Graphical analysis of protein interactions in yeast

The aim of this practical is to examine some uses of graphical analysis in a biological setting. The analyses are identical to those demonstrated in the London Tube Graph examples in the introduction to systems biology lecture.

The approaches we saw in the London Tube Map example involved visual examination of the coloured nodes and edges of the graph. For large biological (or other) graphs, this is not feasible. The graph we will be looking at today is shown here.
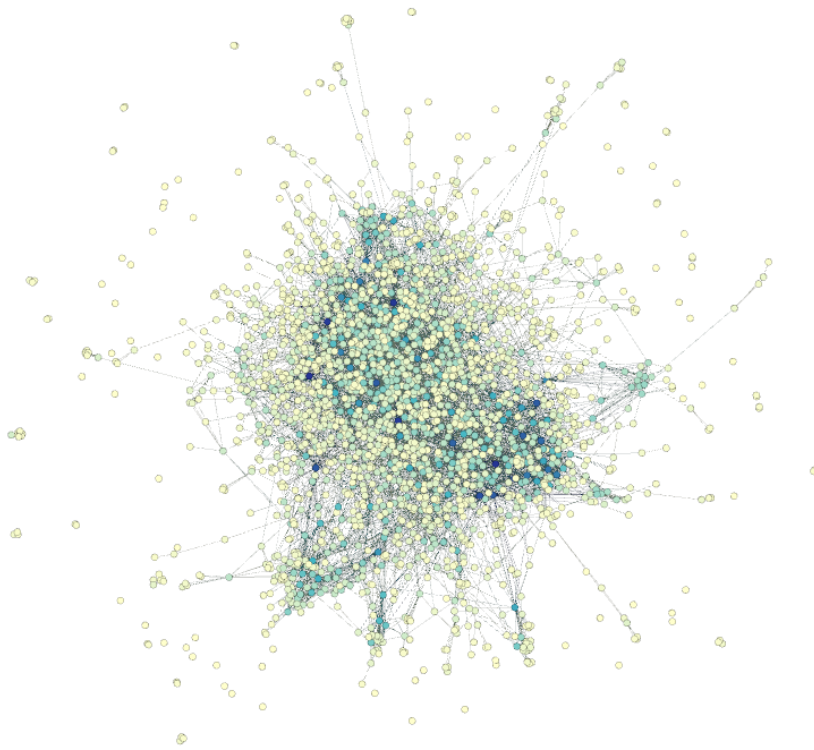

Figure 1: Yeast PIN showing node degree

Questions to be answered for the assignment are written in **bold**.

Programming language

The practical uses the Go programming language. Go is a statically typed, compiled language, but we will be using it through an interactive environment

called Jupyter. This tries to make Go behave as an interpretted language, and this may cause some problems at some stages. If you have any problems, please ask the demonstrators for help.

Setting up the package imports

Go modularises code into packages (much the same way as other languages, though diffferent languages will use different terms for the same concept). Go packages must be imported before they can be used. In a compiled Go program this *must* happen at the beginning of a source code file, though in jupyter, this is not strictly necessary.

The packages that we will use are for printing, `"fmt"`, logging errors, `"log"`, and performing the graph analyses, `"github.com/kortschak/graphprac"`. The first two packages are provided with the language and the last is a small package that wraps graph routines made available through the graph packages of the https://gonum.org project, `graph/...`.

```
import (
    "fmt"
    "log"

    "github.com/kortschak/graphprac"
)
```

The documentation for the `"github.com/kortschak/graphprac"` package is available from the godoc.org website: http://godoc.org/github.com/kortschak/graphprac. You can click on links on the documentation page to see the source of the functions and to link to the definitions of types and externally provided functions. This is an excellent way to get to understand the code.

The routines wrapped by the `"github.com/kortschak/graphprac"` are in `graph/network` and `graph/community`.

Read in a graph

The graph is from a data set used in an analysis of topological structure in the yeast protein interaction network (DOI:10.1093/nar/gkg340).

```
const infile = "./src/github.com/kortschak/graphprac/YeastL.dot"
```

```
g, err := graphprac.NewGraph(infile)
if err != nil {
    log.Fatalf("failed to read graph: %v", err)
}
fmt.Printf("G has %d nodes and %d edges.\n", len(g.Nodes()), len(g.Edges()))
```

```
G has 2361 nodes and 6646 edges.
```

Notice that the number of nodes agrees with the data set summary linked above, but the number of edges disagress. Suggest why this might be.

Examining the nodes

There is no `head` function in Go (we could write one, but the code is short, so there is no need).

