BayMAP: A Bayesian hierarchical model for the analysis of PAR-CLIP data

Eva-Maria Huessler, HHU Düsseldorf

October 08, 2018

1 Getting started - Requirements

BayMAP will be shortly available as an R package at `https://cran.r-project.org/`. For instance, the package is availabe at our homepage `http://stat.math.uni-duesseldorf.de/baymap`. **BayMAP** as well as **R2WinBUGS**, on which **BayMAP** is based, can be installed in R by

```
> install.packages("BayMAP", contriburl="http://stat.math.uni-duesseldorf.de/baymap")
> install.packages("R2WinBUGS")
```

BayMAP depends on WinBUGS. On Windows, WinBUGs can be installed directly. Linux User can use WinBUGS via Wine.

1.1 Windows

The **BayMAP** package calls WinBUGS. WinBUGS can be downloaded from

`https://www.mrc-bsu.cam.ac.uk/software/bugs/the-bugs-project-winbugs/`

Please download WinBUGS *and* the patch for WinBUGS 1.4.3. The installation of the patch is explained in the corresponding text file itself. WinBUGS should either be stored in the program file directory or the directory should be declared in the `baymap` function in R using the argument `bugs.directory = "my/path/to/WINBUGS14"`.

1.2 Linux

On Linux, WinBUGS can be called using Wine. For installation, please follow these steps:

1. First, install the actual version of wine, following for example these instructions:

   `https://wiki.winehq.org/Ubuntu`

2. Download WinBUGS and unzip the folder:

   `https://www.mrc-bsu.cam.ac.uk/software/bugs/the-bugs-project-winbugs/`

3. Navigate to the directory to which you have downloaded WinBUGS before you run

   ```
   wine WinBUGS14.exe
   ```

   WinBUGS should now be installed in the folder ' /.wine/drive_c/Program Files/WinBUGS14' (or alternatively in ' /.wine/drive_c/Program Files (x86)/WinBUGS14').

4. Open WinBUGS via Wine:

   ```
   wine ~/.wine/drive_c/Program\ Files/WinBUGS14/WinBUGS14.exe

   or, alternatively:

   wine ~/.wine/drive_c/Program\ Files\ \(x86\)/WinBUGS14/WinBUGS14.exe
   ```

5. Download the patch for WinBUGS 1.4.3 and open it within WinBUGS:

   `https://www.mrc-bsu.cam.ac.uk/wp-content/uploads/WinBUGS14_cumulative_patch_No3_06_08_07_RELEASE.txt`

   Follow the instructions in the text file. Then, you can close WinBUGS and reopen it as previously described.

6. Download the immortality key, open it within WinBUGS, and follow the instructions in the text file:

   `https://www.mrc-bsu.cam.ac.uk/wp-content/uploads/WinBUGS14_immortality_key.txt`

Please note that if WinBUGS is not installed automatically in the folder ' /.wine/drive_c/Program Files/WinBUGS14' by Wine, but in ' /.wine/drive_c/Program Files (x86)/WinBUGS14', the directory has to be specified in `baymap` by the argument `bugs.directory = "c:/Program Files (x86)/WinBUGS14"`. Then the `baymap` function can be called via R, as described in the following Section.

2 Running WinBUGS via BayMAP

First, the package **BayMAP** should be loaded into `R`:

```
> library(BayMAP)
```

Then, BayMAP can be applied to PAR-CLIP data. The main function in **BayMAP** is `baymap`, which takes as only required argument the data. The data should be stored as a data frame with at least 3 columns: one for the number of substitutions for a position (called `count`), one for the total number of reads for this position (`coverage`), and one with the mutation type (`mutation`), as for example the following data set:

```
> data(data_test)
> head(data_test)

      chr       pos mutation count coverage
2411    1  28282101       AG     1      109
4558    1  52089293       TC     1       21
5363    1  67701668       TC     2       76
11568   1 171593171       TC     5       79
16572   1 236208598       TG     2       77
11136   1 163081654       AG   208      225
```

BayMAP can be applied in `R` by

```
> res <- baymap(data = data_test)
```

If additional variables shall be implemented as prior information to the model, the argument `covariates` should be specified by a vector containing the variable names, e.g. `covariates = c('tpUTR', 'cds', 'fpUTR')` (each of these covariates must be contained in a column of the data frame, where the column name must match the name specified in `covariates`).

