Overview

In this vignette, I demonstrate the FCD sampler for the Λ_k parameters in the covariance model with dense eigenvectors. If we consider all of the other parameters in the model as fixed, then the FCDs are essentially the posterior distributions for the unknown Λ_k values. In this case, we can compare a transformation to an exact distribution. Thus, I will show that the sampler reproduces the exact density, and also that the posterior means and credible intervals recover known parameter values. Additional code is included in the final section, for reference.

Model description

Data model: Data matrices Y_k are $P \times P$ Hermitian positive definite. n_k is the number of samples corresponding to data matrix Y_k.

$$Y_k|\sigma_k^2, U_k, \Lambda_k \sim \text{ComplexWishart}(n_k, \sigma_k^2[U_k\Lambda_k U_k^H + I_P])$$

Prior: Denote $diag(\Lambda_k) = (\lambda_{1,k}, \ldots, \lambda_{d,k})'$, and let $\omega_{j,k} = \lambda_{j,k}/(1+\lambda_{j,k})$. We use independent Uniform(0, 1) priors on $\omega_{j,k}$.

FCD: It can be shown that the data likelihood is proportional to a product of terms like $\exp[a_{j,k}\omega_{j,k}](1-\omega_{j,k})^{n_k}$, where $a_{j,k} = (u_{j,k}^H Y_k u_{j,k)})/\sigma_k^2$. Define $\xi_{j,k} = 1 - \omega_{j,k}$, and observe

$$\begin{aligned}\exp[a_{j,k}(1-\xi_{j,k})](\xi_{j,k})^{n_k} &= \exp[a_{j,k} - a_{j,k}\xi_{j,k}](\xi_{j,k})^{n_k} \\ &\propto \exp[-a_{j,k}\xi_{j,k}](\xi_{j,k})^{n_k}.\end{aligned}$$

Thus, each $\xi_{j,k}$ has an independent Gamma$(n_k+1, a_{j,k})$ FCD, truncated to (0, 1). With a sampled $\xi_{j,k}$, one can calculate $\lambda_{j,k} = (1/\xi_{j,k}) - 1$.

Compare sampler to known posterior

In this section, I will show that the sampler function does indeed sample from the correct distribution, which we know exactly (in terms of $\xi_{j,k}$). If we generate samples with the function, their empirical density matches the exact density that I expect.

In terms of estimating the actual parameters, the posterior means at least seem somewhat plausible, and the true parameter values are all contained in 95% credible intervals. In the subsequent section, I will show how the credible intervals and posterior means perform across multiple observations of data.

```
## [1] "k = 1"
```

