1 A Toy Example

In this section, we define a simple objective function called *eval*() which calculates the sum of a penalty term and the squared error between the DataRemix reconstruction and the original input matrix. The input matrix is a 100-by-9 matrix with random values. In this case, we know that when k=9,p=1 or $\mu = 1$, p=1, DataRemix reconstruction is the same as the orginial matrix and the objective function achieves the minimal value which is qual to the penalty term we add.

```
> library(DataRemix)
> eval <- function(X_reconstruct, X, penalty){
+   return(-sum((X-X_reconstruct)^2)+penalty)
+ }#eval
```

First we genrate a random matrix with dimension 100-by-9 and perform the SVD decomposition.

```
> set.seed(1)
> num_of_row <- 100
> num_of_col <- 9
> X <- matrix(rnorm(num_of_row*num_of_col), nrow = num_of_row, ncol = num_of_col)
> svdres <- svd(X)
```

Set mt to be 2000.

```
> basis_short <- omega[1:2000,]
```

Infer the optimal combinations of k, p and μ. Here X and *penalty* are additional inputs for the *eval*() function.

```
> DataRemix.res <- DataRemix(svdres, eval, k_limits = c(1, length(svdres$d)),
+                  p_limits = c(-1,1), mu_limits = c(1e-12,1),
+                  num_of_initialization = 5, num_of_thompson = 50,
+                  basis = basis_short, xi = 0.1, full = T, verbose = F,
+                  X = X, penalty = 100)
> knitr::kable(cbind(1:55,DataRemix.res$para), align = "l",
+              col.names = c("Iteration","k","p","mu","Eval"))
```

Iteration	k	p	mu	Eval
1	8	0.9343941	0.8669163	80.133470
2	4	-0.6161244	0.0822944	-774.549343
3	6	-0.8592770	0.5276627	-674.508131
4	5	-0.9036173	0.5945408	-595.209680
5	8	0.1977374	0.0279159	-608.454077
6	5	-0.3638044	0.0000241	-813.670282
7	2	-0.8046778	0.0000000	-853.465623
8	9	1.0000000	0.0002110	100.000000

9	9	0.6661921	0.0154153	-184.706775
10	8	0.6531390	0.0000000	-244.330065
11	8	1.0000000	0.2141172	62.417337
12	9	1.0000000	0.0000423	100.000000
13	1	0.8418133	0.0000000	-717.557847
14	9	1.0000000	0.5691544	100.000000
15	9	0.6548131	1.0000000	-197.416773
16	4	1.0000000	0.0000000	-312.285772
17	9	0.8373235	0.0000000	2.505949
18	7	1.0000000	0.0000000	-36.839791
19	9	0.9417058	0.0015476	84.135849
20	5	1.0000000	1.0000000	100.000000
21	8	1.0000000	0.0000000	39.148404
22	8	1.0000000	0.0900272	49.611804
23	3	0.9365716	0.0889231	-349.709165
24	6	1.0000000	0.9720235	99.830010
25	8	0.9251656	0.0003053	15.257530
26	9	1.0000000	0.0211273	100.000000
27	5	0.8231541	0.0000024	-282.890004
28	8	1.0000000	0.0024223	39.442852
29	9	-0.0882469	0.0000000	-716.342373
30	9	1.0000000	0.0000002	100.000000
31	5	1.0000000	0.0749025	-158.267025
32	3	0.7020952	0.3133165	-267.634044
33	8	0.9529303	0.0001027	29.061410
34	5	0.7020183	1.0000000	-76.158754
35	1	0.1021742	1.0000000	-28.785900
36	5	0.8742549	0.5679676	-2.442805
37	7	0.7783341	0.0000000	-176.730064
38	7	0.9761727	0.0000957	-39.355145
39	9	-0.7851174	0.0000025	-830.610836
40	9	0.6304672	0.0000001	-224.357930
41	1	0.7011276	0.9561123	53.287770
42	7	0.7868272	0.0000064	-168.649452
43	8	1.0000000	0.0000009	39.148512
44	9	1.0000000	0.0000000	100.000000
45	7	1.0000000	0.0000000	-36.839787
46	9	0.8495929	0.7043951	14.351652
47	9	1.0000000	0.0000013	100.000000
48	6	1.0000000	0.0000000	-117.188654
49	9	1.0000000	0.0000000	100.000000
50	9	0.9622713	0.0000000	93.028830
51	9	0.9252662	0.0000251	74.898253
52	8	0.9912851	0.0000000	38.769336
53	4	1.0000000	1.0000000	100.000000
54	7	1.0000000	0.0000000	-36.839791

```
|55        |5  |0.3441199  |0.0000000 |-621.586219 |
```

