# bayesNMF: Understanding Output

Jenna Landy

```r
library(bayesNMF)
data <- readRDS("examples/3_64_1_cosmic.rds")
refit <- FALSE
```

## Understanding .log, .res, and .pdf files

Three files will be created and updated every `MAP_every` iterations:

### Log

The file named `<file>.log` will log the start time and the progress of the Gibbs sampler, which is useful to estimate the total run time if using a large dataset or a lot of iterations.

The example below shows the first few lines of the log file `res_fixedN.log` from an example ran in this GitHub repo's `README`. The header notes the date and start time of the bayesNMF run, as well as parameters including the maximum iterations specified by the convergence control, whether the "fast" updates are being used (either a Normal likelihood or the Metropolis updates for a Poisson likelihood), and whether rank is being learned (indicated by "learn_A").

After `MAP_over` iterations is hit, the current status is recorded every `MAP_every` iterations. The format for this status is: > `<iter> / (up to) 5000 - <time since last status report> - <% change in MAP metric> - <# reports> no best - <# reports> no change`

Recall from above that we say the MAP "hasn't changed" if it's MAP metric has changed by less than 100*`tol`% since the previous computed MAP – this is recorded by `<# reports> no change`. Recall also that we say that the MCMC has converged if it hasn't improved in `Ninarow_nobest` computations–the number of computations since the last improvement is reported by `# reports> no best`.

```
[1] "2025-03-04 11:24:04 EST"
[1] "maxiters = 5000"
[1] "fast TRUE"
[1] "learn_A FALSE"
[1] "starting iterations, 2025-03-04 11:24:04.841436"
1100 / (up to) 5000 - 13.474 seconds - -0.0111% change - 0 no best - 1 no change
1200 / (up to) 5000 - 2.4105 seconds - -0.1644% change - 0 no best - 0 no change
1300 / (up to) 5000 - 2.2189 seconds - -0.2574% change - 0 no best - 0 no change
```

In the case that a range of latent ranks is provided and either BFI or SBFI is used to learn rank, the log file will look a bit different. The example below shows the first few lines of the log file `res_learnN.log` from an example ran in this GitHub repo's `README`. Besides the change to the `learn_A` flag in the header, there are two key changes.

First, status update information regarding percent change in MAP metric or number of reports since best or with no change will be excluded until the tempering stage is complete (~2/5 of the way through sampling), so only iteration number and time is reported in these early reports.

Second, each report also comes with the current status of samples of the $A$ signature inclusion matrix. This reports the current values along the diagonal of the $A$ matrix (e.g., `11011` says the third signature is excluded while the rest are included) as well as the number of occurrences (e.g., `784` samples of `11011` in posterior samples between iterations 100 and 1100). This gives an idea of the current MAP estimates of rank, and gives the user an idea of how quickly this rank estimate is converging. Unique $A$ matrices are sorted from most to least common within that window of samples, and only the top five most frequent $A$ matrices are reported. If there are fewer than five unique $A$ matrices in that window of samples, you'll see `<NA>` in the table, as seen below.

```
[1] "2025-03-04 11:24:42 EST"
[1] "maxiters = 5000"
[1] "fast TRUE"
[1] "learn_A TRUE"
[1] "starting iterations, 2025-03-04 11:24:42.293999"
1100 / (up to) 5000 - 19.9187 seconds
top_counts
11011 11111 10011 10111  <NA>
  784   175    24    17

1200 / (up to) 5000 - 2.2781 seconds
top_counts
11011 11111  <NA>  <NA>  <NA>
  855   145
```

### PDF

The file named `<file>.pdf` updates plots of MAP metrics from each status report: RMSE, KL Divergence, BIC, log posterior, log likelihood, the latent rank, and the number of samples contributing the MAP (i.e., number of samples with the MAP $A$ matrix). Recall that these are not metrics corresponding to individual samples, but to the MAP estimates from a sliding window of `map_over` samples (see "Iterations to Convergence" section below for details). Note that for log likelihood and log posterior, values from the Poisson models are not comparable to those from Normal models.

The small applications run in this repo's README were so easy that metrics were very good from the very beginning, meaning their metrics plots looked very wobbly because of the tight scale. Instead, we present here an example pdf from a real data application.

The green vertical line indicates when the tempering procedure has completed (i.e., when the model is eligable for convergence). The blue line indicates the final model selected for convergence.

```
<img src="images/pdf-1.png" width="45%"/>
<img src="images/pdf-2.png" width="45%"/>
<img src="images/pdf-3.png" width="45%"/>
```

### Res

The file named `<file>.rds` periodically records results, which is be useful if your run is cut short (the dreaded OOM error). Once the run is complete, this records complete results for future access.

```r
res <- readRDS("../examples/convergence_example.rds")
names(res)
```

```
##  [1] "MAP"                "metrics"            "model"
##  [4] "totaliters"         "converged_at"       "final_Theta"
##  [7] "time"               "credible_intervals" "posterior_samples"
## [10] "logs"
```

Details on `MAP`, `posterior_samples`, `converged_at`, and `totaliters` are described in detail in the next few sub-sections. Here is an overview of the rest:

- `model`: includes original data `M`, specified prior parameters, likelihood, prior, convergence control, initial values, fixed values, dimensions of the decomposition, and the tempering schedule of the temperature parameter gamma.
- `final_Theta`: the final value of Theta, a named list holding all current parameter values, at the final iteration.
- `time`: holds total time for the full run and average seconds per iteration
- `logs`: holds all posterior samples for all parameters
- `posterior_samples`: holds all posterior samples considered when computing the MAP (last `map_over` samples before the `converged_at` iteration).

