The SEMinR Package

Contents

- 3.1 The Corporate Reputation Model – 52**
- 3.2 Loading and Cleaning the Data – 53**
- 3.3 Specifying the Measurement Models – 56**
- 3.4 Specifying the Structural Model – 59**
- 3.5 Estimating the Model – 59**
- 3.6 Summarizing the Model – 62**
- 3.7 Bootstrapping the Model – 64**
- 3.8 Plotting, Printing, and Exporting Results to Articles – 66**
- References – 73**

Learning Objectives

After reading this chapter, you should understand:

1. Loading and cleaning data for use in model estimation
2. Specifying measurement models in SEMinR syntax
3. Specifying the structural model in SEMinR syntax
4. Estimating a PLS path model using SEMinR syntax
5. Summarizing a PLS path model in SEMinR
6. Bootstrapping a PLS path model in SEMinR
7. Accessing the contents of the summary objects
8. Exporting the PLS-SEM results for reporting

SEMinR is a software package developed for the R statistical environment (R Core Team, 2021) that brings a user-friendly syntax to creating and estimating structural equation models. SEMinR is open source, which means that anyone can inspect, modify, and enhance the source code. SEMinR is distributed under a GNU General Public License version 3 (GPL-3), implying it is completely free for personal, academic, and commercial use – as long as any changes made to it, or applications built using it, are also open source.

SEMinR is hosted on GitHub (► <https://github.com/sem-in-r/seminr>). We encourage users to follow the GitHub page for SEMinR and contribute to this project or use the *issues* feature to report bugs or problems. Users of SEMinR can also interact with the developers and each other at the Facebook group (► <https://www.facebook.com/groups/seminr>). Participants regularly discuss recent developments, best practices, and tutorials on basic functionality of SEMinR. We also encourage you to follow this Facebook group for updates on bugs, issues, and new features.

The SEMinR syntax enables applied practitioners of PLS-SEM to use terminology that is very close to their familiar modeling terms (e.g., reflective, composite, and interactions), instead of specifying underlying matrices and covariances. Specifically, the syntax was designed to:

- Provide a domain-specific language to build and estimate PLS path models in R
- Use both variance-based PLS-SEM and covariance-based SEM (CB-SEM) to estimate composite and common factor models (► Chap. 1)
- Simply and quickly specify model relationships and more complex model elements, such as interaction terms (see ► Chap. 8) and higher-order constructs (Sarstedt, Hair Jr, Cheah, Becker, & Ringle, 2019)

SEMinR uses its own PLS-SEM estimation engine and integrates with the *lavaan* package (Rosseel, 2012) for CB-SEM estimation. SEMinR supports the state of the art of PLS-SEM and beyond. The development team regularly improves the program, incorporates new methods, and supports the users with useful reporting options in their analyses.

In ► Chap. 2, we introduced R and RStudio. After reading that chapter, you should now be familiar with writing scripts, creating objects, and installing pack-

ages. In this chapter, we discuss how to use the SEMinR package. The first step is to install the SEMinR package and load it into the RStudio environment (■ Fig. 3.1). SEMinR was built using R version 4.0.3 – depending on how recently you installed R and RStudio, you might need to update these software files to the latest version before installing SEMinR. Refer to ► Chap. 2 for instructions on installing the latest versions of R and RStudio.

```
# Download and install the SEMinR package
# You only need to do this once to equip
# Rstudio on your computer with SEMinR
install.packages("seminr")

# Make the SEMinR library ready to use
# You must do this every time you restart Rstudio and wish to
# use SEMinR
library(seminr)
```

```
Console Terminal x Jobs x
~/seminr/

installing the source package 'seminr'

trying URL 'https://cran.rstudio.com/src/contrib/seminr_2.0.2.tar.gz'
Content type 'application/x-gzip' length 1667097 bytes (1.6 MB)
=====
downloaded 1.6 MB

* installing *source* package 'seminr' ...
** package 'seminr' successfully unpacked and MD5 sums checked
** using staged installation
** R
** data
*** moving datasets to lazyload DB
** demo
** inst
** byte-compile and prepare package for lazy loading
** help
*** installing help indices
*** copying figures
** building package indices
** installing vignettes
** testing if installed package can be loaded from temporary location
** testing if installed package can be loaded from final location
** testing if installed package keeps a record of temporary installation path
* DONE (seminr)

The downloaded source packages are in
  '/private/var/folders/6s/4n4lx86j74575xn4ylsyj_180000gn/T/RtmpIwok8s/downloaded_
packages'
> |
```

