The `XdeParameter` Class

Håkon Tjelmeland and Robert B. Scharpf

October 30, 2017

1 Introduction

The goal of this vignette is to briefly describe the Bayesian hierarchical model to estimate differential expression across multiple studies. This vignette necessarily assumes familiarity with hierarchical models and Bayesian methods of computation, such as MCMC. After reading this vignette, one should be able to:

- create an instance of the `XdeParameter` class that will provide all the necessary parameters for fitting the Bayesian model, including starting values of all the MCMC chains
- change values from their default setting to custom setting. This includes
  - changing the values of any of the hyperparameters
  - changing the starting values of the MCMC chains
  - change the number of updates per MCMC iteration for any of the parameters
  - store all, some, or none of the MCMC chains

Apart from these pragmatic goals, the vignette provides a means to look-up the role of any parameters in our formulation of the Bayesian model. A more detailed discussion of the Bayesian model is provided elsewhere [**?**]. If you are content with the default settings provided when creating an instance of the `XdeParameter` class, fitting the model, assessing convergence, and generating alternative measures of differential expression are discussed in the *XDE* vignette.

2 The Stochastic Model

Let x_{gsp} denote observed expression value for gene $g \in \{1, \ldots, G\}$ and sample (array) $s \in \{1, \ldots, S_q\}$ in study $p \in \{1, \ldots, P\}$, where $P \geq 2$. We assume that some clinical variable (with two possible values) is available for each of the arrays in all studies. We denote this by $\psi_{sp} \in \{0, 1\}$ for sample (array) number s in study number p. We will assume that the studies have been somehow standardized so that the values in each study is centered around zero and are approximately Gaussian distributed.

Our model defined below is based on the following assumptions, (i) for some of the genes, the expression values x_{gsp} are differentially expressed (have different mean values) for arrays where $\psi_{sp} = 0$ and arrays where $\psi_{sp} = 1$, and (ii) any gene is either differentially expresses in all studies or in no studies (the

"δ_g" model). Assumption (ii) can be relaxed by fitting the "δ_{gp}" model that allows for a given gene to be differentially expressed in some studies but not in others. The following description of the Bayesian model is for the δ_g" model. The δ_{gp} model is very similar, substituting δ_{gp} for δ_g and independent beta priors with parameter ξ_p instead of ξ.

We assume the following hierarchical Bayesian model for the expression data. Conditionally on underlying unobserved parameters we assume the expression values to be independently Gaussian distributed,

$$x_{gsp}|\ldots \sim \mathrm{N}(\nu_{gp} + \delta_g(2\psi_{sp} - 1)\Delta_{gp}, \sigma^2_{g\psi_{sp}p}), \tag{1}$$

where $\delta_g \in \{0, 1\}$ indicates whether gene g is differentially expressed ($\delta_g = 1$) or not ($\delta_g = 0$). Thus, if $\delta_g = 0$, the mean of x_{gsp} is ν_{gp}, whereas if $\delta_g = 1$, the mean is $\nu_{gp} - \Delta_{gp}$ and $\nu_{gp} + \Delta_{gp}$ for samples with $\psi_{sp} = 0$ and $\psi_{sp} = 1$, respectively.

Given hyper-parameters, we assume $\boldsymbol{\nu}_g = (\nu_{g1}, \ldots, \nu_{gP})^T$ and $\boldsymbol{\Delta}_g = (\Delta_{g1}, \ldots, \Delta_{gP})^T$ to be multi-Gaussian distributed,

$$\boldsymbol{\nu}_g|\ldots \sim \mathrm{N}(0, \Sigma) \qquad \text{and} \qquad \boldsymbol{\Delta}_g|\ldots \sim \mathrm{N}(0, R), \tag{2}$$

where $\Sigma = [\Sigma_{pq}] \in \Re^{P\times P}$ and $R = [R_{pq}] \in \Re^{P\times P}$ with

$$\Sigma_{pq} = \gamma^2 \rho_{pq} \sqrt{\tau_p^2 \tau_q^2 \sigma_{gp}^{2a_p} \sigma_{gq}^{2a_q}} \tag{3}$$

and

$$R_{pq} = c^2 r_{pq} \sqrt{\tau_p^2 \tau_q^2 \sigma_{gp}^{2b_p} \sigma_{gq}^{2b_q}}. \tag{4}$$

