Network analysis of the Lymphome proteomics data set

Julien Chiquet

September 2019

Contents

Requirements

```
library(tidyverse)
library(corrplot)
library(GGally)
library(igraph)
library(ggfortify)
library(blockmodels)
library(QUIC)
library(stabs)
library(simone)
library(PLNmodels)
library(RColorBrewer)
pal <- brewer.pal(10, "Set3") # a fancy palette for graph
```

1 Basic proteomics data analysis

We first load both proteomics and covariates

```
proteomics <- readRDS("proteins.rds")
covariates <- readRDS("covariates.rds")
```

Let us have a quick look on both data frames:

1.1 Covariates

First a quick look,

```
covariates %>% head() %>% knitr::kable()
```

	disease	type	sex	age	bmi
NS001BC	BC	CO	F	60.36413	25.46938
NS001LY	LY	CO	M	40.19165	25.85463
NS002BC	BC	CA	F	60.87885	30.85938
NS002LY	LY	CA	M	39.95072	23.51556
NS003BC	BC	CO	F	50.08350	24.34175
NS003LY	LY	CO	M	40.36413	26.27840

Only women are subject to this disease, so let us remove men:

```
proteomics <- proteomics[covariates$sex == "F", ]
covariates <- covariates %>% filter(sex == "F") %>% dplyr::select(-sex)
```

Then do a pairs plot:

```
GGally::ggpairs(covariates, columns = 2:4, aes(colour = disease))
```

```
## `stat_bin()` using `bins = 30`. Pick better value with `binwidth`.
## `stat_bin()` using `bins = 30`. Pick better value with `binwidth`.
```

1.2 Proteomics

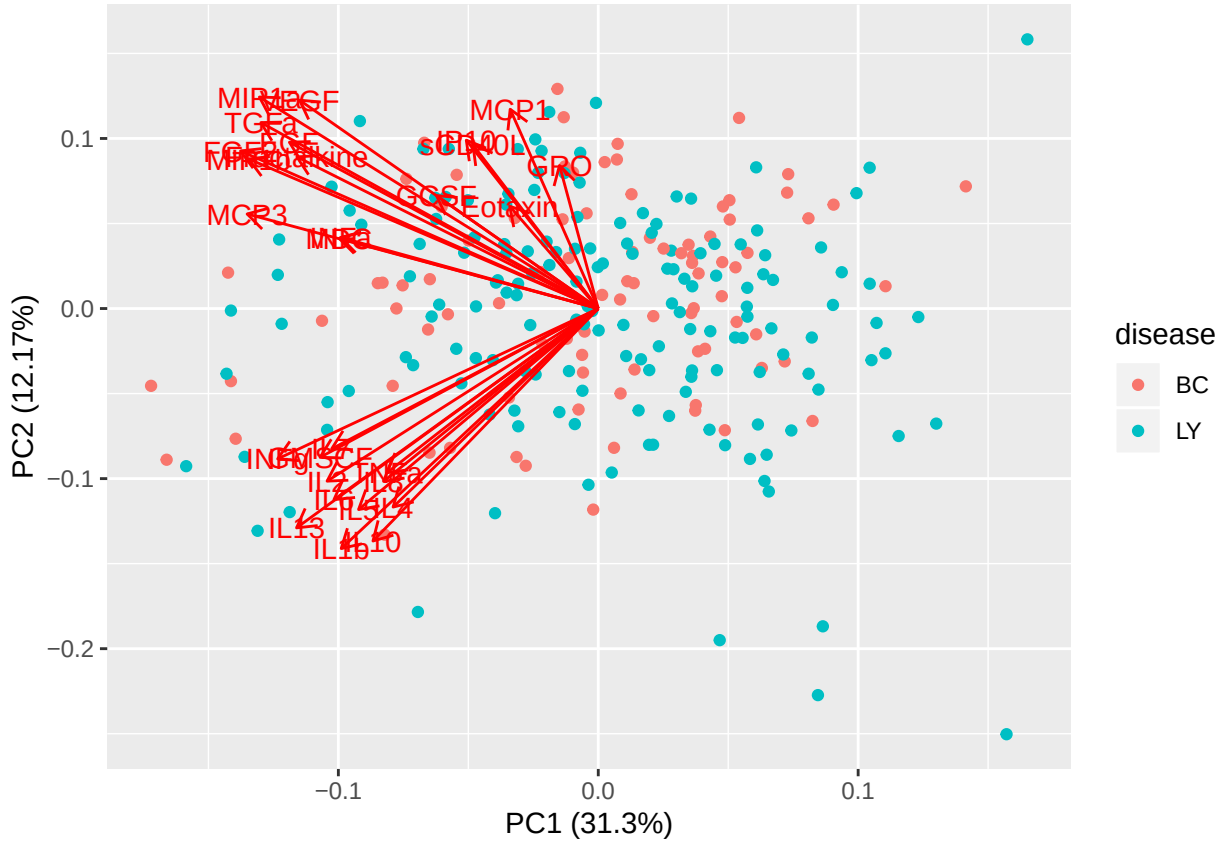
Again first a quick look:

```
proteomics %>% head() %>% knitr::kable()
```

	EGF	Eotaxin	FGF2	Fractalkine	GCSF	GRO	INFa	IP10	MC
NS001BC	0.0055664	-0.4618951	-1.409762	-1.7219935	-1.0901432	0.3206727	-4.062710	-0.3409295	-0.6031
NS002BC	1.3193453	0.3758365	-1.840780	2.0508717	-1.6820340	0.4068460	-4.607592	0.0947143	-0.0293
NS003BC	1.3047518	0.5055310	1.200365	-0.4681025	-0.8453924	0.0313707	-4.073082	-0.1651572	0.2724
NS004BC	-3.0146088	0.0229324	-2.166404	-2.8086296	-1.0874199	0.3461147	-3.380521	-0.1324629	-0.4450
NS006BC	2.7218530	0.4267605	2.818803	2.3179051	2.5723636	-0.4683967	7.315315	0.0987970	0.6490
NS007BC	0.6952752	0.0001863	1.703755	1.6672615	0.9866069	-0.1940195	5.787370	-0.2015096	-0.7642

Simple PCA does not seem to show a strong structuration of the data due to the disease status:

