NDFSM Data Analysis

Siddhesh Kulkarni, Subhadip Pal, Jeremy Gaskins

2023-09-22

This document demonstrates the code used in the manuscript "Bayesian Method for Sparse Canonical Analysis" by Siddhesh Kulkarni, Subhadip Pal, Jeremy T Gaskins.

First we load the source file and other packages on which our code is dependent upon. These files are available on Github at https://github.com/SiddheshKulkarni-source/CCA. The user should download these R files and place them in their directory so that they can be openned and run as source files.

```
source("NDFSM.R")

source("DFSM.R")

source("Bayesian_Summary_Data_Analysis.R")

load("Breast Cancer Data Demonstration.Rdata")
```

Breast Cancer Data Set

To demonstrate the use of our code, we consider the subset of the breast cancer data as discussed in Section 6 of the manuscript.

We apply our method to the breast cancer data available from https://tibshirani.su.domains/PMA/. There are $n = 89$ samples/observation on which DNA and RNA data are available. For the view 1 data, we consider the matrix of DNA copy numbers (DNA) for genes located on the 1^{st} chromosome, yielding $p^{(1)} = 136$ responses per sample. The data source also contains genetic expression levels (RNA) for $19,672$ genes, and we create the view 2 data by selecting the 50 genes located on chromosome 1 with the greatest variability (based on interquartile range). Also we select an additional 200 genes across the other 22 chromosome sites with the highest interquartile range to serve as genes that are likely to be unrelated to the view 1 data. This yields $p^{(2)} = 250$. We standardize each column in both data views.

The following code chunk performs the data cleaning steps for obtaining the DNA and RNA views described above. The required source file for this chunk is "databuild.R" which is loaded in the code chunk before.

We have also provided the obtained data in the form an RData file. We demonstrate the key features.

```
X_1 <- Data_Set_1$X_1_dna #View 1 data of copy numbers

dim(X_1)
```

```
## [1]  89 136
```

```
X_1[1:5,1:10] ## Print the data for first 5 patients and first 10 copy numbers
```

```
##         [,1]       [,2]       [,3]       [,4]       [,5]       [,6]       [,7]
##  -1.51522841 -1.8763116 -2.2522721 -0.1463239 -1.8439237 -1.8551605 -1.6234971
##  -1.08001176 -0.8680335 -1.1316450 -1.0917276  0.3386091 -0.7682492 -1.0485470
##  -1.06477981 -1.1279677 -0.9957609 -0.5070604 -0.7015775 -1.1569762 -0.5342318
```

```
##  -0.08022344  1.1883253  0.6708390  1.3373235 -0.9799449  1.5010204  0.9593438
##   0.17445959 -0.2594383 -0.6419829 -0.5798809  0.7453039 -0.4717568 -0.2871632
##        [,8]       [,9]      [,10]
##  -1.4998537 -1.6462667 -1.5183150
##  -0.6780226 -0.8900621 -1.2666292
##  -0.8166667 -0.9628216 -0.6203015
##   2.2192247  1.5972277  0.9955847
##  -0.5819052 -0.0852466 -0.4613476
```

```
X_2 <- Data_Set_1$X_2_rna #View 2 data with RNA gene expression

dim(X_2)
```

```
## [1]  89 250
```

```
X_2[1:5,1:10] ## Print the data for first 5 patients and first 10 RNA expressions
```

```
##       PRO1489       CDW52        G1P3        P28       CDC20    TSPAN-1
## V1 -0.4767343 -0.58881325 -1.25335395 -1.0105993  1.66108300 -1.3939880
## V2  0.7890941  0.21734443  0.34363490  2.5039883  1.96758356  0.1954492
## V3  0.5219975  0.04829929 -0.16224287  1.1417962  0.13619191 -0.3304225
## V4 -0.2530066  1.50303936 -1.93187490 -1.0020529  1.48180707 -1.2842999
## V5  0.8426768 -0.27194474  0.02675951  0.7280876 -0.01177351  1.0868696
##        CYP4B1       DD96   KIAA0452      PDE4B
## V1 -0.8366050 -1.3055377 -1.0865550 -1.3107094
## V2 -0.4137532 -1.1416697  0.7296188  0.3782305
## V3 -0.5375476  0.4997691 -0.9939956  0.4868516
## V4 -0.8556594 -1.4117527 -1.4249857  0.6038237
## V5 -0.7090564 -1.4913505  0.1163084 -0.7406977
```

```
no_patients <- nrow(X_1)
```

