Overview

```r
# 4x4 real Bingham sampler, with rejection sampler and Gibbs sampler

P <- 4
its <- 1000

set.seed(17042025)

H <- diag(seq(10, 1, length.out = P))

G <- rWishart(1, P + 1, diag(P))[, , 1]
A <- diag(eigen(G)$values)
V <- eigen(G)$vectors

X <- matrix(rnorm(P*P), ncol = P) |> qr() |> qr.Q()

# set up sampling
Us_rst <- array(NA, c(P, P, its))
Covs_rst <- array(NA, c(P, P, its))

if (P <= 4) {
    Us_rj_rst <- array(NA, c(P, P, its))
    Covs_rj_rst <- array(NA, c(P, P, its))
}

# initialize the first entry as a random orthnormal matrix
Us_rst[, , 1] <- matrix(rnorm(P*P), ncol = P) |> qr() |> qr.Q()

# perform sampling
set.seed(17042025)
for (i in 2:its) {
    if (i %% 500 == 0) {print(i)}

    Us_rst[, , i] <- rbing.matrix.gibbs(G, H, Us_rst[, , i-1])
    Covs_rst[, , i] <- Us_rst[, , i] %*% A %*% t(Us_rst[, , i])

    if (P <= 4) {
        Us_rj_rst[, , i] <- rbing.Op(G, H)
        Covs_rj_rst[, , i] <- Us_rj_rst[, , i] %*% A %*% t(Us_rj_rst[, , i])
    }
}
```

```
## [1] 500
## [1] 1000
```

```r
# calculate average covariance matrices
avgCovs_rst <- apply(Covs_rst[, , (its/2) : its], c(1, 2), mean)
avgUs_rst <- eigen(avgCovs_rst)$vectors

if (P <= 4) {
    avgCovs_rj_rst <- apply(Covs_rj_rst[, , 2:its], c(1, 2), mean)
```

Mat1				Mat2			
0.51+0.51i	0.259+0.259i	0.774+0.774i	-0.27-0.27i	-0.515-0.515i	-0.258-0.258i	0.76+0.76i	-0.301-0.301i
-0.188-0.188i	0.905+0.905i	-0.272-0.272i	-0.268-0.268i	0.197+0.197i	-0.904-0.904i	-0.276-0.276i	-0.261-0.261i
0.518+0.518i	-0.236-0.236i	-0.492-0.492i	-0.659-0.659i	-0.522-0.522i	0.229+0.229i	-0.527-0.527i	-0.63-0.63i
0.661+0.661i	0.242+0.242i	-0.29-0.29i	0.649+0.649i	-0.65-0.65i	-0.253-0.253i	-0.263-0.263i	0.666+0.666i

```
    avgUs_rj_rst <- eigen(avgCovs_rj_rst)$vectors
}

# TODO what is the output I need, to make this into a function I can use repeatedly?
# a function for the rejection sampler
# a function for the Gibbs sampler
```

```
# distances
grass_dist(avgUs_rst, V)
```

```
## $sq_thetas
## [1] 0.000000e+00 0.000000e+00 2.220446e-16 8.881784e-16
##
## $sub_dist
## [1] 3.332001e-08
```

```
# in this case, one of the columns is flipped, and almost all of the "distance" is from
↪  that column
frame_distance(avgUs_rst, V)
```

```
## $fnorm
## [1] 8.003869
##
## $diags
## [1] 3.999770807 3.999827109 0.002137925 0.002133385
```

```
# if you flip the column sign, then the distance is much smaller, similar to the others
frame_distance(avgUs_rst %*% diag(c(-1, -1, 1, 1)), V)
```

```
## $fnorm
## [1] 0.004673394
##
## $diags
## [1] 0.0002291925 0.0001728915 0.0021379249 0.0021333847
```

```
# the order of column distances is exactly the same order as the Procrustes distances
procrustes_distance(avgUs_rst, V)
```

```
## $fnorm
## [1] 0.0186834
##
## $diags
## [1] 0.0009164222 0.0006913495 0.0085468915 0.0085287416
```

```r
print("rejection sampler")
```

```
## [1] "rejection sampler"
```

```r
# also compare with the samples from the rejection sampler
# distances
grass_dist(avgUs_rj_rst, V)
```

```
## $sq_thetas
## [1] 0.000000e+00 0.000000e+00 0.000000e+00 4.440892e-16
##
## $sub_dist
## [1] 2.107342e-08
```

```r
# in this case, one of the columns is flipped, and almost all of the "distance" is from
↪  that column
frame_distance(avgUs_rj_rst, V)
```

```
## $fnorm
## [1] 4.00003
##
## $diags
## [1] 1.888753e-05 3.999963e+00 3.141594e-05 1.664342e-05
```

```r
# if you flip the column sign, then the distance is much smaller, similar to the others
frame_distance(avgUs_rj_rst %*% diag(c(1, -1, 1, 1)), V)
```

```
## $fnorm
## [1] 0.0001043932
##
## $diags
## [1] 1.888753e-05 3.744635e-05 3.141594e-05 1.664342e-05
```

```r
# the order of column distances is exactly the same order as the Procrustes distances
procrustes_distance(avgUs_rj_rst, V)
```

```
## $fnorm
## [1] 0.0004175693
##
## $diags
## [1] 7.554960e-05 1.497839e-04 1.256626e-04 6.657327e-05

## [1] 500
## [1] 1000


## $sq_thetas
## [1] 0.000000e+00 0.000000e+00 0.000000e+00 0.000000e+00 0.000000e+00
## [6] 0.000000e+00 0.000000e+00 1.110223e-15
##
## $sub_dist
## [1] 3.332001e-08
```

