PLS-SEM: Model Summary and Evaluation

John Robert Torres

2024-11-07

1 Load required and new packages

```
if (!require("pacman")) install.packages("pacman")
```

```
## Loading required package: pacman
```

```
library(pacman)
pacman::p_load("here", "glue", "crayon","readxl", "writexl", "dplyr", "tidyr", "rstatix")
pacman::p_load("seminr")
```

2 Set data paths and details

```
main.path = here::here()
data.path = file.path(main.path, "02 Data")
output.path = file.path(main.path, "04 Outputs")

file.name = "Data - For Analysis.xlsx"
sheet.name = "Final"
output.name = paste0(format(Sys.Date(), "%m%d%y"), "_OUTPUT", ".xlsx")
```

3 Load dataset

```
df.raw = readxl::read_excel(file.path(data.path, file.name),
                            sheet = sheet.name)
```

4 Process data

```
df.proc = df.raw %>%
  dplyr::select(-PR4)
```

5 Implement methodology

5.1 Define measurement models

```
df.mm = seminr::constructs(
  seminr::composite("PE", seminr::multi_items("PE", 1:3)),
  seminr::composite("EE", seminr::multi_items("EE", 1:3)),
  seminr::composite("SI", seminr::multi_items("SI", 1:2)),
  seminr::composite("FC", seminr::multi_items("FC", 1:3)),
  seminr::composite("PI", seminr::multi_items("PI", 1:4)),
  seminr::composite("PR", seminr::multi_items("PR", 1:3)),
  seminr::composite("BI", seminr::multi_items("BI", 1:2))
)
```

5.2 Define structural models

```
df.sm = seminr::relationships(
  seminr::paths(from = c("PE", "EE", "SI", "FC", "PI", "PR"), to = c("BI"))
)
```

5.3 Estimate PLS-SEM model

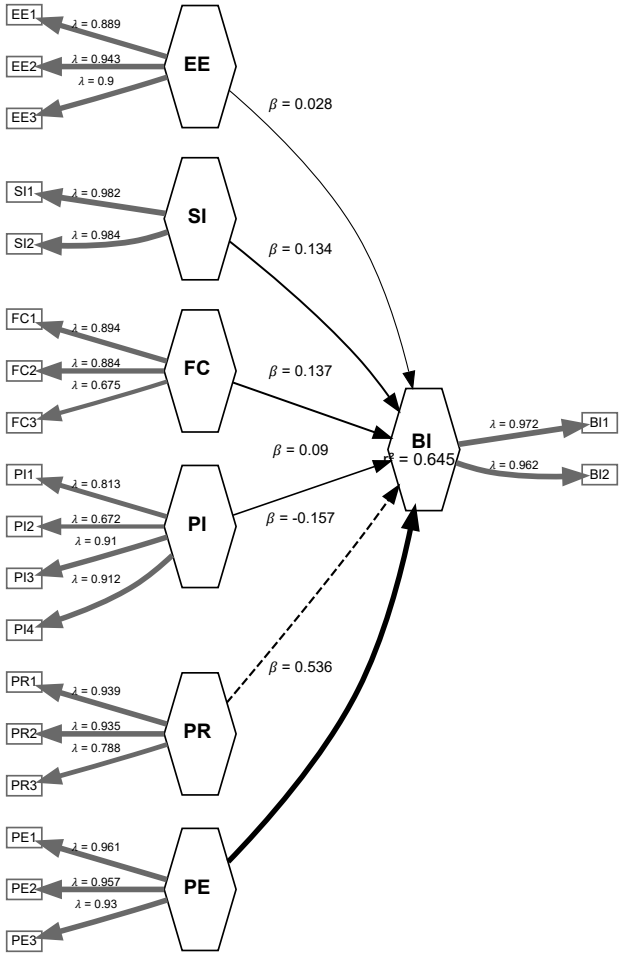
```
pls.model = seminr::estimate_pls(data = df.proc,
                                 measurement_model = df.mm,
                                 structural_model = df.sm,
                                 inner_weights = seminr::path_weighting,
                                 missing = seminr::mean_replacement,
                                 missing_value = -99)
```

```
## Generating the seminr model
```

```
## All 106 observations are valid.
```

