Designing Experiments on Networks

Introduction

This vignette explains a few simple examples on the package `networkDesign` which allows optimal design of experiments on networks under a particular model. All the examples are taken from our research paper "Optimal Design of Experiments on Connected Units with Application to Social Networks" https://doi.org/10.1111/rssc.12170 (preprint)

The aim of the vignette is to provide the means for users to reproduce the results in that paper, and extend them to their own work. This vignette, and indeed the whole package, is very much a draft, and suggestions for changes/improvements are welcomed.

Summary of theory

Consider a simple network as shown in the image. Nodes 1,2,3, and 4 represent people in a network, and we wish to show an advert to each of them.

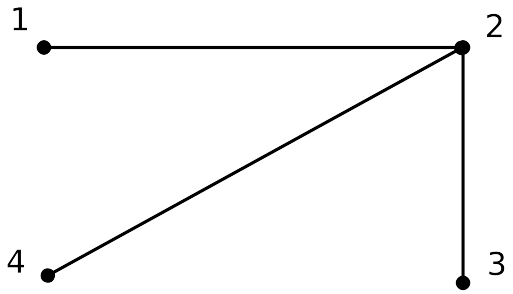


Figure 1: A simple network

The idea behind the model is that if a treatment is given to subject 2, connections of subject 2 might be affected by the treatment, and their response might be altered because of the treatment I gave to subject 1.

The total effect on subject 1 is determined by whatever treatment I give to subject 1 himself, plus an effect due to the treatment I gave to subject 2.

This is formalised in the **Linear network effect model**:

$Y_i = \mu + \tau_{t(i)} + \sum_{k=1} A_{ik}\gamma_{t(k)} + \epsilon_i$

where

- ϵ_i are i.i.d with mean 0, variance σ^2;
- $t(i)$ is the treatment applied to subject i;
- τ_j are treatment effects for $j = 1, \ldots, m$;
- we assume w.l.o.g that $\tau_m = 0$ for uniqueness.

- γ_j is the corresponding *network effect*, the change in the behaviour on a subject due to giving a connected subject a particular treatment.

Getting Started

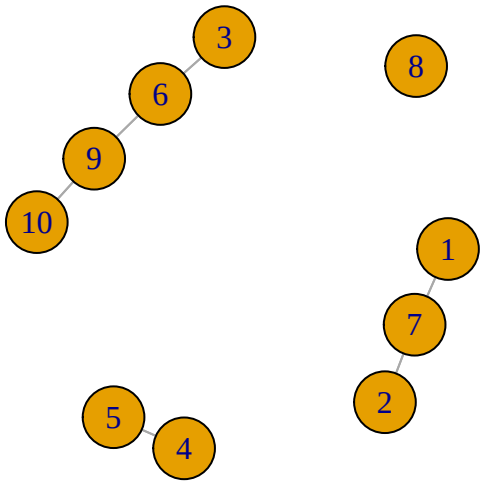
We load the required package `igraph` and then this package `networkDesign`. All the examples are in the data file `JRSSExamples`

```
library(igraph)
library(networkDesign)
attach(JRSSExamples)
```

Example 1

The first command plots the graph (for more on network visualisation see the tutorial https://kateto.net/network-visualization); we can see this toy example is a not very densely connected network.

```
g1<-graph_from_adjacency_matrix(ex1)
l <- layout_nicely(g1)
E(g1)$arrow.mode<-0
plot.igraph(g1,edge.arrow.size=0.4, vertex.size=30,layout=l)
```



Next, we search through all possible designs to find the design which is best.

```
gridSearch(ex1,2)
#> $AOptVal
#> [1] 0.4186047
#>
#> $AOptDesign
#>       [,1] [,2]
#>  [1,]    1    0
```

```
#>  [2,]    1    0
#>  [3,]    1    0
#>  [4,]    0    1
#>  [5,]    0    1
#>  [6,]    1    0
#>  [7,]    0    1
#>  [8,]    1    0
#>  [9,]    1    0
#> [10,]    0    1
#>
#> $numEval
#> [1] 507

gridSearch(ex1,2, networkEffects = TRUE)
#> $AOptVal
#> [1] 0.2369146
#>
#> $AOptDesign
#>       [,1] [,2]
#>  [1,]    1    0
#>  [2,]    1    0
#>  [3,]    1    0
#>  [4,]    0    1
#>  [5,]    0    1
#>  [6,]    0    1
#>  [7,]    1    0
#>  [8,]    1    0
#>  [9,]    1    0
#> [10,]    0    1
#>
#> $numEval
#> [1] 507
```

