MesuermentDev

Teodor Chakarov

2022-05-21

Survey and Measurrement Development in R

Preparing to analyse survey data

```
library(psych)
library(dplyr)
```

```
##
## Attaching package: 'dplyr'

## The following objects are masked from 'package:stats':
##
##     filter, lag

## The following objects are masked from 'package:base':
##
##     intersect, setdiff, setequal, union
```

```
library(likert)
```

```
## Loading required package: ggplot2

##
## Attaching package: 'ggplot2'

## The following objects are masked from 'package:psych':
##
##     %+%, alpha

## Loading required package: xtable

##
## Attaching package: 'likert'

## The following object is masked from 'package:dplyr':
##
##     recode
```

```r
sme <- read.csv("brandloyalty.csv")

# Print beginning of sme data frame
head(sme)
```

```
##   BL1 BL2 BL3 BL4 BL5 BL6 BL7 BL8 BL9 BL10
## 1   5   4   4   4   4   4   3   4   3    3
## 2   4   3   4   3   3   3   3   1   1    1
## 3   5   3   5   3   3   4   2   2   2    3
## 4   4   4   2   2   2   2   2   4   2    2
## 5   4   5   4   4   4   4   3   4   4    4
## 6   4   4   4   4   3   4   2   2   2    2
```

```r
# Correlation matrix of expert ratings
cor(sme)
```

```
##            BL1       BL2       BL3       BL4       BL5       BL6       BL7
## BL1  1.0000000 0.4383758 0.5013256 0.3500438 0.3750830 0.3979041 0.1512149
## BL2  0.4383758 1.0000000 0.5247358 0.3639021 0.4151766 0.3679184 0.1919387
## BL3  0.5013256 0.5247358 1.0000000 0.3842278 0.4855423 0.4481031 0.2751482
## BL4  0.3500438 0.3639021 0.3842278 1.0000000 0.4956118 0.4725727 0.2876030
## BL5  0.3750830 0.4151766 0.4855423 0.4956118 1.0000000 0.6251928 0.3855511
## BL6  0.3979041 0.3679184 0.4481031 0.4725727 0.6251928 1.0000000 0.3635232
## BL7  0.1512149 0.1919387 0.2751482 0.2876030 0.3855511 0.3635232 1.0000000
## BL8  0.2158089 0.2852414 0.2664924 0.3430974 0.3724939 0.4017230 0.5045515
## BL9  0.2223461 0.2711436 0.2982184 0.2994033 0.3851179 0.3900507 0.6355096
## BL10 0.1636942 0.1951917 0.1993999 0.1378767 0.2517390 0.2772448 0.4792299
##            BL8       BL9      BL10
## BL1  0.2158089 0.2223461 0.1636942
## BL2  0.2852414 0.2711436 0.1951917
## BL3  0.2664924 0.2982184 0.1993999
## BL4  0.3430974 0.2994033 0.1378767
## BL5  0.3724939 0.3851179 0.2517390
## BL6  0.4017230 0.3900507 0.2772448
## BL7  0.5045515 0.6355096 0.4792299
## BL8  1.0000000 0.6487062 0.4613553
## BL9  0.6487062 1.0000000 0.6022968
## BL10 0.4613553 0.6022968 1.0000000
```

```r
# Percentage agreement of experts
#irr::agree(sme)
```

```r
# Load psych package
library(psych)

# Check inter-rater reliability
psych::cohen.kappa(sme)
```

```
## Warning in cohen.kappa1(x1, w = w, n.obs = n.obs, alpha = alpha, levels =
## levels): upper or lower confidence interval exceed abs(1) and set to +/- 1.
```

```
## Warning in psych::cohen.kappa(sme): No variance detected in cells 6 5
```

```
## Warning in psych::cohen.kappa(sme): No variance detected in cells 7 1

## Warning in psych::cohen.kappa(sme): No variance detected in cells 7 2

## Warning in psych::cohen.kappa(sme): No variance detected in cells 9 1

## Warning in psych::cohen.kappa(sme): No variance detected in cells 9 2

## Warning in psych::cohen.kappa(sme): No variance detected in cells 9 7

## Warning in psych::cohen.kappa(sme): No variance detected in cells 9 8

## Warning in psych::cohen.kappa(sme): No variance detected in cells 10 9

## At least one item had no variance.  Try describe(your.data) to find the problem.

##
## Cohen Kappa (below the diagonal) and Weighted Kappa (above the diagonal)
## For confidence intervals and detail print with all=TRUE
##        BL1     BL2   BL3  BL4  BL5  BL6   BL7  BL8   BL9  BL10
## BL1  1.000  0.4337 0.485 0.22 0.26 0.32 0.064 0.11 0.093 0.080
## BL2  0.204  1.0000 0.510 0.24 0.29 0.30 0.084 0.15 0.118 0.098
## BL3  0.266  0.2609 1.000 0.29 0.39 0.41 0.145 0.17 0.156 0.121
## BL4  0.063  0.0290 0.076 1.00 0.49 0.45 0.245 0.32 0.257 0.129
## BL5  0.065  0.0931 0.167 0.26 1.00 0.61 0.317 0.34 0.317 0.228
## BL6  0.141  0.0916 0.141 0.22 0.33 1.00 0.267 0.34 0.285 0.227
## BL7  0.011  0.0360 0.047 0.11 0.16 0.16 1.000 0.49 0.635 0.470
## BL8  0.062  0.0531 0.057 0.11 0.18 0.16 0.271 1.00 0.629 0.458
## BL9  0.026 -0.0033 0.041 0.14 0.14 0.11 0.431 0.44 1.000 0.592
## BL10 0.049  0.0210 0.070 0.12 0.14 0.14 0.236 0.26 0.371 1.000
##
## Average Cohen kappa for all raters  0.15
## Average weighted kappa for all raters  0.3
```

```
# Calculate the CVR for each unique item in the data frame
#cvr_by_item <- lawshe %>%
#  group_by(item) %>%
#  summarize(CVR = CVratio(NTOTAL = length(unique(expert)),
#                          NESSENTIAL = sum(rating == 'Essential')))

# See the results
#cvr_by_item
```

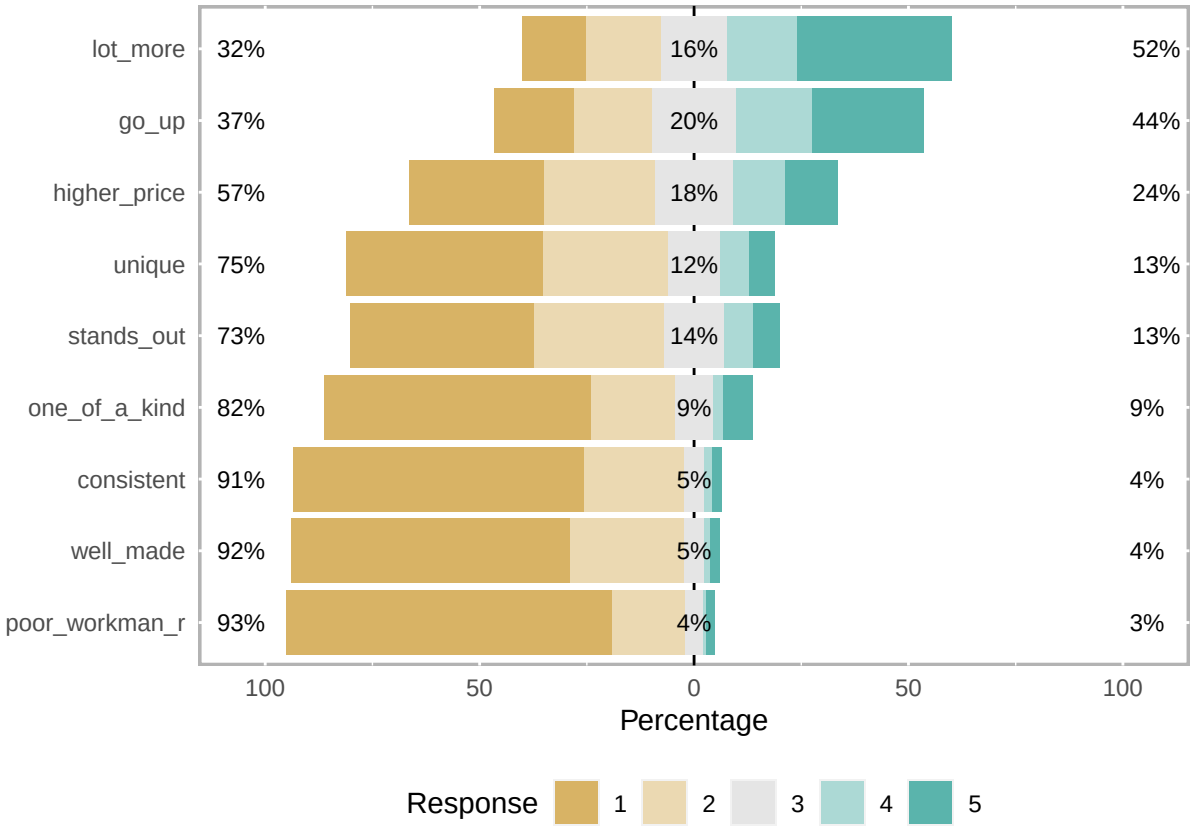
```
brand_rep <- read.csv("brandrep-cleansurvey-extraitem.csv")

# Convert items to factor
b_rep_likert <- brand_rep %>%
                mutate_if(is.integer, as.factor)

# Response frequencies - base R
summary(b_rep_likert)
```

```
##  well_made consistent poor_workman_r higher_price lot_more go_up   stands_out
##  1:363     1:379      1:424          1:175        1: 83    1:103   1:239
##  2:149     2:131      2: 95          2:145        2: 97    2:102   2:170
##  3: 27     3: 26      3: 25          3:103        3: 87    3:110   3: 78
##  4:  8     4: 11      4:  4          4: 67        4: 91    4: 99   4: 38
##  5: 12     5: 12      5: 11          5: 69        5:201    5:145   5: 34
##  unique  one_of_a_kind
##  1:256   1:348
##  2:164   2:109
##  3: 67   3: 50
##  4: 39   4: 13
##  5: 33   5: 39
```

```
# Plot response frequencies
result <- likert(b_rep_likert)
plot(result)
```




```
brand_qual <- read.csv("brandquall11-recodedbutextraitem.csv")

# Get response frequencies from psych
response.frequencies(brand_qual)
```

```
##                    1         2          3           4          5 miss
## trendy        0.8484252 0.1122047 0.01968504 0.007874016 0.01181102    0
## latest        0.8110236 0.1397638 0.01574803 0.013779528 0.01968504    0
```

```
## tired_r        0.8051181 0.1338583 0.03149606 0.015748031 0.01377953    0
## happy_pay      0.6811024 0.2539370 0.03149606 0.011811024 0.02165354    0
## reason_price   0.6968504 0.2244094 0.04133858 0.013779528 0.02362205    0
## good_deal      0.7322835 0.1968504 0.04921260 0.000000000 0.02165354    0
## strong_perform 0.3307087 0.2480315 0.14566929 0.127952756 0.14763780    0
## leader         0.1889764 0.1476378 0.10826772 0.185039370 0.37007874    0
## serious        0.2381890 0.1830709 0.15748031 0.141732283 0.27952756    0
## innovator      0.2204724 0.2263780 0.18700787 0.175196850 0.19094488    0
```

```
# Print item descriptions
brand_qual_items <- read.csv("branddesc.csv")
brand_qual_items
```

```
##                                                                 X1
## 1                                                                2
## 2                                                                3
## 3                                                                4
## 4                                                                5
## 5                                                                6
## 6                                                                7
## 7                                                                8
## 8 9serious = This brand takes its product quality seriously.
##                                          trendy...This.brand.is.trendy.
## 1 latest = This brand offers the latest products.
## 2 tired = This is a tired brand.
## 3 happy_pay = I am happy paying what I do for this brand's products.
## 4 reason_price = This brand's products are reasonably priced.
## 5 good_deal = This brand's products are a good deal.
## 6 strong_perform = This brand's products are strong performers.
## 7 leader = This brand is a leader in its field.
## 8
```

```
# Reverse code the "opposite" item
brand_qual$tired_r <- recode(brand_qual$tired,
                             "1 = 5; 2 = 4; 4 = 2; 5 = 1")
```

```
## Error in recode(brand_qual$tired, "1 = 5; 2 = 4; 4 = 2; 5 = 1"): argument "to" is missing, with no d
```

```
# Check recoding frequencies
brand_qual %>%
    select(tired, tired_r) %>%
    response.frequencies() %>%
    round(2)
```

```
## Error in `select()`:
## ! Can't subset columns that don't exist.
## x Column `tired` doesn't exist.
```

```
library(Hmisc)
```

```
## Loading required package: lattice
```

```
## Loading required package: survival

## Loading required package: Formula

##
## Attaching package: 'Hmisc'

## The following objects are masked from 'package:xtable':
##
##     label, label<-

## The following objects are masked from 'package:dplyr':
##
##     src, summarize

## The following object is masked from 'package:psych':
##
##     describe

## The following objects are masked from 'package:base':
##
##     format.pval, units
```

```
missing_lots <- read.csv("brandquall11-recodedbutextraitem.csv")
# Total number of rows
nrow(missing_lots)
```

```
## [1] 508
```

