EXAM

Knitted from EXAM_CA.Rmd

2024-06-07

Part 1 (Ana Alina)

First load package and csv file All variables should be factors.

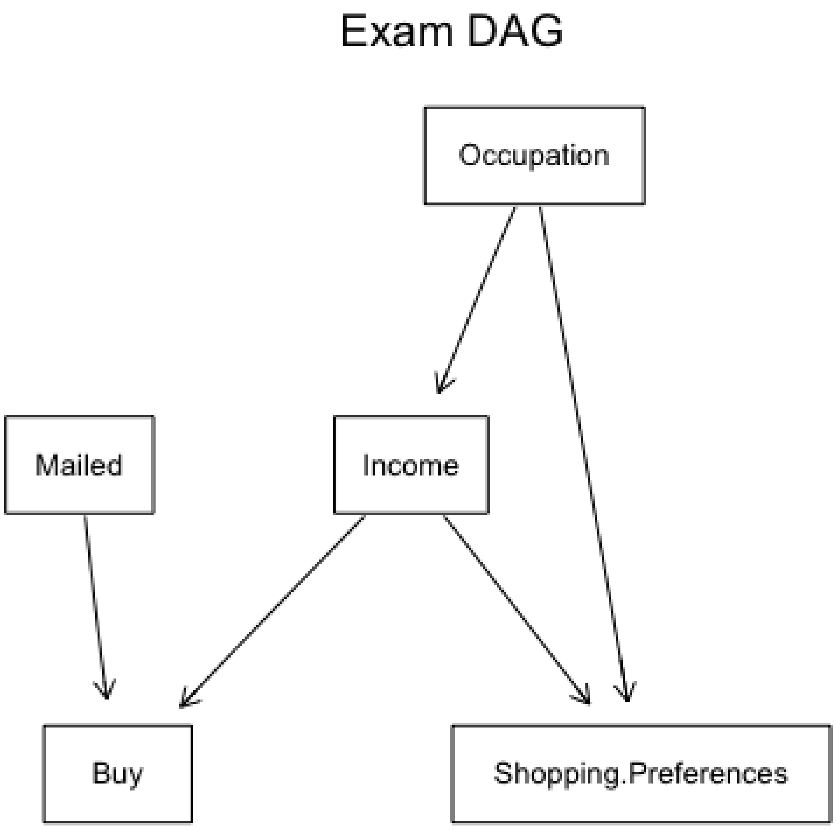
```
data <- read.csv("Data/dataCA2024.csv", sep = ";", stringsAsFactors = TRUE)
str(data)
```

```
## 'data.frame':    4249 obs. of  5 variables:
##  $ Buy                 : Factor w/ 2 levels "No","Yes": 2 2 1 2 2 1 1 2 2
2 ...
##  $ Income              : Factor w/ 9 levels "100k-119k","120k-139k",..: 8
7 6 9 1 6 6 8 4 3 ...
##  $ Mailed              : Factor w/ 2 levels "No","Yes": 1 1 1 2 1 1 2 2 1
2 ...
##  $ Occupation          : Factor w/ 2 levels "Blue-Collar",..: 1 1 1 2 2 1
1 1 2 1 ...
##  $ Shopping.Preferences: Factor w/ 2 levels "In-store","Online": 1 1 1 2 2
1 1 1 2 2 ...
```

Build network and add direction to arcs

```
bn.gs <- gs(data, alpha = 0.05, test ="x2")

# Set direction from Occupation to Income
bn.gs1 <- set.arc(bn.gs, from = "Occupation", to = "Income")

# Set direction from Income to Shopping.Preferences
bn.gs1 <- set.arc(bn.gs1, from = "Income", to = "Shopping.Preferences")

# Set direction from Occupation to Shopping.Preferences
bn.gs1 <- set.arc(bn.gs1, from = "Occupation", to = "Shopping.Preferences")

# Plot network
graphviz.plot(bn.gs1, main = "Exam DAG")
```

```
## Loading required namespace: Rgraphviz
```



Use Maximum Likelihood to fit parameters

```r
bn.mle <- bn.fit(bn.gs1, data = data, method = "mle")
```

