```
> 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, lower_limit = c(1,-1,0),
+                   upper_limit = c(length(svdres$d), 1,1), num_of_initialization = 5,
+                   num_of_thompson = 50, basis = basis_short, xi = 0.1, full = T,
+                   verbose = F, X = X, penalty = 100)
> knitr::kable(cbind(1:55,DataRemix.res$para), align = "l",
+              col.names = c("Iteration","k","p","mu","Eval"))
```

Iteration	k	p	mu	Eval
1	8	0.9343941	0.8669163	80.13347
2	4	-0.6161244	0.0822944	-774.54934
3	6	-0.8592770	0.5276627	-674.50813
4	5	-0.9036173	0.5945408	-595.20968
5	8	0.1977374	0.0279159	-608.45408
6	6	0.7600058	0.7491820	-57.75474
7	1	0.9732232	0.7962543	66.09687
8	2	0.6264308	0.8342587	-30.90262

9	5	0.9392592	0.7591503	69.95399
10	9	1.0000000	1.0000000	100.00000
11	4	0.1871164	0.7129497	-343.61286
12	2	1.0000000	0.9781217	99.68624
13	2	0.6365943	0.4338173	-219.30523
14	1	0.5156579	0.7223355	-41.68571
15	7	1.0000000	0.7208022	89.33314
16	8	0.4294922	0.5433661	-407.64110
17	9	1.0000000	0.6235153	100.00000
18	5	0.5461408	0.9202155	-197.56072
19	8	0.9772540	0.9936968	97.49998
20	4	1.0000000	0.5460449	15.03810
21	1	-0.0173672	1.0000000	-36.59617
22	1	0.9025561	0.6182613	-24.26296
23	8	0.8715479	0.3022851	8.03090
24	8	0.9828520	0.6873251	92.61232
25	8	0.9454568	0.1996929	47.69907
26	1	0.5610817	0.9933571	27.84442
27	3	0.3069788	0.9803562	-189.58190
28	2	0.9187318	0.0457580	-507.27793
29	2	1.0000000	0.7288770	51.81555
30	9	-0.0317688	1.0000000	-697.04950
31	4	0.9620278	0.6540939	46.30062
32	6	1.0000000	0.1181146	-68.91236
33	3	1.0000000	0.8792390	92.25942
34	4	0.9115363	0.8683955	71.85496
35	4	1.0000000	1.0000000	100.00000
36	6	0.9603441	0.5743332	54.42460
37	7	1.0000000	0.4080560	52.05165
38	1	0.8956883	1.0000000	91.35505
39	9	1.0000000	0.1709873	100.00000
40	9	1.0000000	0.4663211	100.00000
41	8	1.0000000	0.1373955	54.72115
42	9	0.9790435	0.0000000	97.76288
43	8	1.0000000	0.5319136	86.66712
44	8	0.7708451	0.6429106	-66.45776
45	3	0.9880974	0.9615625	98.85134
46	5	1.0000000	0.9478538	99.17939
47	1	0.4824335	0.8224709	-10.65728
48	6	0.9907425	0.8281591	93.22178
49	7	0.9866542	0.7967908	93.53178
50	4	1.0000000	0.9221451	97.50097
51	7	0.9224287	0.9049580	75.03276
52	9	1.0000000	0.2967019	100.00000
53	9	0.9003347	0.0943662	57.83476
54	9	1.0000000	0.0935604	100.00000

55	1	0.8545062	0.9146200	78.93407