Table 1: Sample average and True V eigenvectors, cols. 1 - 4

Avg				True V			
-0.299	0.291	0.406	0.734	-0.309	-0.292	0.403	0.733
0.240	0.834	0.012	-0.254	0.246	-0.835	0.005	-0.243
-0.579	0.011	-0.447	-0.116	-0.576	-0.019	-0.462	-0.088
-0.476	0.248	0.429	-0.431	-0.476	-0.252	0.427	-0.442
0.052	0.182	-0.471	0.329	0.051	-0.182	-0.473	0.332
0.343	0.167	0.029	0.213	0.340	-0.172	0.037	0.232
0.264	-0.301	0.460	-0.083	0.255	0.294	0.447	-0.048
0.319	0.075	-0.134	-0.190	0.321	-0.070	-0.129	-0.184

Table 2: Sample average and True V eigenvectors, cols. 5 - 8

Avg				True V			
-0.112	-0.217	0.146	0.203	-0.096	-0.222	0.140	0.204
-0.038	0.307	0.145	0.255	-0.031	0.307	0.139	0.264
-0.639	0.109	0.142	0.107	-0.635	0.140	0.118	0.103
0.026	-0.238	-0.467	-0.258	0.012	-0.227	-0.462	-0.259
0.080	-0.004	-0.776	0.162	0.132	-0.017	-0.771	0.133
-0.500	0.043	-0.071	-0.743	-0.483	0.066	-0.072	-0.746
-0.507	0.257	-0.330	0.436	-0.490	0.294	-0.363	0.433
-0.253	-0.850	0.064	0.219	-0.310	-0.833	0.041	0.222

```
## $fnorm
## [1] 4.016951
##
## $diags
## [1] 0.0002292590 3.9998116696 0.0005297416 0.0026223825 0.0070202621
## [6] 0.0034890599 0.0023079215 0.0009410019
```

```
## $fnorm
## [1] 0.01732796
##
## $diags
## [1] 0.0002292590 0.0001883304 0.0005297416 0.0026223825 0.0070202621
## [6] 0.0034890599 0.0023079215 0.0009410019
```

```
## $fnorm
## [1] 0.06918858
##
## $diags
## [1] 0.0009166368 0.0007522891 0.0021168919 0.0104699609 0.0280262503
## [6] 0.0139306888 0.0092155229 0.0037603406
```

Appendix

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

# Lambdak_gibbs_densCovar
```