D-separation

When looking at the plot we see no arrows between e.g. “Mailed” and “Income”. This means they are independent, insofar we do not set any evidence. However, if we suddenly set evidence for “Buy” the two becomes dependent.

```r
dsep(bn.mle, x = "Mailed", y = "Income") # Returns TRUE -> Independence

## [1] TRUE

dsep(bn.mle, x = "Mailed", y = "Income", z = "Buy") # Returns FALSE -> 
Dependence

## [1] FALSE
```

We can also use d-separation to investigate conditional indepencence (the example above could be called conditional dependence). For example, does a change in “Occupation” affect “Buy” if we already know “Income”?

First, we can inspect if there is dependence between “Occupation” and “Buy” without setting evidence.

```
dsep(bn.mle, x = "Occupation", y = "Buy") # Returns FALSE

## [1] FALSE
```

This means there is dependence without setting evidence. But let us control for the value of "Income":

```
dsep(bn.mle, x = "Occupation", y = "Buy", z = "Income") # Returns TRUE

## [1] TRUE
```

Suddenly, we have independence. How so? The concept is very important to DAGs, and quite intuitive. First, let's inspect levels of the "Occupation" factor.

```
levels(data$Occupation)

## [1] "Blue-Collar"  "White-Collar"
```

The person can either be Blue-Collar or White-Collar.

Based on the learnt DAG, there is a relation between this variable and the probability of actually buying the product.

However, the effect from Occupation is indirect. Switching from Blue to White yields an increase in Income, which is what directly increases the probability.

So, knowing the income of the person, we can "disregard" their occupation.

Arc strength

We can measure this in different ways. I will look first at significance:

```
# options(scipen=999)
arc.strength (bn.gs1, data = data, criterion = "x2") %>%.[order(.$strength),]

##          from                     to      strength
## 2     Income Shopping.Preferences  0.000000e+00
## 3     Mailed                    Buy  0.000000e+00
## 1     Income                    Buy 2.770147e-267
## 4 Occupation                 Income 1.887394e-266
## 5 Occupation Shopping.Preferences  1.991713e-02
```

All relations are significant. It means that the probability of each "target node" is very conditional on the "source node". So for example, the probability of having Shopping.Preferences == "Online" is very conditional on the persons Income.

Now I look at the effect on BIC of removing an arc:

```
arc.strength(bn.gs1, data = data, criterion = "bic") %>%.[order(.$strength),]

##          from                     to    strength
## 2     Income Shopping.Preferences -1005.77420
## 3     Mailed                    Buy  -915.95433
## 4 Occupation                Income  -821.09356
## 1     Income                   Buy  -641.52869
## 5 Occupation Shopping.Preferences    27.79382
```

Here, we could be inclined to remove the arc between “Occupation” and “Shopping.Preferences” since, accoding to this function, that would improve BIC wit 27.79.

K-fold CV

```
netcv = bn.cv(data = data, bn.gs1, loss ="pred", k = 5, loss.args =
list(target = "Buy"), debug = FALSE)
netcv

##
##   k-fold cross-validation for Bayesian networks
##
##   target network structure:
##    [Mailed][Occupation][Income|Occupation][Buy|Income:Mailed]
##    [Shopping.Preferences|Income:Occupation]
##   number of folds:                       5
##   loss function:                         Classification Error
##   training node:                         Buy
##   expected loss:                         0.1812191
```

We see expected loss = 0.18 meaning an Accuracy of 1 - 0.18 = 0.82. In other words, this network is able to correctly predict the level of Buy 82% of the time.

New evidence

We simply SET the value of one/more of the nodes, and inspect how the probabilities of the other nodes react to this.

For example, if we expect a high-income person to always BUY, and we set the evidence that a new person is high-income, then we know they will buy. This is an example of forward inference, which I will carry out in this exercise.

