Journal of Applied Structural Equation Modeling

eISSN: 2590-4221 Journal of Applied Structural Equation Modeling: 2(5),1-35, 2021
DOI: 10.47263/JASEM.5(2)01

PLS-SEM USING R: AN INTRODUCTION TO cSEM AND SEMinR

Francis Chuah*[1], Mumtaz Ali Memon[2], T. Ramayah[3]
Jun-Hwa Cheah[4], Hiram Ting[5], and Tat Huei Cham[6]

[1]*School of Business Management, Universiti Utara Malaysia, Kedah, Malaysia*
[2]*NUST Business School, University of Sciences and Technology (NUST), Islamabad, Pakistan*
[3]*School of Management, Universiti Sains Malaysia, Penang, Malaysia*
[4]*School of Business and Economics, Universiti Putra Malaysia, Selangor, Malaysia*
[5]*Faculty of Hospitality and Tourism Management, UCSI University, Sarawak, Malaysia*
[6]*Graduate Business School, UCSI University, Kuala Lumpur, Malaysia*

[*]*francischuah@uum.edu.my*

ABSTRACT

In this editorial, we continue our discussion on the use of PLS-SEM statistical applications by introducing a few useful packages implemented in R. R is a free and open-source programming software that is used for statistical computing and graphics. To the best of our knowledge, most established PLS-SEM statistical applications have undergone rigorous testing and development in the R environment before being compiled and packaged as a standalone software. This editorial presents two new and well-maintained PLS-SEM packages in R, namely SEMinR and cSEM, to assist readers who prefer to work in a syntax-based environment and seek more flexibility to test their research models.

Keywords: *PLS-SEM, PLS Path Modeling, SEMinR, cSEM, Composite Modeling*

INTRODUCTION

Since its introduction in the 70s by Karl Jöreskog as the Jöreskog, Keesing and Wiley model, and later as the Linear Structural Relationship model (LISREL) (Ramayah et al., 2017), structural equation modeling (SEM) has become widely regarded as an important statistical tool in the social and behavioral sciences (Benitez et al., 2020).

To date, two variations of SEM are observed in the literature: covariance-based SEM (CB-SEM) and variance-based SEM. Both variations consist of two components, the measurement model and the structural model. While the role of the structural model is identical between the two variations, that is, to assess the path coefficient or relationship between two constructs, the composition of the measurement model varies between the two SEM approaches.

When measuring a theoretical concept, its observable indicators in CB-SEM are said to be the manifestation of the concept itself. This occurs under the assumption that a theoretical concept is the common cause of its indicator, thus implying a reflective measurement model, also known as

a common factor model (Hubona et al., 2021). In the variance-based SEM context, on the other hand, theoretical concepts are said to be formed or composed by the linear combination of its observable indicators. This suggests that variance-based estimators predict a composite model (Cho & Choi, 2020; Dijkstra, 2017).

CB-SEM is far more advanced and established than its variance-based SEM counterpart, given the research attention it has received since LISREL was introduced in the 1970s. Nonetheless, in recent years, variance-based SEM has begun to draw research interest and application in the social and behavioral sciences due to its flexibility and relaxed assumptions on distribution. The most widely known variance-based SEM approach is partial least squares path modeling (PLS-PM), which has been subjected to much scholarly debate since the early 2010s (Henseler et al., 2014; Rönkkö & Evermann, 2013). The debates, arguments, and counter-arguments from proponents and opponents of PLS-PM has, over the years, advanced PLS-PM estimators. Henseler (2018) advocates that the continuous debate over the use of PLS-PM has resulted in its transformation into a full-fledged SEM approach that can be used to conduct confirmatory research, explanatory research, exploratory research, descriptive research, and predictive research (p. 2-4). Notably, two types of PLS-PM research streams stand out from the rest. The first stream aims at using PLS-PM for causal-predictive research (Chin et al., 2020; Hwang et al., 2020), while the other seeks to use PLS-PM for confirmatory-explanatory research (Benitez et al., 2020).

Our previous editorial discussed three different commercial "stand-alone" PLS-SEM applications with graphical user interfaces available in the market. We provided a summary of the similarities and dissimilarities among the software to keep our readers informed about the uniqueness of each one.

In this editorial, we extend our discussion on the application of PLS-PM to the free open-source software, R, as an alternative to commercially available software, in case affordability is a concern for our readers. Applying PLS-PM in R requires minimal programming knowledge, which we think is manageable and does not require a huge learning effort. This editorial introduces two variations of PLS-PM packages, cSEM and SEMinR. To the best of our knowledge, these two packages are user-friendly for readers who do not have a basic syntax background. In addition, there are two books users can refer to for the full application of the packages. In "Composite-based structural equation modeling: analyzing latent and emergent variables" (Henseler, 2020), the author demonstrates the application of cSEM in model assessment while in "Partial Least Squares Structural Equation Modeling (PLS-SEM) using R (Hair et al., 2021), the authors demonstrate the application of SEMinR in model estimation.

cSEM

The cSEM package (https://cran.r-project.org/web/packages/cSEM/index.html) available in R is a statistical package that can be used to estimate, analyze, test, and study linear, nonlinear, hierarchical, and multigroup structural equation models using composite-based approaches and procedures, including estimation techniques such as PLS-PM, PLSc (Dijkstra & Henseler, 2015), OrdPLSc (Schuberth et al., 2018), robustPLSc (Schamberger et al., 2020), generalized structured component analysis (GSCA) (Hwang & Takane, 2004), generalized structured component analysis with uniqueness terms (GSCAm) (Hwang et al., 2017), generalized canonical correlation analysis (GCCA) (Kettenring, 1971), principal component analysis (PCA), as well as other several other tests and typical postestimation procedures (Rademaker, 2021; Rademaker & Schuberth, 2021).

This package was developed and is mainly maintained by Dr. Manuel Rademaker and Dr. Florian Schuberth. It is updated periodically, in line with the latest development of the PLS-PM technique. The overview on how to use cSEM is depicted in the following diagram.

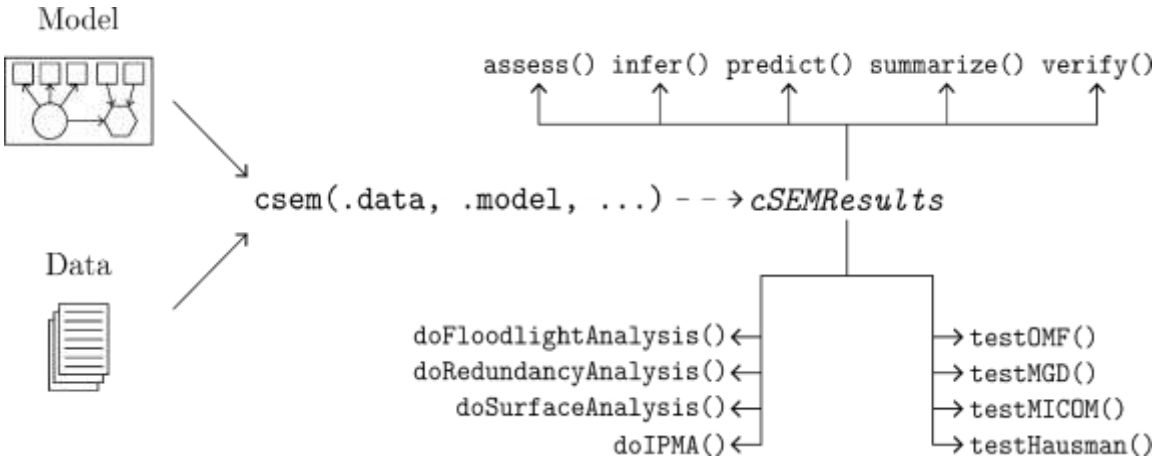


Figure 1: The basic usage of cSEM

Source: https://m-e-rademaker.github.io/cSEM/

To begin with, cSEM users are expected to have a model and a dataset to be incorporated into cSEM. In the R environment, to estimate a model, the model has to be specified using an equation syntax. cSEM uses lavaan syntax (https://lavaan.ugent.be/tutorial/syntax1.html) for model specification (Rosseel, 2012). Following through, the model is compiled and estimated using built-in `csem()` functions. Finally, users of cSEM can use one of the postestimation functions to assess the result output.

In short, cSEM follows the three to four-step procedure below to estimate and analyze a model:

1. Prepare and load the data into the R environment (preferably in .csv, .xlsx, or .rda format)
2. Specify a model using the lavaan syntax
3. Use `csem (.data, .model)` to compile and estimate the model
4. Apply one of the postestimation functions to view the result output

Appendix A of the online supplements for this editorial provides a step-by-step guideline on the specification and estimation of a simple model using cSEM.

Model specification and philosophy of cSEM

The cSEM package guides users in selecting the appropriate PLS approach for the specified model. Specifically, latent variables (common factor) and emergent variables (composite) can be specified. cSEM applies, by default, a correction for attenuation to obtain consistent parameter estimates for a latent variable (common factor). Against this background, cSEM is well-suited for confirmatory-explanatory research. In addition, cSem can be used in exploratory research, descriptive research, predictive research, and auxiliary theory (see Henseler, 2018; Henseler, 2021). For more details about the choice between latent and emergent variables to model abstract

concepts, including corresponding auxiliary theories, interested readers are referred to Henseler (2021).

Given the aforementioned discussion, we use the following figure of a syntax excerpt to introduce some of the model specification syntax used in cSEM.

```
model <- "
LOY ~ IMG + SAT                  # Structural model

IMG <~ imag1 + imag2 + imag3     # Composite
SAT =~ sat1 + sat2 + sat3        # Common factor
LOY =~ loy1 + loy2 + loy3        # Common factor
"
```

The above syntax excerpt presents a model in which **LOY** (endogenous variable) is predicted by **IMG** (exogenous variable 1) and **SAT** (exogenous variable 2). The first exogenous variable, **IMG**, is an emergent variable (composite) made up of three indicators: *imag1*, *imag2*, and *imag3*, while the second exogenous variable, **SAT**, and the endogenous variable, **LOY**, are latent variables (common factor) measured by three indicators each: *sat1*, *sat2*, *sat3* and *loy1*, *loy2*, and *loy3*, respectively.

cSEM applies different types of operators to represent model specification. Specifically, the operator "<~" tells cSEM that the construct to its left is modelled as a **composite.** The operator "=~" tells cSEM that the construct to its left is modelled as a **common factor** and finally, the operator "~" denotes a regression equation which identifies the endogenous variable (left-hand side) and exogenous variable (right-hand side) for cSEM.

cSEM also permits the specification of hierarchical (second-order) models using these operators, which is demonstrated in the following syntax excerpt.

```
model <- "
VAL ~ SAT + QUA                  # Structural model

VAL =~ val1 + val2 + val3        # Common factor
SAT =~ sat1 + sat2 + sat3        # Common factor

IMG =~ imag1 + imag2 + imag3     # First-order common factor
EXP =~ exp1 + exp2 + exp3        # First-order common factor

QUA <~ IMG + EXP                 # Second-order composite
"
```

The above syntax presents a model in which **VAL** (endogenous variable) is predicted by two exogenous variables, **SAT** and **QUA**. The endogenous variable, **VAL**, is a common factor measured by three indicators: *val1*, *val2*, and *val3*. **SAT** (exogenous variable 1) is a common factor measured by three indicators (*sat1*, *sat2*, and *sat3*), while **QUA** (exogenous variable 2) is a hierarchical (second order) composite made up of two first-order common factor variables, **IMG** and **EXP**. Both **IMG** and **EXP** are modelled as common factors measured by three indicators each.