■ Fig. 3.1 Installing and loading the SEMinR package. (Source: authors' screenshot from RStudio)

3.1 The Corporate Reputation Model

With the SEMinR package installed and loaded to the environment, we now introduce the example dataset and model that will be used throughout this textbook. We draw on Eberl's (2010) corporate reputation model, which is also used in *A Primer on Partial Least Squares Structural Equation Modeling (PLS-SEM)* (Hair, Hult, Ringle, & Sarstedt, 2022). The goal of the model is to explain the effects of corporate reputation on customer satisfaction (*CUSA*) and, ultimately, customer loyalty (*CUSL*). Corporate reputation represents a company's overall evaluation by its stakeholders (Helm, Eggert, & Garnefeld, 2010). This construct is measured using two dimensions. One dimension represents cognitive evaluations of the company, which is the company's competence (*COMP*). The second dimension captures affective judgments, which determine the company's likeability (*LIKE*). Research has shown that the model performs favorably (in terms of convergent validity and predictive validity) compared to alternative reputation measures (Sarstedt, Wilczynski, & Melewar, 2013).

In summary, the simple corporate reputation model has two main theoretical components: (1) the target constructs of interest – namely, *CUSA* and *CUSL* (endogenous constructs) – and (2) the two corporate reputation dimensions *COMP* and *LIKE* (exogenous constructs), which are key determinants of the target constructs. ■ Figure 3.2 shows the constructs and their relationships.

Each of these constructs is measured by means of multiple indicators, except satisfaction. For instance, the endogenous construct *COMP* is reflectively measured by three indicator variables, *comp_1*, *comp_2*, and *comp_3*. Respondents answered on a scale from 1 (totally disagree) to 7 (completely agree) to evaluate the statements that these three items represent (■ Table 3.1). Two other constructs in the simple model (*CUSL* and *LIKE*) can be described in a similar manner, and the third (*CUSA*) has only a single indicator. ■ Table 3.1 summarizes the indicator wordings for the four constructs considered in this simple corporate reputation

■ Fig. 3.2 Simple corporate reputation model. (Source: authors' own figure)

Table 3.1 Indicators for the reflectively measured constructs of corporate reputation model

Competence (COMP)	
<i>comp_1</i>	[The company] is a top competitor in its market
<i>comp_2</i>	As far as I know, [the company] is recognized worldwide
<i>comp_3</i>	I believe [the company] performs at a premium level
Likeability (LIKE)	
<i>like_1</i>	[The company] is a company I can better identify with than other companies
<i>like_2</i>	[The company] is a company I would regret more not having if it no longer existed than I would other companies
<i>like_3</i>	I regard [the company] as a likeable company
Customer satisfaction (CUSA)	
<i>cusa</i>	I am satisfied with [the company]
Customer loyalty (CUSL)	
<i>cusl_1</i>	I would recommend [company] to friends and relatives
<i>cusl_2</i>	If I had to choose again, I would choose [company] as my mobile phone service provider
<i>cusl_3</i>	I will remain a customer of [company] in the future
Source: Hair et al. (2022), Chap. 2; used with permission by Sage	

model. In ► Chap. 5, we will extend the simple model by adding four formatively measured constructs.

Now that you are familiar with the reputation model, we will demonstrate the syntax used by SEMinR. Briefly, there are four steps to specify and estimate a structural equation model using SEMinR:

1. Loading and cleaning the data
2. Specifying the measurement models
3. Specifying the structural model
4. Estimating, bootstrapping, and summarizing the model

3.2 Loading and Cleaning the Data

When estimating a PLS-SEM model, SEMinR expects you to have already loaded your data into an object. This data object is usually a `data.frame` class object, but SEMinR will also accept a `matrix` class object. For more information about these objects, you can access the R documentation using the `?` operator (e.g., `?matrix`). The `read.csv()` function allows you to load data into R if the data file is in a .csv

(comma-separated value) or .txt (text) format. Note that there are other packages that can be used to load data in Microsoft Excel's .xlsx format or other popular data formats.