However, we can also perform backwards inference. Continuing on the example above, if we instead set the evidence BUY = NO, then we know the person is not high-income. Obviously, in the model the probabilities are not 0% or 100%, but I think you get my point.

```
# levels(data$Shopping.Preferences)

junction <- compile(as.grain(bn.mle))

## Warning in from.bn.fit.to.grain(x): NaN conditional probabilities in
## Shopping.Preferences, replaced with a uniform distribution.

# See probability of customer buying in general
querygrain(junction, nodes = "Buy")

## $Buy
## Buy
##         No       Yes
## 0.5381609 0.4618391

# Set Shopping.Preferences == "In-store"
InBlue <- setEvidence(junction, nodes = c("Shopping.Preferences",
"Occupation"), states = c("In-store", "Blue-Collar"))

# See probability of customer buying given the preferences
Buy <- querygrain(InBlue, nodes = "Buy")
Buy

## $Buy
## Buy
##         No       Yes
## 0.6610307 0.3389693
```

We see the probability of a customer buying the product decreasing from approx. 46% to approx. 33% when setting this evidence. We might interpret this as following: When a person is blue-collar, their income is low. Furthermore, in-store prices are higher (I assume). Hence, this combination leads to a lower probability of buying the product.

Expected Lift in Profit

```
# levels(data$Income)

c <- 0.5
r_s <- 8
r_u <- 10

PopulationYes <- setEvidence(junction,
                             nodes = c("Shopping.Preferences", "Occupation",
"Income", "Mailed"),
                             states = c("In-store", "Blue-Collar", "100k-
119k", "Yes"))

PopulationNo <- setEvidence(junction,
                             nodes = c("Shopping.Preferences", "Occupation",
"Income", "Mailed"),
                             states = c("In-store", "Blue-Collar", "100k-
119k", "No"))

YesProb <- querygrain(PopulationYes, nodes = "Buy")$Buy[[2]]
NoProb <- querygrain(PopulationNo, nodes = "Buy")$Buy[[2]]

ELP = YesProb * r_s - NoProb * r_u - c
ELP

## [1] 5.218989
```

ELP = Expected Lift in Profit. Since ELP is positive it is a good idea to target this segment with an ad campaign.

The results show that we expect a lift in profit of 5 dollars per targeted customer. This result occurs because of the high (conditional) probability of buying when being contacted via mail. This is evident from a YesProb value of 0.97 and a NoProb value of 0.2. This increased probability of buying the product when contacted via email outweighs the cost associated with the mail (0.5 dollars) and also the lower return from solicited customers (8 vs 10).

Part 2 (Morten)

Cronbach's Alpha

So the statement in pl_1 is: "I dislike this product". Compare this to the other statements for this construct, e.g. "The product is appealing to me". And let us consider a respondent who rates the product highly.

The respondent would then have a low rating for pl_1 and a high rating for the others. This is intuitively not what we want, since a lot of summary statistics would be misleading (e.g. calcualting averages).

It also has an impact on Cronbach's alpha. Cronbach's alpha is in essence a ratio between two important quantities: 1) The sum of variances of the indicators (numerator) 2) The variance of the sum of the indicators (denominator)