The results of the Markov chains are stored in the object `res` of class "baymap", similar to class "bugs". This implies that basic functions available for the class "bugs" can be applied on this object, too, such as `print(res)`. Basic statistics for the different parameters can be obtained as follows:

```
> res$summary

                 mean           sd         2.5%       25%       50%         75%        97.5%
q        9.275374e-01 9.114258e-03 9.086000e-01 9.219e-01 9.278e-01 9.34000e-01 9.440025e-01
u[1]     2.430884e-01 3.158658e-03 2.371975e-01 2.409e-01 2.431e-01 2.45100e-01 2.491025e-01
u[2]     9.847661e-01 3.120548e-04 9.842000e-01 9.845e-01 9.848e-01 9.85000e-01 9.854000e-01
u[3]     5.078647e-03 1.034771e-04 4.870775e-03 5.009e-03 5.077e-03 5.15125e-03 5.270000e-03
p[1,1]   3.356567e-01 2.132950e-02 2.970900e-01 3.208e-01 3.352e-01 3.50000e-01 3.785399e-01
deviance 1.214825e+04 1.186500e+01 1.213000e+04 1.214e+04 1.215e+04 1.21600e+04 1.217000e+04
```

The complete chains for the parameters can be obtained by `res$sims.array`. The first 6 iterations of μ_{exp} can be, e.g., obtained by:

```
> head(object$sims.array[t,1,"u[1]"])

[1] 0.2426 0.2404 0.2378 0.2468 0.2402 0.2399
```

All other values, that are returned by `baymap` can be shown by `res`. They are also described on the help page for the `bugs` function (see "Value" in `?bugs`).

The results obtained by calling `baymap` can then be used for binding site prediction (see Section 4).

More details to all possible arguments to the function `baymap` can be obtained by:

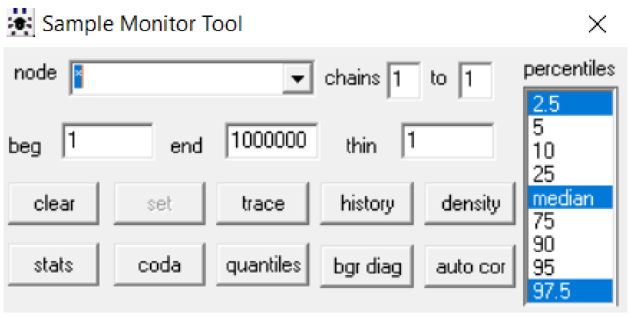
```
> ?baymap
> ?bugs
```

All arguments from the `bugs` function can be specified in `baymap`, too, as `baymap` calls `bugs` from the **R2WinBUGS** package.

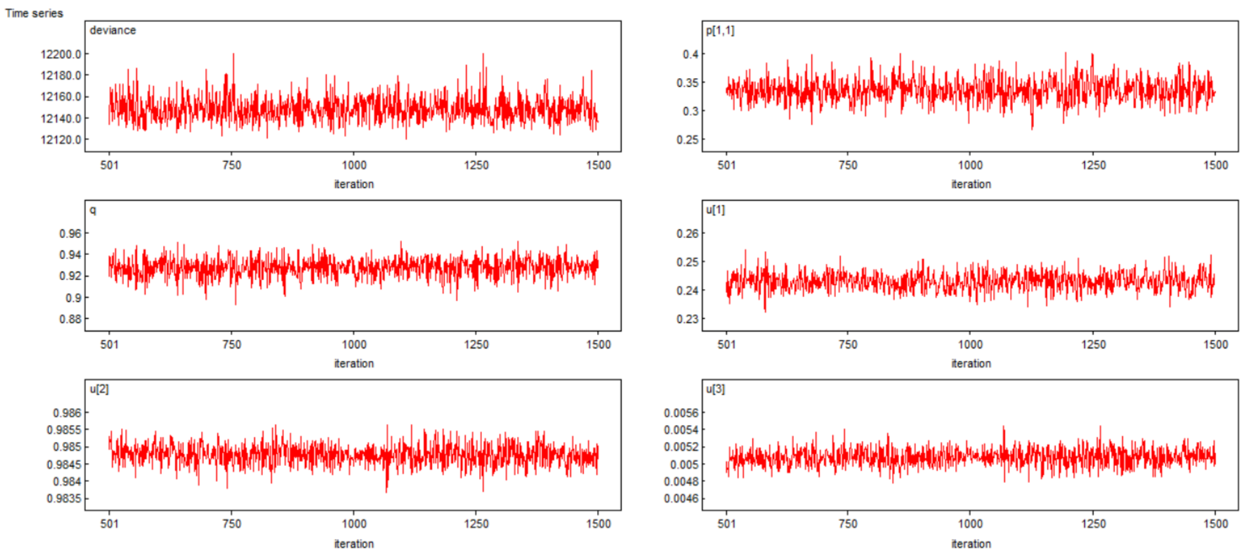
After running the model and obtaining Markov chains, the results should be checked. It is possible that `baymap` should be run again, e.g., with more iterations or a thinning rate (see Section 3).