Comma-separated value (CSV) files are a type of text file, whose lines contain the data of each subject or case of your dataset. The values in each line correspond to the different variables of interest (e.g., the first, second, or third value of a line corresponds with the first, second, or third variable in the dataset, from left to right). These values are typically separated by commas but can also be separated by other special characters (e.g., semicolons). The first line of the file typically consists of variable names, called the header line, and is also separated by commas or other special characters. Thus, a variable will have its name in the first row at a certain position (e.g., fifth data entry), and its values will be in all the following lines of data at the same position (e.g., also at the fifth data entry position). Files in a .csv format are a popular way of storing datasets, and we will use it as an example in this chapter. Many software packages, such as Microsoft Excel and SPSS, can export data into a .csv format.

We can load data from a .csv file using the `read.csv()` function. Remember that you can use the `?` operator to find help about a function in R (e.g., use `?read.csv`) at any time. ■ Table 3.2 shows several arguments for the `read.csv()` function as included in the help file.

In this section, we will demonstrate how to load a .csv file into the RStudio global environment. The file we will use is called Corporate Reputation Data.csv and can be downloaded from the book's website at ► <https://www.pls-sem.net/downloads/>. Once you have downloaded the Corporate Reputation Data.csv file, transfer it to your R project working directory as discussed in ► Chap. 2. If you inspect the Corporate Reputation Data.csv file in a text editor, it should appear as in the screenshot in ■ Fig. 3.3. Note that this .csv file uses semicolons instead of commas to separate variable names and values.

In ■ Fig. 3.3, we see that this sample data has a header row consisting of the variable names (columns). In addition, the semicolon (;) is used as a separator

■ Table 3.2 A (shortened) list of arguments for the `read.csv()` function

Argument	Value
<i>file</i>	The name of the file to be uploaded from the working directory
<i>header</i>	A logical value indicating whether the file contains column headers as the first line. Default is "TRUE"
<i>sep</i>	The character used as a separator between fields in the data file. Default is a comma ","
<i>dec</i>	The character used in the file for decimal points. Default is a period "."

Note: Use `?read.csv()` for the full documentation

Source: authors' own table

3.2 · Loading and Cleaning the Data

■ **Fig. 3.3** The Corporate Reputation Data.csv file viewed in a text editor. (Source: authors' screenshot from R)

character, and the missing values are coded as `-99`. If you wish to import this file to the global environment, you can use the `read.csv()` function, specifying the arguments `file = "Corporate Reputation Data.csv"`, `header = TRUE`, and `sep = ";"` and assigning the output to the `corp_rep_data` variable:

```
# Load the corporate reputation data
corp_rep_data <- read.csv(file = "Corporate Reputation Data.
csv", header = TRUE, sep = ";")
```

When clicking on the `corp_rep_data` object in the environment panel of RStudio, the source window opens at the top left of the screen (■ Fig. 3.4).

➤ Important

Inspect the loaded data to ensure that the correct numbers of columns (indicators), rows (observations or cases), and column headers (indicator names) appear in the loaded data. Note that SEMinR uses the asterisk ("*") character when naming interaction terms as used in, for example, moderation analysis, so please ensure that asterisks are not present in the indicator names. Duplicate indicator names will also cause errors in SEMinR. Finally, **missing values** should be represented with a missing value indicator (such as `-99`, which is commonly used), so they can be appropriately identified and treated as missing values.

We encourage you to follow the above steps to download and read a dataset. Alternatively, you can also access that particular dataset directly from SEMinR. To help demonstrate its features, SEMinR comes bundled with two datasets, the corporate reputation dataset (Hair et al., 2022; `corp_rep_data`) and the European

	serviceprovider	servicetype	csor_1	csor_2	csor_3	csor_4	csor_5	csor_global
1	3	2	3	3	3	3	3	3
2	3	2	2	5	6	4	6	6
3	3	2	3	1	2	2	4	4
4	3	2	3	3	5	3	5	5
5	3	2	4	3	4	4	4	4
6	3	2	3	3	4	3	3	4
7	1	1	7	5	7	3	3	7
8	1	1	4	1	3	3	2	4
9	3	1	7	5	6	4	6	7
10	3	2	4	1	5	2	4	5
11	1	1	4	6	4	4	4	6
12	1	1	4	3	4	4	3	4
13	2	2	4	2	3	2	2	4
14	1	2	4	4	4	4	3	4

Showing 1 to 14 of 344 entries, 50 total columns

■ Fig. 3.4 Inspecting the **corp_rep_data** object. (Source: authors' screenshot from RStudio)

Customer Satisfaction Index (ECSI) dataset (Tenenhaus, Esposito Vinzi, Chatelin, & Lauro, 2005; **mobi**). When the SEMinR library has been loaded to the global environment (**library(seminr)**), the data are accessible by simply calling the object names (**corp_rep_data** or **mobi**).

