EXAM

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

```
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.

```
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
## 2      Income Shopping.Preferences
## 3      Mailed                   Buy
## 1      Income                   Buy
## 4 Occupation                Income
## 5 Occupation Shopping.Preferences
##
strength
## 2
0.000000000000000000000000000000000000000000000000000000000000000000000000000
00000000000000000000000000000000000000000000000000000000000000000000000000000
00000000000000000000000000000000000000000000000000000000000000000000000000000
00000000000000000000000000000000000000000000
## 3
0.000000000000000000000000000000000000000000000000000000000000000000000000000
00000000000000000000000000000000000000000000000000000000000000000000000000000
00000000000000000000000000000000000000000000000000000000000000000000000000000
00000000000000000000000000000000000000000000
## 1
0.000000000000000000000000000000000000000000000000000000000000000000000000000
00000000000000000000000000000000000000000000000000000000000000000000000000000
```

```
00000000000000000000000000000000000000000000000000000000000000000000000000000
00000000000000000000000000000000000002770147
## 4
0.000000000000000000000000000000000000000000000000000000000000000000000000000
00000000000000000000000000000000000000000000000000000000000000000000000000000
00000000000000000000000000000000000000000000000000000000000000000000000000000
000000000000000000000000000000000018873937
## 5
0.019917129174560656290626781128594302572309970855712890625000000000000000000
00000000000000000000000000000000000000000000000000000000000000000000000000000
00000000000000000000000000000000000000000000000000000000000000000000000000000
00000000000000000000000000000000000000000000
```

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 = TRUE)
```

```
## * splitting 4249 data in 5 subsets.
## ----------------------------------------------------------------
## * fitting the parameters of the network from the training sample.
## * applying the loss function to the data from the test sample.
##   > classification error for node Buy is 0.1752941 .
##   @ total loss is 0.1752941 .
## ----------------------------------------------------------------
## * fitting the parameters of the network from the training sample.
## * applying the loss function to the data from the test sample.
##   > classification error for node Buy is 0.1847059 .
##   @ total loss is 0.1847059 .
## ----------------------------------------------------------------
## * fitting the parameters of the network from the training sample.
## * applying the loss function to the data from the test sample.
##   > classification error for node Buy is 0.1764706 .
##   @ total loss is 0.1764706 .
```

```
## ----------------------------------------------------------------
## * fitting the parameters of the network from the training sample.
## * applying the loss function to the data from the test sample.
##   > classification error for node Buy is 0.1835294 .
##   @ total loss is 0.1835294 .
## ----------------------------------------------------------------
## * fitting the parameters of the network from the training sample.
## * applying the loss function to the data from the test sample.
##   > classification error for node Buy is 0.1861013 .
##   @ total loss is 0.1861013 .
## ----------------------------------------------------------------
## * summary of the observed values for the loss function:
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
##  0.1753  0.1765  0.1835  0.1812  0.1847  0.1861

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.

```
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.

levels(data$Shopping.Preferences)

## [1] "In-store" "Online"

# 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
```

```r
Buy <- querygrain(InBlue, nodes = "Buy")
Buy
```

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

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

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

We see the probability of a customer buying the product increasing from approx. 54% to approx. 66% when setting this evidence.

Expected Lift in Profit

```r
# 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
```

Since ELP is positive it is a good idea to target this segment with an ad campaign.

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 - ration" this would lead to a high Alpha -> High reliability.

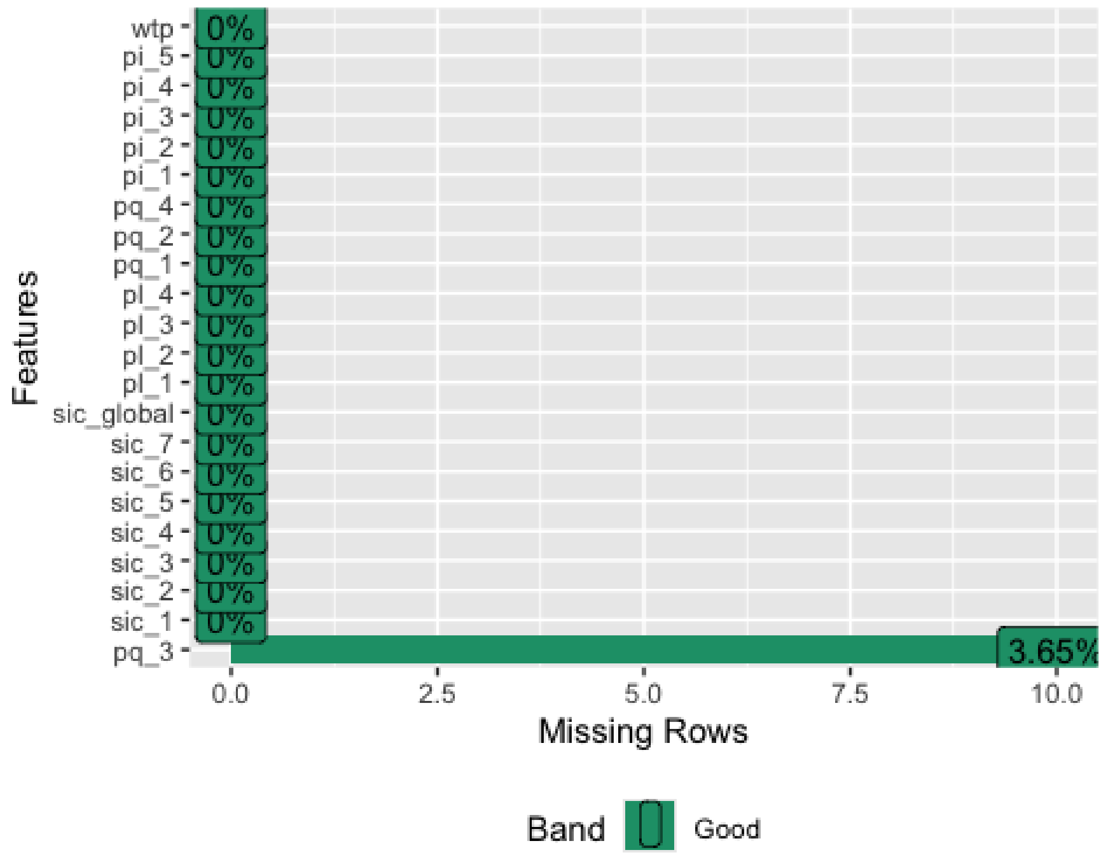
But if we use the original scores of pl_1 the denominator would be smaller, indicating a lower reliability without there necessarily being a lowe reliability.

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