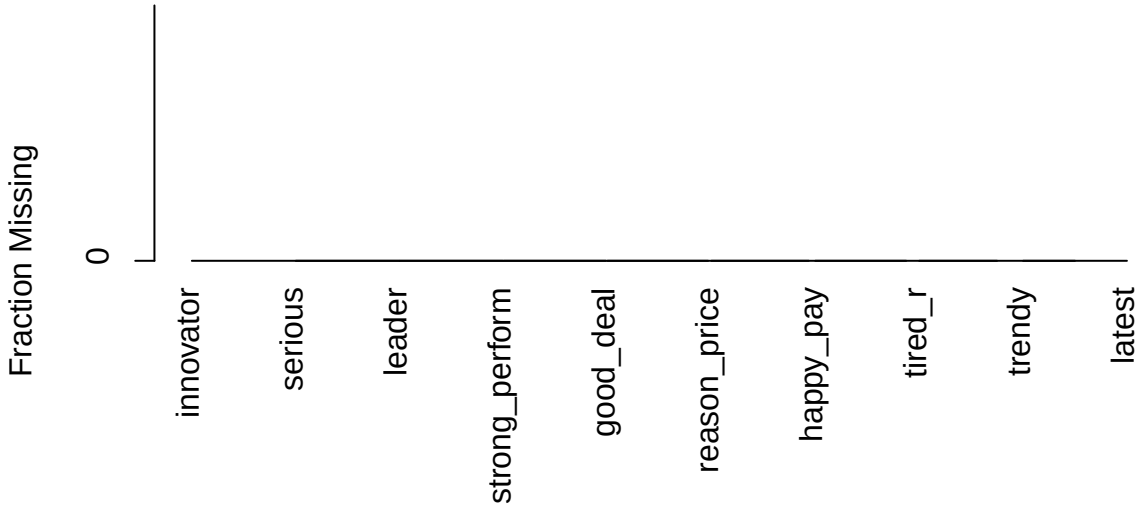
```
# Total number of complete cases
nrow(na.omit(missing_lots))
```

```
## [1] 508
```

```
# Number of incomplete cases by variable
colSums(is.na(missing_lots))
```

```
##         trendy         latest        tired_r      happy_pay   reason_price
##              0              0              0              0              0
##      good_deal strong_perform         leader        serious      innovator
##              0              0              0              0              0
```

```
# Hierarchical plot -- what values are missing together?
plot(naclus(missing_lots))
```




```
# View significance of item correlations
corr.test(missing_lots)
```

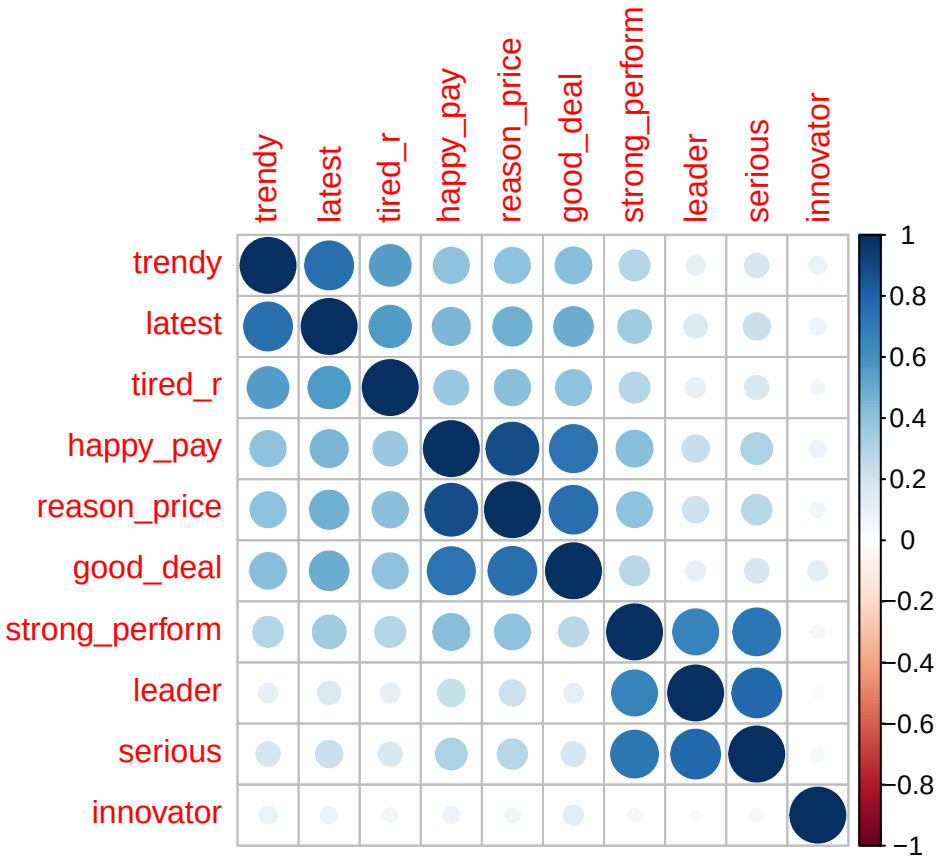
```
## Call:corr.test(x = missing_lots)
## Correlation matrix
##                trendy latest tired_r happy_pay reason_price good_deal
## trendy           1.00   0.76    0.55      0.41         0.41      0.42
## latest           0.76   1.00    0.57      0.46         0.48      0.49
## tired_r          0.55   0.57    1.00      0.38         0.41      0.41
## happy_pay        0.41   0.46    0.38      1.00         0.89      0.73
## reason_price     0.41   0.48    0.41      0.89         1.00      0.75
## good_deal        0.42   0.49    0.41      0.73         0.75      1.00
## strong_perform   0.30   0.35    0.30      0.43         0.40      0.28
## leader           0.12   0.17    0.12      0.23         0.21      0.12
## serious          0.18   0.23    0.18      0.31         0.29      0.19
## innovator        0.09   0.09    0.07      0.09         0.07      0.12
##                strong_perform leader serious innovator
## trendy                   0.30   0.12    0.18      0.09
## latest                   0.35   0.17    0.23      0.09
## tired_r                  0.30   0.12    0.18      0.07
## happy_pay                0.43   0.23    0.31      0.09
## reason_price             0.40   0.21    0.29      0.07
## good_deal                0.28   0.12    0.19      0.12
## strong_perform           1.00   0.67    0.72      0.06
## leader                   0.67   1.00    0.79      0.04
```

```
## serious                  0.72   0.79    1.00      0.06
## innovator                0.06   0.04    0.06      1.00
## Sample Size
## [1] 508
## Probability values (Entries above the diagonal are adjusted for multiple tests.)
##                trendy latest tired_r happy_pay reason_price good_deal
## trendy           0.00   0.00    0.00      0.00          0.0      0.00
## latest           0.00   0.00    0.00      0.00          0.0      0.00
## tired_r          0.00   0.00    0.00      0.00          0.0      0.00
## happy_pay        0.00   0.00    0.00      0.00          0.0      0.00
## reason_price     0.00   0.00    0.00      0.00          0.0      0.00
## good_deal        0.00   0.00    0.00      0.00          0.0      0.00
## strong_perform   0.00   0.00    0.00      0.00          0.0      0.00
## leader           0.01   0.00    0.01      0.00          0.0      0.01
## serious          0.00   0.00    0.00      0.00          0.0      0.00
## innovator        0.03   0.04    0.14      0.05          0.1      0.01
##                strong_perform leader serious innovator
## trendy                    0.0   0.08     0.0      0.26
## latest                    0.0   0.00     0.0      0.31
## tired_r                   0.0   0.08     0.0      0.56
## happy_pay                 0.0   0.00     0.0      0.31
## reason_price              0.0   0.00     0.0      0.51
## good_deal                 0.0   0.08     0.0      0.08
## strong_perform            0.0   0.00     0.0      0.61
## leader                    0.0   0.00     0.0      0.61
## serious                   0.0   0.00     0.0      0.61
## innovator                 0.2   0.39     0.2      0.00
##
##  To see confidence intervals of the correlations, print with the short=FALSE option
```

```
library(corrplot)
```

```
## corrplot 0.92 loaded
```

```
# Visualize item correlations -- corrplot
corrplot(cor(missing_lots), method = "circle")
```




```
# Get response frequencies
response.frequencies(missing_lots)
```

```
##                       1         2          3           4          5 miss
## trendy        0.8484252 0.1122047 0.01968504 0.007874016 0.01181102    0
## latest        0.8110236 0.1397638 0.01574803 0.013779528 0.01968504    0
## tired_r       0.8051181 0.1338583 0.03149606 0.015748031 0.01377953    0
## happy_pay     0.6811024 0.2539370 0.03149606 0.011811024 0.02165354    0
## reason_price  0.6968504 0.2244094 0.04133858 0.013779528 0.02362205    0
## good_deal     0.7322835 0.1968504 0.04921260 0.000000000 0.02165354    0
## strong_perform 0.3307087 0.2480315 0.14566929 0.127952756 0.14763780    0
## leader        0.1889764 0.1476378 0.10826772 0.185039370 0.37007874    0
## serious       0.2381890 0.1830709 0.15748031 0.141732283 0.27952756    0
## innovator     0.2204724 0.2263780 0.18700787 0.175196850 0.19094488    0
```