There are two possible things we might wish to do

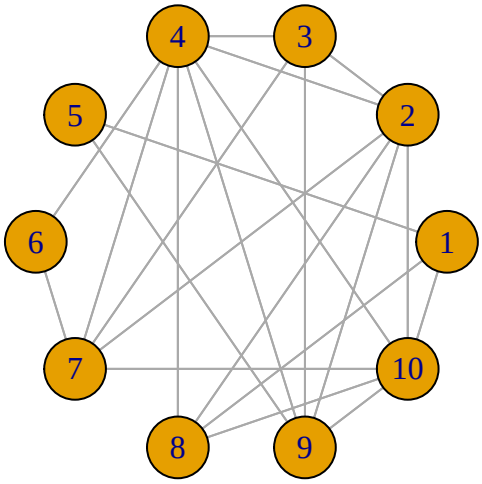
- estimate the pariwise difference between treatment effects, τ_i with minimum variance.
- estimate the pairwise difference between network effects, γ_i with minimum variance.

`gridSearch` with no parameters specified will perform an exhaustive search through every design, and find the best design according to the first of these criteria. Adding 'networkEffexts=TRUE' The output shows the value of the A criterion optimal function value for estimating the treatment effects, and the optimal design, followed by the A criterion optimal design function value for the network effects, the corresponding optimal design, and the number of evaluations done for this. The design is such that we should give treatment 1 to nodes with a 1 in the first column (1,2,3,6,8,9) and treatment 2 to nodes (4,5,7,10).

Example 2

We can do the same for a second example network, which is a slightly more strongly connected network.

```
g2<-graph_from_adjacency_matrix(ex2)
l <- layout_in_circle(g2)
E(g2)$arrow.mode<-0
plot.igraph(g2,edge.arrow.size=0.4, vertex.size=30,layout=l)
```




```
gridSearch(JRSSExamples$ex2,2)
#> $AOptVal
#> [1] 0.40625
#>
#> $AOptDesign
#>       [,1] [,2]
#>  [1,]    1    0
#>  [2,]    1    0
#>  [3,]    0    1
#>  [4,]    0    1
#>  [5,]    0    1
#>  [6,]    1    0
#>  [7,]    0    1
#>  [8,]    1    0
#>  [9,]    0    1
#> [10,]    1    0
#>
#> $numEval
#> [1] 511
gridSearch(JRSSExamples$ex2,2, networkEffects = TRUE)
#> $AOptVal
#> [1] 0.1256614
#>
#> $AOptDesign
#>       [,1] [,2]
#>  [1,]    1    0
```

```
#>  [2,]    0    1
#>  [3,]    1    0
#>  [4,]    0    1
#>  [5,]    1    0
#>  [6,]    0    1
#>  [7,]    0    1
#>  [8,]    1    0
#>  [9,]    0    1
#> [10,]    1    0
#>
#> $numEval
#> [1] 511
```