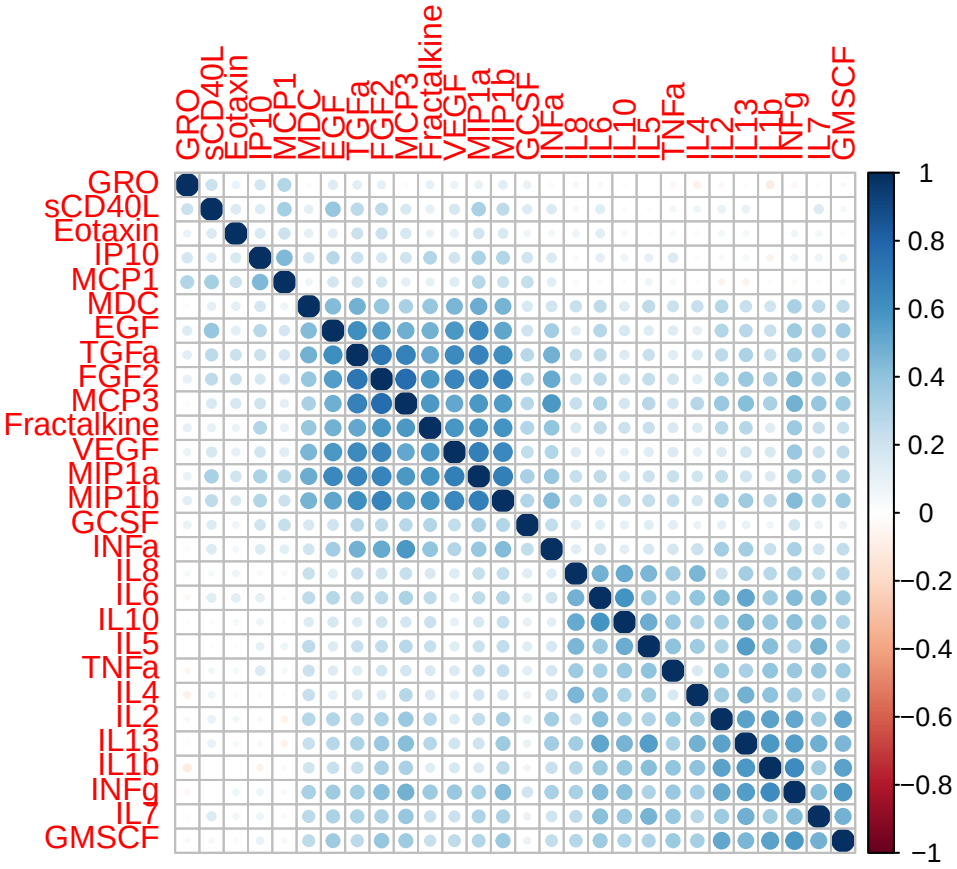
```
pca <- prcomp(proteomics, scale. = TRUE)
for_plot <- as_tibble(proteomics) %>%
  add_column(disease = covariates$disease,
             type    = covariates$type)
autoplot(pca, loadings = TRUE, loadings.label = TRUE,
         data = for_plot, colour = 'disease')
```




```
# autoplot(pca, loadings = TRUE, loadings.label = TRUE,
#          data = for_plot, colour = 'type')
```

Still, there is some structure between the proteines:

```
corrplot(cor(proteomics), order = "hclust")
```



2 Tutorial: proteomics network reconstruction on pooled data

2.1 Neighborhood selection with *simone*

We first retrieve the whole path of inferred networks with *simone*.

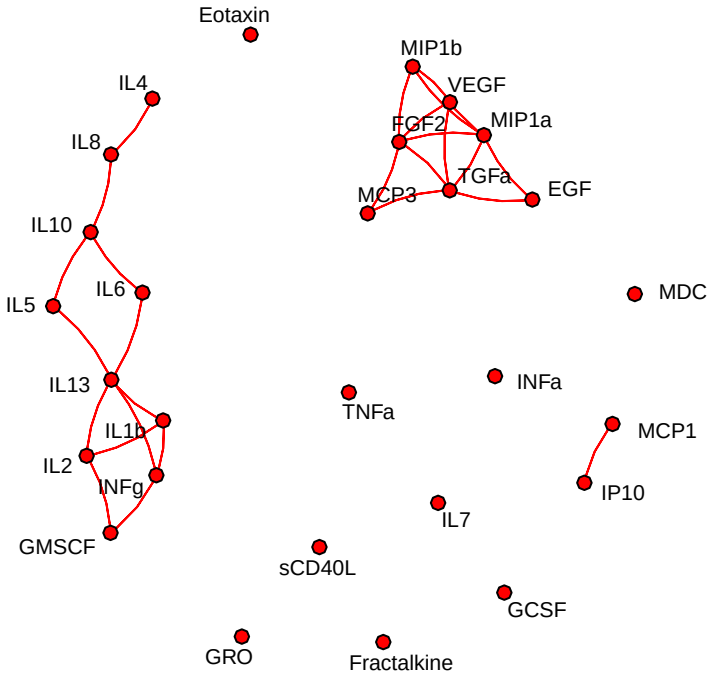
```
pooled_path <- simone(proteomics)
```

And extract one of them:

```
myNetwork <- getNetwork(pooled_path)
```

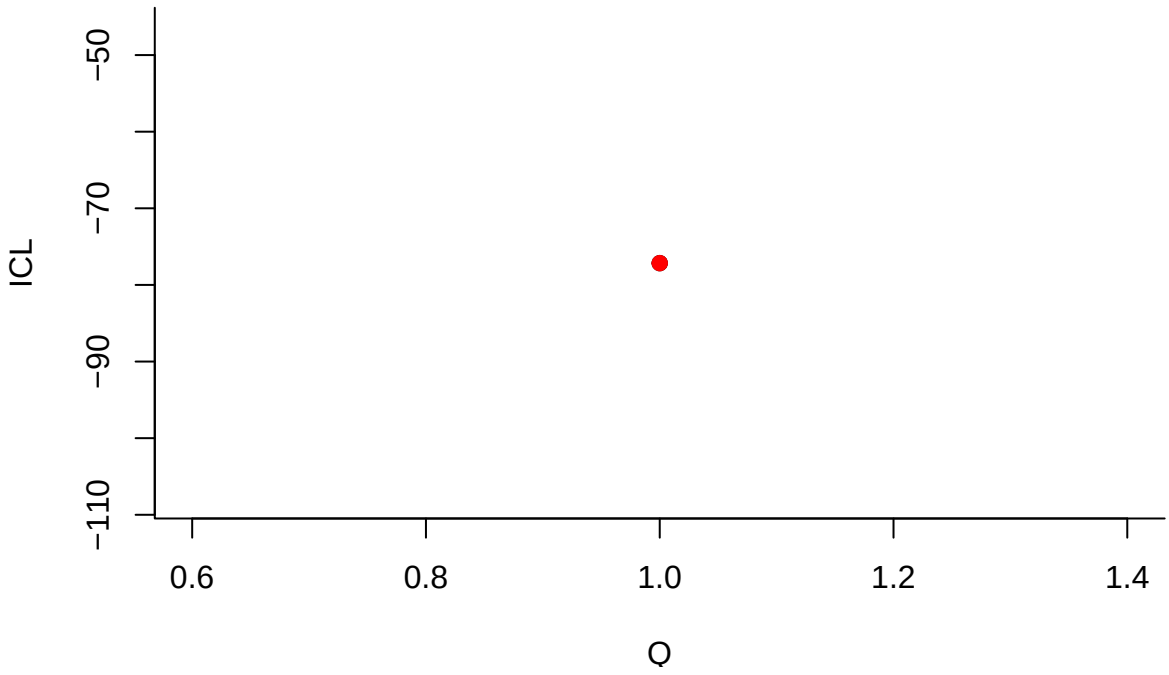
```
##
## Found a network with 27 edges.
```