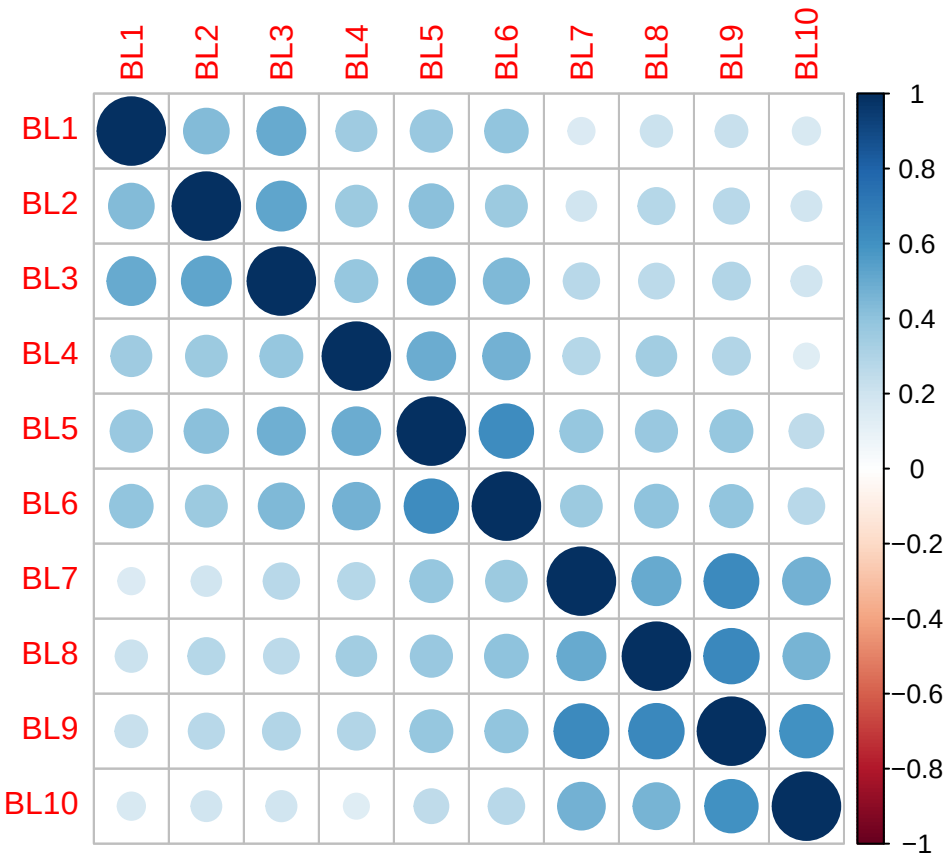
```
# Recode the appropriate item
#b_rep_items
#missing_lots$poor_workman_r <- recode(missing_lots$poor_workman,
#                           "1 = 5; 2 = 4; 4 = 2; 5 = 1")
```

Exploratory factor analysis & survey development

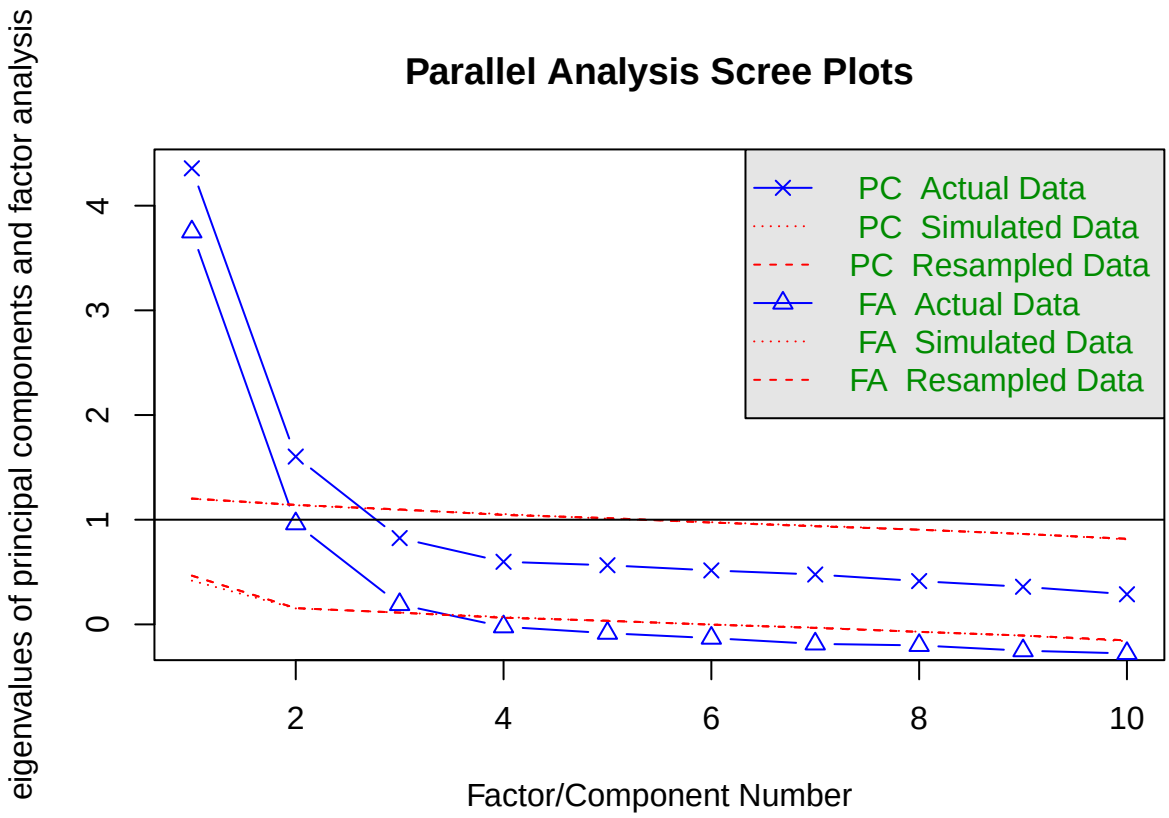
```r
b_loyal_10 <- read.csv("brandloyalty.csv")
corr.test(b_loyal_10)
```

```
## Call:corr.test(x = b_loyal_10)
## Correlation matrix
##       BL1  BL2  BL3  BL4  BL5  BL6  BL7  BL8  BL9 BL10
## BL1  1.00 0.44 0.50 0.35 0.38 0.40 0.15 0.22 0.22 0.16
## BL2  0.44 1.00 0.52 0.36 0.42 0.37 0.19 0.29 0.27 0.20
## BL3  0.50 0.52 1.00 0.38 0.49 0.45 0.28 0.27 0.30 0.20
## BL4  0.35 0.36 0.38 1.00 0.50 0.47 0.29 0.34 0.30 0.14
## BL5  0.38 0.42 0.49 0.50 1.00 0.63 0.39 0.37 0.39 0.25
## BL6  0.40 0.37 0.45 0.47 0.63 1.00 0.36 0.40 0.39 0.28
## BL7  0.15 0.19 0.28 0.29 0.39 0.36 1.00 0.50 0.64 0.48
## BL8  0.22 0.29 0.27 0.34 0.37 0.40 0.50 1.00 0.65 0.46
## BL9  0.22 0.27 0.30 0.30 0.39 0.39 0.64 0.65 1.00 0.60
## BL10 0.16 0.20 0.20 0.14 0.25 0.28 0.48 0.46 0.60 1.00
## Sample Size
## [1] 639
## Probability values (Entries above the diagonal are adjusted for multiple tests.)
##      BL1 BL2 BL3 BL4 BL5 BL6 BL7 BL8 BL9 BL10
## BL1    0   0   0   0   0   0   0   0   0    0
## BL2    0   0   0   0   0   0   0   0   0    0
## BL3    0   0   0   0   0   0   0   0   0    0
## BL4    0   0   0   0   0   0   0   0   0    0
## BL5    0   0   0   0   0   0   0   0   0    0
## BL6    0   0   0   0   0   0   0   0   0    0
## BL7    0   0   0   0   0   0   0   0   0    0
## BL8    0   0   0   0   0   0   0   0   0    0
## BL9    0   0   0   0   0   0   0   0   0    0
## BL10   0   0   0   0   0   0   0   0   0    0
##
##  To see confidence intervals of the correlations, print with the short=FALSE option
```

```r
# Visualize b_loyal_10 correlation matrix
corrplot(cor(b_loyal_10))
```