In the view 2 data, the column names depicts the gene names.
Additionally, this RData file contains `gene_chr_250`, which indicates the chromosome location for each of genes in View 2.

```
table(gene_chr_250)
```

```
## gene_chr_250
##  1  2  3  4  5  6  7  8  9 10 11 12 13 14 15 16 17 18 19 20 21 22 23
## 50  7 15 14 10 17 11 16  6  7 17  6  2  6  8  9  7  4 10  9  4  6  9
```

Using this dataset, we demonstrate how to use our code to fit each model. First, we start with our Non-Diagonal Factor Shrinking Model (NDFSM).

NDFSM and DFSM Model Fitting

Function Arguments

Many of the functions we introduce are based on a common set of inputs/arguments. These include

- *d*: Indicates number of factors used in the factor models.
- *X_1*: Data matrix for View 1.
- *X_2*: Data matrix for View 2. Note that *X_1* and *X_2* must have the same number of rows.
- *CCA_select*: Number of Canonical Correlation coefficients to be estimated/stored. This must be less than or equal to *d*.

- *Burn Iterations*: For MCMC number of iterations are needed to burn before attaining stationary state.
- *MCMC Iterations*: For MCMC, number of iterations are needed to run the model to obtain output.
- *thin*: For MCMC, the iteration that is being saved. For example, if thin is equal to 7, then after completing burn-in, samples from every 7th iteration will be saved.

Run NDFSM and DFSM

The following code runs the MCMC algorithm for the NDFSM and DFSM models. By default, these lines of code are commented out to allow the user to skip to the next block for inference using an existing sample.

```
# Setting up the Burn Iterations and MCMC Iterations

burn_iter <- 25000
thin <- 15
mcmc_iter <- 75000
CCA_select <- 10

# # Running the code for NDFSM
# time.start <- Sys.time()
# NDFSM_Output <-
#   NDFSM(
#     d = 20,
#     burn_iter = burn_iter,
#     mcmc_iter = mcmc_iter,
#     thin = thin,
#     X_1 = X_1,
#     X_2 = X_2,
#     CCA_select = CCA_select
#   )
# save(Output, file=paste0('NDFSM Output_', format(Sys.time(), "%b %e %Y"), '.RData'))
# time.end <- Sys.time()
# print( time.end - time.start)


# # Running the code for DFSM
# time.start <- Sys.time()
# DFSM_Output <-
#   DFSM(
#     d = 20,
#     burn_iter = burn_iter,
#     mcmc_iter = mcmc_iter,
#     thin = thin,
#     X_1 = X_1,
#     X_2 = X_2,
#     CCA_select = CCA_select
#   )
# save(Output, file=paste0('DFSM Output_', format(Sys.time(), "%b %e %Y"), '.RData'))
# time.end <- Sys.time()
# print( time.end - time.start)
```

Note that we save this output file to call later into a dataframe labeled as either `NDFSM_Output` or `DFSM_Output`, along with the data the code was run. This way it can be called later for analysis as needed.

Evaluation of posterior samples

The output of the NDFSM code contains the stored posterior samples of all objects of interest. If using existing code, this block can be used to open the output from these previously run code.

```
load('NDFSM_BC_Output.Rdata')  ## loading output from already ran code for NDFSM

#load('DFSM_BC_Output.Rdata') ## loading output from already ran code for DFSM

names(Output)
```

```
##  [1] "A_1_MCMC"                  "A_2_MCMC"
##  [3] "Mu_1_MCMC"                 "Mu_2_MCMC"
##  [5] "Omega_1_MCMC"              "Omega_2_MCMC"
##  [7] "log_det_MCMC"              "CCA_MCMC"
##  [9] "Direction_CCA_Vec1_MCMC"   "Direction_CCA_Vec2_MCMC"
## [11] "Log_likelihood_MCMC_Grand" "tausqpost_1"
## [13] "tausqpost_2"               "eta"
```

We store a total of G samples, where $G =$`mcmc_iter/thin`. `A_1_MCMC` is a $p^{(1)} \times d \times G$ dimensional array where `A_1_MCMC[ , ,g]` is the g-th sample of the A_1 matrix. Similarly, `Mu_1_MCMC` and `Omega_1_MCMC` store the sampled mean vectors and the generalized specificity matrices for view 1. `A_2_MCMC`, `Mu_2_MCMC`, and `Omega_2_MCMC` contain the samples for the view 2 parameters.