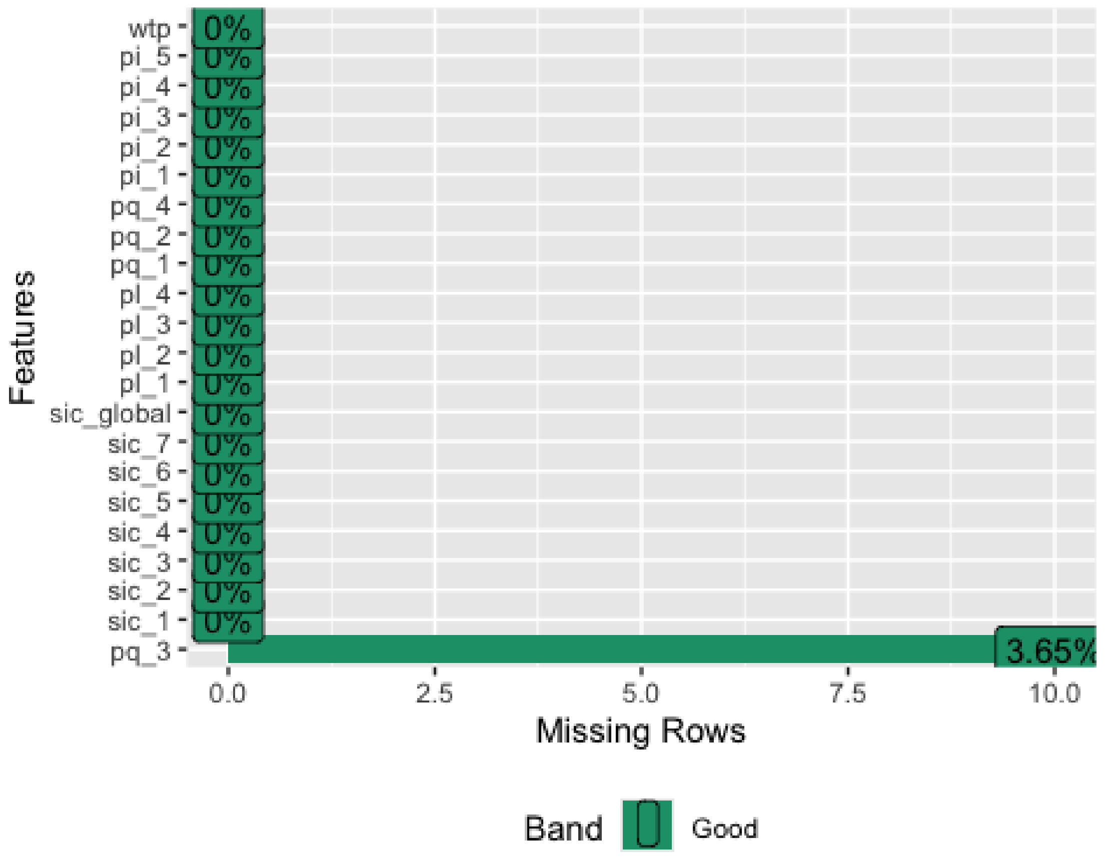
If the indicators have high (positive) correlation, they will never "cancel each other out", leading to a very high denominator. Since we calculate "1 - ratio" this would lead to a high Alpha -> High reliability.

But if we use the original scores of pl_1 the denominator would be smaller, indicating a lower reliability without there necessarily being a lower reliability.

Hence, we reverse the scoring.

Missing Data

```r
pls_data <- read.csv("Data/PLS_data_exam.csv")
pls_data_na <- read.csv("Data/PLS_data_exam.csv", na.strings = "-999")

DataExplorer::plot_missing(pls_data_na)
```



We can see that only pq_3 has missing values.

```
table(pls_data$pq_3)

##
## -999    1    2    3    4    5    6    7
##   10   35   33   31   34   33   34   64
```

It has 10 missing observations.

Reflective Measurement Model

```
influencer_mm <- constructs(
  composite("SIC", multi_items("sic_", 1:7), weights = mode_B),
  composite("PL", multi_items("pl_", 1:4), weights = mode_A),
  composite("PQ", multi_items("pq_", 1:4), weights = mode_A),
  composite("PI", multi_items("pi_", 1:5), weights = mode_A),
  composite("WTP", single_item("wtp")))

# Create structural model
influencer_sm <- relationships(
  paths(from = c("SIC", "PQ", "PL"), to = c("PI")),
  paths(from = c("SIC"), to = c("PL")),
  paths(from = c("SIC"), to = c("PQ")),
  paths(from = c("PI"), to = c("WTP")))

# Estimate the model
sem_model <- estimate_pls(data = pls_data,
                          measurement_model = influencer_mm,
                          structural_model = influencer_sm,
                          missing = mean_replacement,
                          missing_value = "-999")

## Generating the seminr model

## All 274 observations are valid.

summary_model <- summary(sem_model)
```

Report

I implement the PLS-SEM model in accordance with the illustration provided in the exam document.

This means I have a model with 5 constructs, if we consider WTP as a single-item construct. The constructs are:

- Self-Influencer Connection (SIC)
- Perceived Quality (PQ)
- Product Liking (PL)
- Purchase Intention (PI)
- Willingness To Pay (WTP)

All constructs except SIC are endogenous, as they are all being explained by at least one other construct.