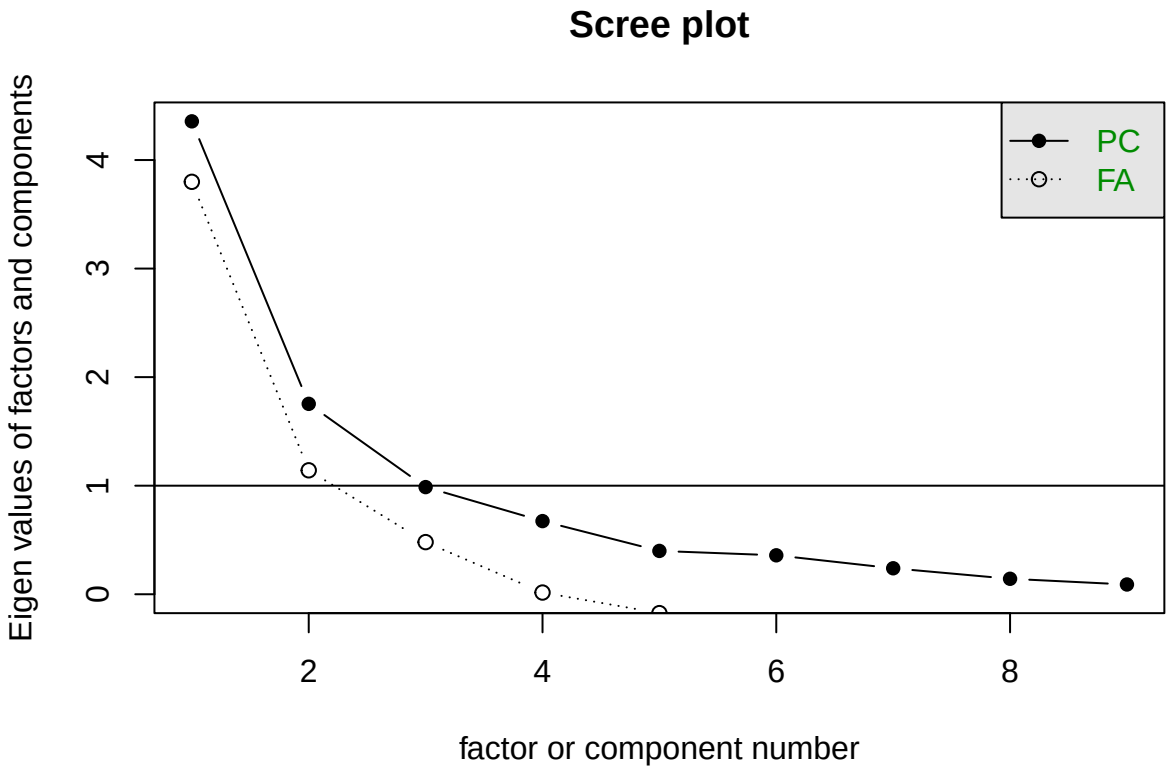



```
# Parallel analysis
fa.parallel(b_loyal_10)
```




```
## Parallel analysis suggests that the number of factors =  3  and the number of components =  2
```

```
brand_rep_9 <- read.csv("brandrep-cleansurvey-extraitem.csv")
scree(brand_rep_9)
```




```
# Conduct three-factor EFA
brand_rep_9_EFA_3 <- fa(brand_rep_9, nfactors = 3)
```

```
## Loading required namespace: GPArotation
```

```
## Warning in fac(r = r, nfactors = nfactors, n.obs = n.obs, rotate = rotate, : I
## am sorry, to do these rotations requires the GPArotation package to be installed
```

```
# Print output of EFA
names(brand_rep_9_EFA_3)
```

```
##  [1] "residual"     "dof"          "chi"          "nh"
##  [5] "rms"          "EPVAL"        "crms"         "EBIC"
##  [9] "ESABIC"       "fit"          "fit.off"      "sd"
## [13] "factors"      "complexity"   "n.obs"        "objective"
## [17] "criteria"     "STATISTIC"    "PVAL"         "Call"
## [21] "null.model"   "null.dof"     "null.chisq"   "TLI"
## [25] "RMSEA"        "BIC"          "SABIC"        "r.scores"
## [29] "R2"           "valid"        "weights"      "rotation"
## [33] "hyperplane"   "communality"  "communalities" "uniquenesses"
## [37] "values"       "e.values"     "loadings"     "model"
## [41] "fm"           "Structure"    "method"       "scores"
## [45] "R2.scores"    "r"            "np.obs"       "fn"
## [49] "Vaccounted"
```

```r
# Summarize results of three-factor EFA
summary(brand_rep_9_EFA_3)
```

```
##
## Factor analysis with Call: fa(r = brand_rep_9, nfactors = 3)
##
## Test of the hypothesis that 3 factors are sufficient.
## The degrees of freedom for the model is 12  and the objective function was  0.08
## The number of observations was  559  with Chi Square =  43.11  with prob <  2.2e-05
##
## The root mean square of the residuals (RMSA) is  0.02
## The df corrected root mean square of the residuals is  0.03
##
## Tucker Lewis Index of factoring reliability =  0.972
## RMSEA index =  0.068  and the 10 % confidence intervals are  0.047 0.091
## BIC =  -32.8
```

```r
# Build and print loadings for a two-factor EFA
brand_rep_9_EFA_2 <- fa(brand_rep_9, nfactors = 2)
```

```
## Loading required namespace: GPArotation
```

```
## Warning in fac(r = r, nfactors = nfactors, n.obs = n.obs, rotate = rotate, : I
## am sorry, to do these rotations requires the GPArotation package to be installed
```

```r
brand_rep_9_EFA_2$loadings
```

```
##
## Loadings:
##                MR1    MR2
## well_made       0.744 -0.504
## consistent      0.725 -0.569
## poor_workman_r  0.570 -0.442
## higher_price    0.704  0.343
## lot_more        0.558  0.495
## go_up           0.641  0.517
## stands_out      0.730  0.145
## unique          0.704  0.105
## one_of_a_kind   0.566
##
##                  MR1   MR2
## SS loadings    3.971 1.436
## Proportion Var 0.441 0.160
## Cumulative Var 0.441 0.601
```

```r
# Build and print loadings for a four-factor EFA
brand_rep_9_EFA_4 <- fa(brand_rep_9, nfactors = 4)
```

```
## Loading required namespace: GPArotation
```

```
## Warning in fac(r = r, nfactors = nfactors, n.obs = n.obs, rotate = rotate, : I
## am sorry, to do these rotations requires the GPArotation package to be installed
```

```
brand_rep_9_EFA_4$loadings
```

```
##
## Loadings:
##                MR1    MR2    MR3    MR4
## well_made       0.719 -0.522  0.112  0.124
## consistent      0.706 -0.599  0.108  0.165
## poor_workman_r  0.551 -0.454         0.106
## higher_price    0.685  0.301  0.140
## lot_more        0.561  0.499  0.262  0.185
## go_up           0.652  0.537  0.266  0.236
## stands_out      0.795  0.164 -0.540
## unique          0.758  0.112 -0.526
## one_of_a_kind   0.657         0.306 -0.686
##
##                  MR1   MR2   MR3   MR4
## SS loadings    4.167 1.506 0.854 0.620
## Proportion Var 0.463 0.167 0.095 0.069
## Cumulative Var 0.463 0.630 0.725 0.794
```

```
# Eigenvalues
brand_rep_9_EFA_3$e.values
```

```
## [1] 4.35629549 1.75381015 0.98701607 0.67377072 0.39901205 0.35865598 0.23915591
## [8] 0.14238807 0.08989556
```

```
# Factor score correlations
brand_rep_9_EFA_3$score.cor
```

```
## NULL
```

```
# Factor loadings
brand_rep_9_EFA_3$loadings
```

```
##
## Loadings:
##                MR1    MR2    MR3
## well_made       0.727 -0.528  0.145
## consistent      0.709 -0.594  0.144
## poor_workman_r  0.556 -0.457  0.123
## higher_price    0.691  0.305  0.160
## lot_more        0.566  0.499  0.316
## go_up           0.650  0.520  0.313
## stands_out      0.812  0.174 -0.553
## unique          0.760  0.114 -0.485
## one_of_a_kind   0.554
##
##                  MR1   MR2   MR3
## SS loadings    4.104 1.496 0.830
## Proportion Var 0.456 0.166 0.092
## Cumulative Var 0.456 0.622 0.714
```

```r
# Create brand_rep_8 data frame
brand_rep_8 <- select(brand_rep_9, -one_of_a_kind)

# Create three-factor EFA
brand_rep_8_EFA_3 <- fa(brand_rep_8, nfactors = 3)
```

```
## Loading required namespace: GPArotation
```

```
## Warning in fac(r = r, nfactors = nfactors, n.obs = n.obs, rotate = rotate, : I
## am sorry, to do these rotations requires the GPArotation package to be installed
```

```r
# Factor loadings
brand_rep_8_EFA_3$loadings
```

```
##
## Loadings:
##                MR1    MR2    MR3
## well_made       0.721 -0.522  0.157
## consistent      0.713 -0.600  0.166
## poor_workman_r  0.552 -0.455  0.134
## higher_price    0.669  0.297  0.147
## lot_more        0.565  0.501  0.321
## go_up           0.658  0.534  0.337
## stands_out      0.816  0.166 -0.527
## unique          0.774  0.111 -0.488
##
##                  MR1   MR2   MR3
## SS loadings    3.799 1.504 0.825
## Proportion Var 0.475 0.188 0.103
## Cumulative Var 0.475 0.663 0.766
```

```r
# Factor correlations -- 9 versus 8 item model
brand_rep_9_EFA_3$score.cor
```

```
## NULL
```

```r
brand_rep_8_EFA_3$score.cor
```

```
## NULL
```

```r
# Three factor EFA loadings
brand_rep_8_EFA_3$loadings
```

```
##
## Loadings:
##                MR1    MR2    MR3
## well_made       0.721 -0.522  0.157
## consistent      0.713 -0.600  0.166
## poor_workman_r  0.552 -0.455  0.134
## higher_price    0.669  0.297  0.147
```

```
## lot_more        0.565  0.501  0.321
## go_up           0.658  0.534  0.337
## stands_out      0.816  0.166 -0.527
## unique          0.774  0.111 -0.488
##
##                  MR1   MR2   MR3
## SS loadings    3.799 1.504 0.825
## Proportion Var 0.475 0.188 0.103
## Cumulative Var 0.475 0.663 0.766
```

```
# Two factor EFA & loadings
brand_rep_8_EFA_2 <- fa(brand_rep_8, nfactors = 2)
```

```
## Loading required namespace: GPArotation
```

```
## Warning in fac(r = r, nfactors = nfactors, n.obs = n.obs, rotate = rotate, : I
## am sorry, to do these rotations requires the GPArotation package to be installed
```

```
brand_rep_8_EFA_2$loadings
```

```
##
## Loadings:
##                MR1    MR2
## well_made       0.742 -0.497
## consistent      0.732 -0.569
## poor_workman_r  0.571 -0.438
## higher_price    0.685  0.337
## lot_more        0.556  0.499
## go_up           0.644  0.524
## stands_out      0.735  0.150
## unique          0.712  0.111
##
##                  MR1   MR2
## SS loadings    3.652 1.436
## Proportion Var 0.456 0.179
## Cumulative Var 0.456 0.636
```

```
# Four factor EFA & loadings
brand_rep_8_EFA_4 <- fa(brand_rep_8, nfactors = 4)
```

```
## Loading required namespace: GPArotation
```