`CCA_MCMC` contains is a $1 \times$`CCA_select`$\times G$ array where the gth sample is a vector of the stored canonical correlations. `Direction_CCA_Vec1_MCMC` and `Direction_CCA_Vec2_MCMC` contain the corresponding loading/direction vectors with dimension $p^{(m)} \times$`CCA_select`$\times G$.

```
dim(Output$A_1_MCMC)
```

```
## [1]  136   20 5000
```

```
dim(Output$Mu_1_MCMC)
```

```
## [1]    1  136 5000
```

```
dim(Output$Omega_1_MCMC)
```

```
## [1]  136  136 5000
```

```
dim(Output$CCA_MCMC)
```

```
## [1]    1   10 5000
```

```
dim(Output$Direction_CCA_Vec1_MCMC)
```

```
## [1]  136   10 5000
```

Summary Analysis Function

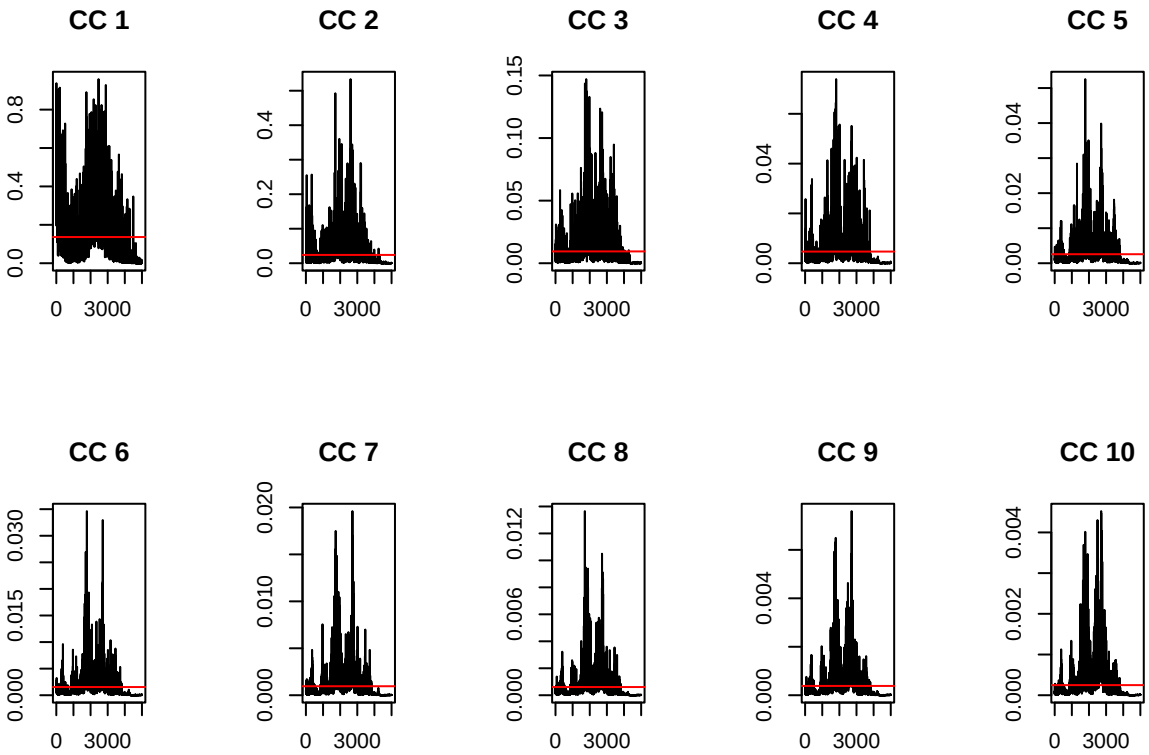
Following function calculates the summary statistics from the MCMC run, as well as producing traceplots to evaluate MCMC convergence. It requires following arguments:

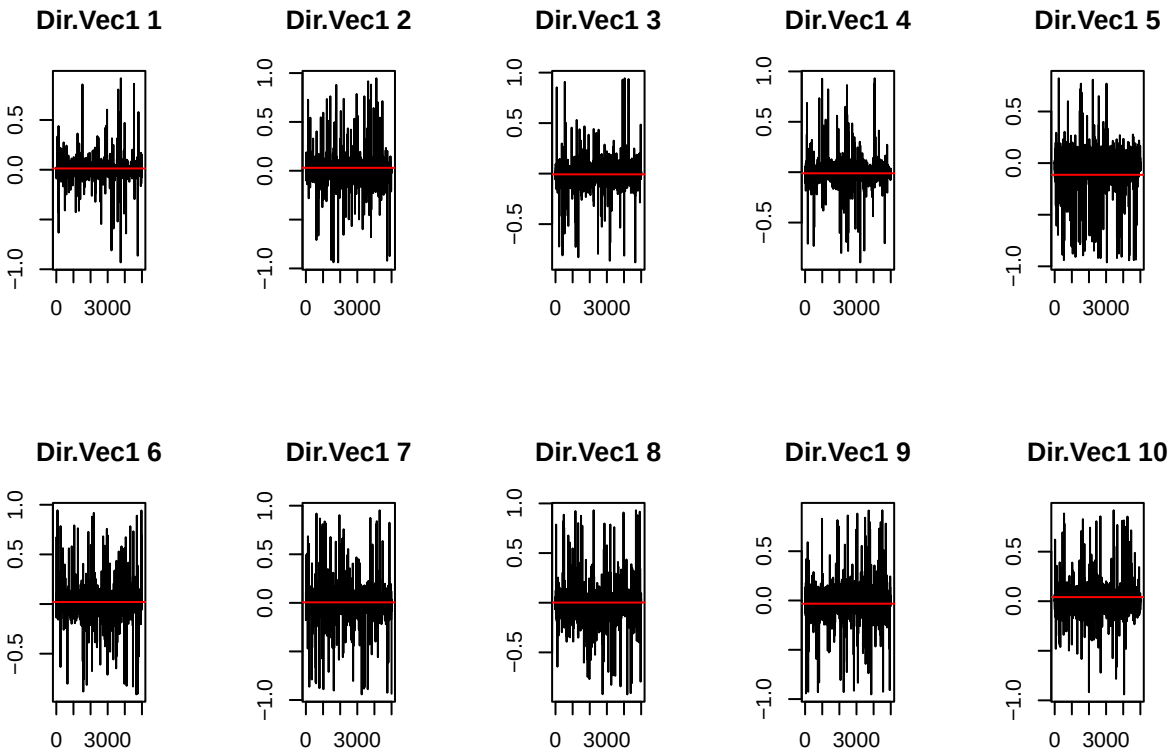
- *Output*: MCMC output from the NDFSM code.
- *Shrinkage Threshold*: A user defined threshold value to assess for potential overshrinking. We evaluate the posterior distribution of the first CC, and if it is smaller than this threshold with posterior probability greater than 50%, then overshrinkage might have occurred.
- *shrinkage_check*: By default TRUE. Determines whether to perform the check of overshrinkage.
- *plot_indicator*: By default TRUE. Generates traceplots.