3 Model checking

The results of the `baymap` function should be checked for convergence and autocorrelation. This is, for example, possible in WinBUGS directly after running the Markov chains. For this purpose, the argument `debug` should be set to `TRUE` in the `baymap` function. Then, the WinBUGS window is not closed automatically after running. Please note that you have to close WinBUGS manually, if `debug` is set to `TRUE`, before you can continue to use `R`. Diagnosis plots can now be displayed in WinBUGS, for example, by selecting "Samples..." in the "Inferences" menu so that the "Sample Monitor Tool" opens.

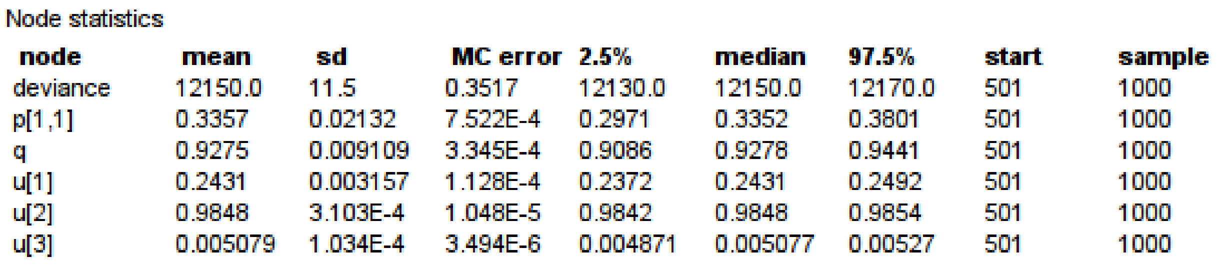


Here, one can, e.g., choose which iterations are shown ("beg", "end", "thin") and display basic results such as trace plots, densities, basic statistics, or autocorrelation plots. Trace plots for all saved iterations can be produced by selecting the "history" button.



If patterns or irregularities are observed, convergence is probably not achieved and chains with more iterations should be generated (higher number for `n.iter`). In the above example, no patterns or irregularities can be observed.

One can also consider, for example, the Monte Carlo (MC) errors. They measure the variability of the arithmetic mean of the sample for each parameter. When the MC error is small in comparison to the posterior standard deviation, the posterior mean is estimated with high precision and iterations could be stopped. One rule of thumb is to stop the iterations when the Monte Carlo error is smaller than 5% of the posterior standard deviation for each parameter. MC error can be displayed by pressing the "stats" button.