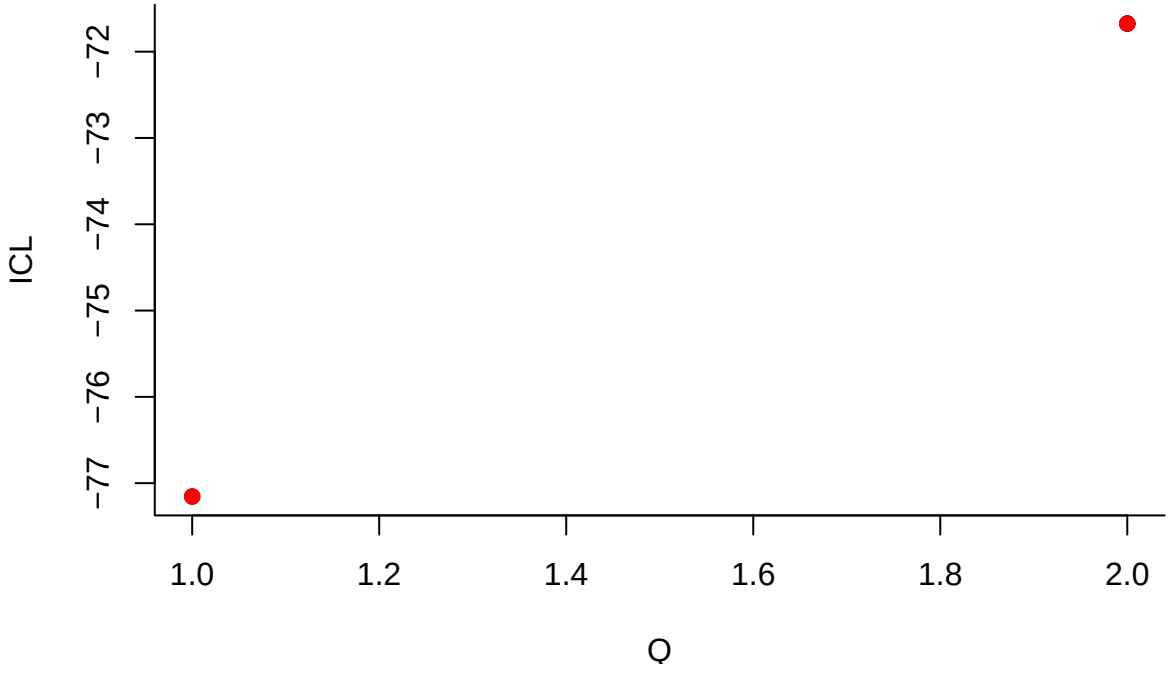
```
plot(myNetwork)
```

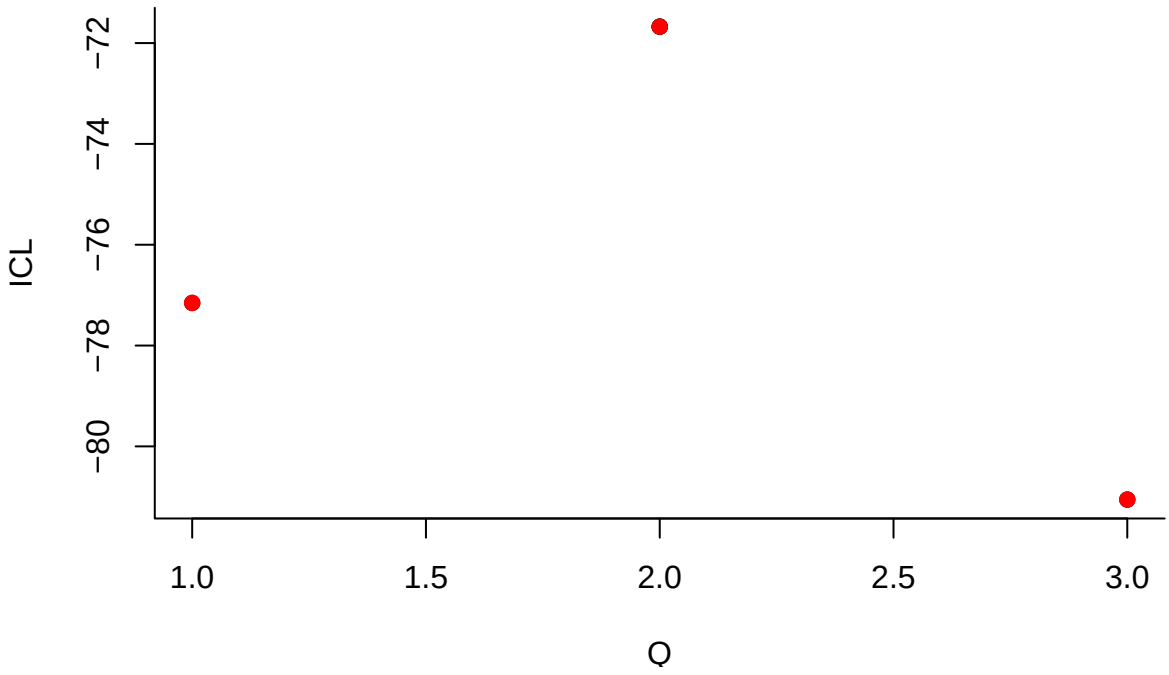


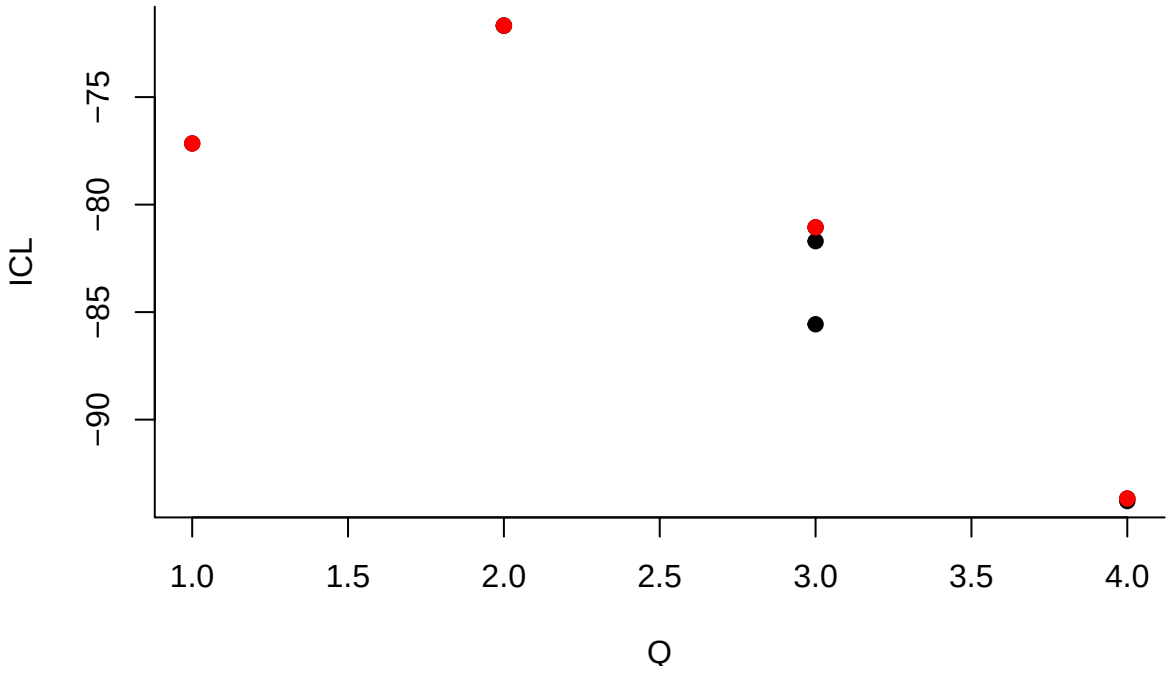
There seems to be two clusters: let us manually adjus a stochastic block model on this network