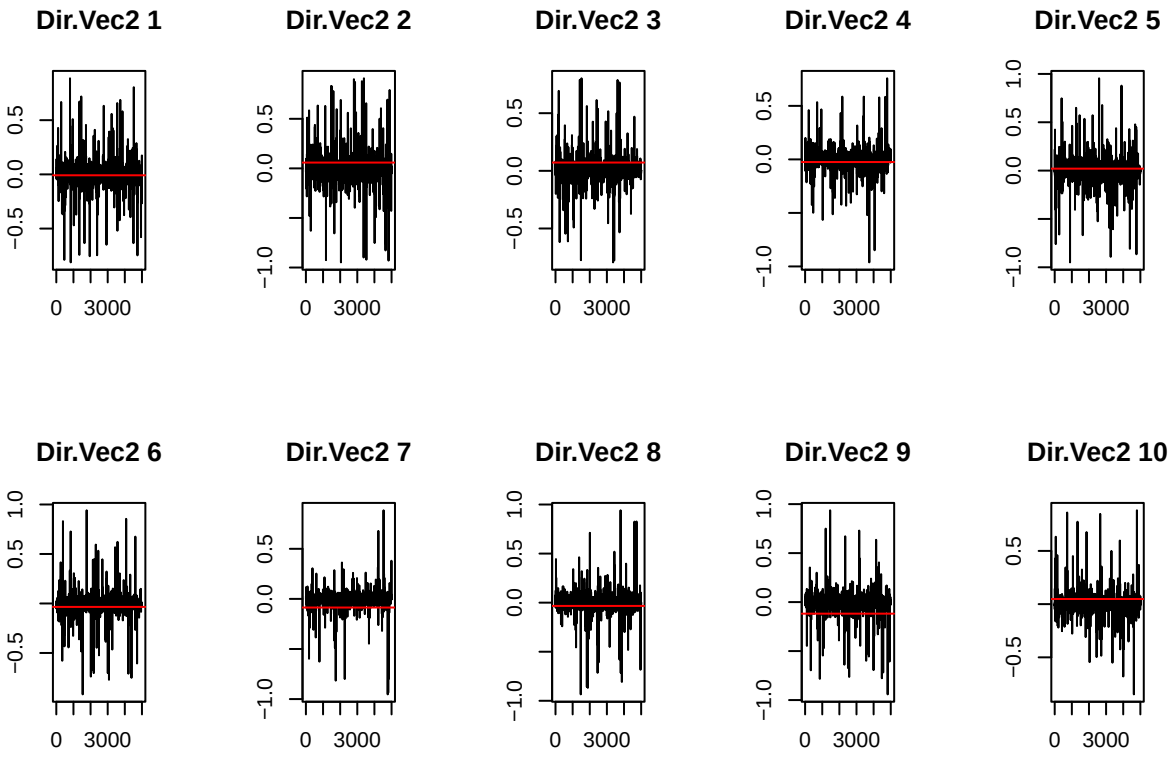
- *plot_components*: Number of components of first vectors of both views needed to be plot.
- *alpha.CC*: determines the level for the credible intervals of the canonical correlations. We generally recommend using 0.05 for 95% intervals.
- *alpha.dir*: determines the level for the credible intervals of the direction vectors. We generally recommend using 0.5 for 50% intervals due to the heavy tailed nature of the horseshoe priors.

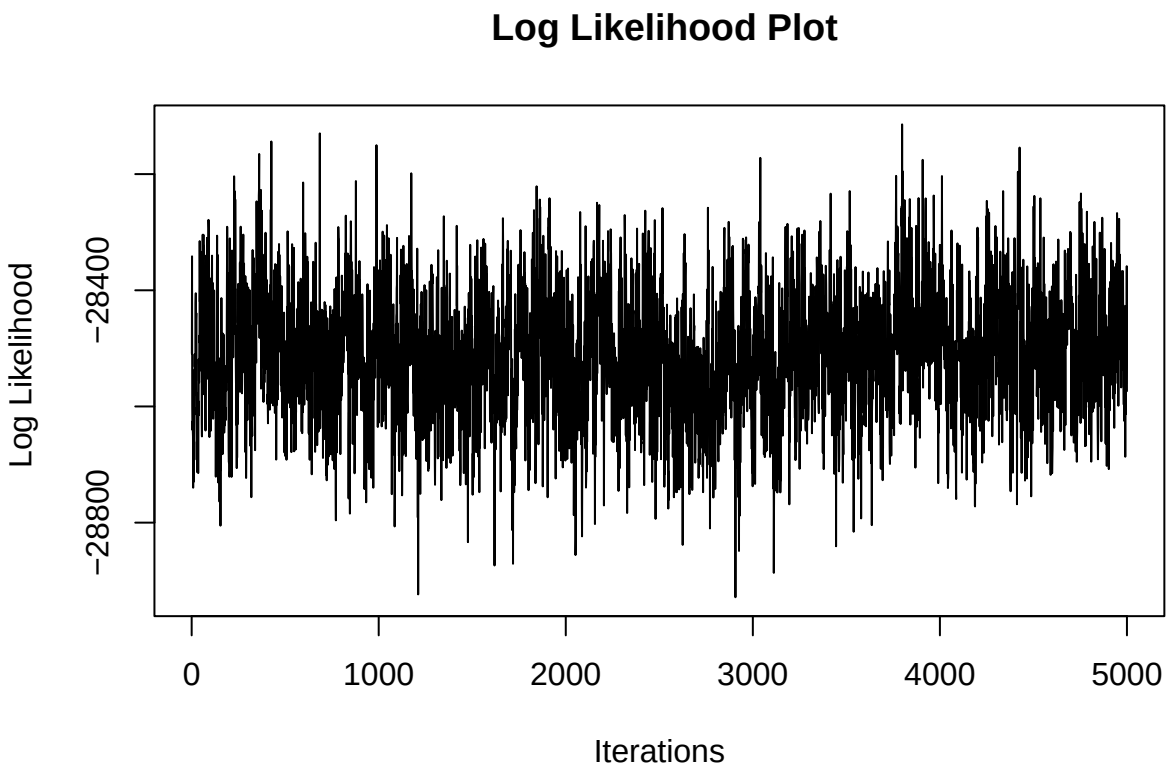
For demonstration, we will plot a selection of traceplots to investigate MCMC convergence. If `plot_indicator=TRUE`, we plot each of canonical correlations (up to `CCA_select`), and the first `plot_components` components of the direction vector of the first CC in both views. The red line designates the posterior estimate of the corresponding parameter. Additionally, traceplots for the log-likelihood function and the log-determinant of the full covariance matrix are considered to further assess overall mixing.

```
Summary_Analysis <-
  Analysis_Summary(
    Output = Output,
    Data=Data_Set_1,
    shrinkage_threshold = 0.2,
    plot_indicator = TRUE,
    plot_components = 10,
    alpha.CC=0.05, alpha.dir=0.50,
    shrinkage_check = TRUE
    )
```