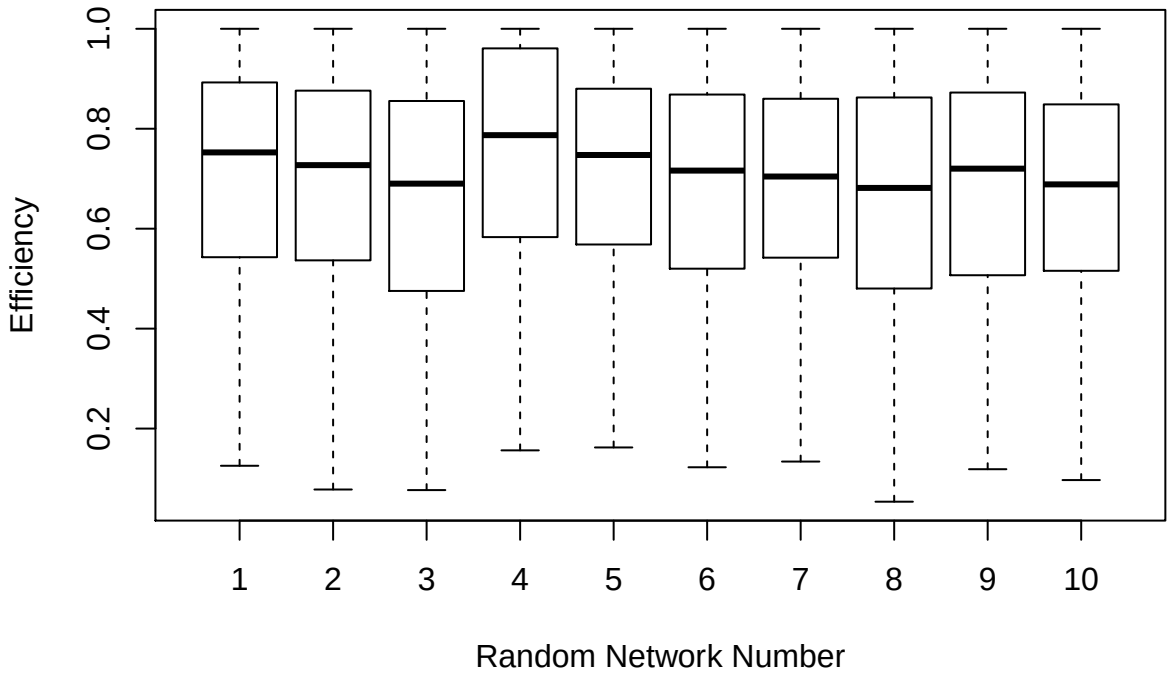
Calculating Efficiencies

We showed (in Section 4.1 of the research paper) that if the Linear Network Effects model were correct, but we ignored it in the design, then we would adopt a Completely Randomized design. For example, for m=2 treatments, we would, at random, give half of our experimental units the first treatment, and half the second.

Here, for each of ten random Erdős-Renyi networks of size $n = 12$, we demonstrate the efficiency of the (incorrect) completely randomised design with respect to the optimal design of the network.

```
n<-12 # Size of network
prob<-0.6 # Probability of link between any two nodes
p<-2 # Number or treatments- we looked at 2 and 3 in the paper
numNetworks<-10 # Number of Networks to investigate

effList<-NULL

for(i in 1:numNetworks){   # For each of numNetworks random networks
ASec4<-as.matrix(as_adjacency_matrix(sample_gnp(n,prob)))   # Generate an Erdos-Renyi network
AOptSec4 <- gridSearch(ASec4,p)   # Find the optimal design

efficiencies<-NULL # Initiate list of efficiencies

### Initiate the design such that every experimental unit is given treatment 1.
testDesign<-rep(1,dim(ASec4)[2])
#flag<-TRUE
while (length(testDesign)==n){
testWeight<-matrix(rep(0,n*p),nrow=n)
for (j in (1:n)){
testWeight[j,testDesign[j]]<-1
}

### Check if design is balanced
if (((max(colSums(testWeight))-0.75)<(n/p)) & ((min(colSums(testWeight))+0.75)>(n/p))){

### if design is balanced, find its corresponding information matrix,
### and hence the efficiency.
trialInfMatrix<-informationMat(ASec4,testWeight)
if (det(trialInfMatrix)>0.0001){
  designEval<-evaluateNetworkDesign(trialInfMatrix,p)
  efficiencies<-rbind(efficiencies,(AOptSec4$AOptVal)/(designEval$ATrial))
}
else{efficiencies<-rbind(efficiencies,0)}
```

```
}

### Consider the next possible design. nextDesign produces next design sequentially
testDesign<-nextDesign(testDesign,p)
}
effList<-cbind(effList,efficiencies) # store the efficiency
}

### Finally plot the efficiencies.
boxplot(effList,xlab="Random Network Number",ylab="Efficiency")
```



Efficiencies of Example 1

We consider example 1 again, where the number of treatements $m = 2$. If we believed there is no network effect in the model, we would conclude that balanced designs with 5 subjects chosen at randomgiven each of treatments 1 and 2 were optimal. We might chose an arbitrary balanced design {1,1,1,1,1,2,2,2,2,2}. We calculate the efficiencies for our two criterion, ϕ_1 for extimating the subject effects and ϕ_2, that for the network effects.

