This document will demonstrate the usage of the Ensemble Network Aggregation (ENA) package available on CRAN.

1 Test Data Setup

We'll first simulate a matrix of gene expression data. This matrix includes 50 samples profiling the expression of 20 genes.

```
> simul <- matrix(rnorm(50*20), ncol=50)
> colnames(simul) <- paste("s", 1:50, sep="")
> rownames(simul) <- paste("g", 1:20, sep="")
```

We'll then inject into this noisy matrix a few obvious correlations; we expect that any network reconstruction technique would be able to identify most of these obvious connections.

```
> #generate a few obvious "trends" which should establish correlations on a few genes.
> simul[5,] <- seq(from=-1, to=1, length.out = ncol(simul))
> simul[10,] <- seq(from=1, to=-1, length.out = ncol(simul))
> simul[15,] <- seq(from=-1, to=1, length.out = ncol(simul))
> simul[20,] <- seq(from=1, to=-1, length.out = ncol(simul))
```

So genes 5, 10, 15, and 20 are all very strongly correlated (either postively or negatively).

2 Bootstrapping

Bootstrapping is a technique in which we randomly select a subset of the available samples in the dataset and the reconstruct the network. We do this multiple times, then merge using Inverse Rank Product (IRP) in order to improve the robustness of the network. The ENA package provides the `bootstrap()` function for this purpose.

We've found that bootstrapping is particularly effective for certain types of network reconstruction techniques —SPACE being one of them. We'll use the `buildSpace()` function, also provided by this package. This command typically creates many lines of output, so we'll suppress them.

```
> library(ENA)
> invisible(capture.output(sp <- bootstrap(simul, "buildSpace")))
> head(sp)

  Source Dest Bootstrapped.buildSpace
1     g1   g2             0.001455563
2     g1   g3             0.001511106
3     g2   g3             0.001507414
4     g1   g4             0.001460790
```

```
5     g2   g4            0.001635446
6     g3   g4            0.001545787
```

The `bootstrap` function produces an adjacency list represnting the aggregation of all networks produced by bootstrapping the selected function ("buildSpace").

3 Method Merging

Now that we have a sample gene expression matrix, we can proceed to use the ENA package. Our first demonstration will involve reconstructing the network from multiple different reconstruction techniques and then merging the resultant networks together at the end.

```
> gn <- buildGenenet(simul)
```

```
Estimating optimal shrinkage intensity lambda (correlation matrix): 0.3968
```

```
> wg <- buildWgcna(simul)
> ar <- buildAracne(simul)
```

These methods return the reconstructed networks in adjacency matrix format. The merging method, however, expects the input in adjacency list format. Additionally, some methods return directed (non-symmetric) matrices which we're not currently handling. So we want to make these matrices symmetric and also convert to adjacency-list format; we can do this simultaneously using the `symmetricize()` function. We'll also want to take the absolute value since we're not interested in the direction of each edge. Note that the `bootstrap` function already takes the absolute value and makes the resultant matrix symmetric before converting to adjacency list.

```
> gn <- symmetricize(abs(gn), "max", adjacencyList=TRUE)
> colnames(gn)[3] <- "GeneNet"
> wg <- symmetricize(abs(wg), "max", adjacencyList=TRUE)
> colnames(wg)[3] <- "WGCNA"
> ar <- symmetricize(abs(ar), "max", adjacencyList=TRUE)
> colnames(ar)[3] <- "Aracne"
```

We're now ready to merge these methods. Often, network reconstruction techniques won't rearrange the order of the genes in the matrix. If this is true, then the first two columns (used for addressing) of each result should be identical for all of the above reconstructed networks. However, you must be cautious, as it's possible that a reconstruction technique could rearrange the order of the genes before returning, in which case the addressing of your adjacency list would be non-identical. To be careful, we'll `merge` the results, rather than merely `cbind`ing the columns together – if you're confident that your addressing columns are all identical, it will be much faster to use `cbind`.

```
> add <- getTableAddressing(rownames(simul))
> #if all the addressing columns are identical, we can use the faster method.
> if (identical(add[,1:2], gn[,1:2]) && identical(add[,1:2], wg[,1:2]) && identical(gn[,1:2]
+         agg <- cbind(add, gn[,3], wg[,3], sp[,3], ar[,3])
+ } else{
+         #build up using merge in case the order of the addressing is incorrect.
+         agg <- merge(add, gn)
+         agg <- merge(agg, wg)
+         agg <- merge(agg, sp)
+         agg <- merge(agg, ar)
+ }
> head(agg)

  Source Dest     gn[, 3]      wg[, 3]     sp[, 3]    ar[, 3]
1     g1   g2 0.021536134 1.839052e-06 0.001455563 0.00000000
2     g1   g3 0.006056372 1.207721e-08 0.001511106 0.00000000
3     g2   g3 0.015786695 1.036934e-07 0.001507414 0.07383545
4     g1   g4 0.039994213 1.331923e-10 0.001460790 0.14639839
5     g2   g4 0.090875664 7.709753e-05 0.001635446 0.01454857
6     g3   g4 0.051416358 1.040683e-06 0.001545787 0.05596465

>
```