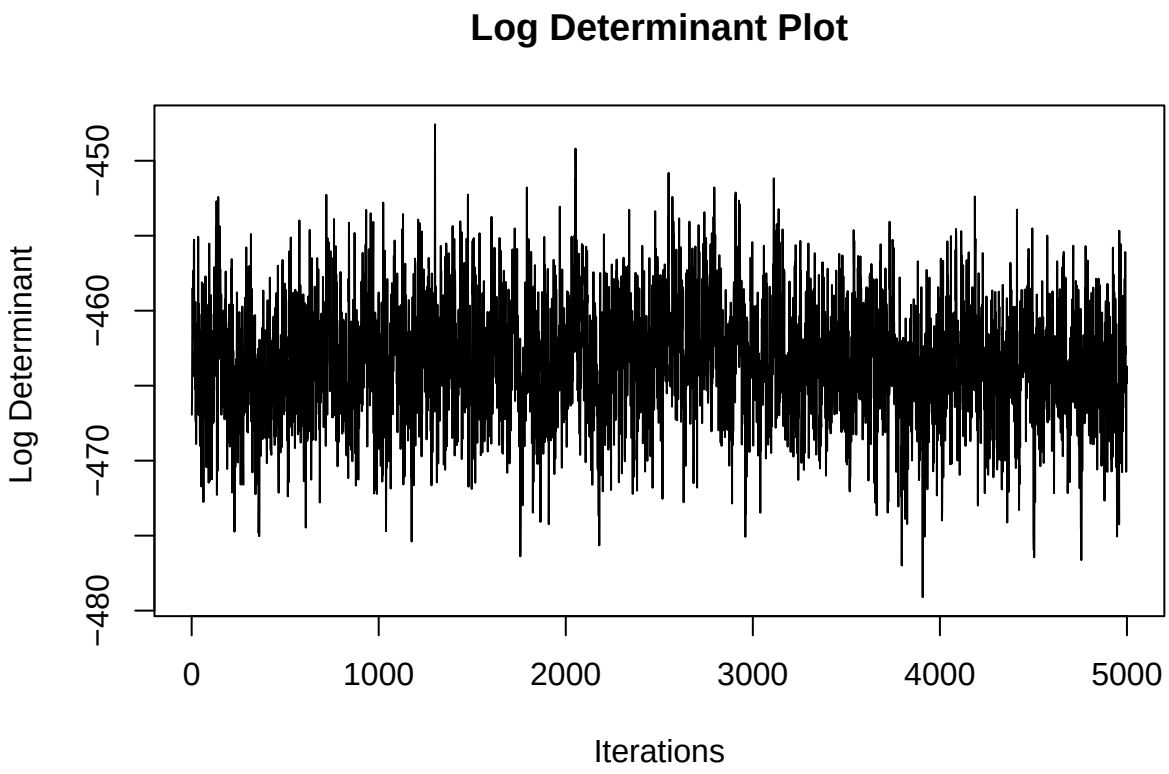



Dir.Vec1 1
Dir.Vec1 2
Dir.Vec1 3
Dir.Vec1 4
Dir.Vec1 5
Dir.Vec1 6
Dir.Vec1 7
Dir.Vec1 8
Dir.Vec1 9
Dir.Vec1 10
0
3000