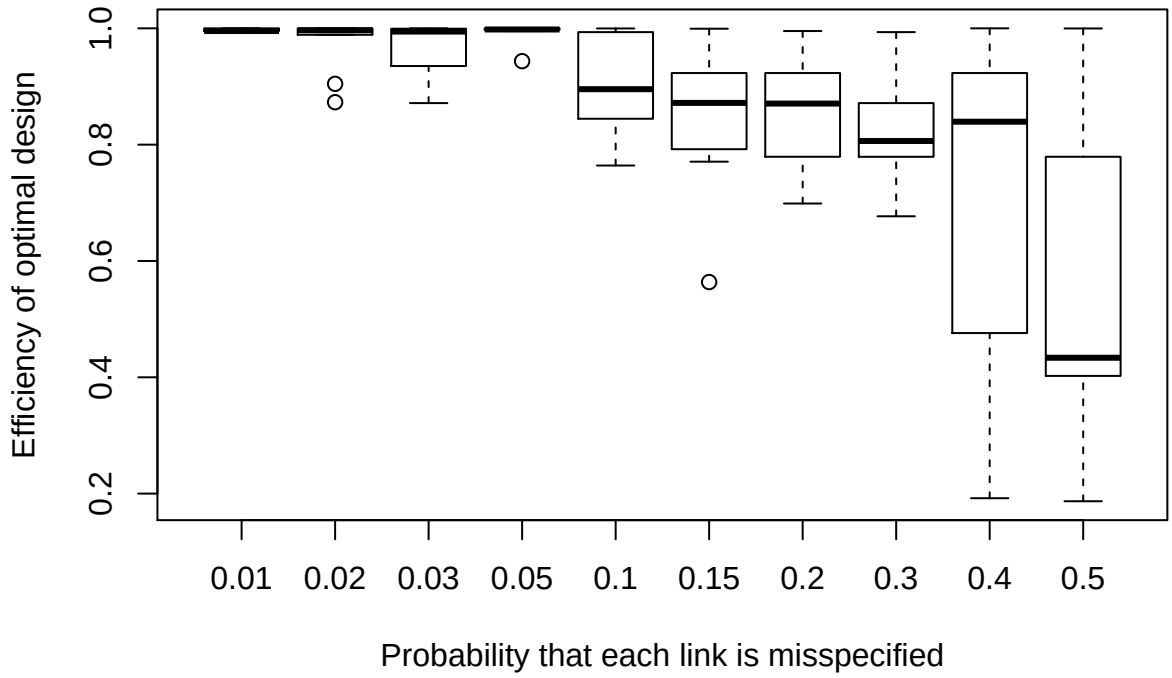
```
testDesign<-c(rep(1,5),rep(2,5))
testWeight<-matrix(rep(0,10*2),nrow=10)
for (j in (1:10)){
  testWeight[j,testDesign[j]]<-1;
}
trialInfMatrix<-informationMat(ex1,testWeight);
designEval<-evaluateNetworkDesign(trialInfMatrix,2)
```

```
example1Opt<-gridSearch(ex1,2)
example1OptNet<-gridSearch(ex1,2,networkEffects = TRUE)
# Efficiency for phi1
1/(designEval$ATrial/example1Opt$AOptVal)
#> [1] 0.9298513

##Efficiency for phi2
1/(designEval$ATrialNet/example1OptNet$AOptVal)
#> [1] 0.8025482
```

Mispescification of network

We are also interested in the case where the network is misspecified- that is we do not know with certainty whether there is a link between two subjects. We simulate this situation by again considering Example 2. For any two nodes in the network given by adjacency matrix A, we flip $A_{ij} = A_{ji}$ from 1 to 0 with probability p, and from 0 to 1 with the same probability. We calculate the efficiency of using the optimal design forthe mispecified network when the network is really as described in example 2, and do this for 1000 simulated networks.

```
n<-10 # Size of matrix
p<-2 # Number of treatments
numRandomMatrices<-10 # numRandomMatrices for each probabilty

efficiencies<-matrix(rep(0,n*numRandomMatrices),nrow=numRandomMatrices)
a1<-gridSearch(ex2,p)$AOptVal # The optimal value for the correctly specified optimal design

probList<-c(0.01,0.02,0.03,0.05,0.1,0.15,0.2,0.3,0.4,0.5)
for (probCount in (1:10)) {
probSwitch<-probList[probCount]
for (count in 1:numRandomMatrices) {
### Generate a binary matrix with based on the original where
### each element has a fixced prob probSwitch of being switched
newA<-ex2
m2 <- matrix(rbinom(n*n,1,probSwitch),n,n)
newA<-(ex2+m2)%%2 # Add this to Adj Matrix Modulo 2
diag(newA)<-0  ## Ensure diagonal is zero

gs<-gridSearch(newA,p) # Find the optimal design for the randomly permuted network

trialInfMatrix<-informationMat(ex2,gs$AOptDesign) # Calculate its information matrix
if (det(trialInfMatrix)>0.01){
  designEval<-evaluateNetworkDesign(trialInfMatrix,p)
}
efficiencies[count,probCount]=a1/designEval$ATrial
}
}

boxplot(efficiencies,names=probList,xlab="Probability that each link is misspecified",ylab="Efficiency
```



We see in the boxplot that efficiency tends to decrease with increasing probability of misspecification. There is a large variation in efficiencies even for small probabilities of misspecification, but the typical (median) design is generally reasonably efficient. In practice, simulations such as this could help experimenters to find designs robust against misspecification.

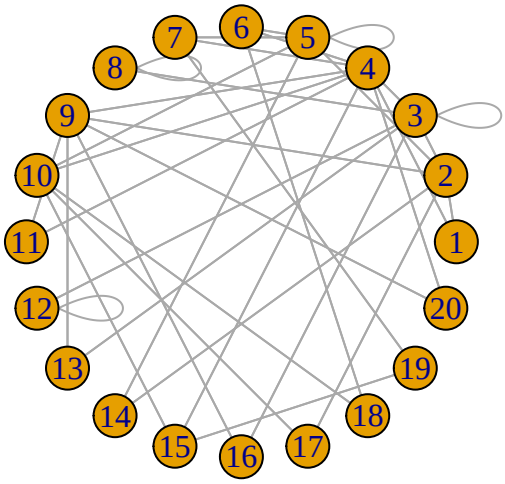
Other examples

Much of the rest of the paper was taken up in showing how the technique could be used in a variety of networks.