I use “Mode A” for PQ, PL and PI since this is synonymous with a reflective measurement model. For SIC I use “Mode B” since it is formatively measured. For the reflectively measured constructs the arrows point from the construct to the indicators. In other words, the indicators reflect the construct. In such a case, we expect high correlation between indicators, which also is part of the evaluation schema.

After having estimated the model we can look into the results for the REFLECTIVE part.

Loadings

```
summary_model$loadings

##         SIC    PQ    PL    PI    WTP
## sic_1 0.334 0.000 0.000 0.000  0.000
## sic_2 0.186 0.000 0.000 0.000  0.000
## sic_3 0.284 0.000 0.000 0.000 -0.000
## sic_4 0.570 0.000 0.000 0.000  0.000
## sic_5 0.277 0.000 0.000 0.000  0.000
## sic_6 0.533 0.000 0.000 0.000  0.000
## sic_7 0.429 0.000 0.000 0.000 -0.000
## pl_1  0.000 0.000 0.849 0.000  0.000
## pl_2  0.000 0.000 0.823 0.000  0.000
## pl_3  0.000 0.000 0.808 0.000  0.000
## pl_4  0.000 0.000 0.842 0.000  0.000
## pq_1  0.000 0.842 0.000 0.000  0.000
## pq_2  0.000 0.811 0.000 0.000  0.000
## pq_3  0.000 0.681 0.000 0.000 -0.000
## pq_4  0.000 0.881 0.000 0.000  0.000
## pi_1  0.000 0.000 0.000 0.270  0.000
## pi_2  0.000 0.000 0.000 0.714  0.000
## pi_3  0.000 0.000 0.000 0.826  0.000
## pi_4  0.000 0.000 0.000 0.875  0.000
## pi_5  0.000 0.000 0.000 0.854  0.000
## wtp   0.000 0.000 0.000 0.000  1.000
```

Looking at the loadings we see no cross loading at all, which is to be expected (it is a paradigmatic model).

For reflectively measured constructs, each indicator should load onto its construct with at least 0.708. Evaluating the loadings using this threshold shows one potential issue:

- pi_1 with a loading on 0.273

Construct Reliability (Cronbach's Alpha)

```
summary_model$reliability

##     alpha  rhoC   AVE  rhoA
## SIC 0.077 0.536 0.157 1.000
## PQ  0.818 0.881 0.652 0.828
## PL  0.850 0.899 0.690 0.855
## PI  0.767 0.848 0.552 0.846
## WTP 1.000 1.000 1.000 1.000
##
## Alpha, rhoC, and rhoA should exceed 0.7 while AVE should exceed 0.5
```

Here, we see a very low alpha for SIC. This is not unexpected since we do not expect the items to correlate too much (it is formatively measured). The other alphas are fine, as we typically say they should exceed 0.7.

Convergent Validity

```
summary_model$reliability

##     alpha  rhoC   AVE  rhoA
## SIC 0.077 0.536 0.157 1.000
## PQ  0.818 0.881 0.652 0.828
## PL  0.850 0.899 0.690 0.855
## PI  0.767 0.848 0.552 0.846
## WTP 1.000 1.000 1.000 1.000
```

```
##
## Alpha, rhoC, and rhoA should exceed 0.7 while AVE should exceed 0.5
```

Same code, but now looking at the Average Variance Extracted (AVE) column. For one indicator, we calculate the amount of variance in that indicator explained by the construct by squaring the loading. The AVE is a construct-level measure where we average the squared loadings across the indicators of a construct. As expected, the AVE is low for SIC. The others are fine (rule of thumb: AVE > 0.5).

Discriminant Validity

```
summary_model$validity$fl_criteria
```

```
##       SIC    PQ    PL    PI   WTP
## SIC 0.396     .     .     .     .
## PQ  0.340 0.807     .     .     .
## PL  0.442 0.255 0.831     .     .
## PI  0.418 0.288 0.550 0.743     .
## WTP 0.146 0.026 0.195 0.303 1.000
##
## FL Criteria table reports square root of AVE on the diagonal and construct
correlations on the lower triangle.
```

The criterion here is that for a given construct sqrt(AVE) must be higher than correlations.