The a_p and $b_p, p = 1, \ldots P$ are assumed apriori independent with discrete probabilities in 0 and 1 and a continuous density on $(0, 1)$. More precisely, we let

$$\mathrm{P}(a_p = 0) = p_a^0 \quad , \quad \mathrm{P}(a_p = 1) = p_a^1 \quad , \quad a_p|a_p \in (0, 1) \sim \mathrm{Beta}(\alpha_a, \beta_a), \tag{5}$$

and

$$\mathrm{P}(b_p = 0) = p_b^0 \quad , \quad \mathrm{P}(b_p = 1) = p_b^1 \quad , \quad b_p|b_p \in (0, 1) \sim \mathrm{Beta}(\alpha_b, \beta_b). \tag{6}$$

To c^2 we assign a uniform prior distribution on $[0, c^2_{\mathrm{Max}}]$, where c^2_{Max} is a parameter specified by the user, and for γ^2 we use an improper uniform distributions on $(0, \infty)$. We restrict $\tau_p^2 > 0, p = 1, \ldots, P$ and $\tau_1^2 \cdot \ldots \cdot \tau_P^2 = 1$ and assume an (improper) uniform distribution for $(\tau_1^2, \ldots, \tau_P^2)$ under this restriction. The $[\rho_{pq}] \in \Re^{P\times P}$ and $[r_{pq}] \in \Re^{P\times P}$ are restricted to be correlation matrices and we assign apriori independent Barnard et al (2000) distributions for them, with ν_ρ and ν_r degrees of freedom for $[\rho_{pq}]$ and $[r_{pq}]$, respectively.

We assume the indicators $\delta_1, \ldots, \delta_G$ to be apriori independent, given a hyper-parameter ξ, with

$$\mathrm{P}(\delta_g = 1|\xi) = \xi, \qquad \text{where} \qquad \xi \sim \mathrm{Beta}(\alpha_\xi, \beta_\xi). \tag{7}$$

We assume σ^2_{gp} in (3) to be given from $\sigma^2_{g\psi p}$ in (1) via the following relations

$$\sigma^2_{g0p} = \sigma^2_{gp}\varphi_{gp} \qquad \text{and} \qquad \sigma^2_{g1p} = \frac{\sigma^2_{gp}}{\varphi_{gp}}. \tag{8}$$

For σ^2_{gp} we assume independent prior distributions, given hyper-parameters,

$$\sigma^2_{gp}|\ldots \sim \mathrm{Gamma}\left(\frac{l_p^2}{t_p}, \frac{l_p}{t_p}\right), \tag{9}$$

where the mean l_p and variance t_p for $p = 1, \ldots, P$ are assigned independent (improper) uniform distributions on $(0, \infty)$. The φ_{gp} are assigned independent gamma priors, given hyper-parameters,

$$\varphi_{gp}| \ldots \sim \text{Gamma}\left(\frac{\lambda_p^2}{\theta_p}, \frac{\lambda_p}{\theta_p}\right), \tag{10}$$

where the mean λ_p and the variance θ_p for $p = 1, \ldots, P$ are assigned independent (improper) uniform distributions on $(0, \infty)$.