**Inference** The maximum a-posterior estimates of all model parameters is held in `res$MAP`, and element-wise 95% credible intervals are stored in `res$credible_intervals`. Recall that NMF has scale non-identifiability ($P * E = (P/2) * (2E)$), so the scale of these estimates are meaningless.

```
names(res$MAP)
```

```
##  [1] "A"                "P"                "P_acceptance"   "E"
##  [5] "E_acceptance"     "q"                "prob_inclusion" "idx"
##  [9] "top_counts"       "sigmasq"
```

```
res$MAP$P[1:5, ]
```

```
##            [,1]      [,2]       [,3]
## [1,] 0.20539031 3.7766615 0.41056998
## [2,] 0.10622790 0.9631511 0.23091809
## [3,] 0.08898680 1.3739272 5.04198848
## [4,] 0.29156800 0.9870912 0.04422617
## [5,] 0.08094637 0.1495512 0.84516288
```

```
res$credible_intervals$P[[1]][1:5, ]
```

```
##            [,1]      [,2]        [,3]
## [1,] 0.09228335 3.5347367 0.238794286
## [2,] 0.03490238 0.8170258 0.100513691
## [3,] 0.01040053 1.1657659 4.779045636
## [4,] 0.19267407 0.8562229 0.001038827
## [5,] 0.02037022 0.0491731 0.705718342
```

```
res$credible_intervals$P[[2]][1:5, ]
```

```
##           [,1]      [,2]      [,3]
## [1,] 0.3367366 4.0118378 0.6134592
## [2,] 0.1910894 1.1071678 0.3866296
## [3,] 0.1982755 1.5863842 5.3016708
## [4,] 0.3914920 1.1439105 0.1413284
## [5,] 0.1527140 0.2683338 0.9968394
```

For a more meaningful scale, you can use the `rescale_bayesNMF` function. Now, columns of P sum to 1 and can be interpreted as a probability mass distribution, and E is on the scale of number of mutations attributed to each signature.

```
res_rescaled <- rescale_bayesNMF(res)
colSums(res_rescaled$MAP$P)
```

```
## [1] 1 1 1
```

```
res_rescaled$MAP$P[1:5, ]
```

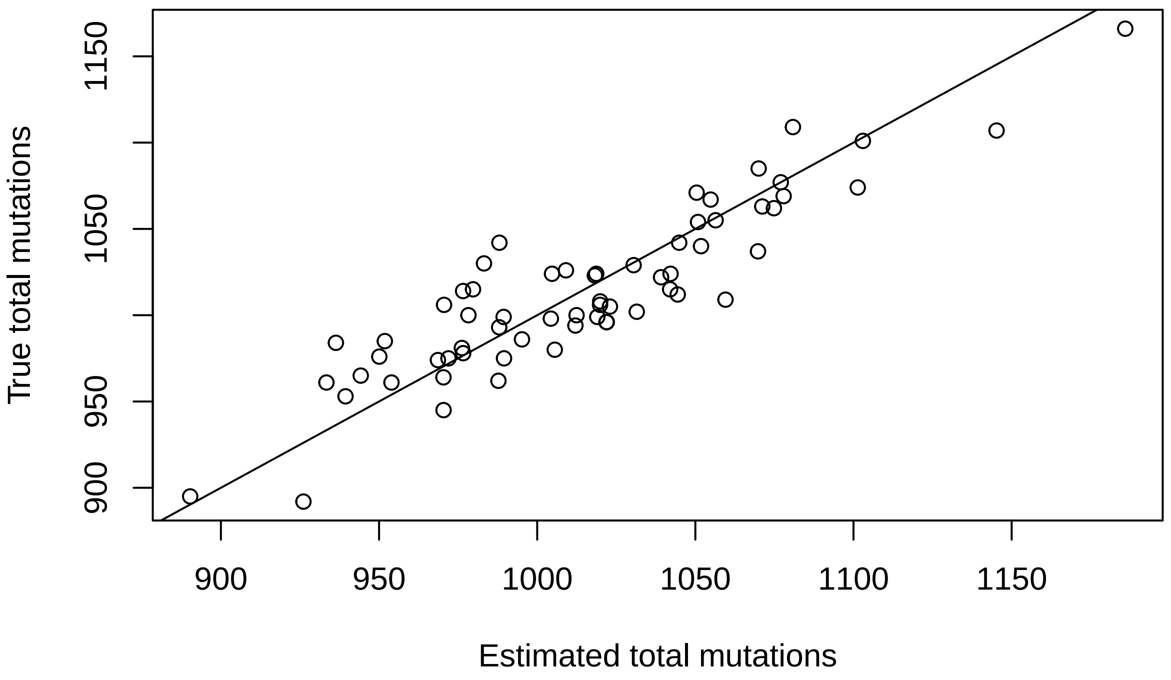
```
##               [,1]         [,2]          [,3]
## [1,] 0.001682604 0.022030520 0.0032250413
## [2,] 0.000870243 0.005618380 0.0018138695
## [3,] 0.000729000 0.008014573 0.0396049925
## [4,] 0.002388591 0.005758030 0.0003473981
## [5,] 0.000663131 0.000872382 0.0066387834
```

```
res_rescaled$MAP$E[, 1:2]
```

```
##             [,1]      [,2]
## [1,]  29.02678  33.69284
## [2,] 673.83786 809.70270
## [3,] 315.36813 144.59065
```