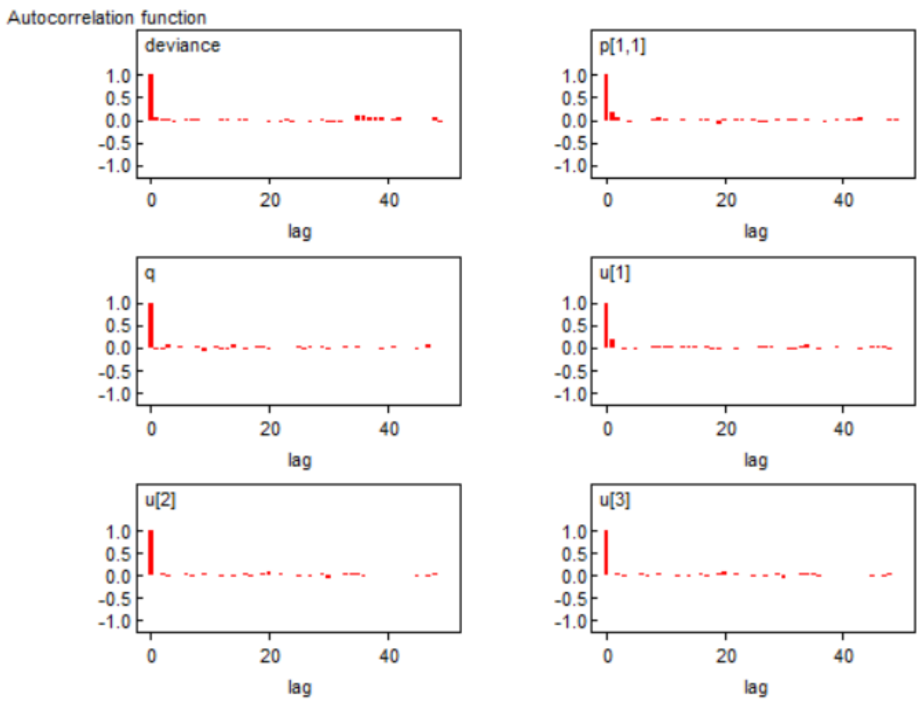


Node statistics

node	mean	sd	MC error	2.5%	median	97.5%	start	sample
deviance	12150.0	11.5	0.3517	12130.0	12150.0	12170.0	501	1000
p[1,1]	0.3357	0.02132	7.522E-4	0.2971	0.3352	0.3801	501	1000
q	0.9275	0.009109	3.345E-4	0.9086	0.9278	0.9441	501	1000
u[1]	0.2431	0.003157	1.128E-4	0.2372	0.2431	0.2492	501	1000
u[2]	0.9848	3.103E-4	1.048E-5	0.9842	0.9848	0.9854	501	1000
u[3]	0.005079	1.034E-4	3.494E-6	0.004871	0.005077	0.00527	501	1000

Not only the posterior standard deviations and the MC errors are displayed, but also basic statistics such as the mean and percentiles. Dividing the MC errors by the standard deviations leads here to values under 5% for each parameter.

In Bayesian statistics, we wish to have independent observations. Autocorrelation can be displayed in WinBUGS by selecting the "auto cor" button.



These plots show the correlations between one iteration and its lag. Correlations should be close to 0 for all parameters, which is the case here. If autocorrelations were only low when considering a higher lag, an independent sample could obtained by rerunning `baymap` with a thinning parameter for which autocorrelation was low (higher value to `n.thin`). In this example, no additional thinning is needed for an independent sample.

For more details, see for example Ntzoufras, I. (2011). *Bayesian modeling using WinBUGS* (Vol. 698). John Wiley & Sons. Diagnostics can, of course, be obtained in `R`, too, by using, for example, packages such as **coda** or **boa**.

4 Prediction

The main goal of **BayMAP** is the prediction of potential microRNA binding sites on the mRNA. To this end, the function `predict` can be applied on the results of class "baymap":

```
> data_new <- predict(res, data_test)
> head(data_new)
```

```
      chr       pos mutation count coverage  BayesFactor
2411    1  28282101       AG     1      109 3.616895e-12
4558    1  52089293       TC     1       21 2.218728e-02
5363    1  67701668       TC     2       76 1.345604e-06
11568   1 171593171       TC     5       79 1.503717e-01
16572   1 236208598       TG     2       77 1.035913e-06
11136   1 163081654       AG   208      225 1.276347e-95
```

Here, one column "BayesFactor" is added to the old data set. If covariates were added to the model, also prior odds and posterior odds would be added to the data set. If the Bayes factor (or the posterior odds when covariates are existent) is larger than 1, it is more likely that the corresponding position is a a binding site than a non-binding site. Bayes factors (as well as posterior odds) are calculated for all mutation types, even if we are only interested in one (e.g., T-to-C). The other mutation types that are not induced by the PAR-CLIP method, enable us to estimate the rate of the falsely detected binding sites.

```
> prop.table(table(data_new$BayesFactor[data_new$mutation == "TC"] > 1))

    FALSE      TRUE
0.6570881 0.3429119

> prop.table(table(data_new$BayesFactor[data_new$mutation != "TC"] > 1))

     FALSE       TRUE
0.97280335 0.02719665
```

These results show us, that about 34% of the T-to-C substitution positions are declared as binding sites, and therefore, PAR-CLIP induced substitution. Only 3% of the other substitution positions are (falsely) declared as binding sites (estimated false positive rate).