3 Metropolis-Hastings algorithm

To simulate from the posterior distribution resulting from the above specified Bayesian model, we use a Metropolis–Hastings algorithm. Default values for the tuning parameters in this algorithm and how these tuning parameters may be modified are discussed in Section 4. The Metropolis–Hastings algorithm uses the following moves:

1. The ν_{pg}'s are updated in a Gibbs step.
2. The Δ_{pg}'s are updated in a Gibbs step.
3. Each $a_p, p = 1, \ldots, P$ is updated in turn, where a potential new value for a_p, $\widetilde{a}_p$, is generated as follows. If $a_p = 0$, $\widetilde{a}_p \sim \text{Uniform}(0, \varepsilon)$, if $a_p = 1$, $\widetilde{a}_p \sim \text{Uniform}(1 - \varepsilon, 1)$, and if $a_p \in (0, 1)$ we use

$$\text{P}(\widetilde{a}_p = 0) = \max\{-(a_p - \varepsilon), 0\} \cdot \text{I}(p_a^0 \neq 0), \tag{11}$$

$$\text{P}(\widetilde{a}_p = 1) = \max\{a_p + \varepsilon - 1, 0\} \cdot \text{I}(p_a^1 \neq 0) \tag{12}$$

   and

$$\widetilde{a}_p | \widetilde{a}_p \in (0, 1) \sim \text{Uniform}(\max\{a_p - \varepsilon, 0\}, \min\{a_p + \varepsilon, 1\}). \tag{13}$$

4. Each $b_p, p = 1, \ldots, P$ is updated in turn, where the proposal distribution is of the same type as in 3.
5. A block Gibbs update for c^2 and $\mathbf{\Delta}_g$ for g's where $\delta_g = 0$.
6. The γ^2 is updated in a Gibbs step.
7. A block update for the correlation matrix $[r_{pq}]$ and the variance c^2. First, potential new values for $[r_{pq}]$, $[\widetilde{r}_{pq}]$, is set by

$$\widetilde{r}_{pq} = (1 - \varepsilon) r_{pq} + \varepsilon T_{pq}, \tag{14}$$

   where $[T_{pq}]$ is a correlation matrix which with probability a half is generated from the prior for $[r_{pq}]$, or with probability a half set equal to unity on the diagonal and with all off diagonal elements set equal to the same value b. In the latter case, the value b is sampled from a uniform distribution on $(-1/(P-1), 1)$. Second, the potential new value for c^2 is sampled from the corresponding full conditional (given the potential new values $[\widetilde{r}_{pq}]$). The ε is a tuning parameter.
8. A block update for the correlation matrix $[\rho_{pq}]$ and the variance γ^2. The potential new values are generated similar to what is done in 7.
9. For each $g = 1, \ldots, G$ in turn, a block update for δ_g and $\mathbf{\Delta}_g$. First, the potential new value for δ_g is set equal to $\widetilde{\delta}_g = 1 - \delta_g$. Second, the potential new value for $\mathbf{\Delta}_g$ is sampled from the full conditional (given the potential new value $\widetilde{\delta}_g$). No tuning parameter for this update.

10. The ξ is updated in a Gibbs step.

11. Each σ_{gp}^2 is updated in turn, where the potential new value is given as $\widetilde{\sigma}_{gp}^2 = \sigma_{gp}^2 \cdot u$, where $u \sim \text{Uniform}(1/(1+\varepsilon), 1+\varepsilon)$. The ε is a tuning parameter.

12. Each $t_p, p = 1, \ldots, P$ is updated in turn. The potential new value is given as $\widetilde{t}_p = t_p \cdot u$, where $u \sim \text{Uniform}(1/(1+\varepsilon), 1+\varepsilon)$. The ε is a tuning parameter.

13. Each $l_p, p = 1, \ldots, P$ is updated in turn. The potential new value is given as $\widetilde{l}_p = l_p \cdot u$, where $u \sim \text{Uniform}(1/(1+\varepsilon), 1+\varepsilon)$. The ε is a tuning parameter.