The following code loops over the first 10 elements of the nodes slice (essentially and array - there are differences in Go, but that is not important here) and prints out the node value stored in `n`.

Note that running this multiple times will result in different sets of nodes being printed as the nodes returned by `g.Nodes()` are selected in a random order, so the first ten will differ. Try this out.

Why is it not important that a node list be returned in a specific order?

```
for _, n := range g.Nodes()[:10] {
    fmt.Println(n)
}
```

```
&{1150 "SEC23" [{desc "component of COPII coat of ER-golgi vesicles"}]}
&{430 "LTV1" [{desc "low-temperature viability protein"}]}
&{472 "FAT2" [{desc "AMP-binding protein, peroxisomal"}]}
&{609 "YDR383c" [{desc "weak similarity to S.pombe paramyosin"}]}
&{670 "LSM1" [{desc "Sm-like (Lsm) protein"}]}
&{848 "MYO4" [{desc "myosin heavy chain, unconventional (class V) isoform"}]}
&{919 "PBP2" [{desc "PAB1 binding protein"}]}
&{1070 "RPB5" [{desc "DNA-directed RNA polymerase I, II, III 25 KD subunit"}]}
&{1697 "YLR243w" [{desc "strong similarity to YOR262w"}]}
&{152 "Unknown0153" [{desc "Unknow protein !!!"}]}
```

Network analysis

We are going to look at nodes that have a high connectivity or potential for information flow through the network.

Two measure that we can use to do this (*very roughly*) are node betweenness centrality and PageRank.

The routines provided in the **graphprac** package write their analysis results into the graph that is provided as a parameter. This is not how we normally do this kind of analysis, but it make the practical simpler.

3

The writting into the graph is done via a set of attributes that have names that can be queried using the functions in the package.

How can an attribute be queried? Use an example to demonstrate.

The first analysis is for node betweenness of G.

```
graphprac.Betweenness(g)
```

```
nodes, err := graphprac.NodesByAttribute("betweenness", g)
if err != nil {
    log.Fatalf("failed to obtain nodes: %v", err)
}
bestBetween := nodes[0]
for _, n := range nodes[:10] {
    attr := n.Attributes
    fmt.Printf("%s %s -- %s\n", n.Name, attr.Get("betweenness"), attr.Get("desc"))
}
```

```
"SRP1" 210338.37466215802 -- "karyopherin-alpha or importin"
"YKU80" 207675.94102270107 -- "component of DNA end-joining repair pathway"
"HRR25" 206584.73991396077 -- "casein kinase I, ser/thr/tyr protein kinase"
"RPC40" 197208.74555698608 -- "DNA-directed RNA polymerase I, III 40 KD subunit"
"SEC27" 193890.08644779155 -- "coatomer complex beta' chain (beta'-cop) of secretory pathway ve
"HHF1" 190514.32109586368 -- "histone H4"
"CKA1" 175190.9830973763 -- "casein kinase II, catalytic alpha chain"
"PRP6" 150055.96591247874 -- "snRNP(U4/U6)-associated splicing factor"
"SEN15" 134647.32568375507 -- "tRNA splicing endonuclease delta subunit"
"BUD20" 129394.70716794087 -- "weak similarity to human zinc finger protein"
```

If you rerun the code block above, does the order/set of nodes change like in the previous example? Why?

Next we perform a PageRank analysis of G. There are two additional parameters here, `damp` and `tol`.

What is the purpose of these two parameters? *Hint: You will need to read about PageRank.*

```
graphprac.PageRank(g, 0.85, 1e-4)
```

Now we print out the ten highest ranked nodes and keep the highest ranked node of all.

```
nodes, err := graphprac.NodesByAttribute("rank", g)
if err != nil {
    log.Fatalf("failed to obtain nodes: %v", err)
}
bestRank := nodes[0]
for _, n := range nodes[:10] {
    attr := n.Attributes
```

```
    fmt.Printf("%s %s -- %s\n", n.Name, attr.Get("rank"), attr.Get("desc"))
}
```

```
"YKU80" 0.00430241971657203 -- "component of DNA end-joining repair pathway"
"SEC27" 0.004233260731267495 -- "coatomer complex beta' chain (beta'-cop) of secretory pathway
"RPC40" 0.004098353755667779 -- "DNA-directed RNA polymerase I, III 40 KD subunit"
"HRR25" 0.0040836456620875741 -- "casein kinase I, ser/thr/tyr protein kinase"
"HHF1" 0.003693951260436326 -- "histone H4"
"PRP6" 0.003616026005779373 -- "snRNP(U4/U6)-associated splicing factor"
"CKA1" 0.0035935483771140503 -- "casein kinase II, catalytic alpha chain"
"SRP1" 0.0035588630340266784 -- "karyopherin-alpha or importin"
"SEN15" 0.0035385921686267143 -- "tRNA splicing endonuclease delta subunit"
"PWP2" 0.003129708615274176 -- "periodic tryptophan protein"
```

Look at the two sets of highest ranked nodes from the betweenness and PageRank analyses. How well do they agree? How does this situation compare to the case of the London Tube Graph example? Why do you think this is?