5.4 Summarize PLS-SEM model

```
graphics::plot(pls.model,
               edge.display = TRUE,
               node.color = "lightblue",
               node.frame.color = "black")
```




```
pls.summary = summary(pls.model)

items.desc = pls.summary$descriptives$statistics$items
items.desc.df = cbind("Indicators" = rownames(items.desc), as.data.frame(items.desc)) %>%
  dplyr::mutate(dplyr::across(dplyr::where(is.numeric), ~ round(., 4))) %>%
  dplyr::select(-`No.`, -Missing)
items.desc.df
```

```
##     Indicators   Mean Median Min Max Std.Dev. Kurtosis Skewness
## PE1        PE1 4.5472      5   2   5   0.6919   4.1530  -1.3831
## PE2        PE2 4.5943      5   2   5   0.6441   4.9998  -1.5371
## PE3        PE3 4.4811      5   3   5   0.7333   2.6038  -1.0266
## EE1        EE1 4.5283      5   1   5   0.7199   7.3127  -1.7922
## EE2        EE2 4.4340      5   3   5   0.7435   2.3604  -0.8900
## EE3        EE3 4.6038      5   3   5   0.6276   3.6038  -1.3255
## SI1        SI1 4.5377      5   2   5   0.7061   4.9364  -1.5204
## SI2        SI2 4.5660      5   2   5   0.6764   5.6184  -1.6326
## FC1        FC1 4.7170      5   3   5   0.4732   3.1806  -1.2316
## FC2        FC2 4.6038      5   3   5   0.5803   3.3093  -1.1447
## FC3        FC3 4.6226      5   1   5   0.6684  10.2282  -2.2820
## PI1        PI1 4.3491      5   1   5   0.8949   5.4468  -1.5431
## PI2        PI2 3.2075      3   1   5   0.9229   3.1092   0.2358
## PI3        PI3 4.2736      4   3   5   0.7750   1.8487  -0.5125
## PI4        PI4 4.1698      4   1   5   0.8888   4.5783  -1.1554
## PR1        PR1 4.7264      5   1   5   0.6551  13.0600  -2.9213
## PR2        PR2 4.6604      5   1   5   0.7156  14.1891  -3.0042
## PR3        PR3 4.1038      4   1   5   0.8273   6.5485  -1.4111
## BI1        BI1 4.6321      5   2   5   0.6373   5.5806  -1.7241
## BI2        BI2 4.6415      5   2   5   0.6353   5.7491  -1.7743
```

```
constructs.desc = pls.summary$descriptives$statistics$constructs
constructs.desc.df = cbind("Constructs" = rownames(constructs.desc), as.data.frame(constructs.de
sc)) %>%
  dplyr::mutate(dplyr::across(dplyr::where(is.numeric), ~ round(., 4))) %>%
  dplyr::select(-`No.`, -Missing)
constructs.desc.df
```

```
##    Constructs Mean  Median     Min    Max Std.Dev. Kurtosis Skewness
## PE         PE    0  0.6962 -3.4853 0.6962        1   3.6824  -1.2464
## EE         EE    0  0.7492 -2.4261 0.7492        1   2.8029  -1.0207
## SI         SI    0  0.6592 -3.7615 0.6592        1   5.4600  -1.5895
## FC         FC    0  0.7499 -2.2732 0.7499        1   2.0792  -0.8329
## PI         PI    0 -0.0169 -2.4060 1.3181        1   2.2964  -0.5413
## PR         PR    0  0.3491 -5.6041 0.6472        1  13.9079  -2.9583
## BI         BI    0  0.5906 -4.2827 0.5906        1   5.7314  -1.7004
```

5.5 Perform bootstrapping

```
pls.boot = seminr::bootstrap_model(seminr_model = pls.model,
                                   nboot = 10000,
                                   cores = NULL,
                                   seed = 102899)
```