Whichever way you have loaded the **corp_rep_data**, we can now inspect the dataset by using the **head()** function. **head()** is a useful function that outputs the first few fields of an object:

```
# Show the first several rows of the corporate reputation data
head(corp_rep_data)
```

It is clear from inspecting the head of the **corp_rep_data** object (■ Fig. 3.5) that the file has been loaded correctly and has the value “-99” set for the missing values. With the data loaded correctly, we now turn to the measurement model specification.

3.3 Specifying the Measurement Models

Path models are made up of two elements: (1) the measurement models (also called outer models in PLS-SEM), which describe the relationships between the latent variables and their measures (i.e., their indicators), and (2) the structural model (also called the inner model in PLS-SEM), which describes the relationships

```

> # Show the first several rows of the corporate reputation data
> head(corp_rep_data)
  serviceprovider servicetype csor_1 csor_2 csor_3 csor_4 csor_5 csor_global attr_1
1              3           2      3      3      3      3      3           3      5
2              3           2      2      5      6      4      6           6      6
3              3           2      3      1      2      2      4           4      5
4              3           2      3      3      5      3      5           5      3
5              3           2      4      3      4      4      4           4      6
6              3           2      3      3      4      3      3           4      4
  attr_2 attr_3 attr_global perf_1 perf_2 perf_3 perf_4 perf_5 perf_global qual_1
1      1      3           5      5      4      2      5      6           6      5
2      6      6           6      6      6      6      6      6           6      6
3      6      5           6      4      4      2      5      4           5      3
4      7      5           7      5      5      5      3      6           6      5
5      6      6           6      5      6      7      6      5           7      6
6      1      5           5      6      6      4      5      6           6      5
  qual_2 qual_3 qual_4 qual_5 qual_6 qual_7 qual_8 qual_global like_1 like_2 like_3
1      6      4      2      4      4      2      5           6      3      1      2
2      6      6      6      5      6      6      5           6      6      6      6
3      4      3      4      3      5      4      4           5      5      5      5
4      6      6      7      5      4      5      6           7      6      5      6
5      5      7      7      6      6      6      7           7      6      6      7
6      5      5      6      5      6      3      4           6      6      7      7
  comp_1 comp_2 comp_3 cusl_1 cusl_2 cusl_3 cusa age education occupation nphh
1      4      5      5      5      3      3      5      2      3      11      5
2      6      7      6      7      7      7      7      6      3     -99      2
3      4      5      2      7      7      5      6      2      3      11      2
4      6      4      4      7      7      7      6      3      4      11      1
5      6      4      6      6      7      7      6      2      2      12      5
6      3      4      4      7      7      7      6      2      2      11      1
  sample_type mga_1 mga_2 mga_3 mga_4 switch_1 switch_2 switch_3 switch_4
1           1      2      1      1      1      3      1      3      2
2           1      2      1      1      1      5      5      4      4
3           1      2      1      1      1      4      3      2      3
4           1      2      1      1      1      3      4      4      2
5           1      2      1      1      1      5      5      5      4

```

Fig. 3.5 The head of the corporate reputation dataset. (Source: authors' screenshot from RStudio)

between the latent variables. We begin with describing how to specify the **measurement models**.

The basis for determining the relationships between constructs and their corresponding indicator variables is measurement theory. A sound measurement theory is a necessary condition to obtain useful results from any PLS-SEM analysis. Hypothesis tests involving the structural relationships among constructs will only be as reliable or valid as the construct measures.

SEMinR uses the **constructs()** function to specify the list of all construct measurement models. Within this list, we can then define various constructs:

- **composite()** specifies the measurement of individual constructs.
- **interaction_term()** specifies interaction terms.
- **higher_composite()** specifies hierarchical component models (higher-order constructs; Sarstedt et al., 2019).