The next step is to identify sets of nodes that interact more strongly within the set than they do between sets. These sets are called communities.

The `graphprac.Communities` function take a single extra parameter, `resolution`. We are using a resolution of 10.

```
graphprac.Communities(g, 10)
```

Now we are going to identify the community that the highest PageRanked node is in. Note that the algorithm used to identify communities is a randomised algorithm, the Louvain Algorithm, and so different runs will produce a different name for the community and may include slightly different community memberships.

Community detection is an NP-hard problem and the Louvain Algorithm gives us a reasonable approximation in reasonable time.

```
nodes, err := graphprac.NodesByAttribute("community", g)
if err != nil {
    log.Fatalf("failed to get community: %v", err)
}
comm := ""
for _, n := range nodes {
    attr := n.Attributes
    if n.Name == bestRank.Name {
        comm = attr.Get("community")
        break
    }
}
fmt.Printf("%s is in community %s\n", bestRank.Name, comm)
```
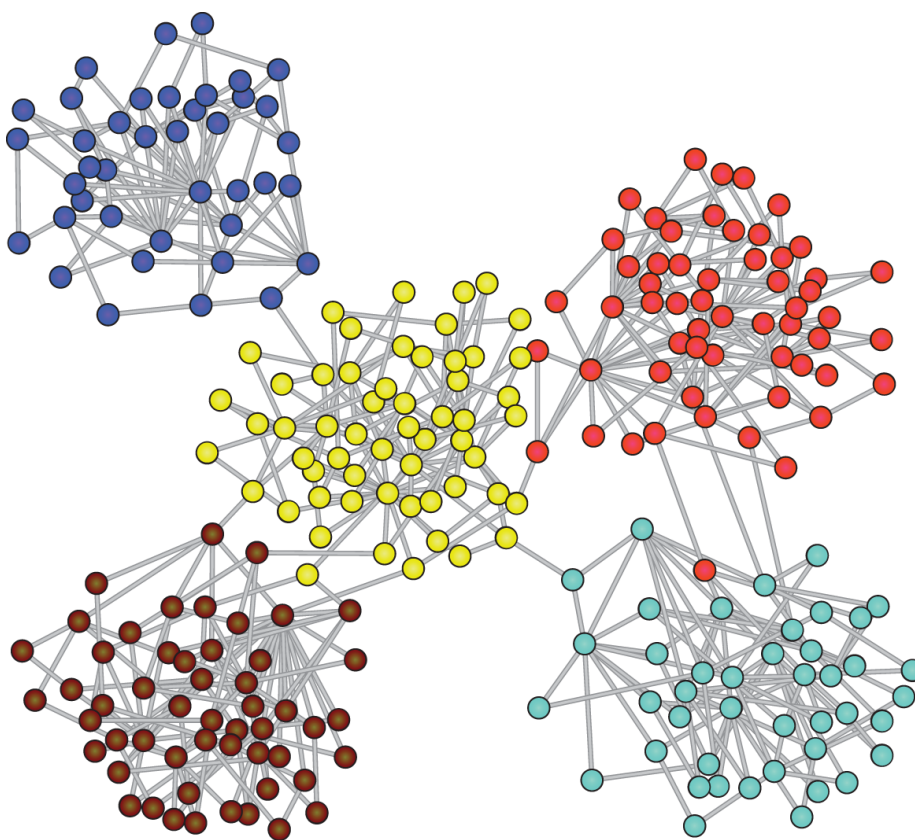
Figure 2: Modular graph showing communities

```
"YKU80" is in community 77

for _, n := range nodes {
    attr := n.Attributes
    if attr.Get("community") == comm {
        fmt.Printf("%s -- %s\n", n.Name, attr.Get("desc"))
    }
}

"YKR051w" -- "similarity to C.elegans hypothetical protein"
"YLR271w" -- "weak similarity to hypothetical protein T04H1.5 C. elegans"
"YML020w" -- "hypothetical protein"
"YBT1" -- "yeast bile transporter"
"YKU80" -- "component of DNA end-joining repair pathway"
"YPR003c" -- "similarity to sulphate transporter proteins"
"SLX1" -- "similarity to hypothetical A.thaliana protein"
"YGR266w" -- "hypothetical protein"
"MET16" -- "3'-phosphoadenylylsulfate reductase"
"APT1" -- "adenine phosphoribosyltransferase"
"RRP3" -- "required for maturation of the 35S primary transcript"
"MSU1" -- "3'-5' exonuclease for RNA 3' ss-tail, mitochondrial"
```

What are the functions of these proteins? Would you expect them to be in the same community? Hint:http://www.yeastgenome.org/

What happens when you alter the resolution parameter to `graphprac.Communities`?