Now that we have a conslidated data.frame representing the adjacency list of all the networks produced by each different method, we can merge them using the `ena()` function. We'll then add it back into the aggregated data.frame for further inspection.

```
> ena <- ena(agg[,3:6])
> agg$ENA <- ena
```

This network is small enough for manual inspection. You can print out the results using the following statement. We'll find that all methods should have very good performance on this trivial network.

```
> cbind(agg[,1:2], round(agg[,3:7], 2))

    Source Dest gn[, 3] wg[, 3] sp[, 3] ar[, 3]  ENA
1       g1   g2    0.02       0    0.00    0.00 0.05
2       g1   g3    0.01       0    0.00    0.00 0.05
3       g2   g3    0.02       0    0.00    0.07 0.06
4       g1   g4    0.04       0    0.00    0.15 0.06
5       g2   g4    0.09       0    0.00    0.01 0.06
6       g3   g4    0.05       0    0.00    0.06 0.06
7       g1   g5    0.03       0    0.00    0.00 0.06
8       g2   g5    0.01       0    0.00    0.10 0.05
9       g3   g5    0.00       0    0.00    0.00 0.05
```

10	g4	g5	0.01	0	0.00	0.00	0.05
11	g1	g6	0.04	0	0.00	0.00	0.05
12	g2	g6	0.13	0	0.00	0.00	0.07
13	g3	g6	0.11	0	0.00	0.00	0.06
14	g4	g6	0.09	0	0.00	0.02	0.06
15	g5	g6	0.01	0	0.00	0.00	0.05
16	g1	g7	0.04	0	0.00	0.06	0.06
17	g2	g7	0.15	0	0.00	0.00	0.07
18	g3	g7	0.14	0	0.00	0.08	0.08
19	g4	g7	0.12	0	0.00	0.02	0.07
20	g5	g7	0.02	0	0.00	0.10	0.06
21	g6	g7	0.07	0	0.00	0.04	0.06
22	g1	g8	0.02	0	0.00	0.00	0.05
23	g2	g8	0.14	0	0.00	0.00	0.07
24	g3	g8	0.05	0	0.00	0.02	0.06
25	g4	g8	0.04	0	0.00	0.12	0.06
26	g5	g8	0.02	0	0.00	0.00	0.05
27	g6	g8	0.03	0	0.00	0.00	0.05
28	g7	g8	0.04	0	0.00	0.00	0.05
29	g1	g9	0.05	0	0.00	0.16	0.06
30	g2	g9	0.04	0	0.00	0.00	0.05
31	g3	g9	0.07	0	0.00	0.00	0.06
32	g4	g9	0.04	0	0.00	0.00	0.05
33	g5	g9	0.00	0	0.00	0.16	0.06
34	g6	g9	0.02	0	0.00	0.04	0.05
35	g7	g9	0.00	0	0.00	0.00	0.05
36	g8	g9	0.03	0	0.00	0.00	0.05
37	g1	g10	0.03	0	0.00	0.00	0.05
38	g2	g10	0.01	0	0.00	0.10	0.05
39	g3	g10	0.00	0	0.00	0.00	0.05
40	g4	g10	0.01	0	0.00	0.00	0.05
41	g5	g10	0.26	1	0.01	2.44	0.18
42	g6	g10	0.01	0	0.00	0.00	0.05
43	g7	g10	0.02	0	0.00	0.13	0.06
44	g8	g10	0.02	0	0.00	0.06	0.05
45	g9	g10	0.00	0	0.00	0.18	0.06
46	g1	g11	0.06	0	0.00	0.00	0.06
47	g2	g11	0.09	0	0.00	0.00	0.06
48	g3	g11	0.06	0	0.00	0.00	0.06
49	g4	g11	0.08	0	0.00	0.00	0.06
50	g5	g11	0.01	0	0.00	0.00	0.05
51	g6	g11	0.01	0	0.00	0.03	0.05
52	g7	g11	0.06	0	0.00	0.00	0.06
53	g8	g11	0.15	0	0.00	0.00	0.07
54	g9	g11	0.07	0	0.00	0.08	0.06
55	g10	g11	0.01	0	0.00	0.00	0.05