```
str(pls_data)

## 'data.frame':    274 obs. of  22 variables:
##  $ sic_1     : int  1 5 7 1 6 6 3 3 3 2 ...
##  $ sic_2     : int  7 3 7 7 5 6 2 6 2 7 ...
##  $ sic_3     : int  7 7 1 2 1 3 6 7 2 1 ...
##  $ sic_4     : int  2 7 7 7 4 3 2 5 4 5 ...
##  $ sic_5     : int  7 3 4 4 4 2 4 2 1 7 ...
##  $ sic_6     : int  3 7 6 5 1 4 7 4 7 2 ...
##  $ sic_7     : int  7 7 7 7 4 4 3 3 4 1 ...
##  $ sic_global: int  5 6 7 6 2 4 7 1 3 3 ...
##  $ pl_1      : int  2 5 7 6 6 7 5 4 1 5 ...
##  $ pl_2      : int  1 7 6 7 2 7 2 5 1 7 ...
```

```
##  $ pl_3       : int  1 7 7 4 3 7 1 6 2 7 ...
##  $ pl_4       : int  4 7 7 7 6 5 1 4 1 6 ...
##  $ pq_1       : int  2 7 4 7 1 4 6 1 4 2 ...
##  $ pq_2       : int  3 5 4 7 2 4 5 1 7 1 ...
##  $ pq_3       : int  4 3 2 7 2 5 7 3 6 1 ...
##  $ pq_4       : int  1 5 4 7 1 5 4 1 6 1 ...
##  $ pi_1       : int  3 7 3 2 5 2 5 7 7 6 ...
##  $ pi_2       : int  4 7 7 5 2 5 3 7 1 3 ...
##  $ pi_3       : int  2 7 7 5 3 7 7 5 5 7 ...
##  $ pi_4       : int  1 7 7 5 4 7 7 6 2 6 ...
##  $ pi_5       : int  2 7 7 5 6 7 6 6 3 7 ...
##  $ wtp        : int  6 7 6 4 5 5 6 5 7 7 ...

influencer_mm <- constructs(
  composite("SIC", multi_items("sic_", 1:7), weights = mode_A),
  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 - Product Liking - Purchase Intention - Willingness To Pay

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

I use “Mode A” since this is synonymous with a reflective measurement model. This means 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.

Loadings

```
summary_model$loadings
```

```
##         SIC    PQ    PL    PI    WTP
## sic_1 0.280 0.000 0.000 0.000  0.000
## sic_2 0.149 0.000 0.000 0.000  0.000
## sic_3 0.314 0.000 0.000 0.000 -0.000
## sic_4 0.576 0.000 0.000 0.000  0.000
## sic_5 0.318 0.000 0.000 0.000  0.000
## sic_6 0.562 0.000 0.000 0.000  0.000
## sic_7 0.408 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 0.843 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 0.880 0.000 0.000  0.000
## pi_1  0.000 0.000 0.000 0.273  0.000
## pi_2  0.000 0.000 0.000 0.713  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 0.853  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). Each indicator should load onto its construct with at least 0.708. Evaluating the loadings using this threshold gives us a lot of potential issues: - sic_1 with a loading on 0.280 - sic_2 with a loading on 0.149 - sic_5 with a loading on 0.318 - sic_7 with a loading on 0.408 - 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 0.159 0.120
## PQ  0.818 0.881 0.652 0.827
## 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 after having evaluated the loadings. 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 0.159 0.120
## PQ  0.818 0.881 0.652 0.827
## 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 too 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 0.399     .     .     .     .
## PQ  0.340 0.807     .     .     .
## PL  0.442 0.255 0.831     .     .
## PI  0.413 0.287 0.550 0.743     .
## WTP 0.143 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.
```

Rerun the model

I think it is already clear that the model is not very good.