Example 3

Suppose we design an experiment to determine the effectiveness of a new advertisement. Ten people are sent a video advertisement for a type of soft drink (treatment 1), and ten are shown no advertising (treatment 2). The amount of soft drink each person buys in the following week is then monitored. The marketing executives are interested not only in how effective the advertisement is, but also how effective it is at being conveyed to friends of those in the group which originally saw the advertisement (this is known as viral marketing), so 20 people are chosen such that the connectivity structure of an online social network of the subjects is known.

```
### Plot the Network.
g3<- graph_from_adjacency_matrix(ex3)
l <- layout_in_circle(g3)
E(g3)$arrow.mode<-0
plot.igraph(g3,edge.arrow.size=0.4, vertex.size=20,layout=l)
```




```
### Find the optimal design
ex3.opt<-gridSearch(ex3,p=2)
ex3.opt$AOptVal
#> [1] 0.2003026
ex3.net<-gridSearch(ex3,p=2, networkEffects = TRUE)
ex3.net$AOptVal
#> [1] 0.02542984
```

If our linear network effects model is true, we find $\phi_1^* = 0.2002$. The bottom number in each node shows the optimal design for ϕ_2 for estimating the network effect (the viral effect of the advert), with $\phi_2^* = 0.0284$.

(Actually, in the paper we accidentally specified some links such that some nodes were linked to themselves- the diagonal of the adjacency matrix was not zero. If we correct this we get a very slightly different output)

```
diag(ex3)<-0
ex3.opt<-gridSearch(ex3,2)
ex3.opt$AOptVal
#> [1] 0.2002865
```

Example 4: Field Trials

Druilhet (1999) shows that this example design is optimal for estimating $m = 4$ treatment effects:

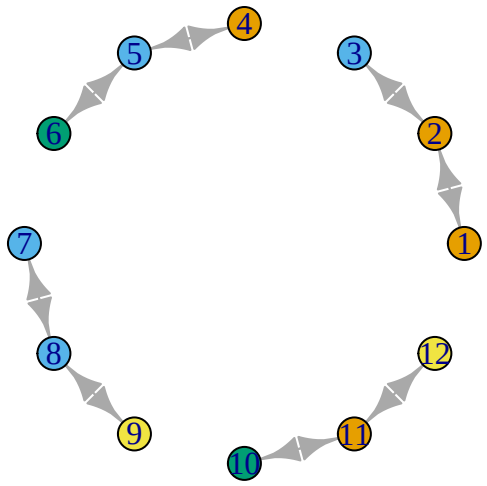
$$\begin{pmatrix} 4 \\ 3 \\ 2 \\ 1 \end{pmatrix} \begin{pmatrix} 2 & 3 & 4 \\ 1 & 4 & 3 \\ 4 & 1 & 2 \\ 3 & 2 & 1 \end{pmatrix} \begin{pmatrix} 2 \\ 1 \\ 4 \\ 3 \end{pmatrix}$$

Each number represents a treatment applied to a plot, and the treatment will affect its left and right

neighbours within the block, each block being a horizontal line of plots. There is a block effect for all subjects in a block. The plots in the left and right brackets are "guard" plots, and no response is gathered from these plots.

We represent the plots as a network and find the optimal design.

```
ex4.opt<-gridSearch(ex4,4)
ex4.opt$AOptVal
#> [1] 0.8333333
(ex4.des<-as.vector(ex4.opt$AOptDesign%*%c(1:4)))# A shorter way to write the optimal design
#>  [1] 1 1 2 1 2 3 2 2 4 3 1 4

### Now plot the graph
g4<-graph_from_adjacency_matrix(ex4)
l <- layout_in_circle(g4)
E(g4)$arrow.mode<-0
### Plot the network with different nodes coloured according to which treatment they receive.
plot(graph_from_adjacency_matrix(ex4),vertex.color=ex4.des,layout=l)
```



Without guard plots, note that the optimal design is not balanced, let alone neighbour balanced; for instance, treatment 2 appears 4 times, but treatment 1 only twice. However, Druilhet's model assumes a block effect which we do not have in our present model, so a direct comparison may not be fair; with our method we can design an experiment without guard plots.