14. Each φ_{gp} is updated in turn, where the proposal distribution is of the same type as in 11.

15. Each $\theta_p, p = 1, \ldots, P$ is updated in turn, where the proposal distribution is of the same type as in 12.

16. Each $\lambda_p, p = 1, \ldots, P$ is updated in turn, where the proposal distribution is of the same type as in 13.

17. The $(\tau_1^2, \ldots, \tau_P^2)$ is updated by first uniformly at random drawing a pair $p, q \in \{1, \ldots, P\}$ so that $p \neq q$. Potential new values for τ_p^2 and τ_q^2 are generated by setting

$$\widetilde{\tau}_p^2 = \tau_p^2 \cdot u \quad \text{and} \quad \widetilde{\tau}_q^2 = \tau_q^2 / u, \tag{15}$$

where $u \sim \text{Uniform}(1/(1+\varepsilon), 1+\varepsilon)$. The ε is again a tuning parameter.

4 The `XdeParameter` class

The Bayesian hierarchical model can be tuned and output modified in a number of ways. The `XdeParameter` class is an effort to organize these options and to facilitate tweaking. In our experience, it is easier to change and keep track of the parameters in the class than to provide a function for fitting the model with numerous arguments. The `XdeParameter` class is not a container for the gene expression data (see `ExpressionSetList` in the *XDE* vignette).

4.1 Initializing the `XdeParameter` class

There are numerous options for customizing the fit of the Bayesian model as well as the output. Default values are defined in the initialization method for the `XdeParameter` class. Initialization requires an object of class `ExpressionSetList` and the name of the classification variable used to define differential expression. (The initialization method will produce an error if the supplied phenotypeLabel is not present in the `varLabels` of each element of the `ExpressionSetList`). Using the example dataset discussed in the *XDE* vignette, we have defined the covariate "adenoVsquamous" that takes the value 0 for adenocarcinomas and 1 for squamous carcinomas. The `show` method only lists the first few parameters in each slot of the `XdeParameter` object.

```
> library(XDE)
> data(expressionSetList)
> xlist <- expressionSetList
> params <- new("XdeParameter", esetList=xlist, phenotypeLabel="adenoVsquamous")
> params
```

```
Instance of XdeParameter

hyperparameters:
alpha.a  beta.a    p0.a    p1.a alpha.b  beta.b    p0.b
    1.0     1.0     0.1     0.1     1.0     1.0     0.1
   p1.b
    0.1
...

updates (frequency of updates per MCMC iteration):
   nu Delta     a     b    c2
    1     1     3     3     1
...

tuning (the epsilon for Metropolis-Hastings proposals):
   nu Delta     a     b    c2
 0.01  0.01  0.04  0.04  0.01
...

output (parameters to save (0 = not saved, 1 = saved to log file):
 potential acceptance         nu     DDelta          a
         1          1          1          1          1
...

iterations:  1000
thin:  1
seed:  661325
notes:
firstMcmc:
List of 5
 $ Nu    : num [1:1500] -1.125 0.331 -0.247 1.075 1.602 ...
 $ DDelta: num [1:1500] -0.603 0.494 -0.998 0.872 -1.047 ...
 $ A     : num [1:3] 0.189 0.213 0.331
 $ B     : num [1:3] 0.221 0.33 0.345
 $ C2    : num 0.474
...

showIterations:  TRUE
specifiedInitialValues:  TRUE
directory (where to save the MCMC chains):  ./
phenotypeLabel:  adenoVsquamous
studyNames:  study1 study2 study3
one.delta:  TRUE
```