Dir.Vec2 1
Dir.Vec2 2
Dir.Vec2 3
Dir.Vec2 4
Dir.Vec2 5
Dir.Vec2 6
Dir.Vec2 7
Dir.Vec2 8
Dir.Vec2 9
Dir.Vec2 10
0
3000




```
## Run DFSM Model; Overshrinkage is suspected
```



We consider the effective sample size of the first CCA as well as log-likelihood as a marker of the amount of information contained in a posterior sample. We typically seek for this value to be at least 1000. These effective samples sizes are given in

```
Summary_Analysis$effective_size_first_CCA
```

```
##     var1
## 17.22524
```

```
Summary_Analysis$effective_size_log_likelihood
```

```
##     var1
## 635.2769
```

Summary Analysis Output and Interpretation

The output of the `Analysis_Summary` function contains the following objects:

```
names(Summary_Analysis)
```

```
##  [1] "Shrinkage_calculator"
##  [2] "first_direction_significant_features_View1"
##  [3] "first_direction_significant_features_View2"
##  [4] "V1.dir.data.frame"
##  [5] "V2.dir.data.frame"
##  [6] "Direction_Vector_View_1"
##  [7] "Direction_Vector_View_2"
##  [8] "effective_size_first_CCA"
##  [9] "effective_size_log_likelihood"
```

```
## [10] "CC.data.frame"
```

The objects contain the following information:

- `Shrinkage_calculator`: The posterior probability that the first CC is less than the `shrinkage_threshold`.
- `first_direction_significant_features_View1` and `_View2`: These provide indicator vectors representing the significant features in direction vector 1 for View 1 and View 2.
- `V1.dir.data.frame` and `V2.dir.data.frame`: These are *p1* by 3 and *p2* by 3 data frames containing the estimated first direction vector and its credible interval for each view.
- `Direction_Vector_View_1` and `Direction_Vector_View_2`: These are *p1* by *G* and *p2* by *G* data frames containing the samples for the first direction vectors for view 1 and view 2 after applying the identifiability adjustment.
- `effecticve_size_first_CCA` and `effecticve_size_log_likelihood`: These provide the effective sample sizes for the first canonical correlation and for the log-likelihood.
- `CC.data.frame`: The summary file contains the posterior means of the first `CCA_select` canonical correlations, and the credible interval bounds for each.

Once, we have established that MCMC convergence is appropriate from the traceplots and ESS, we next need to determine if the NDFSM output is compromised due to potential overshrinkage. (This is not relevant if using the DFSM model.) To that end, we consider `Shrinkage_calculator`, the posterior probability that the first CC is less than the `shrinkage_threshold`.

```
Summary_Analysis$Shrinkage_calculator
```

```
## [1] 0.7458
```

If this exceeds alpha, which is set 0.5 here, there is a high probability given to low CC suggesting overshrinkage. The Summary function gives out the message which tells whether to continue with NDFSM (if `shrinkage_check` is set to TRUE). In this case as the `Shrinkage_calculator` suggest the possibility of overshrinkage, we need to run diagonal version of the model. Note that running the summary function will automatically output one of following two message to instruct the use on how to proceed:

- Continue with NDFSM; No evidence of overshrinkage
- Run DFSM Model; Overshrinkage is suspected

We also note that in cases with high levels of over-shrinkage, there are often issues with MCMC convergence with poor ESS.

Posterior Inference (Point Estimates and Intervals)

If we pass the overshrinkage criteria, we will perform inference using these samples. Firstly, we consider the posterior means and the credible intervals for the canonical correlations.

```
Summary_Analysis$CC.data.frame
```

```
##    Estimated_value  Lower_bound Upper_bound
## 1     0.1363402204 3.010593e-04 0.570019652
## 2     0.0240522364 8.641021e-05 0.126178966
## 3     0.0094450021 4.123263e-05 0.049448709
## 4     0.0046815503 2.263411e-05 0.023548223
## 5     0.0025749269 1.254158e-05 0.013019764
## 6     0.0015382006 7.370991e-06 0.007947647
## 7     0.0009562268 3.751904e-06 0.005259952
## 8     0.0006073286 2.072342e-06 0.003234042
## 9     0.0003833727 1.112521e-06 0.002044053
## 10    0.0002482929 6.765963e-07 0.001397225
```