Table 3.3 The arguments for the `composite()` function

Argument	Value
<i>construct_name</i>	The name of the construct to be created
<i>item_names</i>	A vector of the item names, usually created by the <code>multi_items()</code> or <code>single_item()</code> functions
<i>weights</i>	Defines the estimation mode for each measurement model. Mode_A or correlation_weights for reflectively specified measurement models and mode_B or regression_weights for formatively specified measurement models Mode_A is the default value if weights are not specified. ► Sect. 3.4 contains explanations of mode A and mode B

Source: authors' own table

The `constructs()` function compiles the list of constructs and their respective measurement model definitions. We must supply it with any number of individual `composite()`, `interaction_term()`, or `higher_composite()` constructs using their respective functions. Note that neither a dataset nor a structural model is specified in the measurement model stage, so we can reuse the measurement model object across different datasets and structural models.

The `composite()` function describes the measurement model of a single construct and takes the arguments shown in Table 3.3.

SEMinR strives to make specification of measurement items shorter and cleaner using `multi_items()`, which creates a vector of multiple measurement items with similar names or `single_item()` that describes a single measurement item. For example, we can use `composite()` for PLS path models to describe the reflectively measured *COMP* construct with its indicator variables *comp_1*, *comp_2*, and *comp_3*:

```
composite("COMP", multi_items("comp_", 1:3), weights = mode_A);
```

► Sect. 3.5 contains explanations of mode A and mode B. When no measurement weighting scheme is specified, the argument default is set to **mode_A**. Similarly, we can use `composite()` to define the single-item measurement model of *CUSA* as `composite("CUSA", single_item("cusa"))`. Combining the four measurement models within the `constructs()` function, we can define the measurement model for the simple model in Fig. 3.2. Note: If an error occurs, make sure you used the `library(seminr)` command in R to load the SEMinR package before executing the program code.

```
# Create measurement model
simple_mm <- constructs(
  composite("COMP", multi_items("comp_", 1:3)),
  composite("LIKE", multi_items("like_", 1:3)),
  composite("CUSA", single_item("cusa")),
  composite("CUSL", multi_items("cusl_", 1:3)))
```

The program code above facilitates the specification of standard measurement models. However, the `constructs()` function also allows specifying more complex models, such as interaction terms (Memon et al., 2019) and higher-order constructs (Sarstedt et al., 2019). We will discuss the `interaction_term()` function for specifying interactions in more detail in ► Chap. 8.

3.4 Specifying the Structural Model

With our measurement model specified, we now specify the **structural model**. When a structural model is being developed, two primary issues need to be considered: the sequence of the constructs and the relationships between them. Both issues are critical to the concept of modeling because they represent the hypotheses and their relationships to the theory being tested.

In most cases, researchers examine linear independent–dependent relationships between two or more constructs in the path model. Theory may suggest, however, that model relationships are more complex and involve mediation or moderation relationships. In the following section, we briefly introduce these different relationship types. In ► Chaps. 7 and 8, we explain how they can be estimated and interpreted using SEMinR.

SEMinR makes structural model specification more human readable, domain relevant, and explicit by using these functions:

- `relationships()` specifies all the structural relationships between all constructs.
- `paths()` specifies relationships between sets of antecedents and outcomes.

The simple model in ■ Fig. 3.2 has five relationships. For example, to specify the relationships from *COMP* and *LIKE* to *CUSA* and *CUSL*, we use the `from` and `to` arguments in the path function: `paths(from = c("COMP", "LIKE"), to = c("CUSA", "CUSL"))`.

```
# Create structural model
simple_sm <- relationships(
  paths(from = c("COMP", "LIKE"), to = c("CUSA", "CUSL")),
  paths(from = c("CUSA"), to = c("CUSL")))
```

Note that neither a dataset nor a measurement model is specified in the structural model stage, so we can reuse the structural model object `simple_sm` across different datasets and measurement models.

3.5 Estimating the Model

After having specified the measurement and structural models, the next step is the **model estimation** using the **PLS-SEM algorithm**. For this task, the algorithm needs to determine the scores of the constructs that are used as input for (single

and multiple) partial regression models within the path model. After the algorithm has calculated the construct scores, the scores are used to estimate each partial regression model in the path model. As a result, we obtain the estimates for all relationships in the measurement models (i.e., the indicator weights/loadings) and the structural model (i.e., the path coefficients).

