20220404_miss_data_model

I do *not* know how to specify these missing data models.

Let's try anyways!

```
# loading packages
library(sna)
library(ergm)
```

Generate some toy networks

```
# fix seed for now
set.seed(123)

# just generating a random graph
testNet = rgraph(n = 20,
                 m = 1,
                 tprob = 0.5,
                 mode = "graph")

# number of ties
sum(testNet)
```

```
## [1] 206
```

```
# since it's symmetric (undirected), the unique ties are half of that sum.
```

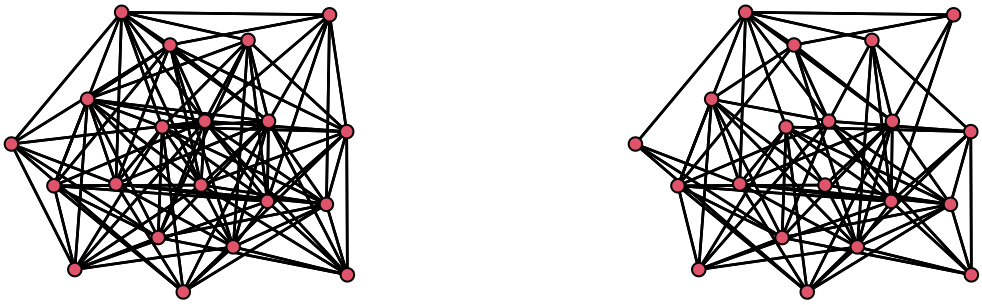
Missing completely at random

```
# let's do an MCAR example for comparison
missThresh = matrix( data = runif(20^2, min = 0, max = 1),
                     nrow = 20,
                     ncol = 20 )

## NOTE: for an undirected network, this needs to be symmetric so
# overwrite the upper triangle with the lower triangle
missThresh[upper.tri(missThresh)] = t(missThresh)[upper.tri(missThresh)]

# set a proportion of missing entries, 0.2 would imply that 20% of all tie variables are missing.
propMiss = 0.2

# index which tie variables are missing
```

```
missTieVars = missThresh <= propMiss

# degrade the network
degradedNet = testNet
degradedNet[missTieVars] = NA

# check how many are missing (should be ~propMiss * n^2)
sum(is.na(degradedNet))
```

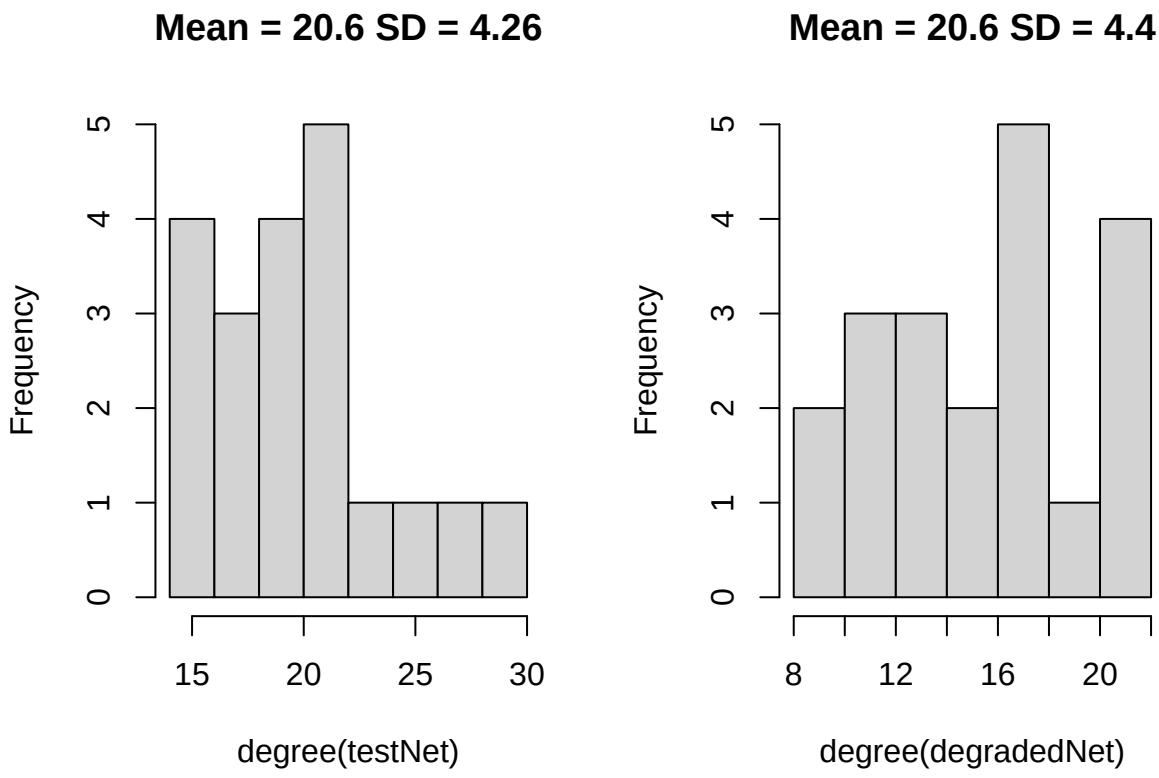
```
## [1] 83
```

```
# diagonal of 0
diag(degradedNet) = 0

# plot side by side
par(mfrow = c(1,2))
coord = gplot(testNet, xlab = "Toy net", gmode = "graph")
gplot(degradedNet, xlab = "Degraded net", gmode = "graph", coord = coord)
```