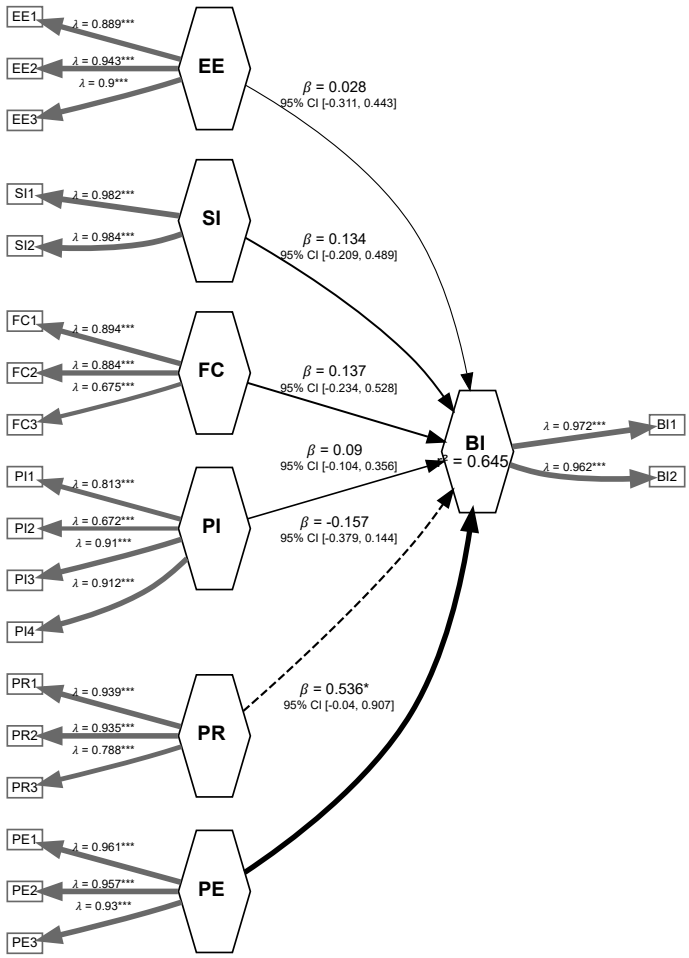
```
## Bootstrapping model using seminr...
```

```
## SEMinR Model successfully bootstrapped
```

```
pls.boot.summary = summary(pls.boot, alpha = 0.05)
pls.boot.summary$bootstrapped_paths
```

```
##           Original Est. Bootstrap Mean Bootstrap SD T Stat. 2.5% CI 97.5% CI
## PE  ->  BI         0.536          0.487        0.243   2.210  -0.040    0.907
## EE  ->  BI         0.028          0.043        0.190   0.146  -0.311    0.443
## SI  ->  BI         0.134          0.143        0.179   0.748  -0.209    0.489
## FC  ->  BI         0.137          0.141        0.192   0.714  -0.234    0.528
## PI  ->  BI         0.090          0.102        0.117   0.765  -0.104    0.356
## PR  ->  BI        -0.157         -0.119        0.133  -1.179  -0.379    0.144
```

```
graphics::plot(pls.boot,
               edge.display = TRUE,
               node.color = "lightblue",
               node.frame.color = "black")
```



5.6 Evaluate measurement models

5.6.1 Evaluate indicator reliability

```
mm.ir = pls.summary$loadings
mm.ir.df = cbind("Indicators" = rownames(mm.ir), as.data.frame(mm.ir)) %>%
  dplyr::mutate(dplyr::across(dplyr::where(is.numeric), ~ round(., 4)))
mm.ir.df
```