2 GTex Correlation Network

In this section, we define a different task of optimizing the known pathway recovery based on the GTex gene expression data. $corMatToAUC()$ is the main objective function with two inputs: *data* and GS. We formally define the objective as the average AUC across pathways and we also keep track of the average AUPR value. You can refer to the $corMatToAUC()$ document for more information.

```
> library(DataRemix)
```

Load the data. $GTex\_cc$ stands for the GTex gene correlation matrix with dimension 7294-by-7294 and *canonical* represents the canonical mSigDB pathways with dimension 7294-by-1330. It takes time to decompose $GTex\_cc$, thus we pre-compute the SVD decomposition of $GTex\_cc$ and load it as $GTex\_svdres$.

```
> load(url("https://www.dropbox.com/s/o949wkg76k0ccaw/GTex_cc.rdata?dl=1"))
> load(url("https://www.dropbox.com/s/wsuze8w2rp0syqg/GTex_svdres.rdata?dl=1"))
> load(url("https://github.com/wgmao/DataRemix/blob/master/inst/extdata/canonical.rdata?raw=
> #svdres <- svd(GTex_cc)
```

Run corMatToAUC() on the default correlation matrix $GTex\_cc$.

```
> GTex_default <- corMatToAUC(GTex_cc, canonical)
> GTex_default
```

```
[1] 0.0450869 0.7238648
```

Set mt to be 2000.

```
> basis_short <- omega[1:2000,]
```

Infer the optimal combinations of k, p and μ. Here GS is the additional input for the $corMatToAUC()$ function.

```
> DataRemix.res <- DataRemix(GTex_svdres, corMatToAUC,
+                            k_limits = c(1, length(GTex_svdres$d)%/%2),
+                            p_limits = c(-1,1), mu_limits = c(1e-12,1),
+                            num_of_initialization = 5, num_of_thompson = 150,
+                            basis = basis_short, xi = 0.1, full = T, verbose = F,
+                            GS = canonical)
> knitr::kable(cbind(1:15,DataRemix.res$full[order(DataRemix.res$para[,4],decreasing = T)
+              [1:15],]), align = "l", col.names = c("Rank","k","p","mu",
+              "mean AUPR", "mean AUC"))
```

Rank	k	p	mu	mean AUPR	mean AUC
1	2228	0.3403299	0.0000463	0.1050642	0.7759018
2	2255	0.3272925	0.2449971	0.1064295	0.7757571
3	2148	0.3457493	0.6718778	0.1046002	0.7757414
4	634	0.3136743	0.0073647	0.0981968	0.7756265
5	1928	0.3217277	0.0000000	0.1060840	0.7755995
6	978	0.3081544	0.3278853	0.1031726	0.7754414
7	2474	0.3426601	0.0000000	0.1052944	0.7753801
8	2539	0.3299483	0.0000000	0.1064598	0.7753504
9	3246	0.3342323	0.0135609	0.1059412	0.7753500
10	1952	0.3131336	0.0010847	0.1068391	0.7753464
11	1639	0.3430526	0.0000000	0.1035629	0.7753172
12	1600	0.3123257	0.0000028	0.1059670	0.7753143
13	1489	0.3283818	0.0000752	0.1039642	0.7752730
14	2591	0.3511484	0.0000021	0.1044931	0.7752570
15	1414	0.3309955	0.0000002	0.1033410	0.7751931