Toy net Degraded net

```
# degree distributions
hist(degree(testNet), main = paste("Mean = ",mean(degree(testNet)), " SD = ", round(sd(degree(testNet))
hist(degree(degradedNet),main = paste("Mean = ",mean(degree(testNet)), " SD = ", round(sd(degree(degrad
```



What's going on from the principles side of things?

This is basically saying:

$$\mathbf{D}|\mathbf{X} \sim Bern(\theta)$$

With θ set to 0.2 in the example.

So, given the missing data model, we can then clarify how this is MCAR,

$$\Pr(\mathbf{D}|\mathbf{X}, \theta) = \Pr(\mathbf{D}|\theta) \text{ for all entries X and parameters } \theta.$$

So, the values in $\mathbf{X}$ have absolutely no effect on how likely a tie variable is in being missing.

Therefore, the data are missing completely at random (MCAR).

Missing not at random

Toy example: more central would imply more likely to be missing. From a covert network perspective, this might be for security reasons. Another perspective would be that because this particular figure is so central, nobody nominates them because the nominators implicitly assumed this central figure to already be acknowledged.

If we start by explaining the model,

$$\Pr(\mathbf{D}_{ij} = 1) = logistic(\alpha + \beta(\mathbf{X}_{i+} + \mathbf{X}_{+j}))$$

Note that $\mathbf{X}_{i+} + \mathbf{X}_{+j}$ is meant to reflect degree (centrality). For the undirected case, this should be equivalent to $2 \times \mathbf{X}_{i+}$ or $2 \times \mathbf{X}_{+j}$.

This needs to be set up in a way such that the higher the degree, the more likely for the tie to be missing.

```
# how to start..

# make up some values for alpha and beta
alpha = -5  # an intercept value,

beta = 0.17 # this beta value corresponds to the weight of the variable when assigning the probability

# grab degrees
degCentr = degree(testNet)

# let's figure out standardising this later...
# maxDegree = 20*19
#
# # normalise it with the maximum possible degree so scale is less of an issue (i.e., a form of standar
# normDegCentr = degCentr/maxDegree
#

# finalgle the probability such that the mean is ~0.2

mean(1/(1 + exp(-alpha - (beta * degCentr))))
```

```
## [1] 0.2042103
```

```
# list out the probabilities
missTieProb = 1/(1 + exp(-alpha - (beta * degCentr)))
```

Why did I pick these alpha and beta values?

I started with trying to get an average missing tie variable probability of ~0.2 (similar to the MCAR example),

$$\frac{\sum_{i=1..n} \frac{1}{1+\exp(-\alpha - \mathbf{X}_i\beta)}}{n} = 0.2,$$

where $\mathbf{X}_i$ here represents the (unnormalised) degree centrality.

Also, working backwards,

$$1 + \exp(-\alpha - \mathbf{X}_i\beta) = 5,$$

$$\exp(-\alpha - \mathbf{X}_i\beta) = 4,$$

Let's use the mean degree in place of each individual to make life easier,

$$\exp(-\alpha - \frac{\sum_{i=1..n} \mathbf{X}_i}{n}\beta) = 4.$$

Let's use the selected random graph mean degree so $\frac{\sum_{i=1..n} \mathbf{X}_i}{n} = 20.6$