- PQ: sqrt(AVE) = 0.807, higher than any correlations.
- PL: sqrt(AVE) = 0.831, higher than any correlations.
- PI: sqrt(AVE) = 0.743, higher than any correlations.

So according to the FL Criteria there is discriminant validity.

Refinement

The only thing I noticed was the low loading of pi_1 so I try to remove it and run the model again.

```
influencer_mm_2 <- constructs(
  composite("SIC", multi_items("sic_", 1:7), weights = mode_B),
  composite("PL", multi_items("pl_", 1:4), weights = mode_A),
  composite("PQ", multi_items("pq_", 1:4), weights = mode_A),
  composite("PI", multi_items("pi_", 2:5), weights = mode_A),
  composite("WTP", single_item("wtp")))

# Create structural model
influencer_sm_2 <- relationships(
  paths(from = c("SIC", "PQ", "PL"), to = c("PI")),
  paths(from = c("SIC"), to = c("PL")),
  paths(from = c("SIC"), to = c("PQ")),
  paths(from = c("PI"), to = c("WTP")))
```

```
# Estimate the model
sem_model_2 <- estimate_pls(data = pls_data,
                            measurement_model = influencer_mm_2,
                            structural_model = influencer_sm_2,
                            missing = mean_replacement,
                            missing_value = "-999")
```

```
## Generating the seminr model

## All 274 observations are valid.
```

```
summary_model_2 <- summary(sem_model_2)

summary_model_2$reliability
```

```
##     alpha  rhoC   AVE  rhoA
## SIC 0.077 0.536 0.156 1.000
## PQ  0.818 0.881 0.652 0.829
## PL  0.850 0.899 0.690 0.855
## PI  0.841 0.894 0.679 0.856
## WTP 1.000 1.000 1.000 1.000
##
## Alpha, rhoC, and rhoA should exceed 0.7 while AVE should exceed 0.5
```

I notice a higher Alpha for PI now that I have removed the indicator pi_1. This could tell us this is a better measurement model, since the indicators forming PI now are more correlated.

AVE is obviously higher, since they are very dependent on the loadings, and I just removed a low loading.

I will conclude the PLS-SEM part here.

Distance measures

```
Var1 <- c(5,6,15,16,25)
Var2 <- c(5,6,13,15,20)
df <- data.frame(Var1 = Var1, Var2 = Var2)
dist <- dist(df, method="euclidean")
dist
```

```
##           1         2         3         4
## 2  1.414214
## 3 12.806248 11.401754
## 4 14.866069 13.453624  2.236068
## 5 25.000000 23.600847 12.206556 10.295630
```

Step 1

Above we see distances between observations. In hierarchical clustering, the observations with the lowest distance are joined first. So when inspecting the matrix, we see the lowest distance between Observation 1 and Observation 2. They are joined first.

Step 2

Now that 1 and 2 are joined, we have 4 “objects” instead of 5. One of the objects consist of 1 and 2 joined.

This introduces ambiguity in how to join. How do we measure the distance between this “joined object” and the other objects? The complete linkage method evaluates all possible distances and bases its approach on the maximum distance.

I am not sure how to do this programatically tho. I guess we can first make the matrix with Obs 1 and then with Obs 2 and find the maximum distances.

```r
obs1 <- df[c(1,3,4,5), ]
obs2 <- df[c(2,3,4,5), ]

obs1_dist <- dist(obs1, method = "euclidian")
obs2_dist <- dist(obs2, method = "euclidian")

obs1_dist

##           1         3         4
## 3 12.806248
## 4 14.866069  2.236068
## 5 25.000000 12.206556 10.295630

obs2_dist

##           2         3         4
## 3 11.401754
## 4 13.453624  2.236068
## 5 23.600847 12.206556 10.295630
```

So we can just use the distance matrix above and see that the distance from (1&2) to the others:

- (1&2) to 3 = 12.806248
- (1&2) to 4 = 14.866069
- (1&2) to 5 = 25.000000

Since 3 & 4 are closest, they are joined next.

