of small-sample clinical genomics studies using multi-parameter shrinkage: application to high-throughput RNA interference screening. *BMC Med. Genom.*, Accepted.

Van de Wiel, M., Leday, G., Pardo, L., Rue, H., van der Vaart, A., and van Wieringen, W. (2012). Bayesian analysis of RNA sequencing data by estimating multiple shrinkage priors. *Biostatistics*, doi: 10.1093/biostatistics/kxs031.

Ventrucci, M., Scott, E. M., and Cocchi, D. (2011). Multiple testing on standardized mortality ratios: a Bayesian hierarchical model for FDR estimation. *Biostatistics*, 12:51–67.