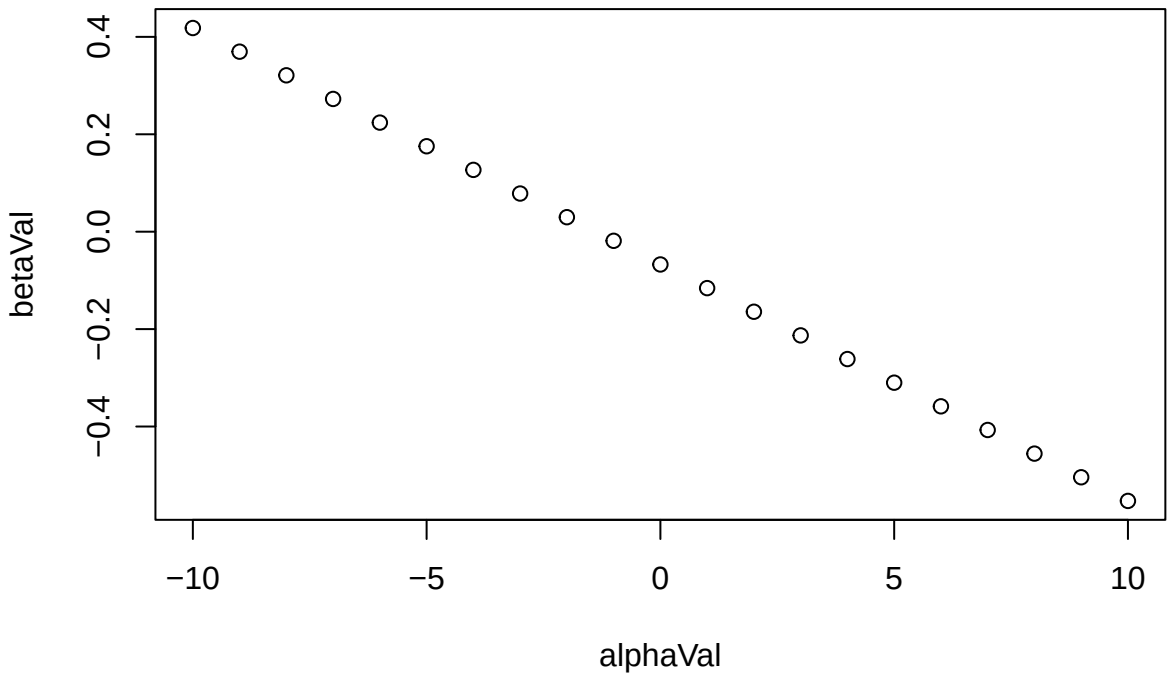
$$\exp(-\alpha - 20.6\beta) = 4,$$

$$-\alpha - 20.6\beta = \log(4),$$

$$\beta = \frac{-\log(4) - \alpha}{20.6}.$$

Let's graph that...

```
# make up some alpha values
alphaVal = seq(from = -10, to = 10, by = 1)

betaVal = (-log(4) - alphaVal)/20.6

plot(x = alphaVal, y = betaVal)
```



Any pair of values here can be used, but we want β to have certain properties,

- The absolute value of beta describes how much weight the variable in question affects the missing probability,
- The sign of beta describes the relationship between the variable in question and the missing probability.

Our variable in question here is degree centrality, we want it to have a pronounced effect, but not *too* extreme. We also want beta's sign to be positive to reflect a positive linear relationship between degree centrality and the probability of a missing tie variable.

Hence we ended up with:

$$\alpha = -5\beta = 0.17$$

Now, apply that to the network degradation,

```
# another seed because randomness.
set.seed(132)

# similar to the previous depletion,
# we have a set of model-predicted missing tie probabilities

# let's first make a matrix of missing data indicators with the probabilities we have
# the rbinom function's just a convenient way of doing this, also each actor technically has an implici
missDatIndicator = t(replicate(20, rbinom(length(missTieProb), size = 1, prob = missTieProb)))

# check how many were indicated to be missing
sum(missDatIndicator)
```

```
## [1] 85
```

```
# make some adjustments because undirected network
missDatIndicator[upper.tri(missDatIndicator)] = t(missDatIndicator)[upper.tri(missDatIndicator)]
diag(missDatIndicator) = 0
isSymmetric(missDatIndicator)
```

```
## [1] TRUE
```

```
# make a table to compare,
compareTab = data.frame(missTieProb = round(missTieProb, digits = 2), IndicatorProbabilities = colMeans
compareTab$difference = compareTab[,1] - compareTab[,2]
print(compareTab)
```

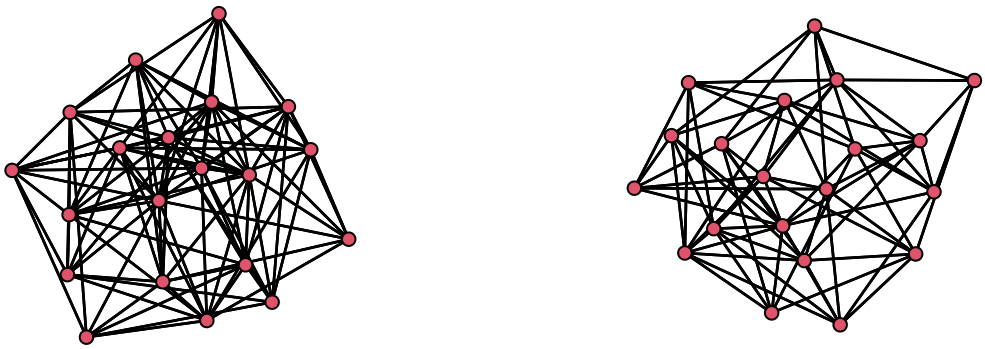
```
##    missTieProb IndicatorProbabilities difference
## 1         0.07                   0.15      -0.08
## 2         0.17                   0.20      -0.03
## 3         0.22                   0.35      -0.13
## 4         0.22                   0.25      -0.03
## 5         0.09                   0.20      -0.11
## 6         0.22                   0.20       0.02
## 7         0.09                   0.20      -0.11
## 8         0.17                   0.15       0.02
## 9         0.13                   0.30      -0.17
## 10        0.17                   0.10       0.07
## 11        0.44                   0.35       0.09
## 12        0.28                   0.15       0.13
## 13        0.13                   0.10       0.03
## 14        0.22                   0.35      -0.13
```

```
## 15        0.13                    0.15      -0.02
## 16        0.52                    0.30       0.22
## 17        0.22                    0.25      -0.03
## 18        0.07                    0.05       0.02
## 19        0.36                    0.20       0.16
## 20        0.17                    0.20      -0.03
```