4.2 Starting values for MCMC chains

A complete listing of initial values for the MCMC can be obtained by

```
> initialValues <- firstMcmc(params)
> str(initialValues)
```

```
List of 18
 $ Nu     : num [1:1500] -1.125 0.331 -0.247 1.075 1.602 ...
 $ DDelta : num [1:1500] -0.603 0.494 -0.998 0.872 -1.047 ...
 $ A      : num [1:3] 0.189 0.213 0.331
 $ B      : num [1:3] 0.221 0.33 0.345
 $ C2     : num 0.474
 $ Gamma2 : num 2.1
 $ R      : num [1:3] 0.14996 0.00601 0.12897
 $ Rho    : num [1:3] 0.161 0.356 0.605
 $ Delta  : int [1:1500] 1 1 1 0 0 0 0 0 0 1 ...
 $ Xi     : num [1:3] 0.487 0.487 0.487
 $ Sigma2 : num [1:1500] 0.809 1.54 1.096 6.498 5.21 ...
 $ T      : num [1:3] 1 1 1
 $ L      : num [1:3] 0.966 1 0.979
 $ Phi    : num [1:1500] 0.2918 0.0978 0.193 1.2305 0.4846 ...
 $ Theta  : num [1:3] 1.02 1 1
 $ Lambda : num [1:3] 1 0.993 1.02
 $ Tau2R  : num [1:3] 0.99 1 1.01
 $ Tau2Rho: num [1:3] 0.978 1.024 0.999
```

The initial values of the MCMC chains stored in `firstMcmc` are generated by random sampling from the prior distributions for these parameters. One may replace any one of the named elements in this list with different starting values. At this time, we do not provide checks that the replacement vectors are of the appropriate length. WARNING: providing a replacement vector of the incorrect length may cause the MCMC algorithm to crash without warning. Note that specifiedInitialValues in the `XdeParameter` object is always `TRUE` as the values created here are the initial values used to run the MCMC, regardless of the starting values were explicitly defined or simulated from the priors.

```
> params@specifiedInitialValues
```

```
[1] TRUE
```

If one were to change `specifiedInitialValues` to `FALSE`, the MCMC would begin at a different set of initial values drawn from the prior and not the values provided in the `XdeParameter` class.

4.3 Hyper-parameters

When initializing `XdeParameter`for three studies as we are here, the default values for the hyper-parameters are as follows:

```
> hyperparameters(params)
```

```
 alpha.a   beta.a     p0.a     p1.a  alpha.b   beta.b
     1.0      1.0      0.1      0.1      1.0      1.0
    p0.b     p1.b     nu.r   nu.rho alpha.xi  beta.xi
     0.1      0.1      4.0      4.0      1.0      1.0
   c2max
    50.0
```

The names used in the software implementation that correspond to the notation used in Section 2 are as follows:

hyperparameter	*XDE* name
α_a	alpha.a
β_a	beta.a
p_a^0	p0.a
p_a^1	p1.a
α_b	alpha.b
β_b	beta.b
p_b^0	p0.b
p_b^1	p1.b
ν_r	nu.r
ν_ρ	nu.rho
α_ξ	alpha.xi
β_ξ	beta.xi
$c^2_{\max}$	c2max

In situations in which there is no signal (all noise), the $c^2_{\max}$ is provided as a precaution to avoid sampling from infinity. However, in simulations of complete noise, the $c^2_{\max}$ parameter was not generally needed and remains in the present model primarily for debugging purposes. Modification of any of the above hyperparameters can be performed in the usual way:

```
> hyperparameters(params)["alpha.a"] <- 1
```

4.4 Tuning parameters for Metropolis-Hastings proposals

A default ϵ for each of the Metropolis-Hastings proposals discussed in Section 3 is created when initializing the `XdeParameter` class. See

```
> tuning(params)
```

```
          nu          Delta              a              b
        0.01           0.01           0.04           0.04
          c2         gamma2         rAndC2   rhoAndGamma2
        0.01           0.01           0.01           0.01
       delta             xi         sigma2          tAndL
        0.01           0.01           0.50           0.10
           l            phi thetaAndLambda         lambda
        0.04           0.40           0.10           0.02
       tau2R        tau2Rho
        0.04           0.04
```

for a named vector of the default values. A replacement method has been defined to facilitate the tuning of the proposals. To illustrate, the ϵ used in the proposal for the parameter a (the power conjugate parameter for ν) could be adjusted by the following command:

```
> tuning(params)["a"] <- tuning(params)["a"]*0.5
```