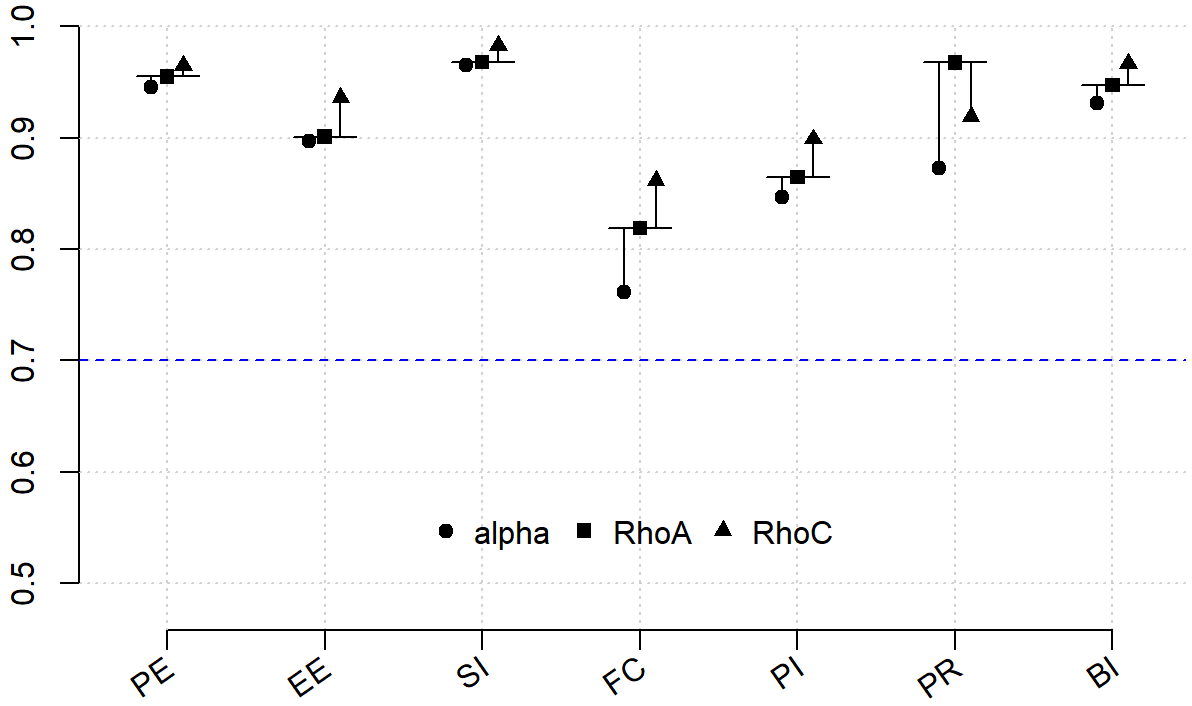
```
##     Indicators     PE     EE     SI     FC     PI     PR     BI
## PE1        PE1 0.9608 0.0000 0.0000 0.0000 0.0000 0.0000 0.0000
## PE2        PE2 0.9566 0.0000 0.0000 0.0000 0.0000 0.0000 0.0000
## PE3        PE3 0.9295 0.0000 0.0000 0.0000 0.0000 0.0000 0.0000
## EE1        EE1 0.0000 0.8893 0.0000 0.0000 0.0000 0.0000 0.0000
## EE2        EE2 0.0000 0.9428 0.0000 0.0000 0.0000 0.0000 0.0000
## EE3        EE3 0.0000 0.8997 0.0000 0.0000 0.0000 0.0000 0.0000
## SI1        SI1 0.0000 0.0000 0.9817 0.0000 0.0000 0.0000 0.0000
## SI2        SI2 0.0000 0.0000 0.9839 0.0000 0.0000 0.0000 0.0000
## FC1        FC1 0.0000 0.0000 0.0000 0.8941 0.0000 0.0000 0.0000
## FC2        FC2 0.0000 0.0000 0.0000 0.8835 0.0000 0.0000 0.0000
## FC3        FC3 0.0000 0.0000 0.0000 0.6749 0.0000 0.0000 0.0000
## PI1        PI1 0.0000 0.0000 0.0000 0.0000 0.8129 0.0000 0.0000
## PI2        PI2 0.0000 0.0000 0.0000 0.0000 0.6718 0.0000 0.0000
## PI3        PI3 0.0000 0.0000 0.0000 0.0000 0.9096 0.0000 0.0000
## PI4        PI4 0.0000 0.0000 0.0000 0.0000 0.9119 0.0000 0.0000
## PR1        PR1 0.0000 0.0000 0.0000 0.0000 0.0000 0.9391 0.0000
## PR2        PR2 0.0000 0.0000 0.0000 0.0000 0.0000 0.9347 0.0000
## PR3        PR3 0.0000 0.0000 0.0000 0.0000 0.0000 0.7876 0.0000
## BI1        BI1 0.0000 0.0000 0.0000 0.0000 0.0000 0.0000 0.9721
## BI2        BI2 0.0000 0.0000 0.0000 0.0000 0.0000 0.0000 0.9618
```

