Solutions for chapter Using Graphs for Interactome Data

Exercise 1

a RobjectlitG is an instance of the *graphNEL* class. You can use the manual page to find out more; type `class?graphNEL`.

b
```
> nodes(litG)[1:5]

[1] "YBL072C" "YBL083C" "YBR009C" "YBR010W" "YBR031W"
```

c The `ccyclered` data is stored in a *data.frame*. Please also have a look at the manual page of this class.

```
> class(ccyclered)
> str(ccyclered)
> dim(ccyclered)
> names(ccyclered)
```

Exercise 2

a Each element of `cc` is a character vector of node names defining one of the connected components of the graph.

```
> cc[[7]]

[1] "YBR118W" "YAL003W" "YLR249W"
```

b There are 2642 connected components. The largest component consists of 88 nodes. There are 2587 singletons.

Exercise 3

Again, we first create the layouts using the function `layoutGraph`. These can then be plotted using `renderGraph`, as above.

```
> lay12neato = layoutGraph(sg1, layoutType="dot")
> renderGraph(lay12neato,
      graph.pars=list(nodes=list(fillcolor="steelblue2")))
> lay12twopi = layoutGraph(sg2, layoutType="twopi")
> renderGraph(lay12twopi,
      graph.pars=list(nodes=list(fillcolor="steelblue2")))
```

Exercise 4

The `sps` object is a list. The manual page for `sp.between` describes its structure.

To plot individual nodes and edges with different colors we have to use the `nodeRenderInfo` and `edgeRenderInfo` functions:

```
> fill = rep("steelblue2", length(pth))
> names(fill) = pth
> nodeRenderInfo(lsg1) = list(fill=fill)
> edges = paste(pth[-length(pth)], pth[-1], sep="~")
> lwd = rep(5, length(edges))
> col = rep("steelblue2", length(edges))
> names(lwd) = names(col) = edges
> edgeRenderInfo(lsg1) = list(col=col, lwd=lwd)
```

```
> renderGraph(lsg1)
```

Exercise 5

`allp` is a matrix of the shortest path distances between all pairs of nodes. You can find the diameter by finding the maximum value in `allp`.

```
> max(allp)
[1] 13
```

The longest shortest path is not unique:

```
> sum(allp == max(allp))
[1] 36
```

Exercise 6

```
> clusts = with(ccyclered, split(Y.name, Cluster))
```

Exercise 7

```
> ccClust = connectedComp(cg)
> length(ccClust)
[1] 30
```

Exercise 8

The return value of the `intersection` method is a new *graph* object containing the common set of nodes and edges between the two input graphs. So the number of common edges between the graphs is simply the number of edges in the returned graph:

```
> numEdges(commonG)
[1] 42
```

Exercise 9

The `nodePerm` function takes two graphs, `g1` and `g2`, as inputs along with the number of permutation-based tests to perform, `B`. The function loops `B` times. For each iteration, the node labels of `g1` are permuted and the number of common edges between the permuted `g1` and `g2` is computed. The return value is a numeric vector with length equal to `B` such that each element gives the number of common edges for the corresponding permutation.

Exercise 10

```
> barplot(table(nPdist))
```

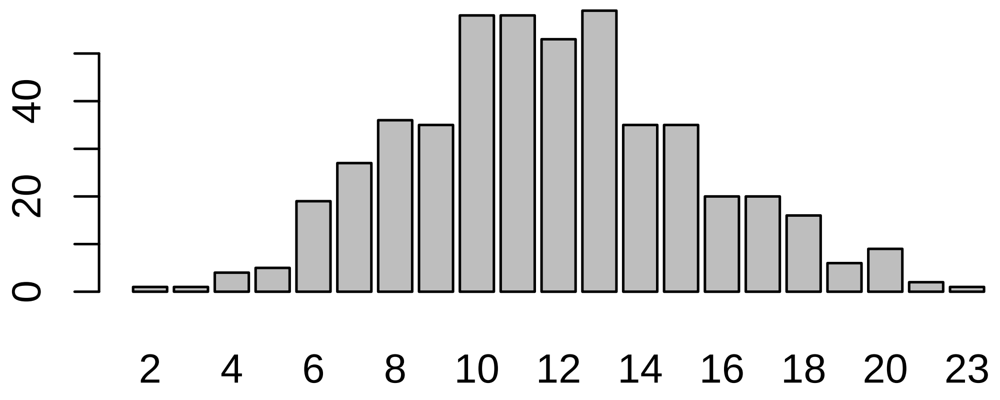

Figure 1. Barplot of frequencies in the permutation distribution of the number of common edges in `nPdist`. The observed number of common edges `numEdges(commonG)=42` is larger than any of the permutation values, which indicates that there are significantly more common edges than what would be expected if co-expression and protein interaction were unrelated.