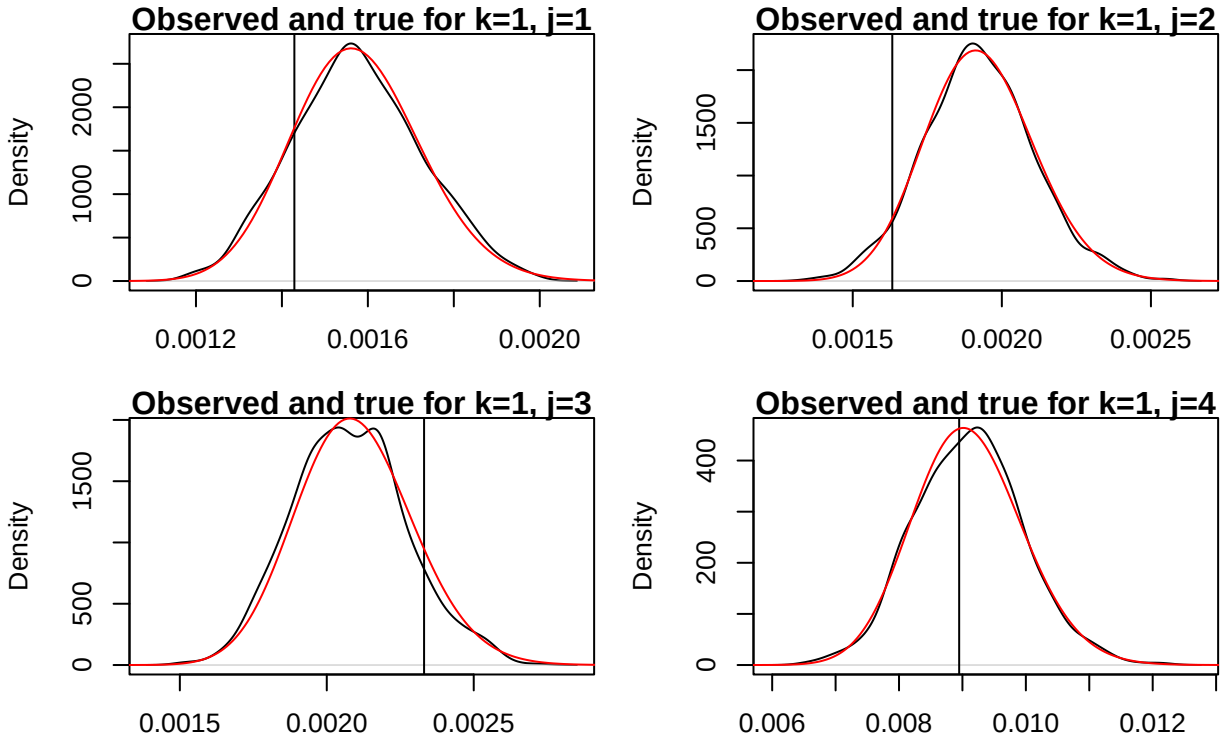
```
##   True Lambda_j Posterior mean 0.025 quant. 0.975 quant.
## 1      698.7977       639.6712    532.70513     766.7820
## 2      611.5314       523.6617    431.59323     636.8176
## 3      427.9837       484.2126    401.19934     576.3148
## 4      110.7485       109.7760     90.96927     132.3727
```

```
## [1] "k = 2"
```

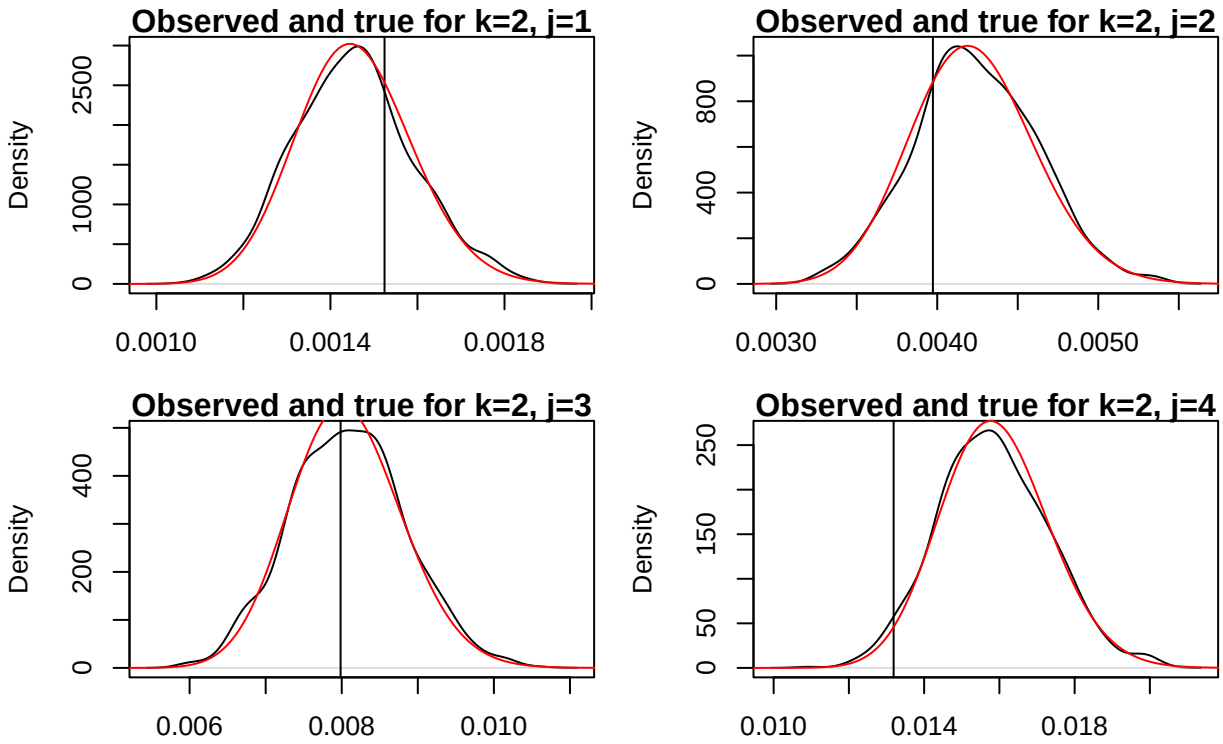
```
##   True Lambda_j Posterior mean 0.025 quant. 0.975 quant.
## 1     655.04532      692.74641    572.11229    827.02129
## 2     250.69164      236.86038    198.48763    284.55142
## 3     124.23930      123.78839    103.49659    149.24040
## 4      74.81969       62.67363     52.13762     75.41182
```