56	g1	g12	0.17	0	0.00	0.00	0.08
57	g2	g12	0.09	0	0.00	0.23	0.08
58	g3	g12	0.04	0	0.00	0.00	0.05
59	g4	g12	0.01	0	0.00	0.00	0.05
60	g5	g12	0.01	0	0.00	0.00	0.05
61	g6	g12	0.02	0	0.00	0.00	0.05
62	g7	g12	0.16	0	0.00	0.02	0.08
63	g8	g12	0.06	0	0.00	0.14	0.06
64	g9	g12	0.05	0	0.00	0.04	0.06
65	g10	g12	0.01	0	0.00	0.00	0.05
66	g11	g12	0.05	0	0.00	0.00	0.06
67	g1	g13	0.16	0	0.00	0.00	0.07
68	g2	g13	0.01	0	0.00	0.07	0.05
69	g3	g13	0.04	0	0.00	0.06	0.05
70	g4	g13	0.13	0	0.00	0.00	0.07
71	g5	g13	0.02	0	0.00	0.05	0.06
72	g6	g13	0.06	0	0.00	0.08	0.06
73	g7	g13	0.11	0	0.00	0.10	0.07
74	g8	g13	0.21	0	0.00	0.08	0.10
75	g9	g13	0.03	0	0.00	0.00	0.05
76	g10	g13	0.02	0	0.00	0.00	0.05
77	g11	g13	0.13	0	0.00	0.00	0.07
78	g12	g13	0.05	0	0.00	0.04	0.06
79	g1	g14	0.04	0	0.00	0.10	0.06
80	g2	g14	0.02	0	0.00	0.00	0.05
81	g3	g14	0.12	0	0.00	0.03	0.07
82	g4	g14	0.02	0	0.00	0.00	0.05
83	g5	g14	0.01	0	0.00	0.00	0.05
84	g6	g14	0.05	0	0.00	0.04	0.06
85	g7	g14	0.07	0	0.00	0.08	0.06
86	g8	g14	0.02	0	0.00	0.18	0.06
87	g9	g14	0.05	0	0.00	0.00	0.05
88	g10	g14	0.01	0	0.00	0.00	0.05
89	g11	g14	0.04	0	0.00	0.00	0.05
90	g12	g14	0.11	0	0.00	0.00	0.06
91	g13	g14	0.13	0	0.00	0.00	0.07
92	g1	g15	0.03	0	0.00	0.00	0.05
93	g2	g15	0.01	0	0.00	0.08	0.05
94	g3	g15	0.00	0	0.00	0.00	0.05
95	g4	g15	0.01	0	0.00	0.00	0.05
96	g5	g15	0.26	1	0.00	2.41	0.12
97	g6	g15	0.01	0	0.00	0.00	0.05
98	g7	g15	0.02	0	0.00	0.13	0.06
99	g8	g15	0.02	0	0.00	0.07	0.05
100	g9	g15	0.00	0	0.00	0.17	0.06
101	g10	g15	0.26	1	0.00	2.53	0.13