Estimation and Postestimation of cSEM

`csem()` is the central function of the cSEM package. Once users complete the model specification, `csem()` is used to compile and estimate the model. The following excerpt depicts the use of `csem()` involving a dataset and a model.

```
model <-"     # An object named "model" comprising the specified model.
LOY ~ IMG + SAT

IMG <~ imag1 + imag2 + imag3
SAT =~ sat1 + sat2 + sat3
LOY =~ loy1 + loy2 + loy3
"

Myplsmodel <- csem(.data = loyalty, .model = model)
```

The above syntax demonstrates the use of the `csem()` function for cSEM to begin estimating the model. `Myplsmodel` denotes an object name given to cSEM so that results and estimations can be generated. `.data = loyalty` denotes that the `csem()` function is reading a dataset named "loyalty", while `.model = model` denotes that the `csem()` function is estimating an object named "model", which can be understood as the theoretical model the user intends to examine in a syntax equation. In general, various arguments can be specified for adjustment in the `csem()` function, such as the PLS inner weighting scheme, the approach used to estimate second-order models, whether a correction for attenuation should be performed, or whether multi-core processing is conducted. For more information, interested readers are referred to the `csem()` function manual.

The cSEM package provides six (6) major postestimation functions, 4 (four) `test_*` family of postestimation functions, and three (3) `do_*` family of postestimation functions, which are explained in the following section.

The six (6) major postestimation functions are:

- `assess()`
- `infer()`
- `predict()`
- `summarize()`
- `verify()`
- `exportToExcel()`

The `assess()` function evaluates the quality of the estimated model. It is noted that statistical tests, such as the test for the overall model, are not conducted via this function. Rather, common aspects of model assessment are reported in this section, including fit indices, reliability estimates, common validity criteria, and other quality-related indices that do not require a formal test procedure.

The `infer()` function calculates common inferential quantities, such as estimated standard errors and/or confidence intervals. Nonetheless, the developer suggests that users opt for the `summarize()` function as it has a better user-friendly print method.

The `predict()` function is used to predict indicator scores of endogenous constructs based on the procedure introduced by Shmueli et al. (2016).

The `summarize()` function summarizes a model. This function provides estimates in a user-friendly data frame and allows for the calculation of various bootstrap confidence intervals for the parameter estimates. The developer acknowledges that this function is more convenient for users who intend to present their results in a paper or presentation.

The `verify()` function verifies the admissibility of the estimated quantities for a given model. Results that violate the estimation assumption are deemed inadmissible.

The `exportToExcel()` function conveniently exports the results from `assess()`, `predict()`, `summarize()` and `testOMF()` to an .xlsx file.

Based on the earlier estimation example, users can execute any of the six (6) postestimation functions by adding the following command:

```
model <-"     # An object named "model" is given for the theoretical model.
LOY ~ IMG + SAT

IMG <~ imag1 + imag2 + imag3
SAT =~ sat1 + sat2 + sat3
LOY =~ loy1 + loy2 + loy3
"
Myplsmodel <- csem(.data = loyalty, .model = model)
Myplsmodel

summarize (Myplsmodel)    # summarizes the model
assess (Myplsmodel)       # assesses the model
predict (Myplsmodel)      # predicts the indicator scores of endogenous
constructs
```

The four (4) test_* family of postestimation functions are:

- `testHausman()`
- `testOMF()`
- `testMGD()`
- `testMICOM()`

`testHausman()` is a regression-based Hausman test for SEM.

`testOMF()`is a boostrap-based overall model fit test suggested by Beran and Srivastava (1985). See also Djikstra and Henseler (2015).

`testMGD()` is a test for group differences using several different approaches, such as the approach described in Klesel et al. (2019). For an overview of group difference tests that are implemented, interested readers can refer to Klesel et al. (in press).

`testMICOM()` is a test of the measurement invariance of composites proposed by Henseler et al. (2016).

Finally, the three (3) `do_*` family of postestimation functions are:

- `doIPMA()`

- `doNonlinearEffectAnalysis()`
- `doRedundancyAnalysis()`

`doIPMA()` performs an importance-performance matrix analysis (IPMA).

`doNonlinearEffectAnalysis()` performs nonlinear effect analysis, such as the floodlight and surface analysis described in Spiller et al. (2013).

`doRedundancyAnalysis()` performs redundancy analysis (RA) to assess the validity of formative constructs, as suggested by Hair et al. (2016) with reference to Chin (1998).

SEMinR

The SEMinR package (https://cran.r-project.org/web/packages/seminr/index.html) allows users to employ common SEM modeling terminology (e.g., reflective, composite, interactions, etc.). This package was developed by Prof. Dr. Soumya Ray and Dr. Nicholas Danks, who subsequently invited André Calero Valdez to be a primary addition to the developer team. This package was also supported by their key contributors, namely Juan Manuel Velasquez Estrada, James Uanhoro, Johannes Nakayama, Lilian Koyan, Laura Burbach, Arturo Heynar Cano Bejar, and Susanne Adler.

SEMinR allows users to apply either PLS-PM or CB-SEM to estimate SEM models. As noted by the developer, SEMinR uses its own PLS-PM estimation engine to assess a PLS-PM model, but integrates with the lavaan package for CB-SEM or confirmatory factor analysis (CFA) estimation.

To use the SEMinR package in R, users are expected to adhere to the following three-step approach to specify and estimate a structural equation model:

1. Describe the measurement model for each construct and its items, including interaction terms (for moderation) and other measurement features.
2. Describe the structural model of causal relationship between constructs (and interaction terms).
3. Bind the measurement model and structural model together to estimate the model using the relevant estimation approach (PLS-PM, CB-SEM, or CFA).

Appendix B of the online supplements for this editorial provides a step-by-step guideline on the specification and estimation of a simple model using SEMinR.

Measurement model description

SEMinR uses the following syntax to describe the measurement model:

- `constructs()`
- `composite()` or `reflective()`
- `interaction_term()` or `higher_composite()`
- `multi_items()` or `single_items()`

`constructs()` gathers all construct in the measurement model. The `composite()` or `reflective()` functions define the measurement mode of individual constructs in the model. `interaction_term()` specifies interactions while `higher_composite()` specifies higher order constructs. Finally, `multi_items()` or `single_items()` define the items of a construct.

The following figure is an excerpt of the syntax used to specify a model in SEMinR. The object `measurements` is used to store the measurement model. It should be noted that the structural model is not specified in the measurement model stage.

```
measurements <- constructs(
       composite("Image",           multi-items("IMAG", 1:5), weights = mode_B
       composite("Expectation",     multi-items("CUEX", 1:3), weights = mode_A
       reflective("Satisfaction",   multi-items("CUSA", 1:3),
       higher_composite("QUA", c("Image", "Expectation"), method = two_stage)
)
```

In the example above, `composite()` is used for the construct "**Expectation**", which has three indicators, *CUEX1*, *CUEX2*, and *CUEX3* to be estimated with composite mode A (correlation weights). In a similar vein, `composite()` is used for the construct "**Image**" which has five indicators, *IMAG1*, *IMAG2*, *IMAG3*, *IMAG4*, and *IMAG5*, to be estimated with composite mode B (regression weights). Alternatively, `reflective()` is used in CB-SEM/CFA/PLSc to describe the reflective common factor measurement of "**Satisfaction**" with three indicators *CUSA1*, *CUSA2*, and *CUSA3*. `higher_composite()`is used to define the higher order construct "**QUA**", which is measured by two lower order constructs, "**Image**" and "**Expectation**".

Structural model description

The following syntax is used to describe the structural model in SEMinR:

- `relationships()`
- `paths()`

The `relationships()` syntax compiles the structural model and structural relationships among all the constructs in the specified model. The `paths()` syntax describes the relationships between the sets of antecedents and outcomes. The following figure depicts the usage of these syntaxes in specifying a structural model. The object `structural` is used to store the structural model. The measurement model is not specified in the structural model syntax.

```
structural <- relationships(
       paths(from = "Image",        to = c("Expectation", "Satisfaction")),
       paths(from = "Expectation",  to = c("Satisfaction", "Complaints")),
       paths(from = "Satisfaction", to = "Complaints"),
       paths(from = "Complaints",          to = "Loyalty")
)
```

In the above example, `relationships()` compiles the following relationships where `paths()`is used:

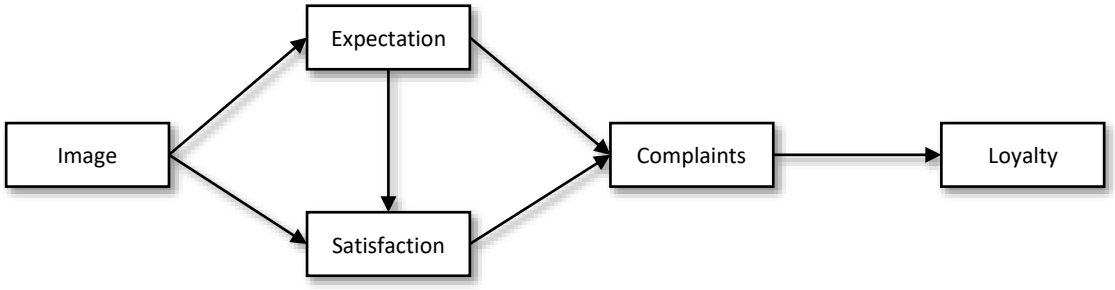


Figure 2: The structural model estimated in SEMinR

Estimation and Bootstrapping

SEMinR uses the following syntaxes to estimate either a full SEM model or to conduct CFA as described by the measurement and structural models.

- `estimate_pls()` - estimates the parameter of a PLS-PM model
- `estimate_cfa()` - estimates the parameter of a CFA model using the lavaan package
- `estimate_cbsem()` - estimates the parameter of a CB-SEM model using the lavaan package

The above-mentioned functions require the combination of the following parameters:

- `data`
- `measurement_model`
- `structural_model`
- `inner_weights`

The `data:` parameter refers to the dataset containing the measurement model items specified in `constructs()`. The `measurement_model` parameter is the measurement model described by the `constructs()` function. The `structural_model` parameter is the structural model described by the `paths()` function, whereas the `inner_weights` parameter represents the weighting scheme for path estimation. Two types of weighting schemes can be applied, namely `path_weighting` for path weighting (default) or `path_factorial` for factor weighting.