Below are comparisons of the observed and exact posterior densities of $\xi_{j,k}$. The vertical lines are the true values. The black curves are the empirical densities, and the red curves are the exact densities.

```
## [1] "k = 1"
```




```
## [1] "k = 2"
```



Comparing estimate over multiple posteriors

In this section, I repeatedly sample posterior distributions after generating new data matrices, to observe the bias in the posterior distributions and the coverage of 95% credible intervals. 200 posterior distributions are sampled, for 2000 iterations each, with 1000 iterations of burn-in. The 95% credible intervals exhibit >90% coverage. As an estimator, the posterior mean does not appear especially biased.

```
## [1] "Coverage of 95% credible intervals"
```

```
##       k=1  k=2   k=3
## j=1 0.935 0.92 0.945
## j=2 0.960 0.94 0.925
## j=3 0.945 0.98 0.900
## j=4 0.965 0.92 0.975
```

```
## [1] "Average posterior means"
```

```
##           k=1      k=2      k=3
## j=1 658.66990 652.0929 503.6927
## j=2 206.64476 586.1589 460.3803
## j=3 176.87846 502.6366 314.4042
## j=4  47.77374 296.1065 252.0765
```

```
## [1] "True Lambda values"
```

```
##           k=1      k=2      k=3
## j=1 661.15640 654.3959 504.5871
## j=2 203.64863 584.3742 458.9767
## j=3 176.94790 501.4732 315.5157
## j=4  48.18718 294.6824 251.9521
```

Large parameter viability

In this section, I briefly demonstrate that the sampler works for large parameter dimensions, like we may see in applications. I set $P = 64$, such as in a 64-channel EEG; $d = 4$, representing reduction to 4 dimensions; and $K = 200$, representing a sample size of 200 EEG signals. The following is a summary of repeated evaluations using `microbenchmark()`. On my laptop, with no parallelization, the time to run a single sampling step is on the order of 10 milliseconds.

```
## Unit: milliseconds
##                                                          expr      min
##  large_sample <- Lambdak_gibbs_densCovar(data_list, param_list) 10.14633
##        lq     mean   median       uq      max neval
##  10.26509 10.93537 10.40083 10.55337 14.55222   100
```

Appendix

```
#####
# Lambdak_gibbs_densCovar
#####

Lambdak_gibbs_densCovar
```

```
## function (data_list, param_list)
## {
##     d <- param_list$d
##     K <- param_list$K
##     result_Lambdas <- array(NA, c(d, K))
##     for (k in 1:K) {
##         Y <- data_list[[k]]
##         n <- param_list$n_k[k]
##         U <- param_list$U_ks[, , k]
##         sigma2 <- param_list$sigma_k2s[k]
##         for (j in 1:d) {
##             tjk <- Re(t(Conj(U[, j])) %*% Y %*% U[, j])/sigma2
##             lb <- 0
##             ub <- 1
##             lp <- pgamma(lb, shape = n, rate = tjk)
##             up <- pgamma(ub, shape = n, rate = tjk)
##             u <- runif(1, lp, up)
##             xi_jk <- qgamma(u, shape = n + 1, rate = tjk)
##             result_Lambdas[j, k] <- 1/xi_jk - 1
##         }
##     }
##     return(result_Lambdas)
## }
## <bytecode: 0x55a92daf5b18>
```