Finally we are going to look for potential targets to disrupt the function of this community. By looking for edges that have a high edge betweenness we may be able to identify candidates for molecular disruption.

Run the two cells below and choose an interaction pair that looks like it might be a good candidate for a druggable target. Write a page (~300 words) explaining why the target was chosen and how it could potentially be investigated further. Give references for information you introduce.

```
graphprac.EdgeBetweenness(g)

edges, err := graphprac.EdgesByAttribute("edge_betweenness", g)
if err != nil {
    log.Fatalf("failed to obtain nodes: %v", err)
}
for _, e := range edges {
    fattr := e.F.Attributes
    tattr := e.T.Attributes
    if fattr.Get("community") != comm && tattr.Get("community") != comm {
        continue
    }
```

```
    attr := e.Attributes
    fmt.Printf("%s--%s %s (%s--%s)\n", e.F.Name, e.T.Name, attr.Get("edge_betweenness"), fa
}

"TPS1"--"YKU80" 17998.851133865064 ("alpha,alpha-trehalose-phosphate synthase, 56 KD subunit"-
"SRP54"--"YKU80" 14825.759518297673 ("signal recognition particle subunit"--"component of DNA
YKU80"--"TEM1" 13820.079410545346 ("component of DNA end-joining repair pathway"--"GTP-bindin
"CLU1"--"YKU80" 13753.683248464687 ("translation initiation factor eIF3 (p135 subunit)"--"comp
YKU80"--"PHO85" 13097.642423104562 ("component of DNA end-joining repair pathway"--"cyclin-de
"TFC7"--"YKU80" 12953.95456623092 ("TFIIIC (transcription initiation factor) subunit, 55 kDa"-
"ARO1"--"YKU80" 12501.64895653774 ("arom pentafunctional enzyme"--"component of DNA end-joinin
YKU80"--"POR1" 11687.8089002464 ("component of DNA end-joining repair pathway"--"mitochondria
"FUN12"--"YKU80" 11501.678795895019 ("general translation factor eIF2 homolog"--"component of
"PRE1"--"YKU80" 10728.427961369243 ("20S proteasome subunit C11(beta4)"--"component of DNA end
"SLC1"--"YKU80" 10177.417198058447 ("fatty acyltransferase"--"component of DNA end-joining rep
YKU80"--"CCT3" 9825.474014995416 ("component of DNA end-joining repair pathway"--"chaperonin
"SIS1"--"YKU80" 9821.675056728662 ("heat shock protein"--"component of DNA end-joining repair p
"LYS12"--"YKU80" 9703.52822295141 ("homo-isocitrate dehydrogenase"--"component of DNA end-join
"LST8"--"YKU80" 9289.382914876924 ("required for transport of permeases from the golgi to the p
"COP1"--"YKU80" 8897.176366428768 ("coatomer complex alpha chain of secretory pathway vesicles"
YKU80"--"ECM10" 8500.804851523682 ("component of DNA end-joining repair pathway"--"heat shock
YKU80"--"GPH1" 8151.344104161202 ("component of DNA end-joining repair pathway"--"glycogen ph
YKU80"--"YDR128w" 8054.340847049892 ("component of DNA end-joining repair pathway"--"weak sim
"RPN12"--"YKU80" 7878.729773843028 ("26S proteasome regulatory subunit"--"component of DNA end
"VMA8"--"YKU80" 6889.36425241908 ("H+-ATPsynthase V1 domain 32 KD subunit, vacuolar"--"compone
YKU80"--"ADR1" 6886.822448775052 ("component of DNA end-joining repair pathway"--"zinc-finger
"CCT5"--"YKU80" 6672.496526122643 ("T-complex protein 1, epsilon subunit"--"component of DNA en
YKU80"--"YHR033w" 6272.305968617097 ("component of DNA end-joining repair pathway"--"strong s
"DPB2"--"YKU80" 6266.169014962364 ("DNA-directed DNA polymerase epsilon, subunit B"--"componen
"DHH1"--"YKU80" 6057.607146202224 ("putative RNA helicase of the DEAD box family"--"component o
YKU80"--"OYE2" 5765.069118236378 ("component of DNA end-joining repair pathway"--"NADPH dehydr
"NOG2"--"YKU80" 5573.892639441366 ("strong similarity to human breast tumor associated autoanti
YKU80"--"ATP3" 5449.634938963042 ("component of DNA end-joining repair pathway"--"F1F0-ATPase
YKU80"--"ILV2" 5424.393255927188 ("component of DNA end-joining repair pathway"--"acetolactat
YKU80"--"PHO86" 5244.937743019643 ("component of DNA end-joining repair pathway"--"inorganic
YKU80"--"STI1" 5210.413342450313 ("component of DNA end-joining repair pathway"--"stress-indu
YKU80"--"YMR226c" 5145.8401418544645 ("component of DNA end-joining repair pathway"--"similar
YKU80"--"MKK2" 5119.151639528138 ("component of DNA end-joining repair pathway"--"protein kin
"PDX1"--"YKU80" 4923.250775978394 ("pyruvate dehydrogenase complex protein X"--"component of D
YKU80"--"YJR072c" 4571.658150553767 ("component of DNA end-joining repair pathway"--"strong s
YKU80"--"YML020w" 4446 ("component of DNA end-joining repair pathway"--"hypothetical protein"
YKU80"--"YLR271w" 4446 ("component of DNA end-joining repair pathway"--"weak similarity to hyp
YKU80"--"MET16" 4446 ("component of DNA end-joining repair pathway"--"3'-phosphoadenylylsulfa
YKU80"--"RRP3" 4446 ("component of DNA end-joining repair pathway"--"required for maturation o
YKU80"--"YKR051w" 4446 ("component of DNA end-joining repair pathway"--"similarity to C.elegan
YKU80"--"YGR266w" 4446 ("component of DNA end-joining repair pathway"--"hypothetical protein"
```