4.5 Frequency of updates for each MCMC iteration

One may control the frequency at which any of the parameters described in Section 3 are updated at each iteration of the MCMC through the method `updates`. See

```
> updates(params)
```

```
            nu          Delta              a              b
             1              1              3              3
             c2         gamma2         rAndC2   rhoAndGamma2
             1              1              3              3
         delta             xi         sigma2          tAndL
             1              1              1              1
             l            phi thetaAndLambda         lambda
             1              1              1              1
         tau2R        tau2Rho
             1              1
```

Specifying zero updates forces a parameter to remain at its initial value. For instance, one may impose conjugacy between location and scale by initializing the power conjugate parameter, b, to 1 and changing the number of updates per MCMC iteration to zero:

```
> firstMcmc(params)$B <- rep(1,3)
> updates(params)["b"] <- 0
```

Alternatively, one could force independence of location and scale by setting b to zero:

```
> firstMcmc(params)$B <- rep(0, 3)
> updates(params)["b"] <- 0
```

By default, b is allowed to vary between zero and 1. Block updates are denoted by 'And'. For instance, r and c^2 are updated as a block (as described above). Therefore, to update the r parameter 3 times per MCMC iteration, one would use the command

```
> updates(params)["rAndC2"] <- 3
```

4.6 MCMC output

XDE provides options to save all, some, or none of the chains produced by MCMC. Options for writing MCMC chains to file are provided in the `output` slot of the `XdeParameter`class.

```
> output(params)
```

```
        thin     potential    acceptance            nu
           1             1             1             1
```

```
      DDelta             a             b            c2
           1             1             1             1
      gamma2             r           rho         delta
           1             1             1             1
          xi        sigma2             t             l
           1             1             1             1
         phi         theta        lambda        tau2R
           1             1             1             1
     tau2Rho     probDelta diffExpressed
           1             1             1
```

The first value, thin, tells the MCMC algorithm how often to write the chain to file. For instance, if thin is two every other iteration is written to file. The remaining numbers in the output vector are either zero (nothing is written to file) or one. If the output is one, the simulated value from the current iteration is added to the log file. The log files are plain text files and are by default stored in the current working directory. One may change the location of where to store the log files by providing a different path

```
> directory(params) <- "logFiles"
```

If the directory "logFiles" does not exist, the above replacement method for directory will automatically create the directory. In general, directory should provide the path relative to the current working directory or the complete path to the desired directory.

An additional slot in the `XdeParameter`class that may be useful is `burnin`. By default, `burnin` is `TRUE` and no chains are written to file. One reason for this setting as a default is to easily check whether the *XDE* model will run with the default values without producing voluminous output in the MCMC chains. For instance, we often set

```
> burnin(params) <- TRUE
> iterations(params) <- 5
```

Note the following behavior of the replacement method for burnin:

```
> output(params)[2:22] <- rep(1, 21)
> output(params)
```

```
        thin     potential    acceptance            nu
           1             1             1             1
      DDelta             a             b            c2
           1             1             1             1
      gamma2             r           rho         delta
           1             1             1             1
          xi        sigma2             t             l
           1             1             1             1
         phi         theta        lambda        tau2R
           1             1             1             1
     tau2Rho     probDelta diffExpressed
           1             1             0
```

```
> burnin(params) <- TRUE
> output(params)
```

```
         thin     potential    acceptance            nu
            1             0             0             0
       DDelta             a             b            c2
            0             0             0             0
       gamma2             r           rho         delta
            0             0             0             0
           xi        sigma2             t             l
            0             0             0             0
          phi         theta        lambda         tau2R
            0             0             0             0
      tau2Rho     probDelta diffExpressed
            0             0             0
```