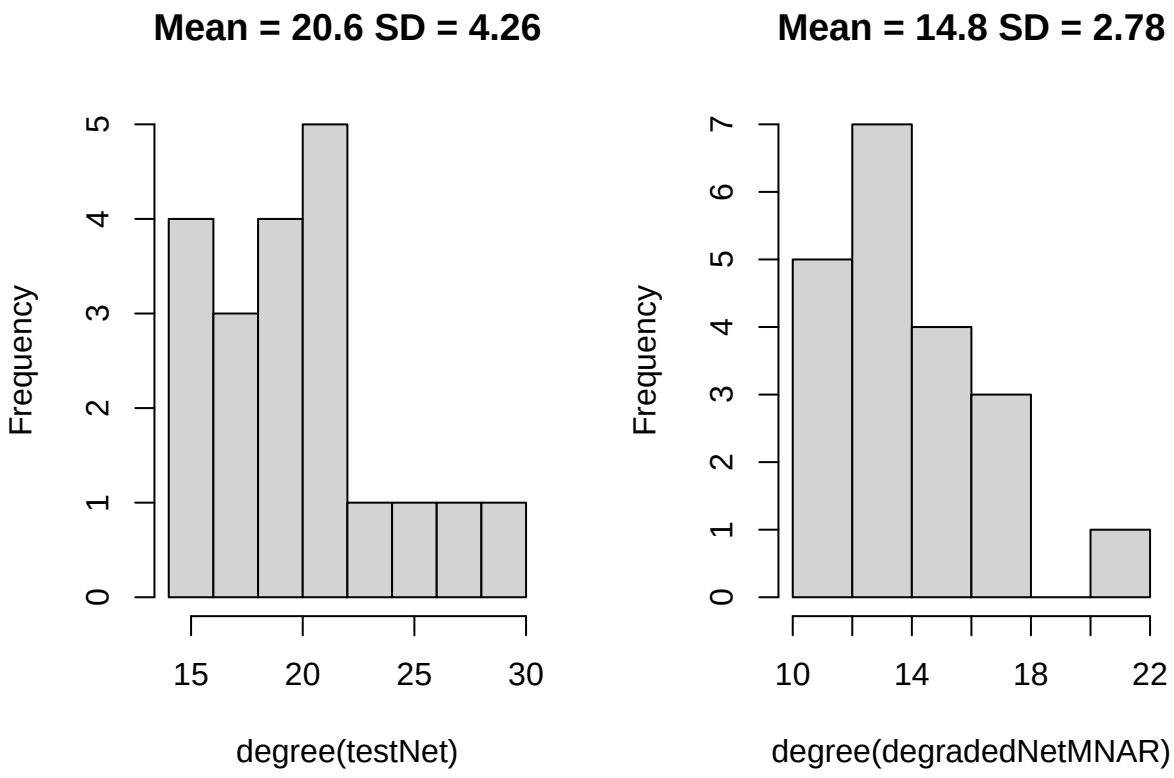
```
# check total amount of missing ties after the adjmat adjustments, should be ~80 (20^2 * 0.2), maybe a
sum(missDatIndicator)
```

```
## [1] 84
```

```
# and we deplete
degradedNetMNAR = testNet
degradedNetMNAR[missDatIndicator==1] = NA

# plot side by side
par(mfrow = c(1,2))
gplot(testNet, xlab = "Toy net", gmode = "graph")
gplot(degradedNetMNAR, xlab = "MNAR degraded net", gmode = "graph")
```




```
# it's quite hard to see, but if we see the average degree centrality and their histograms,
hist(degree(testNet), main = paste("Mean = ",mean(degree(testNet)), " SD = ", round(sd(degree(testNet))
hist(degree(degradedNetMNAR), main = paste("Mean = ",mean(degree(degradedNetMNAR)), " SD = ", round(sd(
```




```
# the mean degree centrality may have decreased more? Hard to tell...
```

What's going on with the MNAR depletion?

First, we use a binomial distribution to generate missing tie variables for each actor with the probability of a missing tie variable corresponding to what we found out before with the missing data model.

So really,

$$\mathbf{D}|\mathbf{X}_i \sim Bi(n, p_i),$$

where the index i here refers to each actor with their own missing tie variable probability which we modelled earlier (or the entire row),

$$\Pr(\mathbf{D}|\mathbf{X}_i) = logistic(\alpha + \beta \times (2 \times \mathbf{X}_{i+})),$$

which can also be written as:

$$p_i = logistic(\alpha + \beta \times (2 \times \mathbf{X}_{i+}).$$

Also reminder that $2 \times \mathbf{X}_{i+}$ here just reflects an actor's degree centralisation in this undirected network.

With that in mind, I made some manual adjustments so the adjacency matrix makes sense (only 0s in the diagonal, symmetric for an undirected network), and we now have a depleted network with a missing not at random pattern of missingness.