Next, we consider the estimated direction vectors. We also have credible intervals for each direction component.

```
#Direction Vectors Summary of View 1

head(round(Summary_Analysis$V1.dir.data.frame,4),20)
```

```
##    Estimated_value Lower_bound Upper_bound
## 1           0.0127     -0.0094      0.0097
## 2           0.0300     -0.0119      0.0126
## 3          -0.0075     -0.0119      0.0122
## 4          -0.0109     -0.0117      0.0110
## 5          -0.1133     -0.0129      0.0168
## 6           0.0209     -0.0131      0.0140
## 7           0.0054     -0.0136      0.0140
## 8           0.0016     -0.0147      0.0147
## 9          -0.0322     -0.0150      0.0128
## 10          0.0414     -0.0142      0.0148
## 11         -0.0356     -0.0159      0.0140
## 12         -0.0089     -0.0132      0.0124
## 13         -0.0442     -0.0142      0.0135
## 14          0.0527     -0.0138      0.0148
## 15          0.0307     -0.0124      0.0125
## 16          0.0184     -0.0145      0.0138
## 17         -0.0084     -0.0144      0.0147
## 18         -0.0077     -0.0152      0.0152
## 19         -0.0394     -0.0118      0.0122
## 20          0.0101     -0.0146      0.0138
```

```
round(Summary_Analysis$V2.dir.data.frame,4)[1:20,]
```

```
##          Estimated_value Lower_bound Upper_bound
## PRO1489          -0.0085     -0.0089      0.0089
## CDW52             0.0585     -0.0118      0.0118
## G1P3              0.0719     -0.0089      0.0085
## P28              -0.0245     -0.0079      0.0075
## CDC20             0.0201     -0.0091      0.0100
## TSPAN-1          -0.0345     -0.0075      0.0073
## CYP4B1           -0.0860     -0.0073      0.0075
## DD96             -0.0334     -0.0080      0.0077
## KIAA0452         -0.1182     -0.0095      0.0086
## PDE4B             0.0481     -0.0080      0.0079
## FLJ23091          0.0133     -0.0078      0.0080
## MGC3184           0.0272     -0.0068      0.0063
## C1orf29          -0.0162     -0.0110      0.0112
## IFI44             0.1263     -0.0079      0.0086
## GBP1              0.1190     -0.0098      0.0105
## VCAM1             0.0121     -0.0092      0.0091
## COL11A1           0.0938     -0.0110      0.0111
## AMY1A            -0.0408     -0.0063      0.0059
## VAV3              0.0177     -0.0067      0.0074
## KIAA1324          0.0066     -0.0071      0.0070
```

We determine which components are significantly associated across views by determining which features have a credible interval that excludes zero.

```
# View 1 Significant features

#number of view 1 significant features
sum(Summary_Analysis$first_direction_significant_features_View1==1)
```

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

```
#location of view 1 significant features
which(Summary_Analysis$first_direction_significant_features_View1==1)
```

```
## [1] 118
```

```
# View 2 Significant features

names(Summary_Analysis$first_direction_significant_features_View2) <-
    colnames(Data_Set_1$X_2_rna)

#number of view 2 significant features
sum(Summary_Analysis$first_direction_significant_features_View2==1)
```

```
## [1] 0
```

```
#location of view 2 significant features
which(Summary_Analysis$first_direction_significant_features_View2==1)
```

```
## named integer(0)
```

Specific to our breast cancer analysis, we also considered how many of these view 2 significant features are associated with the first chromosome. We also compute the weight in the view 2 direction vector estimate that is associated with the genes that are truly located on chromosome 1 (which determines view 1).

```
### Chromosome location for View 2 Significant features

# Significant genes by chromosome
table(gene_chr_250[t(Summary_Analysis$first_direction_significant_features_View2)==1] )
```

```
## < table of extent 0 >
```

```
# Proportion of significant genes on chromosome 1
mean(gene_chr_250[t(Summary_Analysis$first_direction_significant_features_View2)==1]==1 )
```

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

```
# Names of significant genes on chromosome 1
rownames(gene_chr_250)[ which(Summary_Analysis$first_direction_significant_features_View2==1 &
        gene_chr_250==1) ]
```

```
## character(0)
```

```
### View 2 direction vector contribution from chr 1 components
sum((Summary_Analysis$V2.dir.data.frame[,1]^2)[gene_chr_250==1])
```

```
## [1] 0.1979552
```