"YKU80"--"YBT1" 4446 ("component of DNA end-joining repair pathway"--"yeast bile transporter")
"YKU80"--"APT1" 4446 ("component of DNA end-joining repair pathway"--"adenine phosphoribosyltr
"YKU80"--"SLX1" 4446 ("component of DNA end-joining repair pathway"--"similarity to hypothetica
"YKU80"--"YPR003c" 4445.999999999999 ("component of DNA end-joining repair pathway"--"similari
"YKU80"--"PST2" 4281.565503404309 ("component of DNA end-joining repair pathway"--"strong simi
"YKU80"--"YDR453c" 4145.113510914267 ("component of DNA end-joining repair pathway"--"strong s
"YKU80"--"SOD2" 3992.4634834948074 ("component of DNA end-joining repair pathway"--"superoxide
"YOL078w"--"YKU80" 3832.4275308402493 ("similarity to stress activated MAP kinase interacting p
"YKU80"--"VID31" 3830.974200603829 ("component of DNA end-joining repair pathway"--"similarity
"ACC1"--"YKU80" 3805.0375312336846 ("acetyl-CoA carboxylase"--"component of DNA end-joining re
"YKU80"--"IDH1" 3802.3525894641934 ("component of DNA end-joining repair pathway"--"isocitrate
"SIP1"--"YKU80" 3748.5942568338482 ("multicopy suppressor of SNF1"--"component of DNA end-join
"YKU80"--"YER077c" 3648.0147204195855 ("component of DNA end-joining repair pathway"--"hypothe
"PUP3"--"YKU80" 3627.8337619424087 ("20S proteasome subunit (beta3)"--"component of DNA end-jo
"YKU80"--"PUF3" 3418.989072974834 ("component of DNA end-joining repair pathway"--"transcript-
"YKU80"--"FOL2" 3353.2509828413176 ("component of DNA end-joining repair pathway"--"GTP cycloh
"YKU80"--"MSU1" 3327.7472733989575 ("component of DNA end-joining repair pathway"--"3'-5' exon
"YKU70"--"YKU80" 3200.120654849882 ("high-affinity DNA-binding protein"--"component of DNA end
"YKU80"--"LSC1" 3087.8607252755655 ("component of DNA end-joining repair pathway"--"succinate-
"CPA2"--"YKU80" 2948.4145746391328 ("arginine-specific carbamoylphosphate synthase, large chai
"YKU80"--"GAL7" 2644.7063040442877 ("component of DNA end-joining repair pathway"--"UDP-glucos
"MSU1"--"SLD2" 1653.8408453142388 ("3'-5' exonuclease for RNA 3' ss-tail, mitochondrial"--"chr

9