5.6.2 Evaluate internal consistency

```
mm.ic = pls.summary$reliability
mm.ic.df = cbind("Constructs" = rownames(mm.ic), as.data.frame(mm.ic)) %>%
  dplyr::mutate(dplyr::across(dplyr::where(is.numeric), ~ round(., 4))) %>%
  dplyr::select(Constructs, alpha, rhoA, rhoC)
mm.ic.df
```

```
##    Constructs  alpha   rhoA   rhoC
## PE         PE 0.9451 0.9549 0.9646
## EE         EE 0.8973 0.9008 0.9360
## SI         SI 0.9648 0.9677 0.9827
## FC         FC 0.7617 0.8190 0.8618
## PI         PI 0.8463 0.8646 0.8989
## PR         PR 0.8723 0.9675 0.9190
## BI         BI 0.9309 0.9470 0.9664
```

```
graphics::plot(pls.summary$reliability)
```



5.6.3 Evaluate convergent validity

```
mm.cv = pls.summary$reliability
mm.cv.df = cbind("Constructs" = rownames(mm.cv), as.data.frame(mm.cv)) %>%
  dplyr::mutate(dplyr::across(dplyr::where(is.numeric), ~ round(., 4))) %>%
  dplyr::select(Constructs, AVE)
mm.cv.df
```

```
##    Constructs    AVE
## PE         PE 0.9008
## EE         EE 0.8298
## SI         SI 0.9660
## FC         FC 0.6785
## PI         PI 0.6928
## PR         PR 0.7920
## BI         BI 0.9350
```

5.6.4 Evaluate discriminant validity

```
mm.dv = pls.boot.summary$bootstrapped_HTMT
mm.dv.df = cbind("Constructs" = rownames(mm.dv), as.data.frame(mm.dv)) %>%
  dplyr::mutate(dplyr::across(dplyr::where(is.numeric), ~ round(., 4)),
                Constructs = gsub(" -> ", "and", Constructs))
mm.dv.df
```

```
##            Constructs Original Est. Bootstrap Mean Bootstrap SD T Stat. 2.5% CI
## PE  ->  EE  PE and EE        0.8897         0.8898       0.0666 13.3688  0.7460
## PE  ->  SI  PE and SI        0.8680         0.8688       0.0401 21.6564  0.7849
## PE  ->  FC  PE and FC        0.9165         0.9193       0.0387 23.7128  0.8430
## PE  ->  PI  PE and PI        0.8258         0.8252       0.0515 16.0350  0.7147
## PE  ->  PR  PE and PR        0.4922         0.5043       0.0952  5.1686  0.3163
## PE  ->  BI  PE and BI        0.8180         0.8148       0.0586 13.9505  0.6829
## EE  ->  SI  EE and SI        0.7864         0.7854       0.0536 14.6640  0.6724
## EE  ->  FC  EE and FC        0.9873         0.9883       0.0563 17.5456  0.8805
## EE  ->  PI  EE and PI        0.8497         0.8482       0.0684 12.4300  0.7049
## EE  ->  PR  EE and PR        0.2727         0.3014       0.1415  1.9266  0.0725
## EE  ->  BI  EE and BI        0.7737         0.7741       0.0644 12.0180  0.6323
## SI  ->  FC  SI and FC        0.7606         0.7645       0.0605 12.5761  0.6390
## SI  ->  PI  SI and PI        0.6363         0.6360       0.0787  8.0833  0.4734
## SI  ->  PR  SI and PR        0.2932         0.3101       0.1124  2.6088  0.1037
## SI  ->  BI  SI and BI        0.7335         0.7292       0.0817  8.9816  0.5497
## FC  ->  PI  FC and PI        0.9575         0.9555       0.0757 12.6399  0.8073
## FC  ->  PR  FC and PR        0.5556         0.5750       0.1229  4.5214  0.3333
## FC  ->  BI  FC and BI        0.7838         0.7880       0.0750 10.4469  0.6375
## PI  ->  PR  PI and PR        0.5449         0.5472       0.0792  6.8780  0.3832
## PI  ->  BI  PI and BI        0.6920         0.6937       0.0710  9.7469  0.5440
## PR  ->  BI  PR and BI        0.2284         0.2623       0.1362  1.6772  0.0604
##            97.5% CI
## PE  ->  EE   1.0030
## PE  ->  SI   0.9416
## PE  ->  FC   0.9938
## PE  ->  PI   0.9170
## PE  ->  PR   0.6873
## PE  ->  BI   0.9119
## EE  ->  SI   0.8841
## EE  ->  FC   1.1030
## EE  ->  PI   0.9734
## EE  ->  PR   0.6198
## EE  ->  BI   0.8860
## SI  ->  FC   0.8785
## SI  ->  PI   0.7816
## SI  ->  PR   0.5445
## SI  ->  BI   0.8679
## FC  ->  PI   1.1046
## FC  ->  PR   0.8181
## FC  ->  BI   0.9325
## PI  ->  PR   0.6930
## PI  ->  BI   0.8201
## PR  ->  BI   0.5828
```