We can see below that the column sums of the MAP E matrix line up with the total mutations per sample in the orignal dataset M:

```
plot(colSums(res_rescaled$MAP$E), colSums(data$M), xlab = "Estimated total mutations", ylab = "True tota
abline(a = 0, b = 1)
```



If the `fast` option is used (metropolis steps for P and E updates), the average acceptance rates for elements of P and E are also reported. Recall that for a provided rank `1:20`, the full P matrix with 20 columns will be updated on each iteration regardless of what signatures are included. To subset to included signatures, we look at columns matching `res$MAP$A == 1`.

```
res$MAP$P_acceptance[1:5, res$MAP$A == 1]
```

```
##             [,1]      [,2]      [,3]
## [1,] 0.5627547 0.8348990 0.7817097
## [2,] 0.5985164 0.6142779 0.6736077
## [3,] 0.7674479 0.8368916 0.8457764
## [4,] 0.6228861 0.7839573 0.6112739
## [5,] 0.5766717 0.6334378 0.6712449
```

**Checking run is complete** Once the `res` object is loaded, you can check whether the run is complete by looking at `res$converged_at`, which will be an integer representing the final iteration if complete, and

NULL if not complete. If it is not complete, take caution when interpreting any results as the MCM has not yet converged.

```
res$converged_at
```

```
## [1] 3200
```

Sometimes the convergence point is not the final iteration in the case that the samples move away from the MAP. `res$totaliters` will show the total number of iterations sampled.

```
res$totaliters
```

```
## [1] 5000
```

**Viewing metrics from PDF** `res$metrics` is a dataframe holding the same metrics plotted in the PDF (see "PDF" section for details)

```
head(res$metrics)
```

```
##   sample_idx     RMSE       KL    loglik   logpost N n_params MAP_A_counts
## 1       1100 2.421895 2049.951 -11708.52 -12363.24 3      486          166
## 2       1200 2.419609 2046.254 -11704.83 -12344.64 3      486          266
## 3       1300 2.419598 2045.962 -11704.54 -12347.42 3      486          366
## 4       1400 2.420113 2046.571 -11705.14 -12334.68 3      486          466
## 5       1500 2.420500 2046.829 -11705.40 -12339.47 3      486          566
## 6       1600 2.420677 2047.355 -11705.93 -12294.22 3      486          665
##        BIC
## 1 25413.31
## 2 25405.92
## 3 25405.33
## 4 25406.55
## 5 25407.07
## 6 25408.12
```

## Iterations to Convergence

Unlike standard MCMC problems, we cannot use multiple chains to determine convergence because different chains can have different numbers of latent factors which we would be unable to align. We instead determine convergence through an approach rooted in machine learning. The `convergence_control` parameter determines the specifics of this approach. These parameters can be adjusted by the user, but the default values are noted below.

```
convergence_control = new_convergence_control(
    MAP_over = 1000,
    MAP_every = 100,
    tol = 0.001,
    Ninarow_nochange = 10,
    Ninarow_nobest = 20,
    miniters = 1000,
    maxiters = 5000,
    metric = "logposterior"
)
```

We pre-determine that the MAP estimate will be the average over `MAP_over` samples. Starting at `miniters` and at every `MAP_every` samples after, we perform a "status report": we compute the MAP estimate *as if it is the last iteration* and record log likelihood, log posterior, RMSE, and KL Divergence. We say the MAP "hasn't changed" if it's MAP metric has changed by less than 100*`tol`% since the previous report. We say the MCMC has converged when the MAP hasn't changed in `Ninarow_nochange` reports (i.e., Ninarow_nochange*MAP_every samples) OR if it hasn't improved in Ninarow_nobest reports (i.e., Ninarow_nobest*MAP_every samples).

A specified convergence control can be passed to bayesNMF with the convergence_control parameter:

```r
if (refit) {
    res <- bayesNMF(
        data$M, 1:20,
        convergence_control = convergence_control,
        file = "../examples/convergence_example",
        overwrite = TRUE
    )
} else {
    res <- readRDS("../examples/convergence_example.rds")
}
```