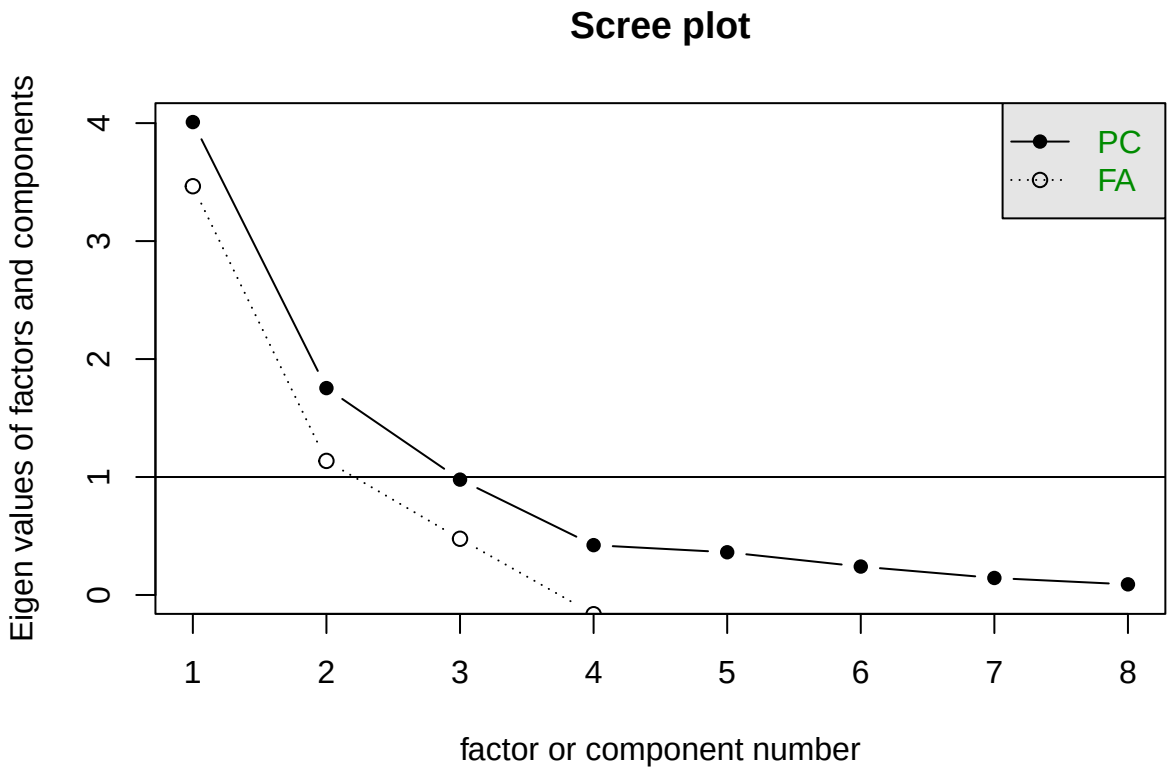
```
## Warning in fac(r = r, nfactors = nfactors, n.obs = n.obs, rotate = rotate, : I
## am sorry, to do these rotations requires the GPArotation package to be installed
```

```
brand_rep_8_EFA_4$loadings
```

```
##
## Loadings:
##                MR1    MR2    MR3    MR4
```

```
## well_made       0.715 -0.497  0.137
## consistent      0.734 -0.627  0.171 -0.184
## poor_workman_r  0.563 -0.463  0.132  0.208
## higher_price    0.668  0.302  0.150
## lot_more        0.563  0.505  0.331
## go_up           0.654  0.531  0.339
## stands_out      0.813  0.181 -0.529
## unique          0.771  0.125 -0.487
##
##                  MR1   MR2   MR3   MR4
## SS loadings    3.813 1.532 0.830 0.089
## Proportion Var 0.477 0.191 0.104 0.011
## Cumulative Var 0.477 0.668 0.772 0.783
```

```
# Scree plot of brand_rep_8
scree(brand_rep_8)
```




```
# Standardized coefficient alpha
psych::alpha(brand_rep_9)$total$std.alpha
```

```
## [1] 0.8648896
```

```
# 3-factor EFA
brand_rep_9_EFA_3 <- fa(brand_rep_9, nfactors = 3)
```

```
## Loading required namespace: GPArotation

## Warning in fac(r = r, nfactors = nfactors, n.obs = n.obs, rotate = rotate, : I
## am sorry, to do these rotations requires the GPArotation package to be installed
```

```
brand_rep_9_EFA_3$loadings
```

```
##
## Loadings:
##                MR1    MR2    MR3
## well_made       0.727 -0.528  0.145
## consistent      0.709 -0.594  0.144
## poor_workman_r  0.556 -0.457  0.123
## higher_price    0.691  0.305  0.160
## lot_more        0.566  0.499  0.316
## go_up           0.650  0.520  0.313
## stands_out      0.812  0.174 -0.553
## unique          0.760  0.114 -0.485
## one_of_a_kind   0.554
##
##                  MR1   MR2   MR3
## SS loadings    4.104 1.496 0.830
## Proportion Var 0.456 0.166 0.092
## Cumulative Var 0.456 0.622 0.714
```

```
# Get names of survey items
colnames(brand_rep_8)
```

```
## [1] "well_made"      "consistent"     "poor_workman_r" "higher_price"
## [5] "lot_more"       "go_up"          "stands_out"     "unique"
```

```
# Create new data frames for each of three dimensions
p_quality <- select(brand_rep_8, 1:3)
p_willingness <- select(brand_rep_8, 4:6)
p_difference <- select(brand_rep_8, 7:8)
```

```
# Get split-half reliability
splitHalf(brand_rep_8)
```

```
## Split half reliabilities
## Call: splitHalf(r = brand_rep_8)
##
## Maximum split half reliability (lambda 4) =  0.93
## Guttman lambda 6                          =  0.92
## Average split half reliability            =  0.86
## Guttman lambda 3 (alpha)                  =  0.86
## Guttman lambda 2                          =  0.87
## Minimum split half reliability  (beta)    =  0.66
## Average interitem r =  0.43  with median =  0.4
```

```r
# Get averages of even and odd row scores
even_items <- colMeans(brand_rep_8[,c(FALSE,TRUE)])
odd_items <- colMeans(brand_rep_8[,c(TRUE,FALSE)])

# Correlate scores from even and odd items
cor(even_items, odd_items)
```

```
## [1] 0.7441724
```

```r
# Get Cronbach's alpha
psych::alpha(brand_rep_8)
```

```
##
## Reliability analysis
## Call: psych::alpha(x = brand_rep_8)
##
##   raw_alpha std.alpha G6(smc) average_r S/N    ase mean   sd median_r
##       0.85      0.86    0.92      0.43 5.9 0.0096  2.2 0.81      0.4
##
##     95% confidence boundaries
##          lower alpha upper
## Feldt     0.83  0.85  0.86
## Duhachek  0.83  0.85  0.87
##
##  Reliability if an item is dropped:
##                raw_alpha std.alpha G6(smc) average_r S/N alpha se var.r med.r
## well_made           0.83      0.83    0.89      0.42 5.0   0.0102 0.042  0.39
## consistent          0.84      0.84    0.89      0.42 5.1   0.0100 0.039  0.41
## poor_workman_r      0.85      0.85    0.92      0.45 5.7   0.0098 0.041  0.41
## higher_price        0.82      0.83    0.91      0.42 5.0   0.0120 0.052  0.39
## lot_more            0.84      0.85    0.91      0.45 5.7   0.0105 0.041  0.41
## go_up               0.82      0.84    0.90      0.43 5.3   0.0115 0.044  0.39
## stands_out          0.81      0.83    0.88      0.41 4.8   0.0116 0.045  0.34
## unique              0.82      0.83    0.88      0.41 4.9   0.0113 0.046  0.38
##
##  Item statistics
##                   n raw.r std.r r.cor r.drop mean   sd
## well_made       559  0.64  0.73  0.72   0.56  1.5 0.83
## consistent      559  0.62  0.71  0.70   0.52  1.5 0.85
## poor_workman_r 559  0.52  0.62  0.56   0.43  1.4 0.77
## higher_price    559  0.78  0.73  0.68   0.68  2.5 1.36
## lot_more        559  0.71  0.62  0.57   0.56  3.4 1.48
## go_up           559  0.76  0.69  0.65   0.64  3.1 1.45
## stands_out      559  0.78  0.77  0.78   0.69  2.0 1.18
## unique          559  0.76  0.76  0.76   0.66  2.0 1.18
##
## Non missing response frequency for each item
##                    1    2    3    4    5 miss
## well_made       0.65 0.27 0.05 0.01 0.02    0
## consistent      0.68 0.23 0.05 0.02 0.02    0
## poor_workman_r 0.76 0.17 0.04 0.01 0.02    0
## higher_price    0.31 0.26 0.18 0.12 0.12    0
## lot_more        0.15 0.17 0.16 0.16 0.36    0
```

```
## go_up          0.18 0.18 0.20 0.18 0.26    0
## stands_out     0.43 0.30 0.14 0.07 0.06    0
## unique         0.46 0.29 0.12 0.07 0.06    0
```

```
# 3 factor EFA
b_loyal_10_EFA_3 <- fa(b_loyal_10, nfactors = 3)
```

```
## Loading required namespace: GPArotation
```

```
## Warning in fac(r = r, nfactors = nfactors, n.obs = n.obs, rotate = rotate, : I
## am sorry, to do these rotations requires the GPArotation package to be installed
```

```
head(b_loyal_10)
```

```
##   BL1 BL2 BL3 BL4 BL5 BL6 BL7 BL8 BL9 BL10
## 1   5   4   4   4   4   4   3   4   3    3
## 2   4   3   4   3   3   3   3   1   1    1
## 3   5   3   5   3   3   4   2   2   2    3
## 4   4   4   2   2   2   2   2   4   2    2
## 5   4   5   4   4   4   4   3   4   4    4
## 6   4   4   4   4   3   4   2   2   2    2
```

```
# 3 factor EFA
b_loyal_10_EFA_3 <- fa(b_loyal_10, nfactors = 3)
```

```
## Loading required namespace: GPArotation
```

```
## Warning in fac(r = r, nfactors = nfactors, n.obs = n.obs, rotate = rotate, : I
## am sorry, to do these rotations requires the GPArotation package to be installed
```

```
# Factor loadings, eigenvalues and factor score correlations
b_loyal_10_EFA_3$loadings
```

```
##
## Loadings:
##      MR1    MR2    MR3
## BL1   0.513  0.361  0.200
## BL2   0.558  0.321  0.241
## BL3   0.627  0.367  0.214
## BL4   0.569  0.222 -0.166
## BL5   0.716  0.232 -0.279
## BL6   0.698  0.188 -0.253
## BL7   0.618 -0.367
## BL8   0.651 -0.308
## BL9   0.741 -0.506
## BL10  0.524 -0.411  0.136
##
##                  MR1   MR2   MR3
## SS loadings    3.920 1.160 0.345
## Proportion Var 0.392 0.116 0.035
## Cumulative Var 0.392 0.508 0.543
```

```
b_loyal_10_EFA_3$e.values
```

```
##  [1] 4.3564537 1.6031839 0.8242739 0.5982539 0.5649528 0.5155507 0.4767388
##  [8] 0.4136564 0.3594158 0.2875202
```

```
b_loyal_10_EFA_3$score.cor
```

```
## NULL
```

```
# 2 factor EFA
b_loyal_10_EFA_2 <- fa(b_loyal_10, nfactors = 2)
```

```
## Loading required namespace: GPArotation
```

```
## Warning in fac(r = r, nfactors = nfactors, n.obs = n.obs, rotate = rotate, : I
## am sorry, to do these rotations requires the GPArotation package to be installed
```

```
# Factor loadings, eigenvalues and factor score correlations
b_loyal_10_EFA_2$loadings
```

```
##
## Loadings:
##      MR1    MR2
## BL1   0.508  0.361
## BL2   0.549  0.316
## BL3   0.620  0.365
## BL4   0.567  0.229
## BL5   0.700  0.226
## BL6   0.687  0.189
## BL7   0.623 -0.362
## BL8   0.657 -0.302
## BL9   0.744 -0.493
## BL10  0.526 -0.399
##
##                  MR1   MR2
## SS loadings    3.877 1.127
## Proportion Var 0.388 0.113
## Cumulative Var 0.388 0.500
```

```
b_loyal_10_EFA_2$e.values
```

```
##  [1] 4.3564537 1.6031839 0.8242739 0.5982539 0.5649528 0.5155507 0.4767388
##  [8] 0.4136564 0.3594158 0.2875202
```

```
b_loyal_10_EFA_2$score.cor
```