Step 3

Now we would have to do the same again. There must be some function for this though :D

```r
obs3 <- df[c(1,2,3,5), ]
obs4 <- df[c(1,2,4,5), ]
obs3_dist <- dist(obs3, method = "euclidian")
obs4_dist <- dist(obs4, method = "euclidian")
```

```
obs1_dist

##           1         3         4
## 3 12.806248                    
## 4 14.866069  2.236068          
## 5 25.000000 12.206556 10.295630

obs2_dist

##           2         3         4
## 3 11.401754                    
## 4 13.453624  2.236068          
## 5 23.600847 12.206556 10.295630

obs3_dist

##           1         2         3
## 2  1.414214                    
## 3 12.806248 11.401754          
## 5 25.000000 23.600847 12.206556

obs4_dist

##           1         2         4
## 2  1.414214                    
## 4 14.866069 13.453624          
## 5 25.000000 23.600847 10.295630
```

Distances:

- From (1&2) to (3&4) = 14.866069
- From (1&2) to 5 = 25.000000
- From (3&4) to 5 = 12.206556

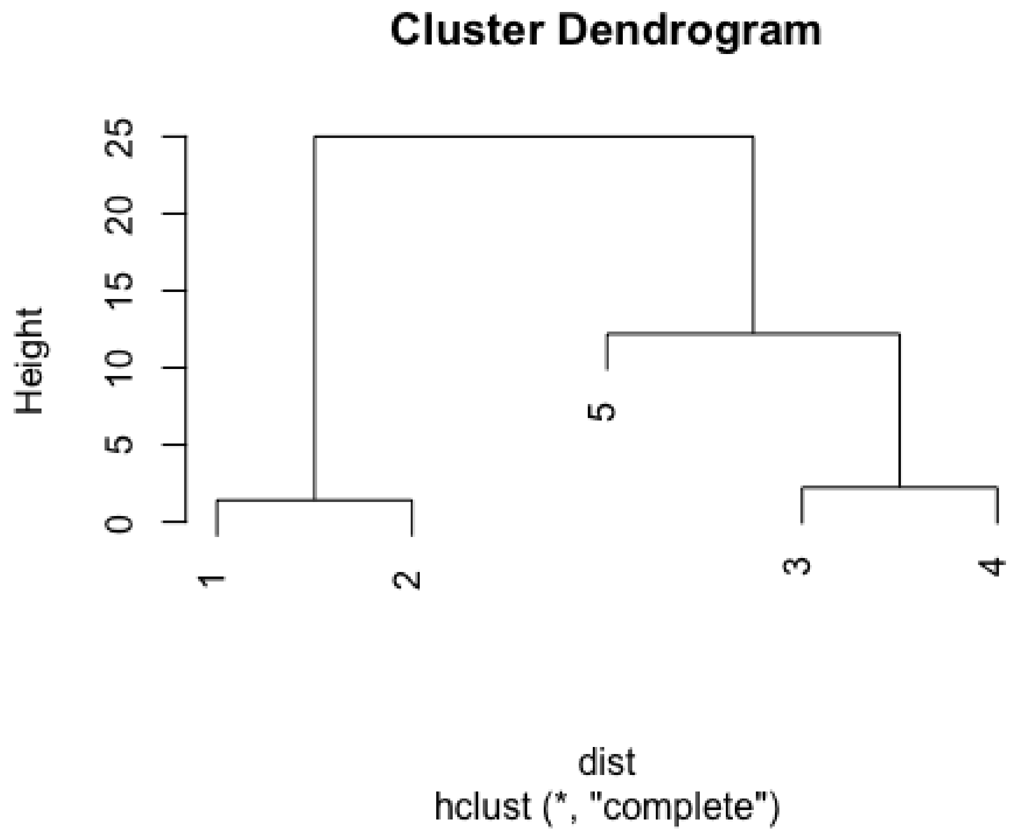
Hence, the next to be joined are (3&4) and 5.

Step 4

Last step. We only have two objects left, so they are joined. In other words, we join (1&2) and (3&4&5)

Using hclust()

```
hclust_fit <- hclust(dist, method = "complete")
plot(hclust_fit)
```



I can see the steps here.

1) Join 1 & 2
2) Join 3 & 4
3) Join (3 & 4) with 5
4) Join (1 & 2) with (3 & 4 & 5)