5.6 Evaluate structural model

5.6.1 Evaluate collinearity issues

```
sm.co = pls.summary$vif_antecedents
sm.co.df = cbind("Constructs" = names(sm.co[[1]]), as.data.frame(sm.co))
sm.co.df
```

```
##    Constructs       BI
## PE         PE 6.149061
## EE         EE 5.530130
## SI         SI 3.390875
## FC         FC 4.776124
## PI         PI 3.027738
## PR         PR 1.584300
```

5.6.2 Evaluate significance and relevance of relationships

```
sm.single.path = pls.boot.summary$bootstrapped_paths
sm.single.path.df = cbind("Paths" = rownames(sm.single.path), as.data.frame(sm.single.path)) %>%
  dplyr::mutate(pvalue = stats::pt(-abs(`T Stat.`), df = nrow(df.proc) - 1) * 2,
                dplyr::across(dplyr::where(is.numeric), ~ round(., 4)),
                Paths = gsub(" -> ", "to", Paths))
sm.single.path.df
```

```
##               Paths Original Est. Bootstrap Mean Bootstrap SD T Stat. 2.5% CI
## PE  ->  BI PE to BI        0.5363         0.4867       0.2427  2.2099 -0.0398
## EE  ->  BI EE to BI        0.0277         0.0430       0.1898  0.1461 -0.3110
## SI  ->  BI SI to BI        0.1337         0.1434       0.1787  0.7484 -0.2091
## FC  ->  BI FC to BI        0.1371         0.1408       0.1919  0.7145 -0.2336
## PI  ->  BI PI to BI        0.0897         0.1017       0.1173  0.7646 -0.1041
## PR  ->  BI PR to BI       -0.1567        -0.1188       0.1329 -1.1791 -0.3786
##            97.5% CI pvalue
## PE  ->  BI   0.9069 0.0293
## EE  ->  BI   0.4429 0.8841
## SI  ->  BI   0.4895 0.4559
## FC  ->  BI   0.5278 0.4765
## PI  ->  BI   0.3559 0.4462
## PR  ->  BI   0.1444 0.2410
```

```
sm.total.path = pls.boot.summary$bootstrapped_total_paths
sm.total.path.df = cbind("Paths" = rownames(sm.total.path), as.data.frame(sm.total.path)) %>%
  dplyr::mutate(pvalue = stats::pt(-abs(`T Stat.`), df = nrow(df.proc) - 1) * 2,
                dplyr::across(dplyr::where(is.numeric), ~ round(., 4)),
                Paths = gsub(" -> ", "to", Paths))
sm.total.path.df
```

```
##              Paths Original Est. Bootstrap Mean Bootstrap SD T Stat. 2.5% CI
## PE  ->  BI PE to BI        0.5363         0.4867       0.2427  2.2099 -0.0398
## EE  ->  BI EE to BI        0.0277         0.0430       0.1898  0.1461 -0.3110
## SI  ->  BI SI to BI        0.1337         0.1434       0.1787  0.7484 -0.2091
## FC  ->  BI FC to BI        0.1371         0.1408       0.1919  0.7145 -0.2336
## PI  ->  BI PI to BI        0.0897         0.1017       0.1173  0.7646 -0.1041
## PR  ->  BI PR to BI       -0.1567        -0.1188       0.1329 -1.1791 -0.3786
##            97.5% CI pvalue
## PE  ->  BI   0.9069 0.0293
## EE  ->  BI   0.4429 0.8841
## SI  ->  BI   0.4895 0.4559
## FC  ->  BI   0.5278 0.4765
## PI  ->  BI   0.3559 0.4462
## PR  ->  BI   0.1444 0.2410
```