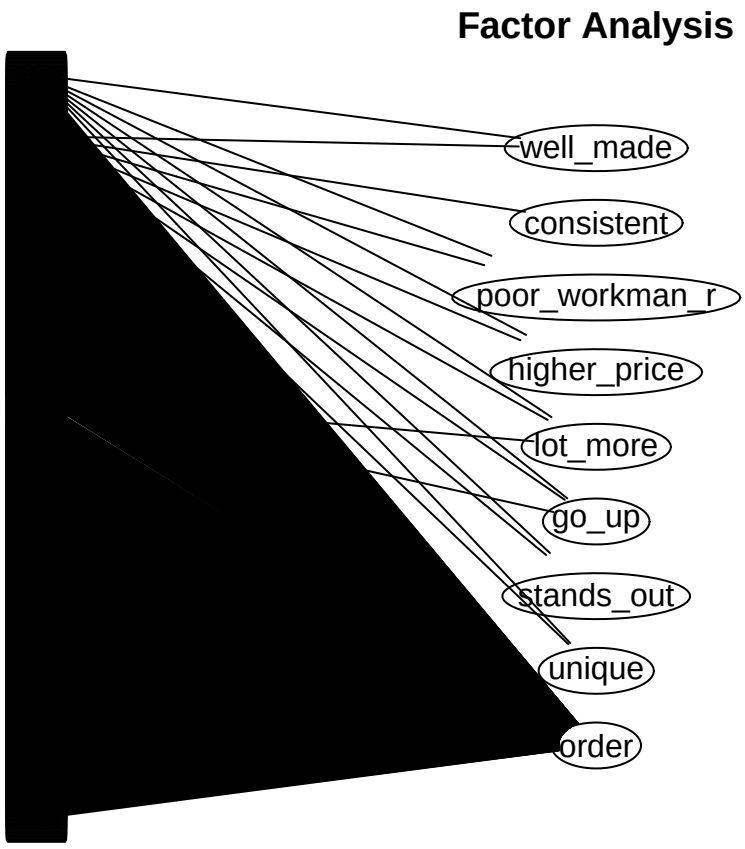
```
## NULL
```

Confirmatory factor analysis & construct validation

```
# Factor loadings -- EFA
brand_rep_8$loadings
```

```
## NULL
```

```
# Plot diagram -- EFA
fa.diagram(brand_rep_8)
```




```
library(lavaan)
```

```
## This is lavaan 0.6-11
## lavaan is FREE software! Please report any bugs.

##
## Attaching package: 'lavaan'

## The following object is masked from 'package:psych':
##
##     cor2cov
```

```r
# Rename items based on proposed dimensions
colnames(b_loyal_10) <- c("ID1", "ID2", "ID3",
                          "PV1", "PV2", "PV3",
                          "BT1", "BT2", "BT3", "BT4")

# Define the model
b_loyal_cfa_model <- 'ID =~ ID1 + ID2 + ID3
                      PV =~ PV1 + PV2 + PV3
                      BT =~ BT1 + BT2 + BT3 + BT4'

# Fit the model to the data
b_loyal_cfa <- cfa(model = b_loyal_cfa_model, data = b_loyal_10)

# Check the summary statistics -- include fit measures and standardized estimates
summary(b_loyal_cfa, fit.measures = TRUE, standardized = TRUE)
```

```
## lavaan 0.6-11 ended normally after 33 iterations
##
##   Estimator                                         ML
##   Optimization method                           NLMINB
##   Number of model parameters                        23
##
##   Number of observations                           639
##
## Model Test User Model:
##
##   Test statistic                                63.953
##   Degrees of freedom                                32
##   P-value (Chi-square)                           0.001
##
## Model Test Baseline Model:
##
##   Test statistic                              2485.786
##   Degrees of freedom                                45
##   P-value                                        0.000
##
## User Model versus Baseline Model:
##
##   Comparative Fit Index (CFI)                    0.987
##   Tucker-Lewis Index (TLI)                       0.982
##
## Loglikelihood and Information Criteria:
##
##   Loglikelihood user model (H0)              -7214.586
##   Loglikelihood unrestricted model (H1)      -7182.610
##
##   Akaike (AIC)                               14475.173
##   Bayesian (BIC)                             14577.751
##   Sample-size adjusted Bayesian (BIC)        14504.727
##
## Root Mean Square Error of Approximation:
##
##   RMSEA                                          0.040
```

```
##   90 Percent confidence interval - lower         0.025
##   90 Percent confidence interval - upper         0.054
##   P-value RMSEA <= 0.05                          0.885
##
## Standardized Root Mean Square Residual:
##
##   SRMR                                           0.030
##
## Parameter Estimates:
##
##   Standard errors                             Standard
##   Information                                 Expected
##   Information saturated (h1) model          Structured
##
## Latent Variables:
##                   Estimate  Std.Err  z-value  P(>|z|)   Std.lv  Std.all
##   ID =~
##     ID1              1.000                               0.450    0.646
##     ID2              1.186    0.090   13.235    0.000    0.534    0.675
##     ID3              1.445    0.102   14.209    0.000    0.651    0.778
##   PV =~
##     PV1              1.000                               0.555    0.626
##     PV2              1.311    0.087   15.012    0.000    0.728    0.800
##     PV3              1.340    0.091   14.765    0.000    0.744    0.772
##   BT =~
##     BT1              1.000                               0.723    0.717
##     BT2              1.106    0.065   17.064    0.000    0.799    0.729
##     BT3              1.174    0.060   19.529    0.000    0.848    0.888
##     BT4              0.886    0.057   15.507    0.000    0.641    0.660
##
## Covariances:
##                   Estimate  Std.Err  z-value  P(>|z|)   Std.lv  Std.all
##   ID ~~
##     PV               0.192    0.020    9.511    0.000    0.768    0.768
##     BT               0.142    0.019    7.423    0.000    0.436    0.436
##   PV ~~
##     BT               0.234    0.026    8.966    0.000    0.583    0.583
##
## Variances:
##                   Estimate  Std.Err  z-value  P(>|z|)   Std.lv  Std.all
##    .ID1              0.282    0.019   14.506    0.000    0.282    0.582
##    .ID2              0.341    0.024   13.918    0.000    0.341    0.545
##    .ID3              0.277    0.026   10.637    0.000    0.277    0.395
##    .PV1              0.479    0.031   15.569    0.000    0.479    0.608
##    .PV2              0.299    0.026   11.301    0.000    0.299    0.361
##    .PV3              0.374    0.030   12.369    0.000    0.374    0.403
##    .BT1              0.492    0.033   14.907    0.000    0.492    0.485
##    .BT2              0.563    0.038   14.670    0.000    0.563    0.468
##    .BT3              0.192    0.024    8.013    0.000    0.192    0.211
##    .BT4              0.530    0.034   15.777    0.000    0.530    0.564
##     ID               0.203    0.025    8.166    0.000    1.000    1.000
##     PV               0.308    0.038    8.097    0.000    1.000    1.000
##     BT               0.522    0.053    9.879    0.000    1.000    1.000
```

```r
c_sat <- read.csv("customersatisfactionclean.csv")

# Two dimensions: odd- versus even-numbered items
c_sat_bad_model <- 'ODD =~ CS1 + CS3 + CS5 + CS7 + CS9
                EVEN =~ CS2 + CS4 + CS6 + CS8 + CS10'

# Fit the model to the data
c_sat_bad_CFA <- cfa(model = c_sat_bad_model, data = c_sat)
```

```
## Warning in lav_object_post_check(object): lavaan WARNING: covariance matrix of latent variables
##                 is not positive definite;
##                 use lavInspect(fit, "cov.lv") to investigate.
```

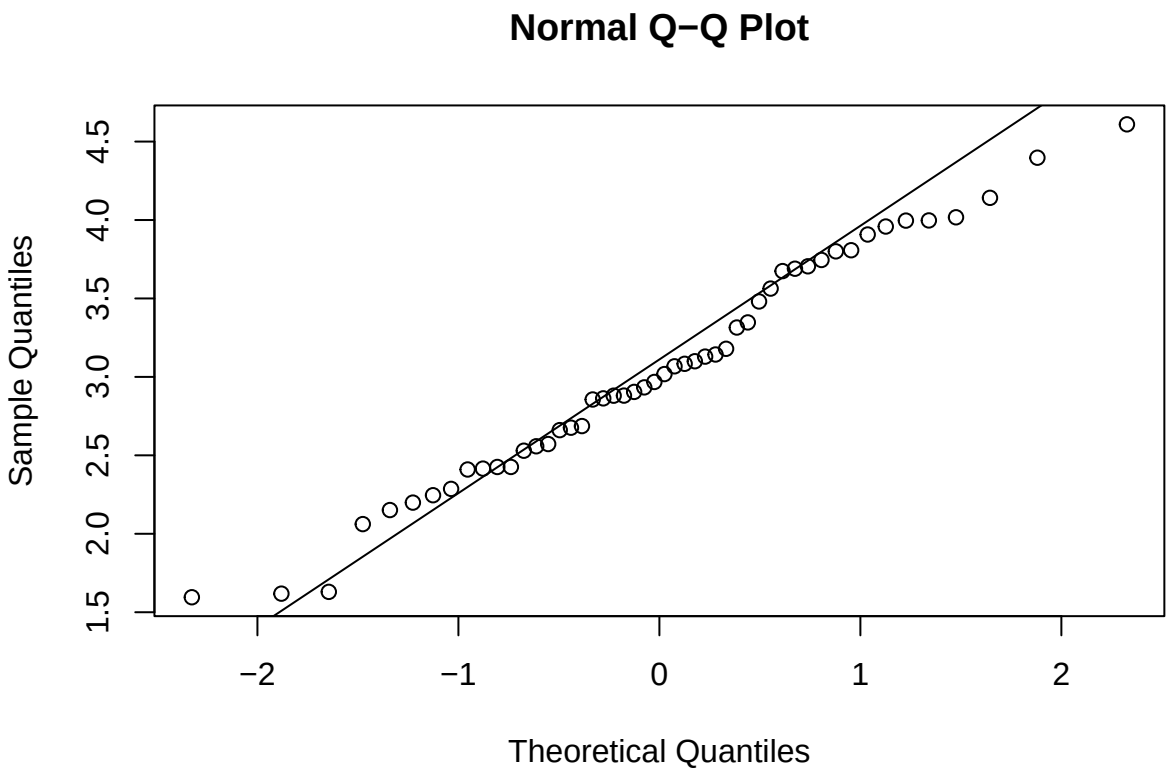
```r
# Summary measures
summary(c_sat_bad_CFA, fit.measures = TRUE, standardized = TRUE)
```

```