The setup of the measurement models depends on whether the construct under consideration is modeled as reflective or formative. When a **reflective measurement model** is assumed for a construct, the indicator loadings are typically estimated through **mode A**. It estimates the relationship from the construct to each indicator based on a reflective measurement model that uses bivariate regressions (i.e., a single indicator variable represents the dependent variable, while the construct score represents the independent variable). As a result, we obtain correlations between the construct and each of its indicators (i.e., **correlation weights**), which become the **indicator loadings**. In contrast, when a **formative measurement model** is assumed for a construct, the indicator weights are typically estimated using multiple regression. More specifically, the measurement model estimation applies PLS-SEM's **mode B**, in which the construct represents a dependent variable and its associated indicator variables are the multiple independent variables. As a result, we obtain **regression weights** for the relationships from the indicators to the construct, which represent the **indicator weights**. While the use of mode A (i.e., correlation weights) for reflective measurement models and mode B (i.e., regression weights) for formative measurement models represents the standard approach to estimate the relationships between the constructs and their indicators in PLS-SEM, researchers may choose a different mode per type of measurement model in special situations (see also Hair et al., 2022; Rigdon, 2012).

Structural model calculations are executed as follows. The partial regressions for the structural model specify an endogenous construct as the dependent variable in a regression model. This endogenous construct's direct predecessors (i.e., latent variables with a direct relationship leading to the specific endogenous construct) are the independent variables in a regression used to estimate the **path coefficients**. Hence, there is a partial regression model for every endogenous construct to estimate all the path coefficients in the structural model.

All partial regression models are estimated by the PLS-SEM algorithm's iterative procedures, which comprise two stages. In the first stage, the construct scores are estimated. Then, in the second stage, the final estimates of the indicator weights and loadings are calculated, as well as the structural model's path coefficients and the resulting R^2 values of the endogenous latent variables. Appendix A of this textbook provides a detailed description of the PLS-SEM algorithm's stages (see also Lohmöller, 1989).

To estimate a PLS path model, algorithmic options and argument settings must be selected. The algorithmic options and argument settings include selecting the structural model path weighting scheme. SEMinR allows the user to apply two structural model **weighting schemes**: (1) the **factor weighting scheme** and (2) the **path weighting scheme**. While the results differ little across the alternative weighting schemes, path weighting is the most popular and recommended approach. This weighting scheme provides the highest R^2 value for endogenous latent variables

■ Table 3.4 Arguments for the `estimate_pls()` function

Argument	Value
<i>data</i>	The dataset containing the indicator data
<i>measurement_model</i>	The measurement model described by the <code>constructs()</code> function
<i>structural_model</i>	The structural model described by the <code>paths()</code> function
<i>inner_weights</i>	The weighting scheme for path estimation. Use either <code>path_weighting</code> for path weighting (default) or <code>path_factorial</code> for factor weighting (see also Chin, 1998; Lohmöller, 1989)
<i>missing</i>	An argument declaring which missing value scheme should be used to replace the missing values. <code>mean_replacement</code> is used by default
<i>missing_value</i>	An argument declaring which value to be used to indicate missing values in the data. NA is used by default
<i>maxIt</i>	The maximum number of iterations to attempt when estimating the PLS path model. 300 is used by default
<i>stopCriterion</i>	The minimum change in the indicator weights/loadings between two consecutive iterations must be smaller than this threshold (or the maximum number of iterations is reached). 7 is used by default and represents 10^{-7} or 0.0000001

Source: authors' own table

and is generally applicable for all kinds of PLS path model specifications and estimations. Chin (1989) provides further details on the different weighting schemes available in PLS-SEM.

SEMinR uses the `estimate_pls()` function to estimate the PLS-SEM model. This function applies the arguments shown in ■ Table 3.4. Please note that arguments with default values do not need to be specified but will revert to the default value when not specified.