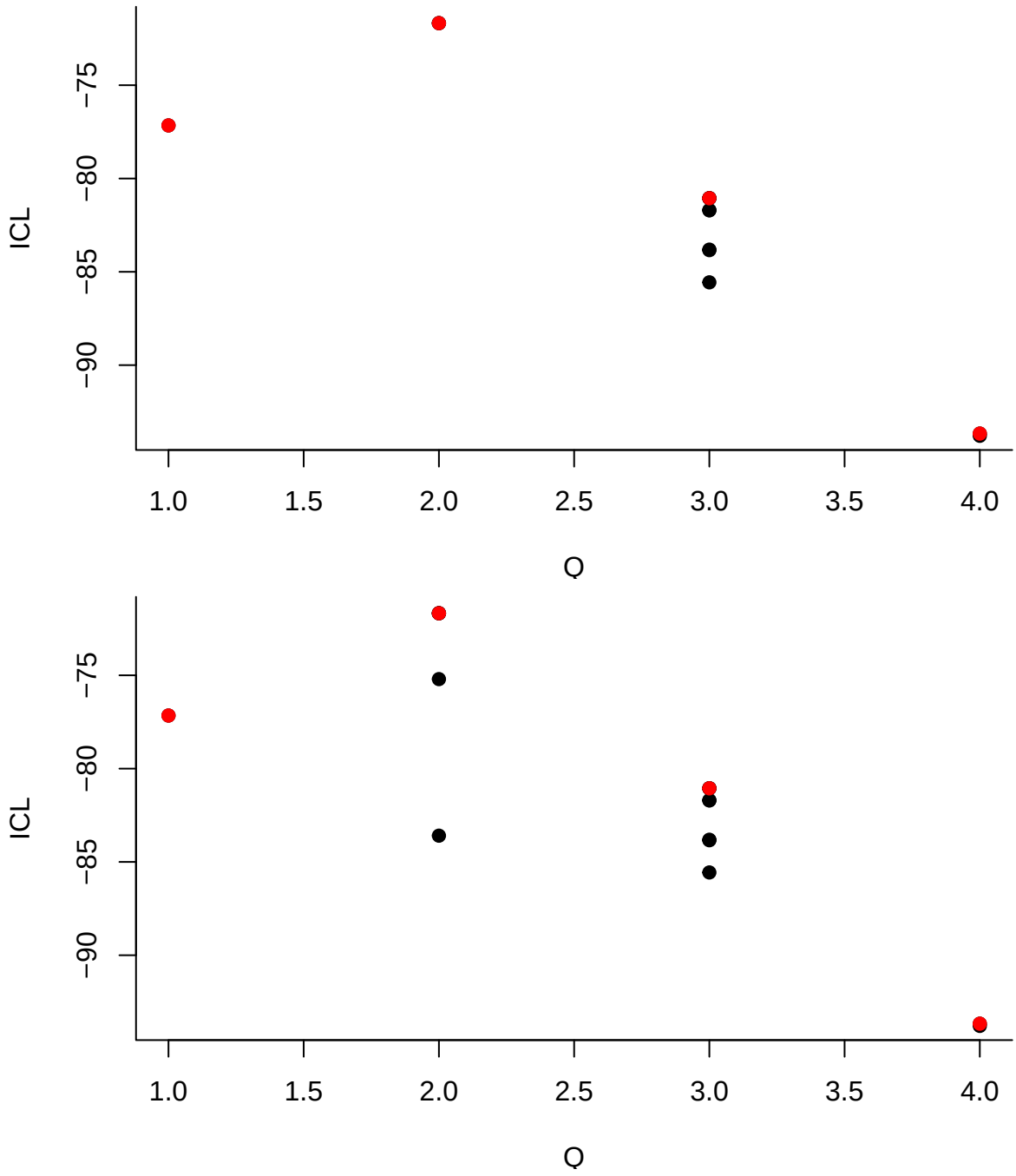
```
subNetwork <- myNetwork$A
to_remove <- rowSums(subNetwork) == 0
subNetwork <- subNetwork[!to_remove, !to_remove]
mySBM_collection <-
  BM_bernoulli(
    "SBM_sym",
    subNetwork,
    verbosity = 0
  )
mySBM_collection$estimate()
```


ICL
−50
−70
−90
−110
0.6
0.8
1.0
1.2
1.4
Q


ICL
−72
−73
−74
−75
−76
−77
1.0
1.2
1.4
1.6
1.8
2.0
Q


ICL
−72
−74
−76
−78
−80
1.0
1.5
2.0
2.5
3.0
Q


ICL
−75
−80
−85
−90
1.0
1.5
2.0
2.5
3.0
3.5
4.0
Q



The ICL criterion choses 3 clusters.

```
clusters <- apply(mySBM_collection$memberships[[2]]$Z, 1, which.max)
```

The following piece of code returns a fancy version of this graph thanks to the *igraph* package ready for plotting:

```
fancy_igraph <- function(graph, clusters, factor = 5) {
  V(graph)$class <- clusters
  V(graph)$size <- degree(graph) * factor
  V(graph)$frame.color <- "white"
  V(graph)$color <- pal[V(graph)$class]
```

```
  E(graph)$arrow.mode <- 0
  graph
}
```