102	g11	g15	0.01	0	0.00	0.00	0.05
103	g12	g15	0.01	0	0.00	0.00	0.05
104	g13	g15	0.02	0	0.00	0.00	0.05
105	g14	g15	0.01	0	0.00	0.00	0.05
106	g1	g16	0.08	0	0.00	0.00	0.06
107	g2	g16	0.05	0	0.00	0.03	0.06
108	g3	g16	0.08	0	0.00	0.00	0.06
109	g4	g16	0.14	0	0.00	0.00	0.07
110	g5	g16	0.04	0	0.00	0.00	0.06
111	g6	g16	0.01	0	0.00	0.18	0.06
112	g7	g16	0.01	0	0.00	0.00	0.05
113	g8	g16	0.05	0	0.00	0.04	0.06
114	g9	g16	0.02	0	0.00	0.00	0.05
115	g10	g16	0.04	0	0.00	0.00	0.05
116	g11	g16	0.15	0	0.00	0.00	0.08
117	g12	g16	0.06	0	0.00	0.04	0.06
118	g13	g16	0.07	0	0.00	0.00	0.06
119	g14	g16	0.05	0	0.00	0.03	0.06
120	g15	g16	0.04	0	0.00	0.00	0.06
121	g1	g17	0.06	0	0.00	0.01	0.06
122	g2	g17	0.09	0	0.00	0.11	0.07
123	g3	g17	0.16	0	0.00	0.00	0.08
124	g4	g17	0.09	0	0.00	0.00	0.06
125	g5	g17	0.02	0	0.00	0.00	0.05
126	g6	g17	0.07	0	0.00	0.00	0.06
127	g7	g17	0.00	0	0.00	0.09	0.05
128	g8	g17	0.04	0	0.00	0.05	0.06
129	g9	g17	0.10	0	0.00	0.06	0.07
130	g10	g17	0.02	0	0.00	0.06	0.05
131	g11	g17	0.04	0	0.00	0.00	0.05
132	g12	g17	0.21	0	0.00	0.06	0.10
133	g13	g17	0.06	0	0.00	0.10	0.06
134	g14	g17	0.09	0	0.00	0.00	0.06
135	g15	g17	0.02	0	0.00	0.06	0.06
136	g16	g17	0.11	0	0.00	0.07	0.07
137	g1	g18	0.10	0	0.00	0.00	0.06
138	g2	g18	0.08	0	0.00	0.00	0.06
139	g3	g18	0.14	0	0.00	0.00	0.07
140	g4	g18	0.14	0	0.00	0.14	0.08
141	g5	g18	0.01	0	0.00	0.00	0.05
142	g6	g18	0.07	0	0.00	0.00	0.06
143	g7	g18	0.01	0	0.00	0.00	0.05
144	g8	g18	0.03	0	0.00	0.00	0.05
145	g9	g18	0.11	0	0.00	0.00	0.06
146	g10	g18	0.01	0	0.00	0.00	0.05
147	g11	g18	0.06	0	0.00	0.16	0.06

148	g12	g18	0.01	0	0.00	0.03	0.05
149	g13	g18	0.06	0	0.00	0.15	0.06
150	g14	g18	0.09	0	0.00	0.00	0.06
151	g15	g18	0.01	0	0.00	0.00	0.05
152	g16	g18	0.14	0	0.00	0.17	0.08
153	g17	g18	0.01	0	0.00	0.00	0.05
154	g1	g19	0.04	0	0.00	0.00	0.06
155	g2	g19	0.11	0	0.00	0.00	0.06
156	g3	g19	0.11	0	0.00	0.00	0.07
157	g4	g19	0.02	0	0.00	0.00	0.05
158	g5	g19	0.05	0	0.00	0.17	0.07
159	g6	g19	0.03	0	0.00	0.00	0.05
160	g7	g19	0.10	0	0.00	0.00	0.06
161	g8	g19	0.00	0	0.00	0.00	0.05
162	g9	g19	0.11	0	0.00	0.00	0.06
163	g10	g19	0.05	0	0.00	0.18	0.07
164	g11	g19	0.04	0	0.00	0.00	0.05
165	g12	g19	0.08	0	0.00	0.00	0.06
166	g13	g19	0.09	0	0.00	0.00	0.06
167	g14	g19	0.18	0	0.00	0.09	0.09
168	g15	g19	0.05	0	0.00	0.16	0.07
169	g16	g19	0.01	0	0.00	0.00	0.05
170	g17	g19	0.03	0	0.00	0.00	0.05
171	g18	g19	0.05	0	0.00	0.00	0.05
172	g1	g20	0.03	0	0.00	0.00	0.05
173	g2	g20	0.01	0	0.00	0.10	0.05
174	g3	g20	0.00	0	0.00	0.00	0.05
175	g4	g20	0.01	0	0.00	0.00	0.05
176	g5	g20	0.26	1	0.00	2.45	0.13
177	g6	g20	0.01	0	0.00	0.00	0.05
178	g7	g20	0.02	0	0.00	0.13	0.06
179	g8	g20	0.02	0	0.00	0.06	0.05
180	g9	g20	0.00	0	0.00	0.15	0.05
181	g10	g20	0.26	1	0.00	2.53	0.11
182	g11	g20	0.01	0	0.00	0.00	0.05
183	g12	g20	0.01	0	0.00	0.00	0.05
184	g13	g20	0.02	0	0.00	0.00	0.05
185	g14	g20	0.01	0	0.00	0.00	0.05
186	g15	g20	0.26	1	0.01	2.55	0.21
187	g16	g20	0.04	0	0.00	0.00	0.05
188	g17	g20	0.02	0	0.00	0.00	0.05
189	g18	g20	0.01	0	0.00	0.00	0.05
190	g19	g20	0.05	0	0.00	0.16	0.07