We now estimate the PLS-SEM model by using the `estimate_pls()` function with arguments `data = corp_rep_data`, `measurement_model = simple_mm`, `structural_model = simple_sm`, `inner_weights = path_weighting`, `missing = mean_replacement`, and `missing_value = "-99"` and assign the output to `corp_rep_simple_model`.

```
# Estimate the model
corp_rep_simple_model <- estimate_pls(data = corp_rep_data,
  measurement_model = simple_mm,
  structural_model = simple_sm,
  inner_weights = path_weighting,
  missing = mean_replacement,
  missing_value = "-99")
```

```

~/seminr/
> # Create measurement model ----
> simple_mm <- constructs(
+   composite("COMP", multi_items("comp_", 1:3)),
+   composite("LIKE", multi_items("like_", 1:3)),
+   composite("CUSA", single_item("cusa")),
+   composite("CUSL", multi_items("cusl_", 1:3)))
> # Create structural model ----
> simple_sm <- relationships(
+   paths(from = c("COMP", "LIKE"), to = c("CUSA", "CUSL")),
+   paths(from = c("CUSA"), to = c("CUSL")))
> # Estimate the model
> corp_rep_simple_model <- estimate_pls(data = corp_rep_data,
+   measurement_model = simple_mm,
+   structural_model = simple_sm,
+   inner_weights = path_weighting,
+   missing = mean_replacement,
+   missing_value = "-99")
Generating the seminr model
All 344 observations are valid.
>

```

■ **Fig. 3.6** The estimated simple corporate reputation model. (Source: authors' screenshot from RStudio)

Note that the arguments for *inner_weights*, *missing*, and *missing_value* can be omitted if the default arguments are used. This is equivalent to the previous code block:

```

# Estimate the model with default settings
corp_rep_simple_model <- estimate_pls(data = corp_rep_data,
  measurement_model = simple_mm,
  structural_model = simple_sm,
  missing_value = "-99")

```

When the PLS-SEM algorithm has converged, the message “Generating the seminr model. All 344 observations are valid” will be shown in the console window (■ Fig. 3.6).

3.6 Summarizing the Model

Once the model has been estimated, we can summarize the model and generate a report of the results using the **summary()** function, which is used to extract the output and parameters of importance from an estimated model. SEMinR supports the use of **summary()** for the **estimate_pls()**, **bootstrap_model()**, and **predict_pls()** functions.

The **summary()** function applied to a SEMinR model object produces a **summary.seminr_model** class object, which can be stored in a variable and contains

■ **Table 3.5** Elements of the `summary.seminr_model` object

Sub-object	Contains
<code>\$meta</code>	The estimation function and version information
<code>\$iterations</code>	The number of iterations for the PLS-SEM algorithm to converge
<code>\$paths</code>	The model's path coefficients and (adjusted) R^2 values
<code>\$total_effects</code>	The model's total effects
<code>\$total_indirect_effects</code>	The model's total indirect effects
<code>\$loadings</code>	The indicator loadings for all constructs
<code>\$weights</code>	The indicator weights for all constructs
<code>\$validity</code>	The metrics necessary to evaluate the construct measures' validity
<code>\$reliability</code>	The metrics necessary to evaluate the construct measures' reliability
<code>\$composite_scores</code>	The estimated scores for constructs
<code>\$vif_antecedents</code>	The metrics used to evaluate structural model collinearity
<code>\$fSquare</code>	The f^2 metric for all structural model relationships
<code>\$descriptives</code>	The descriptive statistics of the indicator data
<code>\$it_criteria</code>	The information theoretic model selection criteria for the estimated model

Source: authors' own table

the sub-objects shown in ■ Table 3.5 that can be inspected using the `$` operator (e.g., `summary_simple_corp_rep$meta`). These sub-objects relate to model estimates, which serve as a basis for the assessment of the measurement and structural models (Hair, Risher, Sarstedt, & Ringle, 2019).

```
# Summarize the model results
summary_simple_corp_rep <- summary(corp_rep_simple_model)

# Inspect the model's path coefficients and the R^2 values
summary_simple_corp_rep$paths

# Inspect the construct reliability metrics
summary_simple_corp_rep$reliability
```



```

Console Terminal x Jobs x
~/seminr/
> # Summarize the model results
> summary_simple_corp_rep <- summary(corp_rep_simple_model)
> # Inspect the structural paths
> summary_simple_corp_rep$paths
      CUSA  CUSL
R^2    0.295 0.562
AdjR^2 0.290 0.558
COMP   0.162 0.009
LIKE   0.424 0.342
CUSA    . 0.504
> # Inspect the construct reliability metrics
> summary_simple_corp_rep$reliability
      alpha rhoC  AVE rhoA
COMP 0.776 0.865 0.681 0.832
LIKE 0.831 0.899 0.747 0.836
CUSA 1.000 1.000 1.000 1.000
CUSL 0.831 0.899 0.748 0.839

Alpha, rhoC, and rhoA should exceed 0.7 while AVE should exceed 0.5
>

```

■ Fig. 3.7 Inspecting the summary report elements. (Source: authors' screenshot from RStudio)

■ Figure 3.7 shows the results stored in the `summary_simple_corp_rep$paths` and `summary_simple_corp_rep$reliability` sub-objects.