But why is it missing not at random and not the other two?

As seen in the missing data model,

$$\Pr(\mathbf{D}|\mathbf{X}_i) = logistic(\alpha + \beta \times (2 \times \mathbf{X}_{i+})),$$

The fact that $\mathbf{X}$ is there at all rules out MCAR.

The reason why this missing data model is not MAR is because of $\mathbf{X}_{i+}$ since this sum would need both the observed and depleted tie variables,

$$\mathbf{X}_{i+} = \mathbf{X}_{\mathbf{obs}i+} + \mathbf{X}_{\mathbf{mis}i+},$$

so conceptually,

$$\Pr(\mathbf{D}|\mathbf{X}, p) = \Pr(\mathbf{D}|\mathbf{X}_{obs}, \mathbf{X}_{mis}, p) \text{ for all X and } p,$$

Which both rules out MAR and would then imply MNAR.

In plain language terms, as we know from the missing data model, the probability of a missing tie variable depends on degree centrality, which can only be calculated if we have all the actors' ties. If we were to use the depleted network's degree centralisation to model the missing data indicator, we would be missing the degrees that were depleted, which would result in a different set of missing tie variable probabilities than the one we found from the missing data model for the full network.

Turning into a function

Another way to think of the depleting process above is the function below,

$$f(\mathbf{X}, \alpha, \beta) = \mathbf{D}.$$

In this case, the inputs are the adjacency matrix $\mathbf{X}$ and model parameters α and β while the output is the missing data indicator $\mathbf{D}$.

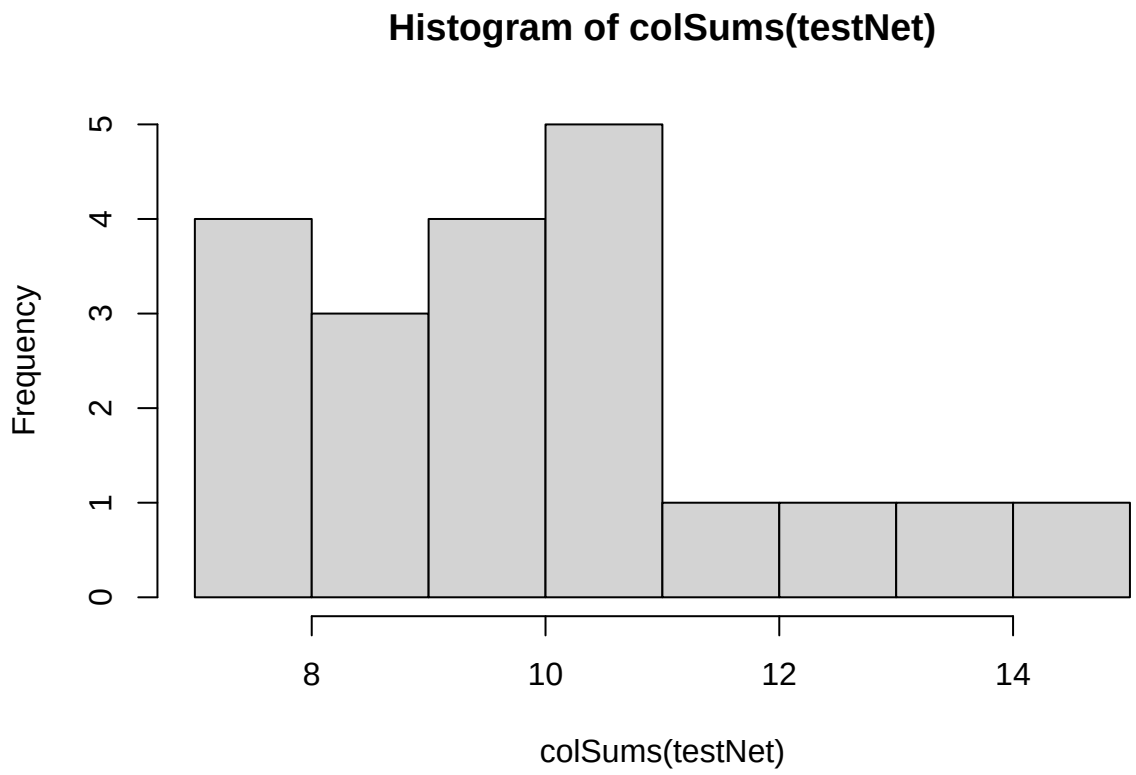
```
# fixing seed to check output
set.seed(132)

## making a function to degrade a net with respect to a logistic model for a binomial dist per actor

## 20220411 update: expanded to include multiple coefficients

degradeNetLogis = function(net, alpha, beta, netStats, directed = TRUE){

  ## degradeNetLogis(net, alpha, beta, directed) takes a graph and replaces the tie variables
  ## with missing values (NA). The mechanism here specifically uses degree (centrality)
  ## to weigh the probability of a missing tie variable. The function can probably be
  ## improved to accommodate different network statistics. It also currently only uses one
  ## variable to weigh the probability of a missing tie variable, but can (and probably will)
  ## be expanded to use more than one variable (and slope).
  ##
  ## Input:
  ## - net:      An adjacency matrix describing the fully observed graph including all tie variables.
  ##             This matrix shouldn't have any missing values.
```

```
  ## - alpha:    The single intercept value for the logistic model.
  ## - beta:     The slope parameter in the logistic model. Its absolute value describes the weight of
  ##             the related variable to the probability of a missing tie variable. Its sign refers to
  ##             relationship between the related variable and the probability of a missing tie variabl
  ##             Can be a p-long vector when the probability of a missing tie variable depends on more
  ##             aspect of the network.
  ## - netStats: An n x p dataframe of network values for each actor. Index-wise, the first column shou
  ##             match the first value in 'beta', the second column the second value in beta, so on.
  ## - directed: Logical value to indicate whether the network is a directed or undirected network.
  ##             Default is set to directed.
  ##
  ## Output:
  ## - A matrix of missing data indicators (i.e., matrix 'D').

  ## spit out an error if the directed argument is misspecified
  if(directed != TRUE & !isSymmetric(net)){
    stop("The undirected network doesn't have a symmetric matrix")
  }

  ## spit out an error if there are not an equal amount of slope weights and network structures
  if(length(beta) != ncol(netStats)){
    stop("The parameter vector has a different length to the network structures")
  }

  ## Get the number of nodes
  n = nrow(net)

  ## Compute the linear combination
  # this line just multiples each column of beta with the matching column in netStats and sums each row
  linComb = rowSums(mapply(FUN = '*', as.data.frame(netStats), beta))

  ## Make the missing tie probabilities for each actor
  missTieProb = 1/(1 + exp(-alpha - (linComb)))

  ## Make the missing tie indicator matrix
  missDatIndicator = t(replicate(n, rbinom(n, size = 1, prob = missTieProb)))

  ## symmetrise if undirected
  if(directed == FALSE){
    missDatIndicator[upper.tri(missDatIndicator)] = t(missDatIndicator)[upper.tri(missDatIndicator)]
  }

  ## Turn diagonal to 0 because adjmat
  diag(missDatIndicator) = 0

  return(missDatIndicator)
}
```