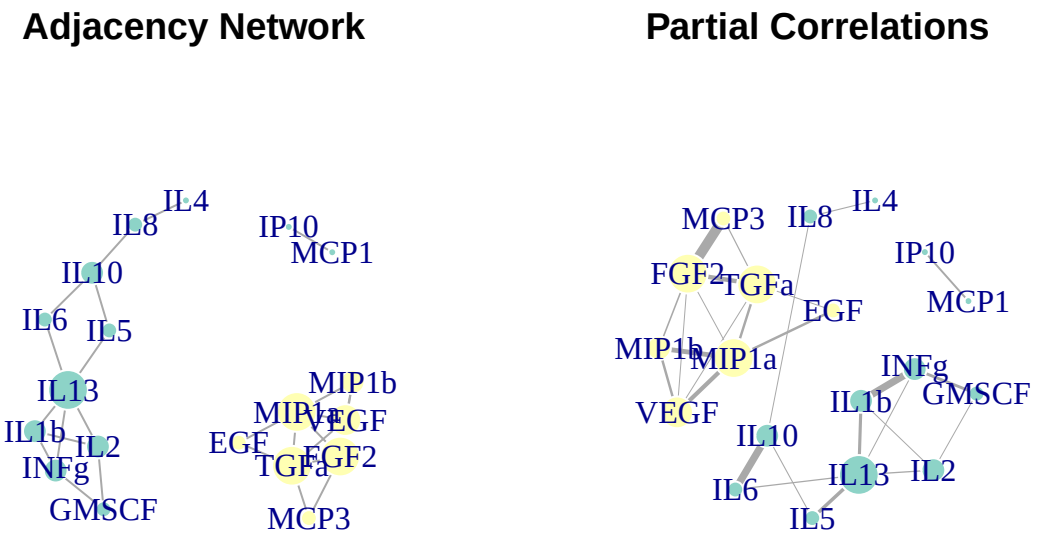
Now, let us represent the graph from the adjacency matrix and a weighted version with partial correlation values :

```
g1 <- graph_from_adjacency_matrix(subNetwork, mode = "undirected", diag = FALSE)

partialCor <- -myNetwork$Theta
partialCor <- partialCor[!to_remove, !to_remove]

g2 <- graph_from_adjacency_matrix(partialCor, weighted = TRUE, mode = "undirected", diag = FALSE)

par(mfrow = c(1,2))
plot(fancy_igraph(g1, clusters), main = "Adjacency Network")
plot(fancy_igraph(g2, clusters), edge.width = 20*E(g2)$weight, main = "Partial Correlations")
```

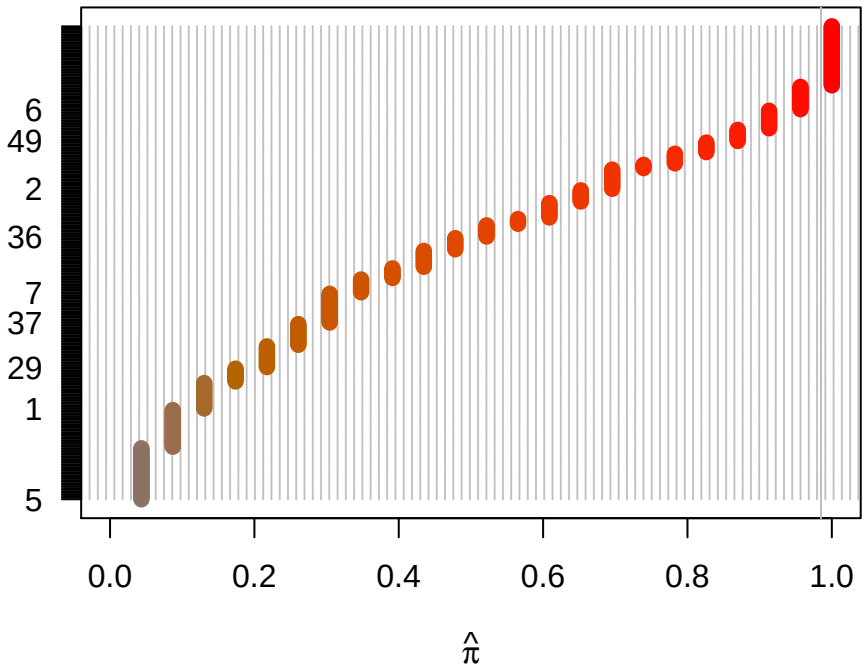


2.2 Graphical-Lasso with Stability Selection

The graphical-Lasso is efficiently implemented by the QUIC algorithm. An alternative to BIC to choose the most stables edges in the network is the stability selection approach. It subsamples the original data set, run the algorithm on each subsample and estimate the probability of selection of each edge along the path.

```
Glasso_stability <- stabsel(x = proteomics, fitfun = "quic.graphical_model", cutoff = 0.985, PFER = 5)
```

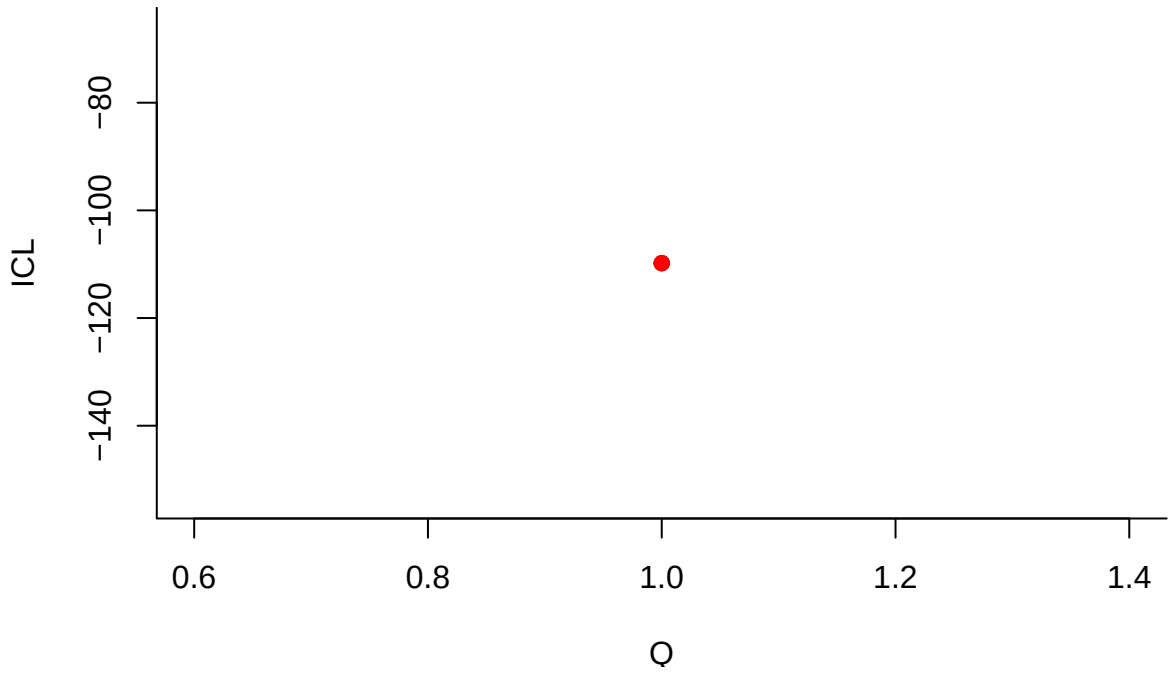
```
plot(Glasso_stability, type = "maxsel", labels = 1:50, main = "selection")
```