5.6.3 Evaluate explanatory power

```
sm.rSquare = pls.summary$paths
sm.rSquare.df = cbind("Constructs" = rownames(sm.rSquare), as.data.frame(sm.rSquare)) %>%
  dplyr::mutate(dplyr::across(dplyr::where(is.numeric), ~ round(., 4)))
sm.rSquare.df
```

```
##        Constructs      BI
## R^2           R^2  0.6446
## AdjR^2     AdjR^2  0.6230
## PE             PE  0.5363
## EE             EE  0.0277
## SI             SI  0.1337
## FC             FC  0.1371
## PI             PI  0.0897
## PR             PR -0.1567
```

```
sm.fSquare = pls.summary$fSquare
sm.fSquare.df = cbind("Constructs" = rownames(sm.fSquare), as.data.frame(sm.fSquare)) %>%
  dplyr::mutate(dplyr::across(dplyr::where(is.numeric), ~ round(., 4)))
sm.fSquare.df
```

```
##    Constructs PE EE SI FC PI PR     BI
## PE         PE  0  0  0  0  0  0 0.1358
## EE         EE  0  0  0  0  0  0 0.0002
## SI         SI  0  0  0  0  0  0 0.0142
## FC         FC  0  0  0  0  0  0 0.0100
## PI         PI  0  0  0  0  0  0 0.0080
## PR         PR  0  0  0  0  0  0 0.0459
## BI         BI  0  0  0  0  0  0 0.0000
```

6 Export necessary data

```
export.list = list(items_desc = items.desc.df,
                   constructs_desc = constructs.desc.df,
                   reliability = mm.ir.df,
                   consistency = mm.ic.df,
                   convergence = mm.cv.df,
                   discriminant = mm.dv.df,
                   collinearity = sm.co.df,
                   single_paths = sm.single.path.df,
                   total_paths = sm.total.path.df,
                   r_square = sm.rSquare.df,
                   effect_size = sm.fSquare.df)

if(length(export.list) != 0){
  if (!file.exists(file.path(output.path, output.name))) {
    writexl::write_xlsx(export.list, file.path(output.path, output.name))
    cat(crayon::green("File successfully written."))
  } else {
    cat(crayon::red(glue::glue("Filename already used: {output.name}")))
    overwrite = readline(prompt = "Overwrite (1 for Yes, 0 for No): ")
    if (overwrite == "1") {
      writexl::write_xlsx(export.list, file.path(output.path, output.name))
      cat(crayon::green("File successfully overwritten"))
    } else {
      cat(crayon::red("File not overwritten"))
    }
  }
}
```

```
## File successfully written.
```