An example with degree and indegree

```
# fix seed to match
set.seed(132)
```

```r
# check the indegree of the test network
hist(colSums(testNet))
```



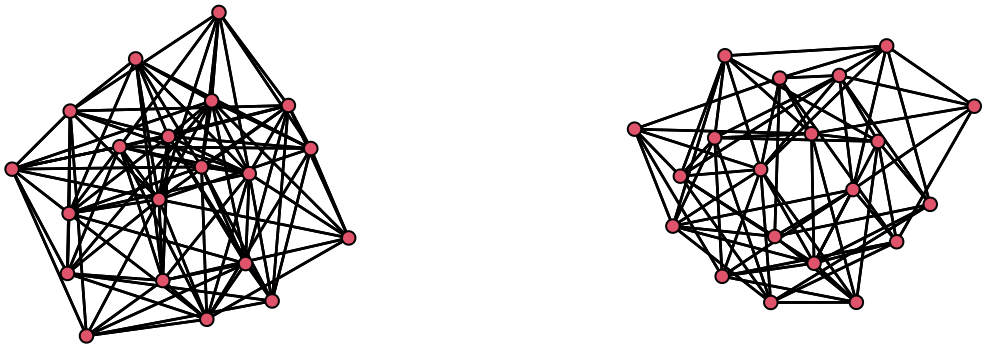

```r
mean(colSums(testNet))
```

```
## [1] 10.3
```

```r
# choosing some parameter values
alpha = -5
beta = c(0.05, 0.23)
netStats = data.frame(degree = degree(testNet), indeg = colSums(testNet))

# make missing data indicator
missDatInd = degradeNetLogis(net = testNet,
                             alpha = alpha,
                             beta = beta,
                             netStats = netStats,
                             directed = FALSE)

# check total amount of missing ties after the adjmat adjustments, should be ~80 (20^2 * 0.2), maybe a
sum(missDatInd)
```