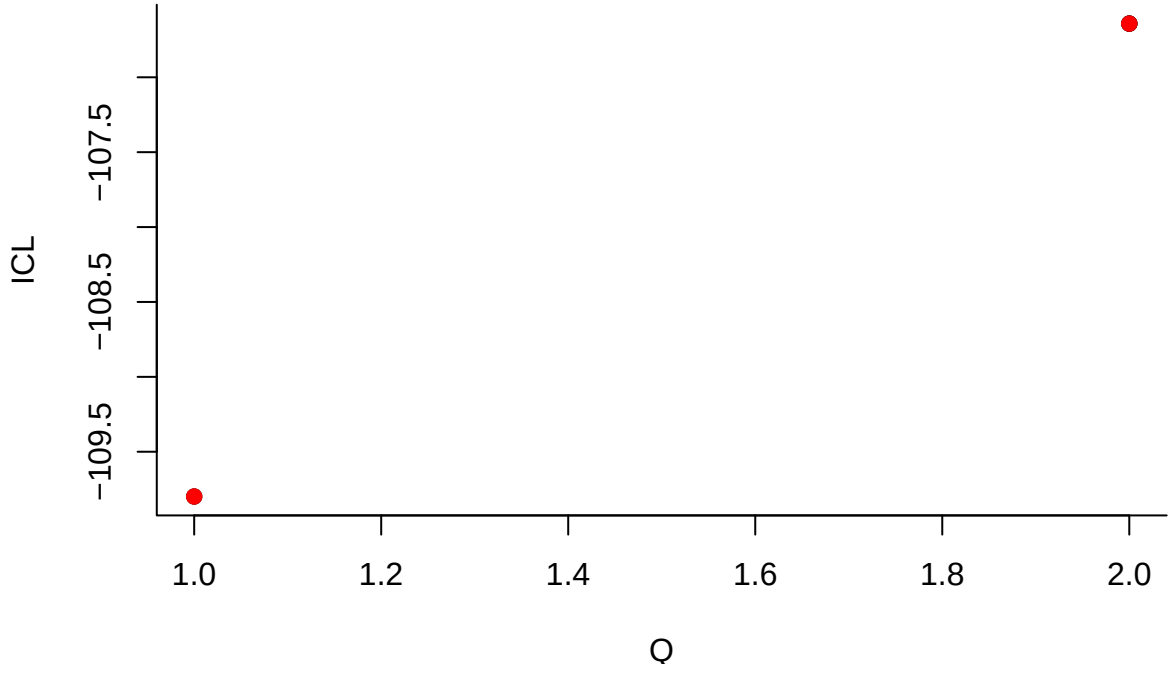


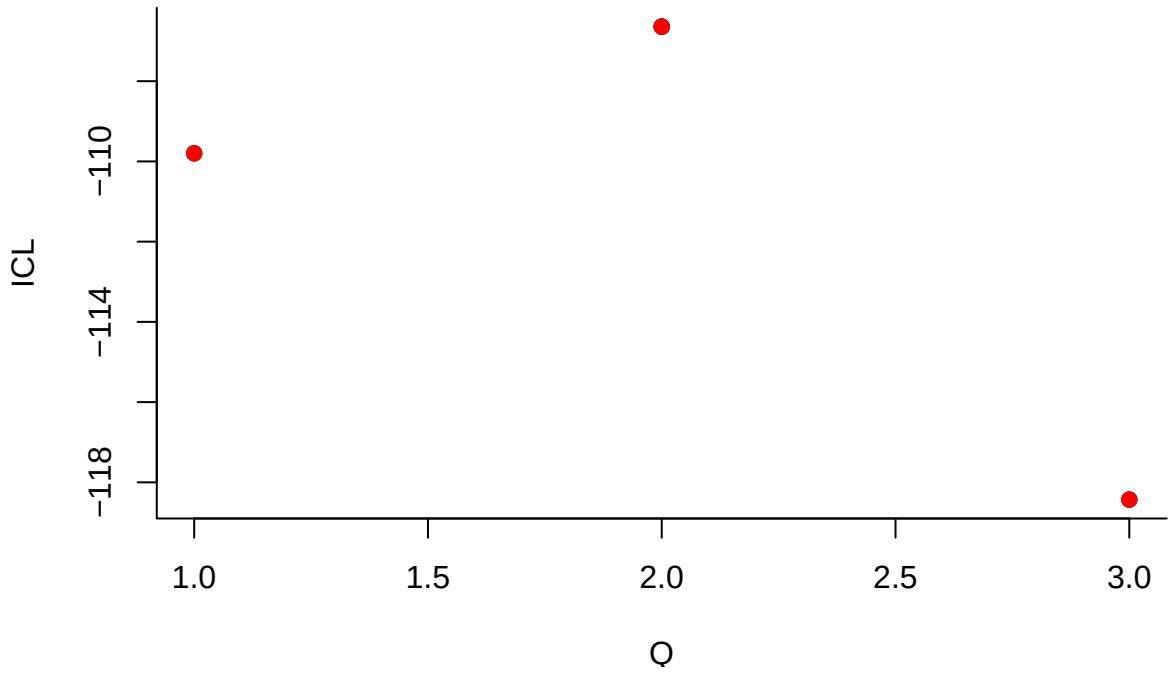

```
g_stabs <-
  graph_from_edgelist(
    do.call(rbind, strsplit(names(Glasso_stability$selected), " : ")),
    directed = FALSE
  )
```

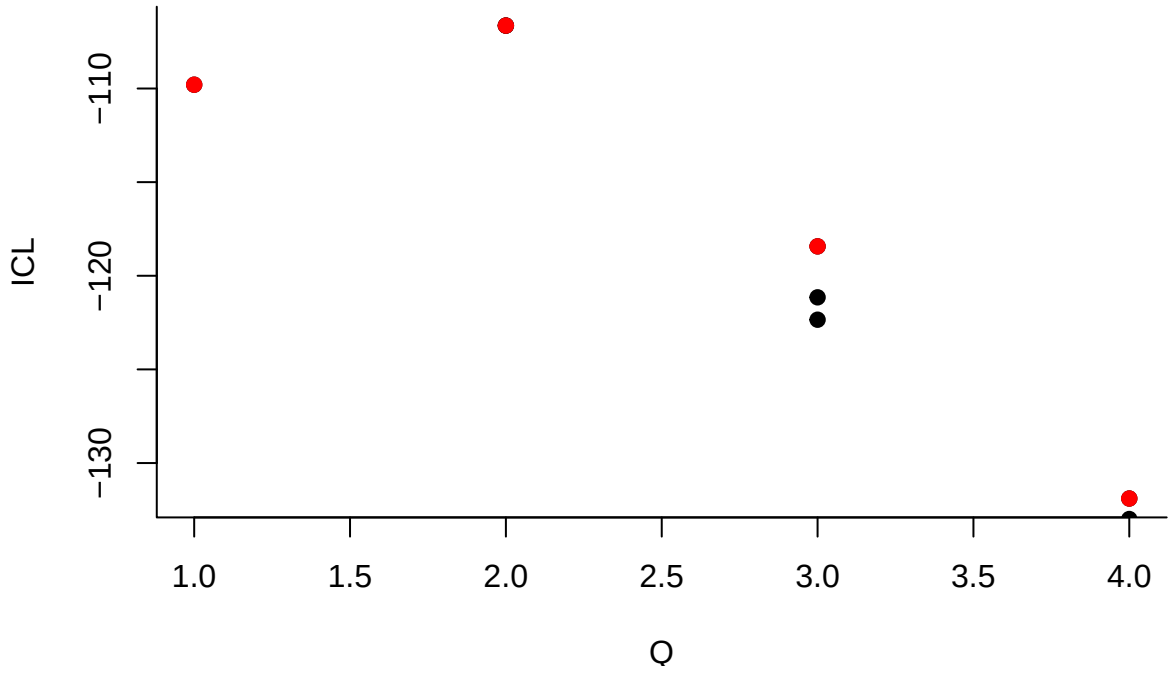
Now try to find some structure thanks to blockmodels:

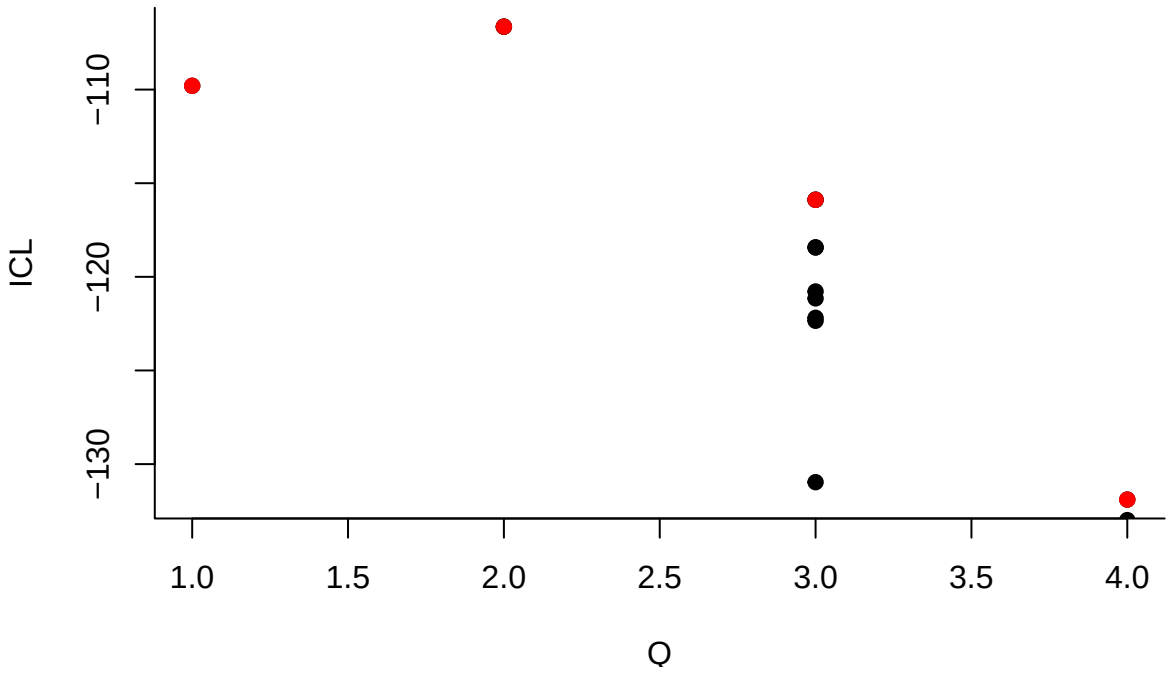
```
mySBM_collection <-
  BM_bernoulli(
    "SBM_sym",
    as_adjacency_matrix(g_stabs, sparse = FALSE),
    verbosity = 0
  )
mySBM_collection$estimate()
```