## lavaan 0.6-11 ended normally after 31 iterations
##
##   Estimator                                         ML
##   Optimization method                           NLMINB
##   Number of model parameters                        21
##
##   Number of observations                           350
##
## Model Test User Model:
##
##   Test statistic                               305.396
##   Degrees of freedom                                34
##   P-value (Chi-square)                           0.000
##
## Model Test Baseline Model:
##
##   Test statistic                              1054.540
##   Degrees of freedom                                45
##   P-value                                        0.000
##
## User Model versus Baseline Model:
##
##   Comparative Fit Index (CFI)                    0.731
##   Tucker-Lewis Index (TLI)                       0.644
##
## Loglikelihood and Information Criteria:
##
##   Loglikelihood user model (H0)              -3803.754
##   Loglikelihood unrestricted model (H1)      -3651.057
##
##   Akaike (AIC)                                7649.509
##   Bayesian (BIC)                              7730.525
##   Sample-size adjusted Bayesian (BIC)         7663.906
##
## Root Mean Square Error of Approximation:
##
##   RMSEA                                          0.151
```

```
##   90 Percent confidence interval - lower         0.136
##   90 Percent confidence interval - upper         0.167
##   P-value RMSEA <= 0.05                          0.000
##
## Standardized Root Mean Square Residual:
##
##   SRMR                                           0.099
##
## Parameter Estimates:
##
##   Standard errors                             Standard
##   Information                                 Expected
##   Information saturated (h1) model          Structured
##
## Latent Variables:
##                   Estimate  Std.Err  z-value  P(>|z|)   Std.lv  Std.all
##   ODD =~
##     CS1              1.000                               0.537    0.626
##     CS3              0.866    0.092    9.382    0.000    0.465    0.581
##     CS5              0.726    0.085    8.534    0.000    0.390    0.519
##     CS7              0.721    0.098    7.354    0.000    0.387    0.438
##     CS9              0.641    0.093    6.915    0.000    0.344    0.409
##   EVEN =~
##     CS2              1.000                               0.453    0.569
##     CS4              1.139    0.118    9.655    0.000    0.516    0.694
##     CS6              0.777    0.102    7.642    0.000    0.352    0.498
##     CS8              0.880    0.114    7.687    0.000    0.398    0.502
##     CS10             1.040    0.123    8.422    0.000    0.471    0.567
##
## Covariances:
##                   Estimate  Std.Err  z-value  P(>|z|)   Std.lv  Std.all
##   ODD ~~
##     EVEN             0.268    0.035    7.699    0.000    1.102    1.102
##
## Variances:
##                   Estimate  Std.Err  z-value  P(>|z|)   Std.lv  Std.all
##    .CS1              0.447    0.038   11.744    0.000    0.447    0.608
##    .CS3              0.424    0.035   12.140    0.000    0.424    0.663
##    .CS5              0.411    0.033   12.503    0.000    0.411    0.730
##    .CS7              0.630    0.049   12.800    0.000    0.630    0.808
##    .CS9              0.588    0.046   12.875    0.000    0.588    0.832
##    .CS2              0.428    0.035   12.175    0.000    0.428    0.676
##    .CS4              0.286    0.026   10.978    0.000    0.286    0.518
##    .CS6              0.376    0.030   12.527    0.000    0.376    0.752
##    .CS8              0.472    0.038   12.512    0.000    0.472    0.748
##    .CS10             0.468    0.038   12.189    0.000    0.468    0.679
##     ODD              0.288    0.047    6.070    0.000    1.000    1.000
##     EVEN             0.205    0.038    5.420    0.000    1.000    1.000
```

```
c_sat_50 <- head(c_sat, 50 )
c_sat_model <- "F1 =~ CS1 + CS2 + CS3 + CS4\nF2 =~ CS5 + CS6 + CS7\nF3 =~ CS8 + CS9 + CS10"

# Mardia's test for multivariate normality
mardia(c_sat_50)
```




```
## Call: mardia(x = c_sat_50)
##
## Mardia tests of multivariate skew and kurtosis
## Use describe(x) the to get univariate tests
## n.obs = 50   num.vars =  10
## b1p =  25.88   skew =  215.65  with probability  <=  0.57
##  small sample skew =  231.06  with probability <=  0.29
## b2p =  115.85   kurtosis =  -0.95  with probability <=  0.34
```

```
# Fit model to the data using robust standard errors
c_sat_cfa_mlr <- cfa(model = c_sat_model, data = c_sat_50, estimator = "MLR")

# Summary including standardized estimates and fit measures
#summary(c_sat_cfa_mlr, standardized = TRUE, fit.measures = TRUE)
```

```
# View current c_sat model
cat(c_sat_model)
```

```
## F1 =~ CS1 + CS2 + CS3 + CS4
## F2 =~ CS5 + CS6 + CS7
## F3 =~ CS8 + CS9 + CS10
```

```
c_sat_model_a <- "F1 =~ CS1 + CS2 + CS3 + CS4\n              F2 =~ CS5 + CS6 + CS7\n          F3 =~
```

```
# Add EU1 to the CSU factor
c_sat_model_b <- "F1 =~ CS1 + CS3 + CS5 + CS7 + CS9\n\t\t\t\tF2 =~ CS2 + CS4 + CS6 + CS8 + CS10"

# Fit Models A and B to the data
c_sat_cfa_a <- cfa(model = c_sat_model_a, data = c_sat)
c_sat_cfa_b <- cfa(model = c_sat_model_b, data = c_sat)
```

```
## Warning in lav_object_post_check(object): lavaan WARNING: covariance matrix of latent variables
##                 is not positive definite;
##                 use lavInspect(fit, "cov.lv") to investigate.
```

```
# Calculate the desired model fit statistics
fitMeasures(c_sat_cfa_a, fit.measures = c("cfi", "tli"))
```

```
##   cfi   tli
## 0.962 0.947
```

```
fitMeasures(c_sat_cfa_b, fit.measures = c("cfi", "tli"))
```

```
##   cfi   tli
## 0.731 0.644
```

```
# Compare the nested models
anova(c_sat_cfa_a, c_sat_cfa_b)
```

```
## Chi-Squared Difference Test
##
##             Df    AIC    BIC   Chisq Chisq diff Df diff Pr(>Chisq)
## c_sat_cfa_a 32 7418.2 7506.9  70.057
## c_sat_cfa_b 34 7649.5 7730.5 305.396     235.34       2  < 2.2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