Note this is not the same optimal design found in the paper, where we used a different ordering for the exhaustive search, but has the same optimal value for phi1. We can check the published design is equivalent with

```
testDesign<-c(1,2,2,3,2,4,1,4,3,2,4,4)
testWeight<-matrix(rep(0,12*4),nrow=12)
for (j in (1:12)){
     testWeight[j,testDesign[j]]<-1;
 }
trialInfMatrix<-informationMat(ex4,testWeight);
designEval<-evaluateNetworkDesign(trialInfMatrix,4)
designEval$ATrial
#> [1] 0.8333333
```

The optimality criterian for phi1 is 0.83333 for both.

Example 5

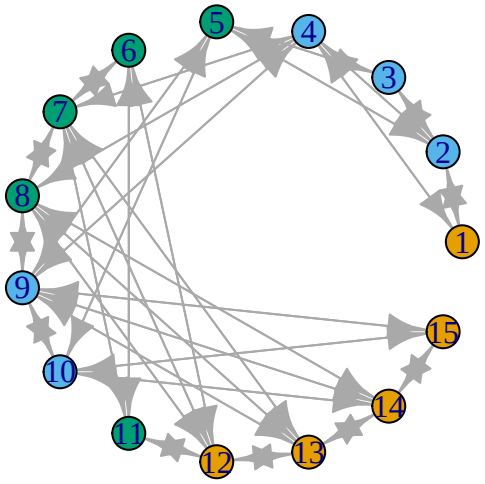
Most field trial experiments focus on rectangular fields. Let us assume we have an irregularly shaped field divided into plots as shown below

| . | . | 1 | 2 | 3 | | . | . | 4 | . | 5| |6 | 7 | 8 | 9 | 10| |11 |12 |13 |14 |15|

An integer represents the number of a plot. We wish to perform a trial on different fertilisers, to estimate the difference between fertilizer (treatment) effects with minimum average variance. We assume that any fertiliser applied might leach to any of the 8 plots that touch the treated plot, including those only touching at corners.

We represent the plots as a network and find the optimal design.

```
ex5.opt<-gridSearch(ex5A,3)
ex5.opt$AOptVal
#> [1] 0.4051849
(ex5.des<-as.vector(ex5.opt$AOptDesign%*%c(1:3)))# A shorter way to write the optimal design
#>  [1] 1 2 2 2 3 3 3 3 2 2 3 1 1 1 1

### Now plot the graph
g5<-graph_from_adjacency_matrix(ex5A)
l <- layout_in_circle(g5)
E(g5)$arrow.mode<-0
plot(graph_from_adjacency_matrix(ex5A),vertex.color=ex5.des,layout=l)
```



This design is balanced, with all treatments equally replicated. Here, $\phi_1^* = 0.4052$.

Example 6: A Crossover trial with planned dropouts

Usually in a crossover trial, including a dropout in the design could be tricky. Let us assume we are performing a crossover trial on four subjects (a, b, c, and d) over four periods with three treatments (labelled 1 to 3), each of which is believed to have a wash-out time of one period. Assume further that it becomes clear that participant b will not be able to take the treatment in the third period of the trial. This is represented as

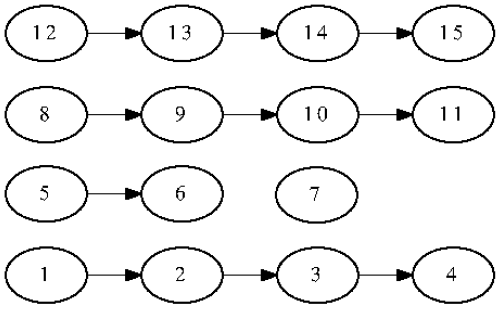
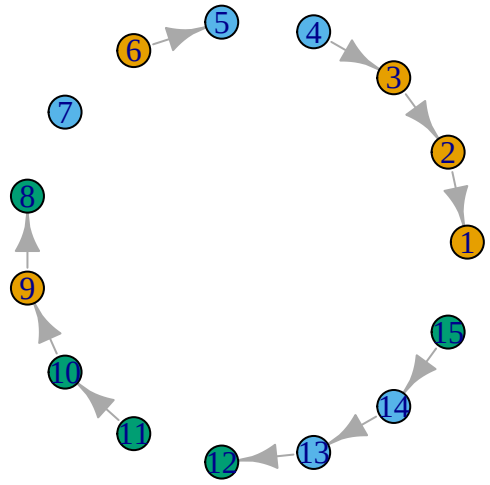