To bootstrap an SEM model, SEMinR incorporates the following syntax to execute high-performance bootstrapping.

- `bootstrap_model()`

The above function requires the combination of the following parameters:

- `seminr_model`
- `nboot`
- `cores`

The `seminr_model` parameter refers to the SEM model provided by `estimate_pls()`. The `nboot` parameter is the number of bootstrap subsamples to generate while the `cores` parameter refers to the multi-core processing of the user's computer. In most cases, SEMinR will automatically detect and utilize all available cores.

The following is a syntax excerpt for the estimation of a simple SEM model alongside syntaxes to develop the measurement and structural models of the SEM model. The bootstrapping

```
# define the measurement model
measurements <- constructs(
       composite("Image",           multi-items("IMAG", 1:5),
       composite("Expectation",     multi-items("CUEX", 1:3),
       composite("Satisfaction",    multi-items("CUSA", 1:3),
       composite("Complaints",      multi-items("COMP", 1:5),
       composite("Loyalty",  multi-items("CUSA", 1:3)
)
# define the structural model
structural <- relationships(
       paths(from = "Image",        to = c("Expectation", "Satisfaction")),
       paths(from = "Expectation",  to = c("Satisfaction", "Complaints")),
       paths(from = "Satisfaction", to = c("Complaints"),
       paths(from = "Complaints",          to = "Loyalty")
)
# syntax to estimate the model
model_est <- estimate_pls(
       data = mydataset,
       measurement model = measurements,
       structural model = structural,
       inner_weights = path_weighting

# syntax to bootstrap the model
boot_model_est <- bootstrap_model(seminr_model = model_est,
                                  nboot = 1000,
                                  cores = 2)

summary(model_est)
model_summary <- summary(model_est)
summary(boot_model_est)
boot_model_summary <- summary(boot_model_est)
```

command is executed after the estimation command.

To report the estimation and bootstrapping results, the following syntax is used at the end of the scripting. Executing this syntax will generate the list of information required to report the assessment of the measurement model as well as the estimation of the structural model.

- `summary()`
- `model_summary <- summary()`

Last but not least, SEMinR incorporates a special function which allows its users to plot all supported models using dot language and the graphViz.js widget from the `DiagrammeR` package. The `plot()` and `save_plot()`commands are used to plot and save a model, respectively.

A Final Note

By introducing two open-source packages in R, namely cSEM and SEMinR, this editorial proposes two alternatives to the existing commercial PLS software we discussed in our previous editorial. We acknowledge the steep learning curve required to use R, especially for readers who do not have a basic understanding of programming language. Nonetheless, we observe that the language itself is becoming more user-friendly and believe that with minimal effort to understand the logic and basics behind the language, one can easily adapt to the syntax environment.

We would like to take this opportunity to thank all the package developers who have worked relentlessly behind the scenes to maintain these packages so they are freely available for our perusal. We also hope that this editorial spurs readers' interest in learning R, which is a powerful tool for data analytics typically involving simulation and rigorous testing of estimation processes.

ACKNOWLEDGEMENT

We would like to thank Prof. Dr. Soumya Ray (National Tsing Hua University, Taiwan), Dr Manuel Rademaker (Universitat Wurzburg), and Dr Florian Schuberth (University of Twente) for their comments to improve the initial draft of this manuscript.

REFERENCES

Beran, R., & Srivastava, M. S. (1985). Bootstrap tests and confidence regions for functions of a covariance matrix. *The Annals of Statistics*, 95-115.

Chin, W. W. (1998). The partial least squares approach to structural equation modeling. *Modern methods for business research*, *295*(2), 295-336.

Chin, W., Cheah, J. H., Liu, Y., Ting, H., Lim, X. J., & Cham, T. H. (2020). Demystifying the role of causal-predictive modeling using partial least squares structural equation modeling in information systems research. *Industrial Management & Data Systems*, *120*(12), 2161-2209

Cho, G., & Choi, J. Y. (2020). An empirical comparison of generalized structured component analysis and partial least squares path modeling under variance-based structural equation models. *Behaviormetrika*, *47*(1), 243-272.

Danks, N. P., & Ray, S. (2020). *SEMinR*. https://cran.r-project.org/web/packages/seminr/vignettes/SEMinR.html

Dijkstra, T. K., & Henseler, J. (2015). Consistent and asymptotically normal PLS estimators for linear structural equations. *Computational statistics & data analysis*, *81*, 10-23.

Dijkstra, T. K. (2017). A perfect match between a model and a mode. In *Partial least squares path modeling* (pp. 55-80). Springer, Cham.

Hair Jr, J. F., Hult, G. T. M., Ringle, C. M., & Sarstedt, M. (2021). *A primer on partial least squares structural equation modeling (PLS-SEM)*. 3rd edition, Sage publications.

Hair, J. F., Hult, G. T. M., Ringle, C. M., Sarstedt, M., Danks, N. P., & Ray, S. (2021). *Partial Least Squares Structural Equation Modeling (PLS-SEM) Using R*. Springer International Publishing. https://doi.org/10.1007/978-3-030-80519-7

Henseler, J. (2018). Partial least squares path modeling: Quo vadis?. *Quality & Quantity*, *52*(1), 1-8.

Henseler, J. (2021). *Composite-based structural equation modeling: analyzing latent and emergent variables*. Guilford Publications.

Henseler, J., Dijkstra, T. K., Sarstedt, M., Ringle, C. M., Diamantopoulos, A., Straub, D. W., ... & Calantone, R. J. (2014). Common beliefs and reality about PLS: Comments on Rönkkö and Evermann (2013). *Organizational research methods*, *17*(2), 182-209.

Henseler, J., Ringle, C. M., & Sarstedt, M. (2016). Testing measurement invariance of composites using partial least squares. *International marketing review*, *33*(3), 405-431

Hwang, H., & Takane, Y. (2004). Generalized structured component analysis. *Psychometrika*, *69*(1), 81-99.

Hwang, H., Takane, Y., & Jung, K. (2017). Generalized structured component analysis with uniqueness terms for accommodating measurement error. *Frontiers in psychology*, *8*, 2137.

Hwang, H., Sarstedt, M., Cheah, J. H., & Ringle, C. M. (2020). A concept analysis of methodological research on composite-based structural equation modeling: bridging PLSPM and GSCA. *Behaviormetrika*, *47*(1), 219-241.

Kettenring, J. R. (1971). Canonical analysis of several sets of variables. *Biometrika*, *58*(3), 433-451.

Klesel, M., Schuberth, F., Henseler, J., & Niehaves, B. (2019). A test for multigroup comparison using partial least squares path modeling. *Internet research*, *33*(3), 405-431.

Klesel, M., Schuberth, F., Niehaves, B., & Henseler, J. (Accepted/In press). Multigroup analysis in information systems research using PLS-PM: A systematic investigation of approaches. *Data Base for Advances in Information Systems*.

Memon, M. A., Ting, H., Cheah, J. H., Thurasamy, R., Chuah, F., & Cham, T. H. (2021). PLS-SEM Statistical Programs: A Review. Journal of Applied Structural Equation Modeling, *5*(1), 1-14.

Rademaker, M. (2021). *Introduction to cSEM*. https://cran.r-project.org/web/packages/cSEM/vignettes/cSEM.html

Rademaker, M., & Schuberth, F. (2021). *cSEM: Composite-based SEM*. https://m-e-rademaker.github.io/cSEM/

Rönkkö, M., & Evermann, J. (2013). A critical examination of common beliefs about partial least squares path modeling. *Organizational Research Methods*, *16*(3), 425-448.

Rosseel, Y. (2012). Lavaan: An R package for structural equation modeling and more. Version 0.5–12 (BETA). *Journal of statistical software*, *48*(2), 1-36.

Schamberger, T., Schuberth, F., Henseler, J., & Dijkstra, T. K. (2020). Robust partial least squares path modeling. *Behaviormetrika*, *47*(1), 307-334.

Schuberth, F., Henseler, J., & Dijkstra, T. K. (2018). Partial least squares path modeling using ordinal categorical indicators. *Quality & Quantity*, *52*(1), 9-35.

Shmueli, G., Ray, S., Estrada, J. M. V., & Chatla, S. B. (2016). The elephant in the room: Predictive performance of PLS models. *Journal of Business Research*, *69*(10), 4552-4564.

Spiller, S. A., Fitzsimons, G. J., Lynch Jr, J. G., & McClelland, G. H. (2013). Spotlights, floodlights, and the magic number zero: Simple effects tests in moderated regression. *Journal of marketing research*, *50*(2), 277-288.

APPENDIX A

Installing and setting up cSEM

You must install the following package in R or RStudio to be able to use it:

```
install.packages("cSEM")
install.packages("matrixStats")
install.packages("listviewer")
```

The cSEM package is installed once, but you need to load it in R / RStudio in every session you want to use it:

```
library(cSEM)
```

Importing / Loading data

You must load your data into a data frame from sources acceptable in R (CSV, Rda, Excel, etc.). The column names must be the names of your items. We use cSEM bundled with a dataset from the Customer Satisfaction Index to help beginners follow the coding/programming process easily. Since the dataset is embedded in the cSEM package, we use the following syntax instead for the rest of the illustration:

```
data(satisfaction)
```

Following through, you can use the following syntax to check if the data has loaded successfully:

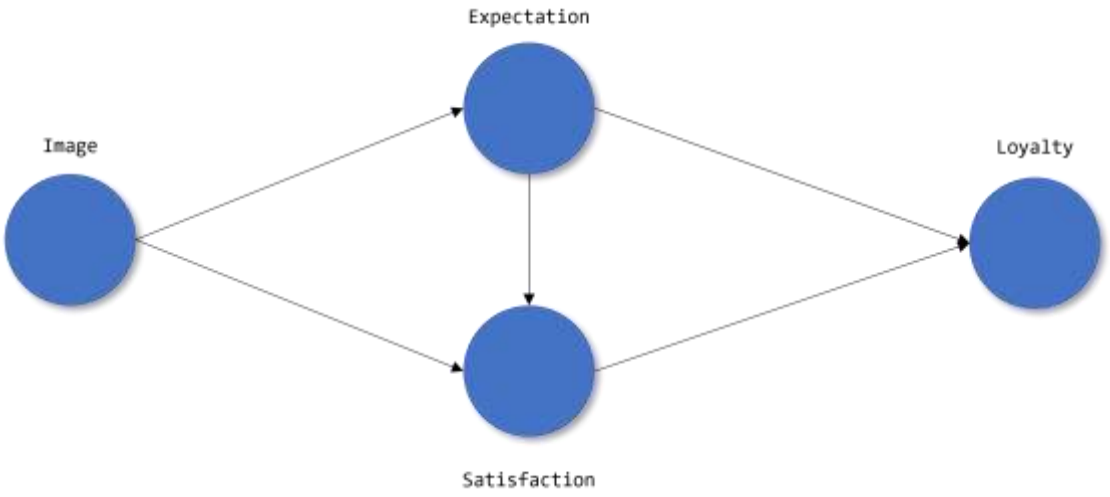
```
dim(satisfaction)  #this will show you the number of rows and columns available in your
dataset

## [1] 250  27

head(satisfaction) #this will show you the first six rows of your dataset

##   imag1 imag2 imag3 imag4 imag5 expe1 expe2 expe3 expe4 expe5 qual1 qual2 qual3
## 1     8     8     9     5     6     9     9     5     8     9     6     8     2
## 2     9     9    10     9     7     9     8    10     8     9     7     9     7
## 3     9     8     8     8     8     8     8     8     9     9     7     9     6
## 4     8     9     8     9     7    10     6     3    10    10     7     3     2
## 5    10    10     8    10     8     7     9     8     9     8     7     9     8
## 6     7     8     8     8     8     8     8     8     9    10     6     6     7
##   qual4 qual5 val1 val2 val3 val4 sat1 sat2 sat3 sat4 loy1 loy2 loy3 loy4
## 1     7     7    7   10    5    6    6    7    6    7    9    9    6    6
## 2     8     7    8    8    7    8    8    7    8    7    8    9    8    8
## 3     8     6    8    9    7    8    9    8    8    8    9    8    9    9
## 4    10     8    6    7    6    4    7    7    7    6    6    6    5    5
## 5     9     8    9   10   10    8    8    8    8    7    8    9    9    9
## 6     7     7    8    7    6    5    7    6    6    7    7    8    8    7
```

We will be using the model depicted below for subsequent illustrations:



satisfaction

Specifying the model

In this illustration, we specify all variables as ***composite***. As highlighted in the manuscript, users need to specify the measurement models as either composite or common factor. To specify a common factor, the term ***reflective*** is used. As stated in the manuscript, cSEM uses the lavaan syntax for model specification, which comprises the following: 1. the `=~` operator is used to represent a latent variable / common factor; 2. the `<~` operator is used to represent a composite; 3. the `~` operator is used to represent a regression; and 4. the `~~` operator is used to represent the error of (co)variances, indicator correlations, or correlations between exogenous constructs

```
model <- "
Expectation ~ Image
Satisfaction ~ Expectation
Satisfaction ~ Image
Loyalty ~ Expectation
Loyalty ~ Satisfaction

Image <~ imag1 + imag2 + imag3 + imag4 + imag5
Expectation <~ expe1 + expe2 + expe3 + expe4 + expe5
Satisfaction <~ sat1 + sat2 + sat3 + sat4
Loyalty <~ loy1 + loy2 + loy3 + loy4
"
# We begin by specifying the structural model, followed by the measurem
ent model
```

Estimating the model and reporting the estimated model

The estimation of the model is done using the `csem()` function. The following syntax is used.

```
est_model <- csem(.data = satisfaction, .model = model)

# est_model refers to the name of the object created to be estimated
# .data refers to the dataset used
# .model refers to the object of the model specified above
```

Applying the postestimation function to get a summary of the results

`cSEM` uses the following postestimation commands to acquire results: * `assess()` * `infer()` * `predict()` * `summarize()` * `verify()`

Given the above example, we can use the following command to call for results:

```
summarize(est_model)
```

```
## ____________________________________________________________________________
## ----------------------------- Overview -----------------------------------
##
##  General information:
##  ------------------------
##  Estimation status                = Ok
##  Number of observations           = 250
##  Weight estimator                 = PLS-PM
##  Inner weighting scheme           = "path"
##  Type of indicator correlation    = Pearson
##  Path model estimator             = OLS
##  Second-order approach            = NA
##  Type of path model               = Linear
##  Disattenuated                    = No
##
##  Construct details:
##  ------------------
##  Name          Modeled as     Order         Mode
##
##  Image         Composite      First order   "modeB"
##  Expectation   Composite      First order   "modeB"
##  Satisfaction  Composite      First order   "modeB"
##  Loyalty       Composite      First order   "modeB"
##
## ------------------------------ Estimates ---------------------------------
##
## Estimated path coefficients:
## ============================
##   Path                          Estimate  Std. error   t-stat.   p-value
##   Expectation ~ Image             0.6107          NA        NA        NA
##   Satisfaction ~ Image            0.5238          NA        NA        NA
##   Satisfaction ~ Expectation      0.3165          NA        NA        NA
##   Loyalty ~ Expectation           0.1007          NA        NA        NA
##   Loyalty ~ Satisfaction          0.6608          NA        NA        NA
##
## Estimated loadings:
## ===================
```

```
##   Loading                 Estimate  Std. error   t-stat.   p-value
##   Image =~ imag1            0.5498          NA        NA        NA
##   Image =~ imag2            0.8240          NA        NA        NA
##   Image =~ imag3            0.8728          NA        NA        NA
##   Image =~ imag4            0.4816          NA        NA        NA
##   Image =~ imag5            0.8194          NA        NA        NA
##   Expectation =~ expe1      0.7096          NA        NA        NA
##   Expectation =~ expe2      0.8628          NA        NA        NA
##   Expectation =~ expe3      0.6813          NA        NA        NA
##   Expectation =~ expe4      0.8190          NA        NA        NA
##   Expectation =~ expe5      0.7882          NA        NA        NA
##   Satisfaction =~ sat1      0.9407          NA        NA        NA
##   Satisfaction =~ sat2      0.9180          NA        NA        NA
##   Satisfaction =~ sat3      0.7265          NA        NA        NA
##   Satisfaction =~ sat4      0.8265          NA        NA        NA
##   Loyalty =~ loy1           0.9308          NA        NA        NA
##   Loyalty =~ loy2           0.5680          NA        NA        NA
##   Loyalty =~ loy3           0.9235          NA        NA        NA
##   Loyalty =~ loy4           0.4841          NA        NA        NA
##
## Estimated weights:
## ==================
##   Weight                  Estimate  Std. error   t-stat.   p-value
##   Image <~ imag1           -0.0451          NA        NA        NA
##   Image <~ imag2            0.1995          NA        NA        NA
##   Image <~ imag3            0.4869          NA        NA        NA
##   Image <~ imag4            0.0605          NA        NA        NA
##   Image <~ imag5            0.4958          NA        NA        NA
##   Expectation <~ expe1      0.0786          NA        NA        NA
##   Expectation <~ expe2      0.4147          NA        NA        NA
##   Expectation <~ expe3      0.1448          NA        NA        NA
##   Expectation <~ expe4      0.3915          NA        NA        NA
##   Expectation <~ expe5      0.2121          NA        NA        NA
##   Satisfaction <~ sat1      0.4335          NA        NA        NA
##   Satisfaction <~ sat2      0.3718          NA        NA        NA
##   Satisfaction <~ sat3     -0.0013          NA        NA        NA
##   Satisfaction <~ sat4      0.3047          NA        NA        NA
##   Loyalty <~ loy1           0.5701          NA        NA        NA
##   Loyalty <~ loy2           0.0655          NA        NA        NA
##   Loyalty <~ loy3           0.5178          NA        NA        NA
##   Loyalty <~ loy4          -0.0951          NA        NA        NA
##
## Estimated indicator correlations:
## =================================
##   Correlation       Estimate  Std. error   t-stat.   p-value
##   imag1 ~~ imag2      0.6437          NA        NA        NA
##   imag1 ~~ imag3      0.5433          NA        NA        NA
##   imag1 ~~ imag4      0.5036          NA        NA        NA
##   imag1 ~~ imag5      0.3459          NA        NA        NA
##   imag2 ~~ imag3      0.7761          NA        NA        NA
##   imag2 ~~ imag4      0.4495          NA        NA        NA
##   imag2 ~~ imag5      0.5010          NA        NA        NA
##   imag3 ~~ imag4      0.4622          NA        NA        NA
##   imag3 ~~ imag5      0.4590          NA        NA        NA
##   imag4 ~~ imag5      0.2603          NA        NA        NA
```