Hence, when setting burnin to `TRUE` we assume that none of the iterations are to be saved. Similarly, when setting `burnin` to FALSE, we assume that one wishes to save all of the parameters:

```
> ##Specify a thin of 1 and save none of the parameters
> output(params)[2:22] <- rep(0, 21)
> output(params)
```

```
         thin     potential    acceptance            nu
            1             0             0             0
       DDelta             a             b            c2
            0             0             0             0
       gamma2             r           rho         delta
            0             0             0             0
           xi        sigma2             t             l
            0             0             0             0
          phi         theta        lambda         tau2R
            0             0             0             0
      tau2Rho     probDelta diffExpressed
            0             0             0
```

```
> burnin(params) <- FALSE
> output(params)
```

```
         thin     potential    acceptance            nu
            1             1             1             1
       DDelta             a             b            c2
            1             1             1             1
       gamma2             r           rho         delta
            1             1             1             1
           xi        sigma2             t             l
            1             1             1             1
          phi         theta        lambda         tau2R
```

```
      1             1             1             1
tau2Rho     probDelta diffExpressed
      1             1             1
```

To save a subset of the parameters, we recommend setting the `burnin` argument to `FALSE` and turning off the parameters that one does not wish to save. Warning: the parameters $\Delta_{gp}, \delta_g, \text{probDelta}_g, \sigma^2_{gp}, \nu_{gp}, \phi_{gp}$ are all indexed by genes and/or study and may therefore require a large amount of memory to save. Use the following commands to avoid writing these chains to file:

```
> burnin(params) <- FALSE
> output(params)[c("nu", "DDelta", "delta", "probDelta", "sigma2", "phi")] <- 0
> output(params)
```

```
        thin     potential    acceptance            nu
           1             1             1             0
      DDelta             a             b            c2
           0             1             1             1
      gamma2             r           rho         delta
           1             1             1             0
          xi        sigma2             t             l
           1             0             1             1
         phi         theta        lambda         tau2R
           0             1             1             1
      tau2Rho     probDelta diffExpressed
           1             0             1
```

5 Session Information

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

- R version 3.4.2 (2017-09-28), `x86_64-pc-linux-gnu`
- Locale: `LC_CTYPE=en_US.UTF-8`, `LC_NUMERIC=C`, `LC_TIME=en_US.UTF-8`, `LC_COLLATE=C`, `LC_MONETARY=en_US.UTF-8`, `LC_MESSAGES=en_US.UTF-8`, `LC_PAPER=en_US.UTF-8`, `LC_NAME=C`, `LC_ADDRESS=C`, `LC_TELEPHONE=C`, `LC_MEASUREMENT=en_US.UTF-8`, `LC_IDENTIFICATION=C`
- Running under: `Ubuntu 16.04.3 LTS`
- Matrix products: default
- BLAS: `/home/biocbuild/bbs-3.6-bioc/R/lib/libRblas.so`
- LAPACK: `/home/biocbuild/bbs-3.6-bioc/R/lib/libRlapack.so`
- Base packages: base, datasets, grDevices, graphics, methods, parallel, stats, utils
- Other packages: Biobase 2.38.0, BiocGenerics 0.24.0, XDE 2.24.0

- Loaded via a namespace (and not attached): AnnotationDbi 1.40.0, DBI 0.7, GeneMeta 1.50.0, IRanges 2.12.0, MASS 7.3-47, Matrix 1.2-11, MergeMaid 2.50.0, RColorBrewer 1.1-2, RCurl 1.95-4.8, RSQLite 2.0, Rcpp 0.12.13, S4Vectors 0.16.0, XML 3.98-1.9, annotate 1.56.0, bit 1.1-12, bit64 0.9-7, bitops 1.0-6, blob 1.1.0, compiler 3.4.2, digest 0.6.12, genefilter 1.60.0, grid 3.4.2, gtools 3.5.0, lattice 0.20-35, memoise 1.1.0, multtest 2.34.0, mvtnorm 1.0-6, rlang 0.1.2, siggenes 1.52.0, splines 3.4.2, stats4 3.4.2, survival 2.41-3, tibble 1.3.4, tools 3.4.2, xtable 1.8-2

References