Figure 2: A crossover trial

We use each subject/period combination as a node in our network, and we modify our methodology such that links are uni-directional by relaxing the restriction that A is symmetric; our Linear Network Effects Model in Equation (**??**) is still valid, and we perform an exhaustive search as before to minimise ϕ_1.

```
ex6.opt<-gridSearch(ex6A,3)
ex6.opt$AOptVal
#> [1] 0.412766
(ex6.des<-as.vector(ex6.opt$AOptDesign%*%c(1:3)))# A shorter way to write the optimal design
#>  [1] 1 1 1 2 2 1 2 3 1 3 3 3 2 2 3

### Now plot the graph
g6<-graph_from_adjacency_matrix(ex6A)
l <- layout_in_circle(g6)
E(g6)$arrow.mode<-0
plot(graph_from_adjacency_matrix(ex6A),vertex.color=ex6.des,layout=l)
```



Example 7 : Large networks

We dealt in this paper with small networks, and a small number of treatments. Current work seeks to extend this to much larger networks.

In this paper, we assumed we have a network of 10000 customers, and want to assess the effectiveness of sending 4 different adverts to four subsets each of 25 customers (before we send the best advertisement to the rest of the customers). An advertisement is deemed effective if more customers click on a hyperlink in a message. The advertisements will be released to a test panel of subjects, with each subject being allocated a single treatment. The response of these subjects to the advert will be measured as the number of times that hyperlinks are clicked in the advertisement. (Alternatively we might choose to measure a longer term response, such as a measure of changes in the behaviour of the subjects.)

We first generate a scale-free (Barabasi-Albert) network of size using igraph package.

```
A<-as_adjacency_matrix(barabasi.game(10000,power=1,m=2,directed=FALSE))
```

We then wish to assess designs with four adverts. We use five treatments: treatments 1 to 4 are coded to mean that adverts 1 to 4 are sent to a customer, and we invent a special treatment 0 which means that the customer is not selected. It is important that we include customers that are not given an advertisement in our experiment, as even though we do not show them an advertisement directly, the advertisement may be passed to them virally and they may click on the hyperlink.

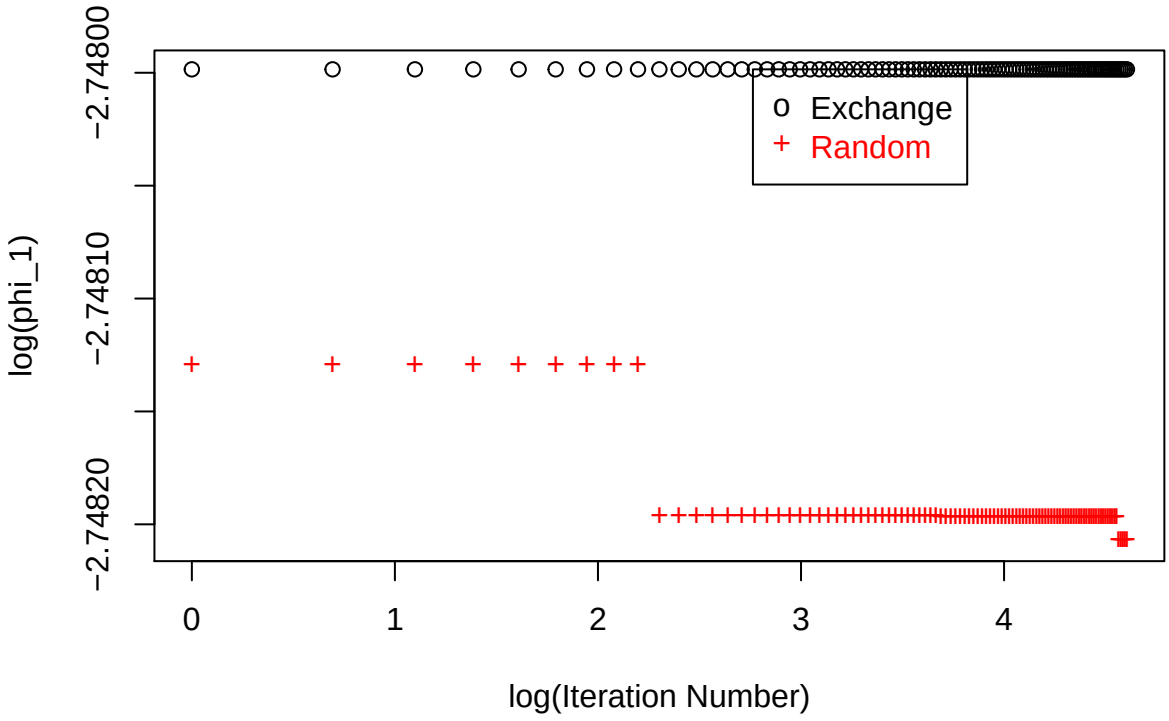
As our network of $n = 10000$ does not allow us to perform an exhaustive search, we compare two methods: 1) the lowest information function found from a number of iterations of the exchange algorithm. Briefly, at each iteration we swap the treatements from a customer given no treatment (Treatment 0) to one given one of the four adverts (treatments 1 to 4) 2) the lowest information function found after evaluating a number of randomly chosen designs.