```
##   expe1 ~~ expe2      0.5353           NA         NA         NA
##   expe1 ~~ expe3      0.4694           NA         NA         NA
##   expe1 ~~ expe4      0.5699           NA         NA         NA
##   expe1 ~~ expe5      0.5562           NA         NA         NA
##   expe2 ~~ expe3      0.5467           NA         NA         NA
##   expe2 ~~ expe4      0.5038           NA         NA         NA
##   expe2 ~~ expe5      0.6116           NA         NA         NA
##   expe3 ~~ expe4      0.4273           NA         NA         NA
##   expe3 ~~ expe5      0.4982           NA         NA         NA
##   expe4 ~~ expe5      0.5279           NA         NA         NA
##   sat1 ~~ sat2        0.8202           NA         NA         NA
##   sat1 ~~ sat3        0.6609           NA         NA         NA
##   sat1 ~~ sat4        0.6663           NA         NA         NA
##   sat2 ~~ sat3        0.6997           NA         NA         NA
##   sat2 ~~ sat4        0.6285           NA         NA         NA
##   sat3 ~~ sat4        0.5942           NA         NA         NA
##   loy1 ~~ loy2        0.4903           NA         NA         NA
##   loy1 ~~ loy3        0.7323           NA         NA         NA
##   loy1 ~~ loy4        0.5315           NA         NA         NA
##   loy2 ~~ loy3        0.5124           NA         NA         NA
##   loy2 ~~ loy4        0.4453           NA         NA         NA
##   loy3 ~~ loy4        0.4771           NA         NA         NA
##
## ------------------------------ Effects ----------------------------------
##
## Estimated total effects:
## ========================
##   Total effect                  Estimate  Std. error   t-stat.   p-value
##   Expectation ~ Image             0.6107          NA        NA        NA
##   Satisfaction ~ Image            0.7171          NA        NA        NA
##   Satisfaction ~ Expectation      0.3165          NA        NA        NA
##   Loyalty ~ Image                 0.5353          NA        NA        NA
##   Loyalty ~ Expectation           0.3099          NA        NA        NA
##   Loyalty ~ Satisfaction          0.6608          NA        NA        NA
##
## Estimated indirect effects:
## ===========================
##   Indirect effect          Estimate  Std. error   t-stat.   p-value
##   Satisfaction ~ Image       0.1933          NA        NA        NA
##   Loyalty ~ Image            0.5353          NA        NA        NA
##   Loyalty ~ Expectation      0.2091          NA        NA        NA
## ____________________________________________________________________________
```

```
assess(est_model)
```

```
## ____________________________________________________________________________
##
##  Construct           AVE           R2          R2_adj
##  Expectation         NA          0.3730        0.3704
##  Satisfaction        NA          0.5770        0.5736
##  Loyalty             NA          0.5315        0.5277
##
## -------------------------- Distance and fit measures ---------------------
-----
##
```

```
##  Geodesic distance           = 0.1378006
##  Squared Euclidian distance  = 0.3724618
##  ML distance                 = 0.6823565
##
##  Chi_square     = 169.9068
##  Chi_square_df  = 1.665753
##  CFI            = 0.973347
##  CN             = 186.4956
##  GFI            = 0.9106742
##  IFI            = 0.9738701
##  NFI            = 0.9370905
##  NNFI           = 0.9600206
##  RMSEA          = 0.05170788
##  RMS_theta      = 0.0390972
##  SRMR           = 0.04667054
##
##  Degrees of freedom    = 102
##
## ---------------------- Model selection criteria --------------------------
##
##  Construct           AIC          AICc          AICu
##  Expectation      -113.6951     138.4025      -111.6871
##  Satisfaction     -210.0917      42.0716      -207.0735
##  Loyalty          -184.5460      67.6172      -181.5279
##
##  Construct           BIC           FPE           GM
##  Expectation      -106.6522      0.6346       259.4321
##  Satisfaction     -199.5273      0.4316       265.9793
##  Loyalty          -173.9816      0.4780       276.2327
##
##  Construct           HQ            HQc       Mallows_Cp
##  Expectation      -110.8605     -110.7494      2.3892
##  Satisfaction     -205.8398     -205.6306      5.4149
##  Loyalty          -180.2942     -180.0849      15.6683
##
## ------------------ Variance inflation factors (VIFs) ----------------------
##
##   Dependent construct: 'Satisfaction'
##
##  Independent construct    VIF value
##  Image                     1.5948
##  Expectation               1.5948
##
##   Dependent construct: 'Loyalty'
##
##  Independent construct    VIF value
##  Expectation               1.6806
##  Satisfaction              1.6806
##
## -------- Variance inflation factors (VIFs) for modeB constructs ----------
##
##   Construct: 'Image'
##
##  Weight    VIF value
##  imag1      1.8956
```

```
##  imag2      3.2181
##  imag3      2.6625
##  imag4      1.4379
##  imag5      1.3586
##
##   Construct: 'Expectation'
##
##  Weight    VIF value
##  expe1      1.8311
##  expe2      1.9611
##  expe3      1.5830
##  expe4      1.7015
##  expe5      1.9420
##
##   Construct: 'Satisfaction'
##
##  Weight    VIF value
##  sat1       3.5128
##  sat2       3.5693
##  sat3       2.1606
##  sat4       1.9661
##
##   Construct: 'Loyalty'
##
##  Weight    VIF value
##  loy1       2.4152
##  loy2       1.4829
##  loy3       2.3314
##  loy4       1.5000
##
## ---------------------- Effect sizes (Cohen's f^2) ------------------------
##
##   Dependent construct: 'Expectation'
##
##  Independent construct       f^2
##  Image                     0.5948
##
##   Dependent construct: 'Satisfaction'
##
##  Independent construct       f^2
##  Image                     0.4066
##  Expectation               0.1485
##
##   Dependent construct: 'Loyalty'
##
##  Independent construct       f^2
##  Expectation               0.0129
##  Satisfaction              0.5545
##
## ------------------------------ Effects --------------------------------
##
## Estimated total effects:
## ========================
##   Total effect                  Estimate  Std. error   t-stat.   p-value
##   Expectation ~ Image             0.6107          NA        NA        NA
```

```
##   Satisfaction ~ Image            0.7171          NA        NA        NA
##   Satisfaction ~ Expectation      0.3165          NA        NA        NA
##   Loyalty ~ Image                 0.5353          NA        NA        NA
##   Loyalty ~ Expectation           0.3099          NA        NA        NA
##   Loyalty ~ Satisfaction          0.6608          NA        NA        NA
##
## Estimated indirect effects:
## ===========================
##   Indirect effect          Estimate  Std. error   t-stat.   p-value
##   Satisfaction ~ Image       0.1933          NA        NA        NA
##   Loyalty ~ Image            0.5353          NA        NA        NA
##   Loyalty ~ Expectation      0.2091          NA        NA        NA
## ___________________________________________________________________________
```

