```
> library(DataRemix)
```

1 A Toy Example

Define a simple objective function 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, it achieves the maximal value which is qual to the penalty term.

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

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	1	-0.8864492	0.0000000	-857.307510
7	9	0.9710026	1.0000000	95.796992

8	9	0.5652128	1.0000000	-294.183178
9	2	0.3076150	0.0000000	-762.471773
10	9	1.0000000	0.0213327	100.000000
11	8	0.8987462	0.2482674	24.395734
12	9	0.7693587	0.0307309	-68.911583
13	4	1.0000000	0.2702496	-119.556864
14	5	0.6702578	0.6061702	-148.365491
15	6	0.9714513	0.4360788	27.620644
16	8	1.0000000	0.0115548	40.546538
17	8	0.9275926	0.0010597	16.749705
18	7	-0.8048691	0.0000000	-837.884176
19	1	0.8895300	0.4589730	-143.745379
20	1	0.2451882	0.5008529	-215.580317
21	5	1.0000000	1.0000000	100.000000
22	6	0.9021402	0.0021875	-149.272093
23	9	1.0000000	0.1065941	100.000000
24	8	1.0000000	0.0000000	39.148407
25	5	0.1647886	0.0208602	-687.531413
26	7	0.7651296	0.0000000	-189.425162
27	4	0.9763708	0.0003240	-313.770572
28	5	0.6015909	1.0000000	-154.939640
29	8	1.0000000	0.0000000	39.148404
30	9	0.8481905	0.0000000	13.019424
31	5	1.0000000	0.0000000	-201.782435
32	6	0.4246590	0.0000000	-536.224365
33	8	0.8861626	0.0000000	-11.472133
34	7	1.0000000	1.0000000	100.000000
35	9	0.8752923	0.3208530	37.626562
36	7	1.0000000	0.0968655	-11.613634
37	2	1.0000000	1.0000000	100.000000
38	3	0.7020952	0.3133165	-267.634044
39	8	0.8898831	0.0000000	-8.621272
40	9	1.0000000	0.0000000	100.000000
41	8	1.0000000	0.0000002	39.148434
42	7	0.9423744	0.0000000	-50.562141
43	9	0.9474766	0.0000000	86.947638
44	5	0.9509870	1.0000000	91.602471
45	1	-0.0133796	0.0000000	-836.453933
46	9	1.0000000	0.0000016	100.000000
47	9	1.0000000	0.0000000	100.000000
48	1	0.9430562	0.0001217	-702.783740
49	7	0.7776872	0.0000448	-177.336161
50	9	0.8566499	0.0000002	20.964082
51	5	0.8830432	0.0253040	-227.421134
52	2	1.0000000	0.1112699	-417.742749
53	9	0.8240095	0.0000018	-10.793403

54	9	1.0000000	0.0000000	100.000000
55	9	1.0000000	0.0773820	100.000000