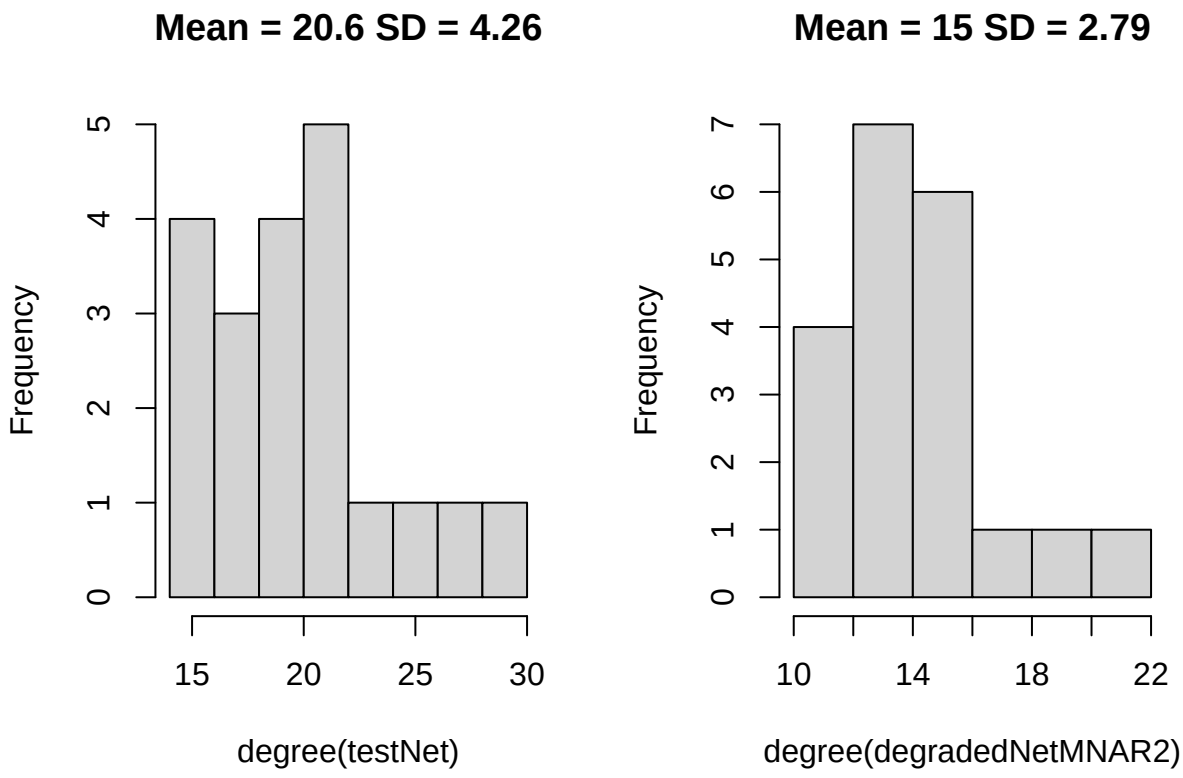
```
## [1] 82
```

```r
# and we deplete
degradedNetMNAR2 = testNet
degradedNetMNAR2[missDatInd==1] = NA

# plot side by side
par(mfrow = c(1,2))
gplot(testNet, xlab = "Toy net", gmode = "graph")
gplot(degradedNetMNAR2, xlab = "MNAR degraded net", gmode = "graph")
```



Toy net MNAR degraded net

```r
# it's quite hard to see, but if we see the average degree centrality and their histograms,
hist(degree(testNet), main = paste("Mean = ",mean(degree(testNet)), " SD = ", round(sd(degree(testNet))
hist(degree(degradedNetMNAR2), main = paste("Mean = ",mean(degree(degradedNetMNAR2)), " SD = ", round(s
```



The model above can be thought of as:

$$\Pr(\mathbf{D}_{ij} = 1) = logistic(\alpha + \beta_1(\mathbf{X}_{i+} + \mathbf{X}_{+j}) + \beta_2(\mathbf{X}_{+j})),$$

and similar to the example before, if we wanted ~20% missingness,

$$\frac{\sum_{i=1..n} \frac{1}{1+\exp(-\alpha-\beta_1(\mathbf{X}_{i+}+\mathbf{X}_{+j})-\beta_2(\mathbf{X}_{+j}))}}{n} = 0.2.$$

After filling in the values with the test network (testNet), where the mean degree is 20.6 (above), and the mean indegree is $\overline{\mathbf{X}}_{j+} = 10.3$

$$\exp(-\alpha - 20.6\beta_1 - 10.3\beta_2) = 4,$$

$$-\alpha - 20.6\beta_1 - 10.3\beta_2 = \log(4),$$

Don't exactly know how to solve this, I just kept guessing numbers to get a mean of ~0.2.

The model itself isn't particularly intuitive either, the higher the degree (which counts indegree, so it's counted twice) and the higher the indegree, the more likely a tie variable is to be missing. But the important(?) part is that I can put multiple things in the model and weigh certain aspects of the network more than other aspects (e.g., I might think indegree matters more than the overall degree)

what can we do from here on?

Two sides:

the modelling front (i.e., wrt to methods)

The modelling front can be improved and extended in a lot of ways... Programming-wise, sensitivity plots where parameters scale to missing data proportions would be nice (i.e., a fn to spit out parameter values for a set proportion of missingness would be useful). Undirected networks also need to acknowledge both ends (i.e., the β model that's just a slightly modified Bernoulli model where we may conceptualise it as the probability of a sender sending a tie with the probability the receiver receives the tie.). We could always use more complicated models, but I believe it's a better idea to neaten our conceptual front before we make things more complex for the sake of being complex.

Maybe there's something to do in the modelling front to disentangle the missing data mechanism (**D**) with whatever decision-making-mechanism the network analyst is doing (similar to the NSF false positive/miss rates?). This may be a meta perspective if we were to account for differences in the way analysts approach the collected resources to represent the network- that is that there may be different ways for the 'archaeological' approach to be done.