Bootstrapping the model and reporting the results

`cSEM` offers two ways to compute resamples (bootstrapping): 1. Setting the `.resample_method` function in `cSEM` to `bootstrap` or `jackknife` and using the postestimation functions `summarize()` or `infer()` to view the results; or 2. using the `resamplescSEMResults()` function and subsequently using the postestimation functions `summarize()` or `infer()` to view the results.

```
# Setting `.resample_method` with 1000 resamples
bootstrap<- csem(.data = satisfaction, .model = model, .resample_method = "bootstrap", .R = 1000)

# Using `resamplescSEMResults()`
bootstrap <- resamplecSEMResults(est_model)

# Using the postestimation command to view the results
summarize(bootstrap)
```

```
## ___________________________________________________________________________
## ---------------------------------- Overview -----------------------------------
##
##  General information:
##  ------------------------
##  Estimation status                = Ok
##  Number of observations           = 250
##  Weight estimator                 = PLS-PM
##  Inner weighting scheme           = "path"
##  Type of indicator correlation    = Pearson
##  Path model estimator             = OLS
##  Second-order approach            = NA
##  Type of path model               = Linear
##  Disattenuated                    = No
##
##  Resample information:
##  ---------------------
##  Resample method                  = "bootstrap"
##  Number of resamples              = 1000
##  Number of admissible results     = 1000
##  Approach to handle inadmissibles = "drop"
##  Sign change option               = "none"
##  Random seed                      = -1261539961
##
##  Construct details:
##  ------------------
##  Name          Modeled as     Order         Mode
##
##  Image         Composite      First order   "modeB"
##  Expectation   Composite      First order   "modeB"
```

```
##  Satisfaction  Composite      First order   "modeB"
##  Loyalty       Composite      First order   "modeB"
##
## ---------------------------------- Estimates ---------------------------------
##
## Estimated path coefficients:
## ============================
##                                                                      CI_percentile
##   Path                          Estimate  Std. error   t-stat.   p-value         95%
##   Expectation ~ Image             0.6107      0.0454   13.4609    0.0000 [ 0.5358; 0.7075 ]
##   Satisfaction ~ Image            0.5238      0.0615    8.5156    0.0000 [ 0.3943; 0.6341 ]
##   Satisfaction ~ Expectation      0.3165      0.0658    4.8079    0.0000 [ 0.1956; 0.4486 ]
##   Loyalty ~ Expectation           0.1007      0.0644    1.5642    0.1178 [-0.0288; 0.2241 ]
##   Loyalty ~ Satisfaction          0.6608      0.0582   11.3527    0.0000 [ 0.5487; 0.7764 ]
##
## Estimated loadings:
## ===================
##                                                                    CI_percentile
##   Loading                 Estimate  Std. error   t-stat.   p-value         95%
##   Image =~ imag1            0.5498      0.0948    5.8001    0.0000 [ 0.3380; 0.7109 ]
##   Image =~ imag2            0.8240      0.0522   15.7784    0.0000 [ 0.7048; 0.9049 ]
##   Image =~ imag3            0.8728      0.0398   21.9282    0.0000 [ 0.7749; 0.9314 ]
##   Image =~ imag4            0.4816      0.1061    4.5386    0.0000 [ 0.2636; 0.6670 ]
##   Image =~ imag5            0.8194      0.0506   16.2081    0.0000 [ 0.6978; 0.8936 ]
##   Expectation =~ expe1      0.7096      0.0956    7.4206    0.0000 [ 0.4768; 0.8523 ]
##   Expectation =~ expe2      0.8628      0.0506   17.0583    0.0000 [ 0.7233; 0.9311 ]
##   Expectation =~ expe3      0.6813      0.0775    8.7890    0.0000 [ 0.4976; 0.8014 ]
##   Expectation =~ expe4      0.8190      0.0612   13.3857    0.0000 [ 0.6730; 0.9097 ]
##   Expectation =~ expe5      0.7882      0.0609   12.9492    0.0000 [ 0.6496; 0.8813 ]
##   Satisfaction =~ sat1      0.9407      0.0222   42.4525    0.0000 [ 0.8848; 0.9712 ]
##   Satisfaction =~ sat2      0.9180      0.0302   30.3989    0.0000 [ 0.8421; 0.9572 ]
##   Satisfaction =~ sat3      0.7265      0.0633   11.4699    0.0000 [ 0.5838; 0.8284 ]
##   Satisfaction =~ sat4      0.8265      0.0575   14.3652    0.0000 [ 0.6920; 0.9216 ]
##   Loyalty =~ loy1           0.9308      0.0393   23.6566    0.0000 [ 0.8278; 0.9797 ]
##   Loyalty =~ loy2           0.5680      0.0950    5.9762    0.0000 [ 0.3797; 0.7339 ]
##   Loyalty =~ loy3           0.9235      0.0329   28.0875    0.0000 [ 0.8360; 0.9661 ]
##   Loyalty =~ loy4           0.4841      0.1097    4.4144    0.0000 [ 0.2651; 0.6944 ]
##
## Estimated weights:
## ==================
##                                                                    CI_percentile
##   Weight                  Estimate  Std. error   t-stat.   p-value         95%
##   Image <~ imag1           -0.0451      0.1129   -0.3995    0.6895 [-0.2858; 0.1633 ]
##   Image <~ imag2            0.1995      0.1279    1.5600    0.1188 [-0.0591; 0.4408 ]
##   Image <~ imag3            0.4869      0.1078    4.5152    0.0000 [ 0.2535; 0.6999 ]
##   Image <~ imag4            0.0605      0.0934    0.6484    0.5167 [-0.1158; 0.2580 ]
##   Image <~ imag5            0.4958      0.0789    6.2814    0.0000 [ 0.3362; 0.6372 ]
##   Expectation <~ expe1      0.0786      0.1437    0.5467    0.5846 [-0.2170; 0.3392 ]
##   Expectation <~ expe2      0.4147      0.1118    3.7106    0.0002 [ 0.1585; 0.6084 ]
##   Expectation <~ expe3      0.1448      0.0974    1.4875    0.1369 [-0.0503; 0.3177 ]
##   Expectation <~ expe4      0.3915      0.1248    3.1368    0.0017 [ 0.1406; 0.6258 ]
##   Expectation <~ expe5      0.2121      0.1297    1.6355    0.1019 [-0.0350; 0.4699 ]
##   Satisfaction <~ sat1      0.4335      0.0953    4.5494    0.0000 [ 0.2472; 0.6218 ]
##   Satisfaction <~ sat2      0.3718      0.1002    3.7123    0.0002 [ 0.1654; 0.5480 ]
##   Satisfaction <~ sat3     -0.0013      0.0780   -0.0162    0.9871 [-0.1546; 0.1600 ]
##   Satisfaction <~ sat4      0.3047      0.0907    3.3576    0.0008 [ 0.1355; 0.4965 ]
##   Loyalty <~ loy1           0.5701      0.1327    4.2964    0.0000 [ 0.3261; 0.8388 ]
##   Loyalty <~ loy2           0.0655      0.0895    0.7320    0.4642 [-0.0929; 0.2593 ]
##   Loyalty <~ loy3           0.5178      0.1233    4.2011    0.0000 [ 0.2440; 0.7262 ]
##   Loyalty <~ loy4          -0.0951      0.0967   -0.9841    0.3251 [-0.2752; 0.1063 ]
##
## Estimated indicator correlations:
## =================================
##                                                          CI_percentile
##   Correlation       Estimate  Std. error   t-stat.   p-value         95%
##   imag1 ~~ imag2      0.6437      0.0660    9.7590    0.0000 [ 0.5079; 0.7570 ]
##   imag1 ~~ imag3      0.5433      0.0708    7.6706    0.0000 [ 0.3978; 0.6768 ]
##   imag1 ~~ imag4      0.5036      0.0498   10.1072    0.0000 [ 0.4010; 0.6020 ]
##   imag1 ~~ imag5      0.3459      0.0619    5.5903    0.0000 [ 0.2247; 0.4609 ]
```

```
##   imag2 ~~ imag3      0.7761      0.0381   20.3440    0.0000 [ 0.6965; 0.8452 ]
##   imag2 ~~ imag4      0.4495      0.0730    6.1589    0.0000 [ 0.2951; 0.5766 ]
##   imag2 ~~ imag5      0.5010      0.0580    8.6387    0.0000 [ 0.3780; 0.6098 ]
##   imag3 ~~ imag4      0.4622      0.0715    6.4673    0.0000 [ 0.3070; 0.5951 ]
##   imag3 ~~ imag5      0.4590      0.0672    6.8327    0.0000 [ 0.3162; 0.5817 ]
##   imag4 ~~ imag5      0.2603      0.0658    3.9548    0.0001 [ 0.1280; 0.3852 ]
##   expe1 ~~ expe2      0.5353      0.0592    9.0426    0.0000 [ 0.4114; 0.6428 ]
##   expe1 ~~ expe3      0.4694      0.0599    7.8418    0.0000 [ 0.3542; 0.5774 ]
##   expe1 ~~ expe4      0.5699      0.0562   10.1325    0.0000 [ 0.4552; 0.6755 ]
##   expe1 ~~ expe5      0.5562      0.0581    9.5762    0.0000 [ 0.4354; 0.6643 ]
##   expe2 ~~ expe3      0.5467      0.0617    8.8667    0.0000 [ 0.4063; 0.6537 ]
##   expe2 ~~ expe4      0.5038      0.0658    7.6549    0.0000 [ 0.3745; 0.6245 ]
##   expe2 ~~ expe5      0.6116      0.0514   11.9036    0.0000 [ 0.5032; 0.7061 ]
##   expe3 ~~ expe4      0.4273      0.0504    8.4744    0.0000 [ 0.3233; 0.5194 ]
##   expe3 ~~ expe5      0.4982      0.0544    9.1602    0.0000 [ 0.3924; 0.6034 ]
##   expe4 ~~ expe5      0.5279      0.0627    8.4262    0.0000 [ 0.4037; 0.6434 ]
##   sat1 ~~ sat2        0.8202      0.0299   27.4409    0.0000 [ 0.7550; 0.8705 ]
##   sat1 ~~ sat3        0.6609      0.0553   11.9469    0.0000 [ 0.5466; 0.7560 ]
##   sat1 ~~ sat4        0.6663      0.0510   13.0624    0.0000 [ 0.5613; 0.7612 ]
##   sat2 ~~ sat3        0.6997      0.0514   13.6056    0.0000 [ 0.5912; 0.7898 ]
##   sat2 ~~ sat4        0.6285      0.0568   11.0746    0.0000 [ 0.5099; 0.7309 ]
##   sat3 ~~ sat4        0.5942      0.0613    9.6904    0.0000 [ 0.4707; 0.7090 ]
##   loy1 ~~ loy2        0.4903      0.0683    7.1769    0.0000 [ 0.3578; 0.6171 ]
##   loy1 ~~ loy3        0.7323      0.0557   13.1391    0.0000 [ 0.6112; 0.8300 ]
##   loy1 ~~ loy4        0.5315      0.0715    7.4309    0.0000 [ 0.3926; 0.6688 ]
##   loy2 ~~ loy3        0.5124      0.0613    8.3592    0.0000 [ 0.3875; 0.6242 ]
##   loy2 ~~ loy4        0.4453      0.0696    6.3941    0.0000 [ 0.3070; 0.5741 ]
##   loy3 ~~ loy4        0.4771      0.0698    6.8357    0.0000 [ 0.3380; 0.6040 ]
##
## ----------------------------------- Effects ------------------------------------
##
## Estimated total effects:
## ========================
##                                                                        CI_percentile
##   Total effect                  Estimate  Std. error   t-stat.   p-value         95%
##   Expectation ~ Image             0.6107      0.0454   13.4609    0.0000 [ 0.5358; 0.7075 ]
##   Satisfaction ~ Image            0.7171      0.0420   17.0615    0.0000 [ 0.6356; 0.7949 ]
##   Satisfaction ~ Expectation      0.3165      0.0658    4.8079    0.0000 [ 0.1956; 0.4486 ]
##   Loyalty ~ Image                 0.5353      0.0476   11.2407    0.0000 [ 0.4488; 0.6349 ]
##   Loyalty ~ Expectation           0.3099      0.0621    4.9860    0.0000 [ 0.1918; 0.4330 ]
##   Loyalty ~ Satisfaction          0.6608      0.0582   11.3527    0.0000 [ 0.5487; 0.7764 ]
##
## Estimated indirect effects:
## ===========================
##                                                                      CI_percentile
##   Indirect effect          Estimate  Std. error   t-stat.   p-value         95%
##   Satisfaction ~ Image       0.1933      0.0431    4.4851    0.0000 [ 0.1255; 0.2845 ]
##   Loyalty ~ Image            0.5353      0.0476   11.2407    0.0000 [ 0.4488; 0.6349 ]
##   Loyalty ~ Expectation      0.2091      0.0474    4.4124    0.0000 [ 0.1281; 0.3060 ]
## ____________________________________________________________________________________
```

APPENDIX B

Installing and setting up SEMinR

You must install the SEMinR package in R or RStudio to be able to use it:

```
install.packages("seminr")
```

The SEMinR package is installed once, but you need to load it in R / RStudio in every session you want to use it:

```
library(seminr)
```

Importing / Loading data

You must load your data into a data frame from sources acceptable in R (.csv, .rda, .xls, etc.). The column names must be the names of your items. We use SEMinR bundled with a dataset from the European Customer Satisfaction Index (ECSI) adapted to the mobile phone market (Tenenhaus et al., 2005) to help beginners follow the coding/programming process easily. The following syntax is used to load the data into the data frame.

```
mobi <- read.csv("mobi_survey_data.csv")
```

Since the dataset is embedded in the SEMinR package, we will use the following syntax instead for the rest of the illustration:

```
data(mobi)
```

Following through, you can use the following syntax to check if the data has loaded successfully:

```
dim(mobi)  #this will show you the number of rows and columns available in your dataset

## [1] 250  24

head(mobi) #this will show you the first six rows of your dataset

##   CUEX1 CUEX2 CUEX3 CUSA1 CUSA2 CUSA3 CUSCO CUSL1 CUSL2 CUSL3 IMAG1 IMAG2 IMAG3
## 1     7     7     6     6     4     7     7     6     5     6     7     5     5
## 2    10    10     9    10    10     8    10    10     2    10    10     9    10
## 3     7     7     7     8     7     7     6     6     2     7     8     7     6
## 4     7    10     5    10    10    10     5    10     4    10    10    10     5
## 5     8     7    10    10     8     8     5    10     3     8    10    10     5
## 6    10     9     7     8     7     7     8    10     3    10     8     9    10
##   IMAG4 IMAG5 PERQ1 PERQ2 PERQ3 PERQ4 PERQ5 PERQ6 PERQ7 PERV1 PERV2
## 1     5     4     7     6     4     7     6     5     5     2     3
## 2    10     9    10     9    10    10     9    10    10    10    10
## 3     4     7     7     8     5     7     8     7     7     7     7
## 4     5    10     8    10    10     8     4     5     8     5     5
## 5     8     9    10     9     8    10     9     9     8     6     6
## 6     8     9     9    10     9    10     8     9     9    10    10
```

We will be using the model depicted in Figure 2 in the manuscript for subsequent illustrations.

Specifying the measurement model

In this illustration, we specify all variables as ***composite***. As highlighted in the manuscript, users need to specify the measurement models as either composite or common factor. To specify a common factor, the term ***reflective*** is used.

```
measurements <- constructs(
  composite("Image",         multi_items("IMAG", 1:5)),
  composite("Expectation",   multi_items("CUEX", 1:3)),
  composite("Satisfaction",  multi_items("CUSA", 1:3)),
  composite("Complaints",    single_item("CUSCO")),
  composite("Loyalty",       multi_items("CUSL", 1:3))
)
```

We do not specify the weight estimation (`correlation_weights` / `regression_weights`) in this illustration. The default weight estimation will be used to estimate the model.