We have encoded these two methods in the `bigNetworkSearch` function.

```
# Select 100 nodes to be given a treatment and find the best design using
# 1) exchange algorithm and
numIterations<-100
y0<-bigNetworkSearch(A,4,100,0,numIterations)

# 2) randomly chosen designs
y1<-bigNetworkSearch(A,4,100,1,numIterations)
```

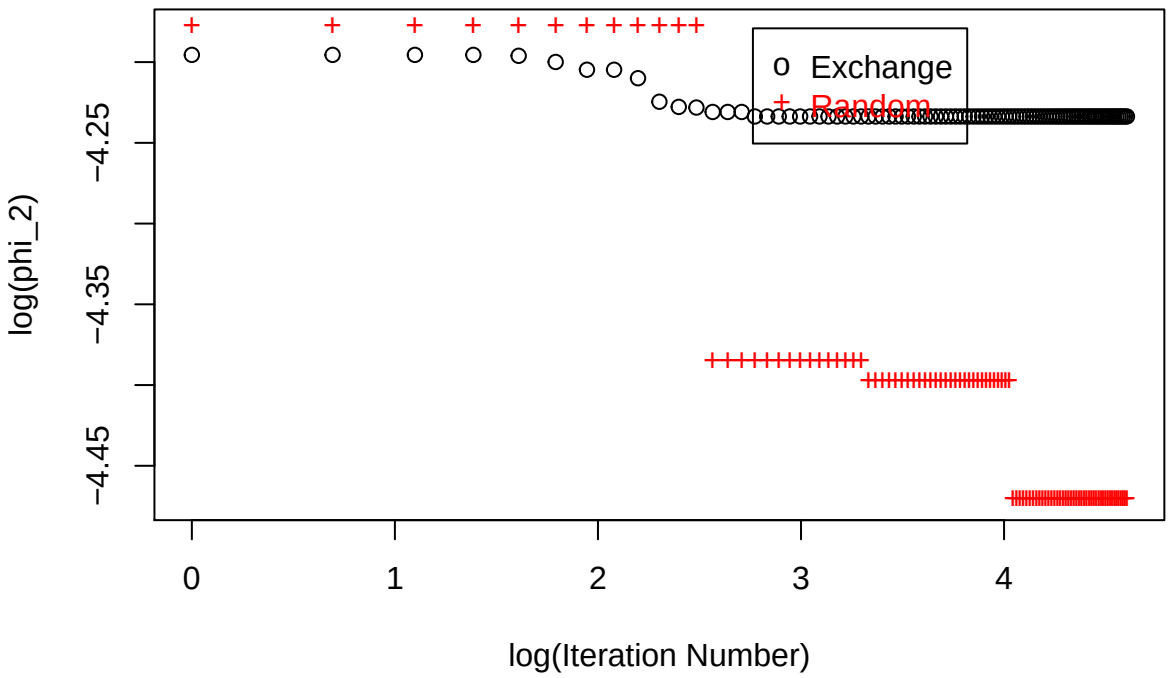
We plot ϕ_1 for exchange algorithm (black circles) and randomly chosen designs (red crosses)

```
ylimMin<-min(min(log(y0$runningAOptVal),log(y1$runningAOptVal)))
ylimMax<-max(max(log(y0$runningAOptVal),log(y1$runningAOptVal)))
plot(log(1:numIterations),log(y0$runningAOptVal),ylim=c(ylimMin,ylimMax),xlab="log(Iteration Number)",yl
points(log(1:numIterations),log(y1$runningAOptVal),pch="+",col="red")
legend(0.6*log(numIterations),ylimMax,c("Exchange","Random"),pch=c("o","+"),col=c("black","red"),text.co
```



We make a similar plot for ϕ_2.

```
ylimNetMin<-min(min(log(y0$runningAOptNetVal),log(y1$runningAOptNetVal)))
ylimNetMax<-max(max(log(y0$runningAOptNetVal),log(y1$runningAOptNetVal)))

plot(log(1:numIterations),log(y0$runningAOptNetVal),ylim=c(ylimNetMin,ylimNetMax),xlab="log(Iteration Nu
points(log(1:numIterations),log(y1$runningAOptNetVal),pch="+",col="red")
legend(0.6*log(numIterations),ylimNetMax,c("Exchange","Random"),pch=c("o","+"),col=c("black","red"),tex
```


log(phi_2)
−4.25
−4.35
−4.45
0
1
2
3
4
log(Iteration Number)
o Exchange
+ Random