Would it be useful (or possible) to simulate how a covert network grows over time from some semantically coded 'data' (e.g., A is suspect, B is connected to A by coarrests, C is A's non-illicit friend in legitimate businesses), similar to how Pete made that gif of a network growing over time, and then introducing missing data via a pre-specified mechanism? This somehow sounds doable in my head, That being said, we're still talking about unknown unknowns when we model MNAR missing data mechanisms, so in terms of 'how useful is this?', I don't really have a clear answer.

Multiplexity, if we were to acknowledge covert networks as another type of social network, they would have their own social dynamics in it and so we may be able to represent valenced ties (e.g., position in the hierarchy perhaps since actual data asking the covert actors how they feel about other actors would be extremely unfeasible). Heterophily may be another aspect to consider here in terms of roles in the covert network. I feel like there's a lot of value in specifying types of ties/relationships in elaborating particular network properties, but I also think that this might grow to be very unfeasible in a ridiculous pace if we wanted to *really* describe multipartite nodes and/or multiplex ties. Also, proximity (or physical distance). How do we deal with that? Using some dodgy surveillance/data collection sources to identify meetings? Maybe ethnographical measures are more sensible than heavy computation semantic analysis solutions here?

the conceptual front (i.e., wrt to plausibility of the mechanisms)

We can consider 'popular' psych effects that replicate-ish to justify the plausibility of some missing data mechanisms (e.g., the 'bystander' effect). Another think to think about is sampling frames for investigators/detectives. How would two people approach and analyse a covert network differently? Given n observers, perhaps the observers systematically 'sample' the network as a fn of 'visibility' or whatever. This would again be similar to having a model for missing data indicator D that depends on characteristics of the network. However, the final observed network(s) would be a product of many models for sampling and actual missing mechanisms... Is some kind of triangulation a benefit of multiple network analysts' perspectives?

What do I need to brainstorm this? I'd love to know how 'data' is 'collected'. What are the nuances that come up when we discuss missing data mechanisms? Let alone from a data analyst's perspective, but from the observed network's perspective as well. How important are brokers? Spies? High-social-capital-actors? 'Extreme' actors? This might just result in a 'betweenness' argument, but maybe something to look into is the brokerage aspects in a conceptual front? I mean, assuming that the actors will act in a way to optimise network efficiency with security, the 'ideal' position probably is being sparsely connected to high-connected nodes. Sparsely connected to decrease visibility, but connected to well-connected nodes so that this particular node has a lot of 'reach'. In that sense, would geodesic distance (or just generally path lengths) be a useful metric? Or inverse average geodesic ('efficiency'). I don't really think centralities are the be-all-end-all in identifying potentially important actors in a network. In simple and ideal cases sure, but when the network

gets complicated (and maybe if the 'authority' is sparsely connected), centrality's usefulness goes down real fast. Do we need abstract algebras for identifying roles now, or is this something we can leave to data??

When acknowledging that the archaelogical approaches to 'representing' a network may differ, how exactly can they differ? Some analysts may have resources that emphasise some nodes more than the other. There's also the time element since investigations are cumulative efforts (i.e., only more information can be gained through the passage of time). It would be neat if we had *something* to describe how a network evolves w.r.t. the progress of an investigation (e.g., perhaps how prime suspects change through time and how this matches up with whatever evidence is recorded). Of course, one other thing is the politics (?) of the situation. I'm assuming resources are limited, so investigations may be ushered to 'wrap up' and prosecute the prime suspect due to various pressures (e.g., political, economical, etc.).

How does the 'investigation' network grow? Is it via a snowball mechanism? Surely informants play a role here to connect potential covert actors... If I were to imagine I were an analyst part of an investigation collating information, I would simply just represent what the most common threads in the evidence are if I were to 'construct' a network. I (the network analyst) feel like I would have no agency on which possible ties I'd want to investigate since it would be somewhat data-driven. Is it a collaborative effort with more than one analyst, all of whom have different sources of information? Or is it usually done under a central figure commanding the analysis like a head analyst?

Can the missing data mechanism 'progress' over time as the investigation proceeds as a response to the investigation? Some kind of defensive mechanism for the covert network to reorganise itself in response to the investigation? In that sense, would I need to acknowledge the missing data mechanism to be a temporal process or from an 'omniscient observer' perspective where I assume I know the entire covert network? How would I even acknowledge how a missing data mechanism can grow? The missingness just gets snowballed?

What about using qualitative aspects of ties? If we knew that certain actors in the covert network share antagonistic ties, would they be more reliable informants? Given the scattered aspect of the 'raw data', multiplexity may be a boon in disguise if we wanted to identify important actors or potential informants for further information. Point is, in the most basic sense, 'ties' can mean different things to different actors and I don't see why this would be different for covert networks (i.e., a covert network is still a network with its own dynamics).

Do covert networks 'grow' in the same way a non-covert network does in a snowball sense?

How do we go about identifying 'extreme actors' (i.e., 'outliers' who are incredibly influential in the network) beyond centrality? What kind of structures would we want to try see? Perhaps path-length measures, alternating n stars (though this would just be a degree centrality...)?