If there is moderation involved, users should specify the interaction in the measurement model by using the `interaction_term()` syntax. SEMinR provides high-level functions for creating simple interactions between constructs. The interaction terms are described in the measurement model function `construct()` using the following command:

- `product_indicator` describes a single interaction composite generated by the scaled product-indicator method described by Henseler and Chin (2010)
- `two_stage` describes a single-item interaction composite that uses a product of the IV and moderator construct scores.
- `orthogonal` describes a single interaction composite generated by the orthogonalization method of Henseler and Chin (2010)

For example, we can describe the interaction between Image and Expectation using the following syntax.

```
# This is the default interaction term using two stage approach
interaction_term(iv = "Image", moderator = "Expectation")

# You can also consider the following syntax
interaction_term(iv = "Image", moderator = "Expectation", method = "two_stage")
interaction_term(iv = "Image", moderator = "Expectation", method = "product_indicator")
interaction_term(iv = "Image", moderator = "Expectation", method = "orthogonal")
```

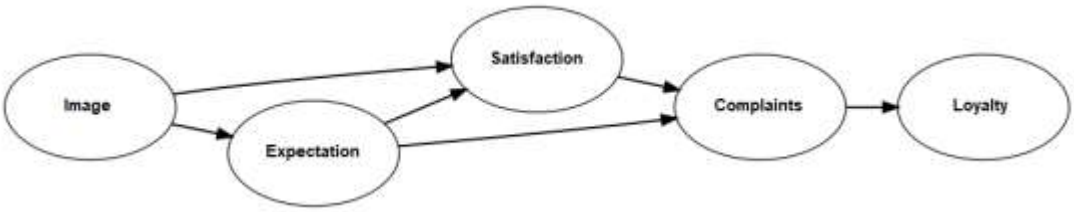
Specifying the structural model

The following syntax specifies the structural model depicted in Figure 2 of the manuscript:

```
structural <- relationships(
  paths(from = "Image",           to = c("Expectation", "Satisfaction"
)),
  paths(from = "Expectation",     to = c("Satisfaction", "Complaints")
),
  paths(from = "Satisfaction",    to = "Complaints"),
  paths(from = "Complaints",      to = "Loyalty")
)
```

We can use the `plot()` function to gain a visualization of the non-estimated structural model. This will help us understand if we have specified the model correctly. Note that the `DiagrammeR` package is required (installed) for plotting.

```
plot(structural)
```



Estimating the model and reporting the estimated model

At this stage, we have defined both the measurement model object `measurement` and the structural model object `structural`, so we can now estimate our model. The `estimate_pls()` syntax is used to estimate the parameters of the model. We then use `summary()` or `model_summary` to obtain the results of the estimated model. The following syntax denotes the full code to estimate and report the estimated model:

```
measurements <- constructs(
  composite("Image",         multi_items("IMAG", 1:5)),
  composite("Expectation",   multi_items("CUEX", 1:3)),
  composite("Satisfaction",  multi_items("CUSA", 1:3)),
  composite("Complaints",    single_item("CUSCO")),
  composite("Loyalty",       multi_items("CUSL", 1:3))
)

structural <- relationships(
  paths(from = "Image",            to = c("Expectation", "Satisfaction")),
  paths(from = "Expectation",      to = c("Satisfaction", "Complaints")),
  paths(from = "Satisfaction",     to = "Complaints"),
  paths(from = "Complaints",       to = "Loyalty")
)

# estimating the model
model_mobi <- estimate_pls(
  data = mobi,
  measurement_model = measurements,
  structural_model = structural
)

# summary of the estimated model
summary(model_mobi)

##
## Results from  package seminr (2.1.0)
##
## Path Coefficients:
##              Expectation Satisfaction Complaints Loyalty
## R^2                0.260        0.515      0.278   0.184
## AdjR^2             0.257        0.511      0.272   0.181
## Image              0.510        0.588          .       .
## Expectation            .        0.210     -0.009       .
## Satisfaction           .            .      0.532       .
## Complaints             .            .          .   0.429
##
## Reliability:
##              alpha  rhoC   AVE  rhoA
## Image        0.723 0.819 0.478 0.739
## Expectation  0.452 0.732 0.480 0.468
## Satisfaction 0.779 0.871 0.693 0.788
## Complaints   1.000 1.000 1.000 1.000
## Loyalty      0.472 0.728 0.511 0.786
```

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

model_summary <- summary(model_mobi)
```

The `model_summary <- summary(model_mobi)` returns an object of class `summary.mobi`, which contains additional information on the model estimation.

- `model_summary$loadings` reports the estimated loadings of the measurement model
- `model_summary$weights` reports the estimated weights of the measurement model
- `model_summary$validity$vif_items` reports the Variance Inflation Factor (VIF) values of the measurement model
- `model_summary$validity$htmt` reports the HTMT values of the constructs
- `model_summary$validity$fl_criteria` reports the Fornell & Larcker criteria scores for the constructs
- `model_summary$validity$cross_loadings` reports the all possible loadings between an indicator and its constructs.
- `model_summary$fSquare` reports the effect size (f^2) of the structural model
- `model_summary$vif_antecedents` reports the Variance Inflation Factor (VIF) values for the structural model
- `model_summary$descriptives` reports the descriptive statistics and correlations for both items and constructs
- `model_summary$composite_scores` reports the construct scores of composites
- `total_effects` reports the total effect of the structural model
- `total_indirect_effects` reports the total indirect effect of the structural model
- `it_criteria` reports the AIC and BIC for the outcome constructs

For example:

```
model_summary$loadings

##       Image Expectation Satisfaction Complaints Loyalty
## IMAG1 0.755       0.000        0.000      0.000   0.000
## IMAG2 0.610       0.000        0.000      0.000   0.000
## IMAG3 0.569       0.000        0.000      0.000   0.000
## IMAG4 0.765       0.000        0.000      0.000   0.000
## IMAG5 0.735       0.000        0.000      0.000   0.000
## CUEX1 0.000       0.755        0.000      0.000   0.000
## CUEX2 0.000       0.740        0.000      0.000   0.000
## CUEX3 0.000       0.570        0.000      0.000   0.000
## CUSA1 0.000       0.000        0.799      0.000   0.000
## CUSA2 0.000       0.000        0.842      0.000   0.000
## CUSA3 0.000       0.000        0.856      0.000   0.000
## CUSCO 0.000       0.000        0.000      1.000   0.000
```

```
## CUSL1 0.000       0.000        0.000      0.000   0.767
## CUSL2 0.000       0.000        0.000      0.000   0.282
## CUSL3 0.000       0.000        0.000      0.000   0.930
```

```
model_summary$validity$htmt
```

```
##              Image Expectation Satisfaction Complaints Loyalty
## Image            .           .            .          .       .
## Expectation  0.888           .            .          .       .
## Satisfaction 0.910       0.865            .          .       .
## Complaints   0.545       0.383        0.588          .       .
## Loyalty      0.867       0.770        0.957      0.561       .
```

```
model_summary$vif_antecedents
```

```
## Expectation :
## Image
##     .
##
## Satisfaction :
##       Image Expectation
##       1.351       1.351
##
## Complaints :
##  Expectation Satisfaction
##         1.35         1.35
##
## Loyalty :
## Complaints
##          .
```

```
model_summary$fSquare
```

```
##              Image Expectation Satisfaction Complaints Loyalty
## Image        0.000       0.351        0.525      0.000   0.000
## Expectation  0.000       0.000        0.067      0.000   0.000
## Satisfaction 0.000       0.000        0.000      0.290   0.000
## Complaints   0.000       0.000        0.000      0.000   0.226
## Loyalty      0.000       0.000        0.000      0.000   0.000
```

At this point, we can also use `plot()` to visualize the estimated model. The following syntax is used to plot the estimated model:

```
plot(model_mobi)
```

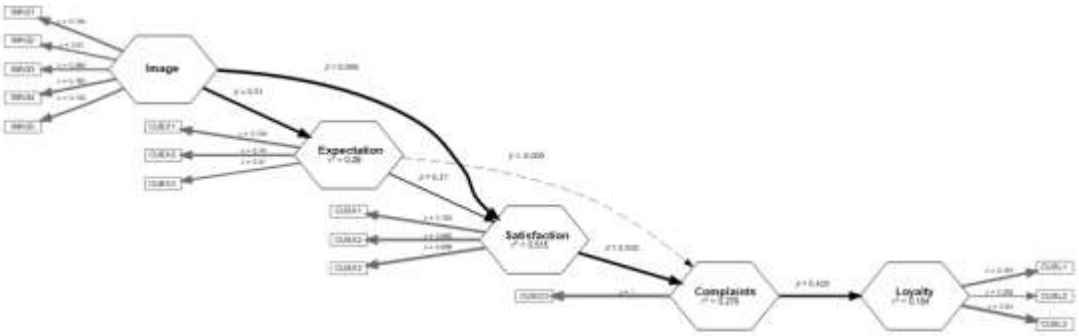


Image
Expectation
Satisfaction
Complaints
Loyalty

Bootstrapping the model and reporting the results

After estimating the model, we can conduct bootstrapping to assess the estimated model. It is worth noting that the `summary(boot_seminr_model)` function returns all estimations for **direct structural paths** in a PLS model. To report a mediated path, the `specific_effect_significance()` function must be included.

```
measurements <- constructs(
  composite("Image",         multi_items("IMAG", 1:5)),
  composite("Expectation",   multi_items("CUEX", 1:3)),
  composite("Satisfaction",  multi_items("CUSA", 1:3)),
  composite("Complaints",    single_item("CUSCO")),
  composite("Loyalty",       multi_items("CUSL", 1:3))
)

structural <- relationships(
  paths(from = "Image",            to = c("Expectation", "Satisfaction")),
  paths(from = "Expectation",      to = c("Satisfaction", "Complaints")),
  paths(from = "Satisfaction",     to = "Complaints"),
  paths(from = "Complaints",       to = "Loyalty")
)

# estimating the model
model_mobi <- estimate_pls(
  data = mobi,
  measurement_model = measurements,
  structural_model = structural
)

# bootstrapping the model
boot_model_mobi <- bootstrap_model( seminr = model_mobi,
                                    nboot = 1000, cores = 2
)

# summary of the bootstrapped model
summary(boot_model_mobi)

##
## Results from Bootstrap resamples:  1000
##
## Bootstrapped Structural Paths:
##                               Original Est. Bootstrap Mean Bootstrap SD T Stat.
## Image  ->  Expectation               0.510          0.523        0.056   9.043
## Image  ->  Satisfaction              0.588          0.594        0.048  12.186
## Expectation  ->  Satisfaction        0.210          0.204        0.062   3.379
## Expectation  ->  Complaints         -0.009         -0.005        0.069  -0.135
## Satisfaction  ->  Complaints         0.532          0.531        0.063   8.488
## Complaints  ->  Loyalty              0.429          0.437        0.061   7.022
##                               2.5% CI 97.5% CI
## Image  ->  Expectation          0.405    0.629
## Image  ->  Satisfaction         0.503    0.694
## Expectation  ->  Satisfaction   0.079    0.321
## Expectation  ->  Complaints    -0.136    0.137
## Satisfaction  ->  Complaints    0.404    0.650
## Complaints  ->  Loyalty         0.318    0.554
##
## Bootstrapped Weights:
##                          Original Est. Bootstrap Mean Bootstrap SD T Stat.
## IMAG1  ->  Image                 0.317          0.315        0.028  11.347
```