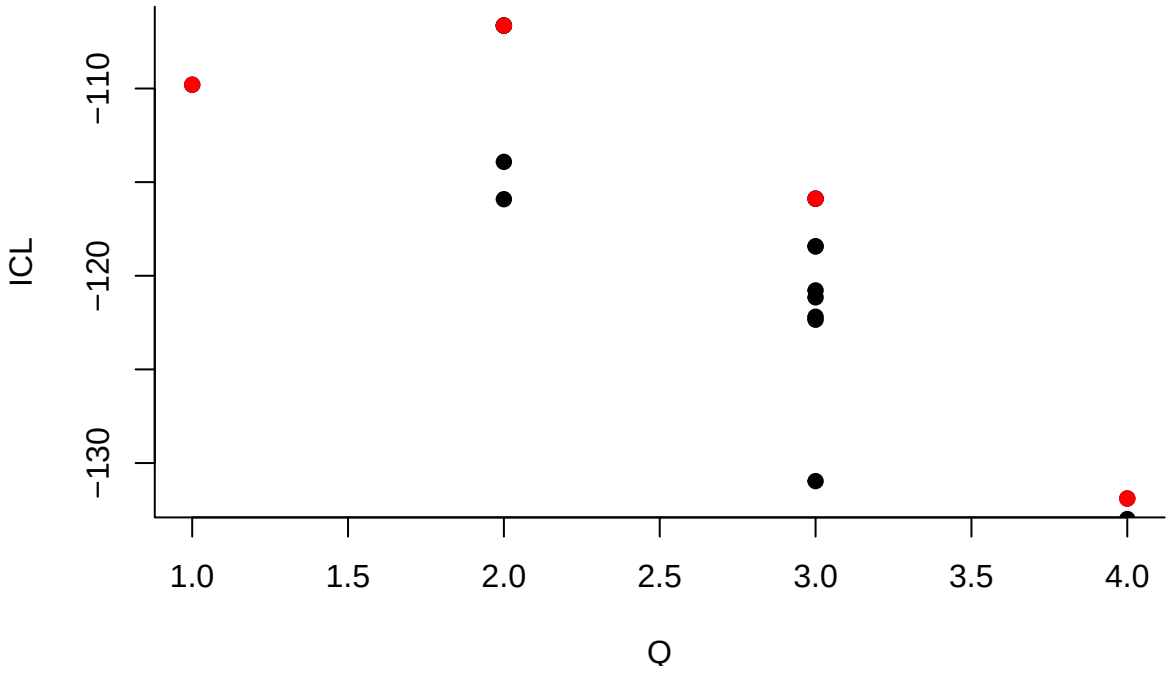

ICL
−80
−100
−120
−140
0.6
0.8
1.0
1.2
1.4
Q

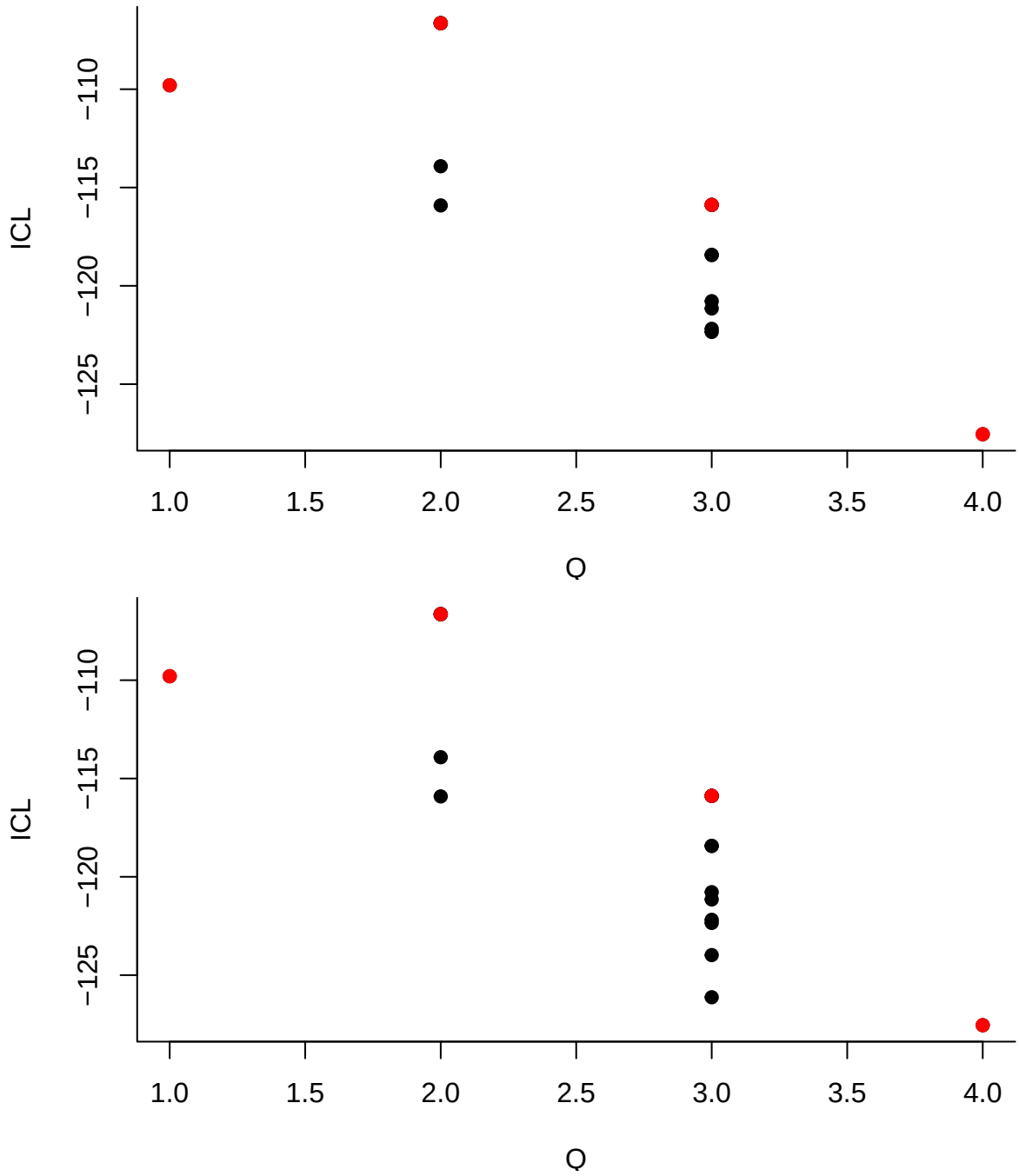

ICL
−107.5
−108.5
−109.5
1.0
1.2
1.4
1.6
1.8
2.0
Q


ICL
−110
−114
−118
1.0
1.5
2.0
2.5
3.0
Q


ICL
−110
−120
−130
1.0
1.5
2.0
2.5
3.0
3.5
4.0
Q


ICL
−110
−120
−130
1.0
1.5
2.0
2.5
3.0
3.5
4.0
Q

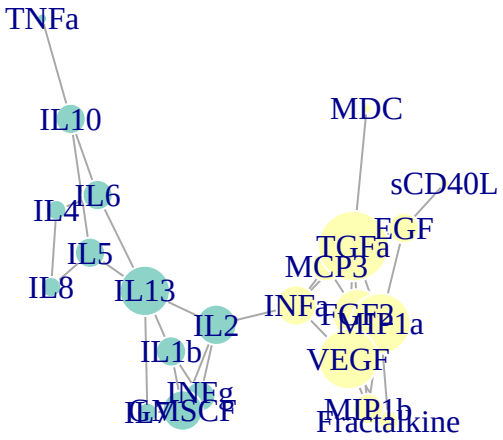

ICL
−110
−120
−130
1.0
1.5
2.0
2.5
3.0
3.5
4.0
Q




```
clusters <- apply(mySBM_collection$memberships[[2]]$Z, 1, which.max)
```

And plot it:

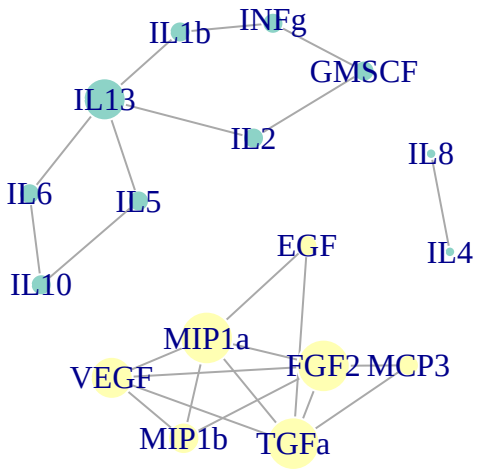
```
plot(fancy_igraph(g_stabs, clusters))
```



What is the intersection with plot inferred with neighborhood selection and *simone*?

```
consensus_graph <- igraph::intersection(g_stabs, g2)
mySBM_collection <-
  BM_bernoulli(
    "SBM_sym",
    as_adjacency_matrix(consensus_graph, sparse = FALSE),
    verbosity = 0, plotting = ""
  )
mySBM_collection$estimate()
```

```
clusters <- apply(mySBM_collection$memberships[[2]]$Z, 1, which.max)
plot(
  delete.vertices(
    fancy_igraph(consensus_graph, clusters),
    degree(consensus_graph) == 0
    )
  )
```



3 Practical: network inference accounting for patient status

3.1 Consensus and differential network from independent network inference

- Infer one network per patient status, with a method of your choice (e.g. *stabsel* with 'quic' fit function, or neighborhood selection with *huge* and STaRS for model selection)
- Compare these two networks according to
  - some descriptive statistics learnt during today's course
  - the clusterings obtained with *blockmodels*
- Build a "differential" and a "consensus" network from the two network associated with each status

3.2 Robustness of the networks

Let us try a "home-made" stability selection for evaluating the robustness of the differential or consensus network: For each patient status, - create K subsamples with, says 90% of the original data - repeat the analysis described in 2.1 - evaluate the stability of the differential and consensus networks

3.3 Consensus network with joint inference

- Use the *simone* multitask framework to infer the networks jointly
- Try to find some edges specific to each patient status
- Compare your results with the first approach (that consider network inference independently)
- If you have time (and multiple cores on your computer...), evaluate the robustness of the consensus network with resampling

3.4 Optional

3.4.1 Weighted penalties

You may put some prior in the network inference model by weighting the penalty associated to each edge:

To this end, build a matrix of weights which depends on the connection probabilities inferred by the SBM ($\propto (1-\hat{\pi}_{ij})$) to refine the inferred networks (parameter *rho* in function 'quic.graphical_model'). Connection probabilities can be extracted from a collection of *blockmodels* objects via `SBM_collection$model_parameters[[1]]$pi`.

3.4.2 Multiattribute network from proteomics and transcriptomics

You may find a consensus network by coupling transcriptomics and proteomics data: first determine the transcripts associated with the set of 20 proteins. Then use the package found here:

```
devtools::install_github("jchiquet/multivarNetwork")
```