```
c_sat_cfa_model_2 <- "F1 =~ CS1 + CS2 + CS3 + CS4 + CS5\n                        + CS6 + CS7\n
c_sat_cfa_model_3 <-"F1 =~ CS1 + CS2 + CS3 + CS4\nF2 =~ CS5 + CS6 + CS7\nF3 =~ CS8 + CS9 + CS10"

# Fit three-factor CFA
c_sat_cfa_3 <- cfa(model = c_sat_cfa_model_3, data = c_sat)

# Inspect key fit measures - three-factor CFA
fitMeasures(c_sat_cfa_3, fit.measures = c("cfi","tli","rmsea"))
```

```
##   cfi   tli rmsea
## 0.962 0.947 0.058
```

```
# Fit two-factor CFA
c_sat_cfa_2 <- cfa(model = c_sat_cfa_model_2, data = c_sat)

# Inspect key fit measures - two-factor CFA
fitMeasures(c_sat_cfa_2, fit.measures = c("cfi","tli","rmsea"))
```

```
##   cfi   tli rmsea
## 0.896 0.862 0.094
```

```
# Compare measures of construct validity for three- versus two-factor models
#reliability(c_sat_cfa_3)
#reliability(c_sat_cfa_2)
```

```
# Store F1 estimates as object loadings
loadings <- standardizedSolution(c_sat_cfa) %>%
 filter(op == "=~", lhs == "F1") %>% select(est.std)
```

```
## Error in standardizedSolution(c_sat_cfa): object 'c_sat_cfa' not found
```

```
# Composite reliability -- the squared sum of all loadings divided by that same figure plus the sum of
com_rel <- sum(loadings) ^ 2 / ((sum(loadings)^ 2)  + sum(1 - loadings ^ 2))
```

```
## Error in sum(loadings): invalid 'type' (closure) of argument
```

```
com_rel
```

```
## Error in eval(expr, envir, enclos): object 'com_rel' not found
```

```
# Average variance extracted -- sum of all factor squares divided by the number of items
avg_var <- sum(loadings ^ 2) / nrow(loadings)
```

```
## Error in loadings^2: non-numeric argument to binary operator
```

```
avg_var
```

```
## Error in eval(expr, envir, enclos): object 'avg_var' not found
```

```
# Compare versus semTools
reliability(c_sat_cfa)
```

```
## Error in reliability(c_sat_cfa): object 'c_sat_cfa' not found
```

```
# Print brand_rep_factors
brand_rep_factors <- read.csv("fact.csv")

# Build model for lavaan
brand_rep_8_cfa_model <- "QUAL =~ consistent + well_made + poor_workman_r
                PRICE =~ go_up + lot_more + higher_price
                UNIQUE =~ stands_out + unique"
```

Criterion validity and replication

```
# Correlate F1, F2 and F3 to spend_f, the 'latentized' spend
brand_rep_model <- 'F1 =~ well_made + consistent + poor_workman_r
                    F2 =~ higher_price + lot_more + go_up
                    F3 =~ stands_out + unique
                    spend_f =~ spend
                    spend_f ~~ F1 + F2 + F3'

# Fit the model to the data -- sem()
#brand_rep_cv <- sem(data = brand_rep_scaled, model = brand_rep_model)

# Print the standardized covariances b/w spend_f and other factors
#standardizedSolution(brand_rep_cv) %>% filter(rhs == "spend_f")

# Plot the model with standardized estimate labels
#semPaths(brand_rep_cv, whatLabels = "est.std", edge.label.cex = .8)
```

```
# Bind & scale the variables
#c_sat_rec_scale <- cbind(c_sat, c_sat_recommend) %>% scale()

# Define the model - Rec_f covaries with F1, F2, F3
#c_sat_rec_model <- 'F1 =~ CS1 + CS2 + CS3 + CS4
#F2 =~ CS5 + CS6 + CS7
#F3 =~ CS8 + CS9 + CS10
#Rec_f =~ Rec_1
#Rec_f ~~ F1 + F2 + F3'

# Fit the model to the data
#c_sat_rec_sem <- sem(model = c_sat_rec_model, data = c_sat_rec)

# Look up standardized covariances
#standardizedSolution(c_sat_rec_sem) %>% filter(rhs == "Rec_f")
```

Predictive validity & factor scores

```
# Define the model
b_q_model <- 'HIP =~ trendy + latest + tired_r
            VALUE =~ happy_pay + reason_price + good_deal
            PERFORM =~ strong_perform + leader + serious
            spend ~ HIP + VALUE + PERFORM'

# Fit the model to the data
#b_q_pv <- sem(data = b_q_scale, model = b_q_model)

# Check fit, r-square, standardized estimates
#summary(b_q_pv, standardized = T, fit.measures = T, rsquare = T)

# Plot the model -- rotate from left to right
#semPaths(b_q_pv, rotation = 2, whatLabels = "est.std", edge.label.cex = .8)
```

```
# Plot the new model
#semPaths(brand_rep_sem, rotation = 2)
```

```
# Get the coefficient information
#standardizedSolution(brand_rep_sem) %>% filter(op == "~")

# Get the r-squared
#r_squared <- inspect(brand_rep_sem, 'r2')["F2"]
#r_squared
```

```
# Linear regression of standardized spending and factor scores
#bq_fs_reg <- lm(spend ~ F1 + F2 + F3, data = bq_fs_spend)

# Summarize results, round estimates
#rounded_summary <- round(summary(bq_fs_reg)$coef, 3)
#rounded_summary

# Summarize the results of CFA model
#summary(brand_qual_pu)

# Compare the r-squared of each
#inspect_rsq <- inspect(brand_qual_pu, "r2")["spend"]
#inspect_rsq
#summary(bq_fs_reg)$r.squared
```