```
## IMAG2  ->  Image                0.272          0.270        0.034   7.916
## IMAG3  ->  Image                0.213          0.213        0.036   5.881
## IMAG4  ->  Image                0.323          0.322        0.032  10.002
## IMAG5  ->  Image                0.308          0.308        0.031   9.816
## CUEX1  ->  Expectation          0.494          0.490        0.055   9.043
## CUEX2  ->  Expectation          0.535          0.531        0.068   7.921
## CUEX3  ->  Expectation          0.407          0.405        0.071   5.702
## CUSA1  ->  Satisfaction         0.378          0.378        0.022  17.079
## CUSA2  ->  Satisfaction         0.372          0.372        0.018  20.293
## CUSA3  ->  Satisfaction         0.450          0.449        0.024  18.883
## CUSCO  ->  Complaints           1.000          1.000        0.000       .
## CUSL1  ->  Loyalty              0.375          0.367        0.057   6.626
## CUSL2  ->  Loyalty              0.193          0.186        0.095   2.040
## CUSL3  ->  Loyalty              0.707          0.709        0.053  13.381
##                           2.5% CI 97.5% CI
## IMAG1  ->  Image            0.259    0.371
## IMAG2  ->  Image            0.202    0.336
## IMAG3  ->  Image            0.141    0.283
## IMAG4  ->  Image            0.258    0.386
## IMAG5  ->  Image            0.250    0.377
## CUEX1  ->  Expectation      0.380    0.599
## CUEX2  ->  Expectation      0.399    0.666
## CUEX3  ->  Expectation      0.267    0.546
## CUSA1  ->  Satisfaction     0.337    0.426
## CUSA2  ->  Satisfaction     0.338    0.407
## CUSA3  ->  Satisfaction     0.406    0.499
## CUSCO  ->  Complaints       1.000    1.000
## CUSL1  ->  Loyalty          0.234    0.461
## CUSL2  ->  Loyalty          0.016    0.369
## CUSL3  ->  Loyalty          0.616    0.815
##
## Bootstrapped Loadings:
##                           Original Est. Bootstrap Mean Bootstrap SD T Stat.
## IMAG1  ->  Image                0.755          0.750        0.039  19.444
## IMAG2  ->  Image                0.610          0.611        0.056  10.802
## IMAG3  ->  Image                0.569          0.569        0.066   8.657
## IMAG4  ->  Image                0.765          0.765        0.045  16.960
## IMAG5  ->  Image                0.735          0.738        0.031  23.999
## CUEX1  ->  Expectation          0.755          0.752        0.053  14.136
## CUEX2  ->  Expectation          0.740          0.734        0.073  10.136
## CUEX3  ->  Expectation          0.570          0.571        0.084   6.758
## CUSA1  ->  Satisfaction         0.799          0.799        0.029  27.839
## CUSA2  ->  Satisfaction         0.842          0.841        0.024  35.524
## CUSA3  ->  Satisfaction         0.856          0.855        0.019  45.190
## CUSCO  ->  Complaints           1.000          1.000        0.000       .
## CUSL1  ->  Loyalty              0.767          0.758        0.060  12.696
## CUSL2  ->  Loyalty              0.282          0.272        0.120   2.358
## CUSL3  ->  Loyalty              0.930          0.928        0.021  45.317
##                           2.5% CI 97.5% CI
## IMAG1  ->  Image            0.663    0.812
## IMAG2  ->  Image            0.492    0.710
## IMAG3  ->  Image            0.424    0.676
## IMAG4  ->  Image            0.664    0.832
## IMAG5  ->  Image            0.668    0.793
## CUEX1  ->  Expectation      0.628    0.837
## CUEX2  ->  Expectation      0.573    0.849
## CUEX3  ->  Expectation      0.395    0.713
## CUSA1  ->  Satisfaction     0.737    0.849
## CUSA2  ->  Satisfaction     0.789    0.884
## CUSA3  ->  Satisfaction     0.812    0.888
## CUSCO  ->  Complaints       1.000    1.000
```

```
## CUSL1  ->  Loyalty        0.606    0.849
## CUSL2  ->  Loyalty        0.033    0.497
## CUSL3  ->  Loyalty        0.881    0.963
##
## Bootstrapped HTMT:
##                               Original Est. Bootstrap Mean Bootstrap SD 2.5% CI
## Image  ->  Expectation                0.888          0.899        0.107   0.702
## Image  ->  Satisfaction               0.910          0.911        0.035   0.837
## Image  ->  Complaints                 0.545          0.545        0.059   0.431
## Image  ->  Loyalty                    0.867          0.888        0.090   0.710
## Expectation  ->  Satisfaction         0.865          0.869        0.093   0.698
## Expectation  ->  Complaints           0.383          0.388        0.097   0.207
## Expectation  ->  Loyalty              0.770          0.791        0.116   0.590
## Satisfaction  ->  Complaints          0.588          0.588        0.059   0.468
## Satisfaction  ->  Loyalty             0.957          0.963        0.093   0.779
## Complaints  ->  Loyalty               0.561          0.558        0.087   0.397
##                               97.5% CI
## Image  ->  Expectation           1.120
## Image  ->  Satisfaction          0.978
## Image  ->  Complaints            0.657
## Image  ->  Loyalty               1.061
## Expectation  ->  Satisfaction    1.063
## Expectation  ->  Complaints      0.588
## Expectation  ->  Loyalty         1.018
## Satisfaction  ->  Complaints     0.701
## Satisfaction  ->  Loyalty        1.145
## Complaints  ->  Loyalty          0.727
##
## Bootstrapped Total Paths:
##                               Original Est. Bootstrap Mean Bootstrap SD 2.5% CI
## Image  ->  Expectation                0.510          0.523        0.056   0.405
## Image  ->  Satisfaction               0.695          0.700        0.034   0.628
## Image  ->  Complaints                 0.365          0.370        0.046   0.278
## Image  ->  Loyalty                    0.157          0.164        0.039   0.097
## Expectation  ->  Satisfaction         0.210          0.204        0.062   0.079
## Expectation  ->  Complaints           0.102          0.102        0.066  -0.022
## Expectation  ->  Loyalty              0.044          0.045        0.030  -0.010
## Satisfaction  ->  Complaints          0.532          0.531        0.063   0.404
## Satisfaction  ->  Loyalty             0.228          0.234        0.051   0.145
## Complaints  ->  Loyalty               0.429          0.437        0.061   0.318
##                               97.5% CI
## Image  ->  Expectation           0.629
## Image  ->  Satisfaction          0.763
## Image  ->  Complaints            0.463
## Image  ->  Loyalty               0.245
## Expectation  ->  Satisfaction    0.321
## Expectation  ->  Complaints      0.227
## Expectation  ->  Loyalty         0.109
## Satisfaction  ->  Complaints     0.650
## Satisfaction  ->  Loyalty        0.343
## Complaints  ->  Loyalty          0.554
```

```
model_summary <- summary(boot_model_mobi)
```

Reporting a mediated bootstrapped structural path

The `specific_effect_significance` syntax is used to obtain the bootstrap estimation of a mediated path. An example is below:

```
measurements <- constructs(
  composite("Image",         multi_items("IMAG", 1:5)),
  composite("Expectation",   multi_items("CUEX", 1:3)),
  composite("Satisfaction",  multi_items("CUSA", 1:3)),
  composite("Complaints",    single_item("CUSCO")),
  composite("Loyalty",       multi_items("CUSL", 1:3))
)

structural <- relationships(
  paths(from = "Image",            to = c("Expectation", "Satisfaction")),
  paths(from = "Expectation",      to = c("Satisfaction", "Complaints")),
  paths(from = "Satisfaction",     to = "Complaints"),
  paths(from = "Complaints",       to = "Loyalty")
)

# estimating the model
model_mobi <- estimate_pls(
  data = mobi,
  measurement_model = measurements,
  structural_model = structural
)

# bootstrapping the model
boot_model_mobi <- bootstrap_model( seminr = model_mobi,
                                    nboot = 1000, cores = 2
)

# calculate, at  the 10% confidence interval (two-tailed), the mediated path from Image
to Complaints
specific_effect_significance(boot_seminr_model = boot_model_mobi,
                 from = "Image",     through = c("Expectation", "Satisfaction"), to =
"Complaints",
                 alpha = 0.10)

##  Original Est. Bootstrap Mean   Bootstrap SD        T Stat.          5% CI
##     0.05693702     0.05650337     0.01860270     3.06068537     0.02704198
##         95% CI
##     0.08840529

# calculate, at the 10% confidence interval (two-tailed), the mediated path from Image
to Satisfaction
specific_effect_significance(boot_seminr_model = boot_model_mobi,
                 from = "Image",     through = "Expectation", to = "Satisfaction",
                 alpha = 0.10)

##  Original Est. Bootstrap Mean   Bootstrap SD        T Stat.          5% CI
##     0.10702812     0.10647136     0.03447965     3.10409558     0.05044081
##         95% CI
##     0.16418313
```

Plotting models

Once you have an estimated and bootstrapped model, you can plot the model and save it to a file (JPEG, PNG, PDF, etc.). Note that the `DiagrammeR` package is required (installed) for plotting.

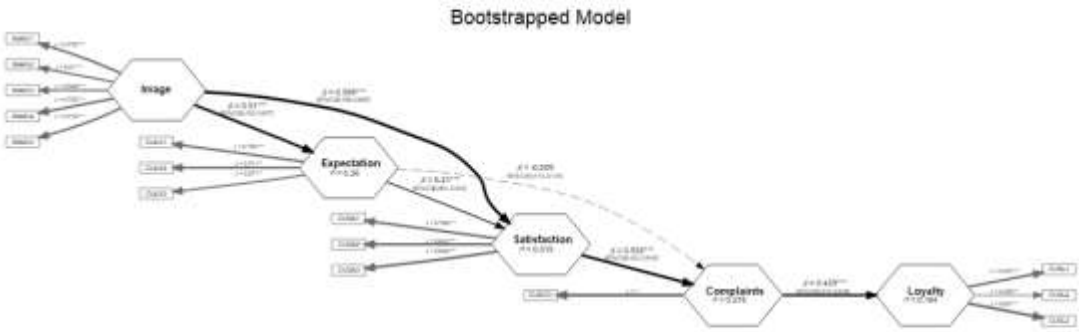
```
measurements <- constructs(
  composite("Image",         multi_items("IMAG", 1:5)),
  composite("Expectation",   multi_items("CUEX", 1:3)),
  composite("Satisfaction",  multi_items("CUSA", 1:3)),
  composite("Complaints",    single_item("CUSCO")),
  composite("Loyalty",       multi_items("CUSL", 1:3))
)

structural <- relationships(
  paths(from = "Image",            to = c("Expectation", "Satisfaction")),
  paths(from = "Expectation",      to = c("Satisfaction", "Complaints")),
  paths(from = "Satisfaction",     to = "Complaints"),
  paths(from = "Complaints",       to = "Loyalty")
)

# estimating the model
model_mobi <- estimate_pls(
  data = mobi,
  measurement_model = measurements,
  structural_model = structural
)

# bootstrapping the model
boot_model_mobi <- bootstrap_model( seminr = model_mobi,
                                    nboot = 1000, cores = 2
)

plot(boot_model_mobi, title = "Bootstrapped Model")
```




```
save_plot("mymodel.pdf")
```