3.7 Bootstrapping the Model

PLS-SEM is a nonparametric method – thus, we need to perform **bootstrapping** to estimate standard errors and compute confidence intervals. Bootstrapping will be discussed in more detail in ► Chaps. 5 and 6, but for now, we introduce the function and arguments.

SEMinR conducts high-performance bootstrapping using parallel processing, which utilizes the full performance of the central processing unit (CPU). The `bootstrap_model()` function is used to bootstrap a previously estimated SEMinR model. This function applies the arguments shown in ■ Table 3.6.

In our example, we use the `bootstrap_model()` function and specify the arguments `seminr_model = corp_rep_simple_model`, `nboot = 1000`, `cores = NULL`, `seed = 123`. In this example, we use 1,000 bootstrap subsamples. However, the final result computations should draw on 10,000 subsamples (Streukens & Leroi-Werelds, 2016). These computations may take a short while (i.e., the R program remains idle). We first assign the output of the bootstrapping to the `boot_simple_corp_rep` variable. We then summarize this variable, assign-

■ **Table 3.6** Arguments for the `bootstrap_model()` function

Argument	Value
<code>seminr_model</code>	A SEMinR model produced by <code>estimate_pls()</code>
<code>nboot</code>	The number of bootstrap subsamples to be generated. The number of subsamples must be at least equal to the number of valid observations in the dataset. Streukens and Leroi-Werelds (2016) suggest drawing at least 10,000 subsamples (see also Hair et al., 2022). For intermediary analyses, the number of subsamples can be considerably smaller (e.g., 1,000) to save computational time
<code>cores</code>	The number of CPU cores to use when performing parallel processing. Default is set to NULL which utilizes the max number of cores available on your computer
<code>seed</code>	The starting seed to use to make the random process replicable. Default is set to 123

Source: authors' own table

■ **Table 3.7** Elements of the `summary.bootstrap_model` object

Sub-object	Contains
<code>\$nboot</code>	The number of bootstrap subsamples generated during bootstrapping
<code>\$bootstrapped_paths</code>	The bootstrap-estimated standard error, <i>t</i> -statistic, and confidence intervals for the path coefficients
<code>\$bootstrapped_weights</code>	The bootstrap-estimated standard error, <i>t</i> -statistic, and confidence intervals for the indicator weights
<code>\$bootstrapped_loadings</code>	The bootstrap-estimated standard error, <i>t</i> -statistic, and confidence intervals for the indicator loadings
<code>\$bootstrapped_HTMT</code>	The bootstrap-estimated standard error, <i>t</i> -statistic, and confidence intervals for the HTMT values
<code>\$bootstrapped_total_paths</code>	The bootstrap-estimated standard error, <i>t</i> -statistic, and confidence intervals for the model's total effects

Source: authors' own table

ing the output of `summary()` to the `sum_boot_simple_corp_rep` variable. The summarized bootstrap model object (i.e., `sum_boot_simple_corp_rep`) contains the elements shown in ■ Table 3.7, which can be inspected using the `$` operator.

```
# Bootstrap the model
boot_simple_corp_rep <- bootstrap_model(seminr_model = corp_
rep_simple_model,
  nboot = 1000,
  cores = NULL,
  seed = 123)

# Store the summary of the bootstrapped model
sum_boot_simple_corp_rep <- summary(boot_simple_corp_rep)

# Inspect the bootstrapped structural paths
sum_boot_simple_corp_rep$bootstrapped_paths

# Inspect the bootstrapped indicator loadings
sum_boot_simple_corp_rep$bootstrapped_loadings
```

■ Figure 3.8 shows the results of the bootstrap procedure for the path coefficients and indicator loadings. Note that bootstrapping is a random process, and your results might be slightly different from those presented here.

3.8 Plotting, Printing, and Exporting Results to Articles

When model estimation, evaluation, and analysis have been completed, it is often necessary to export the results generated in R to a report, such as an Apache OpenOffice writer document (.odt) or Microsoft PowerPoint presentation (.ppt or .pptx). Throughout this book, we provide screenshots for demonstrating the code outputs to the console in RStudio. However, we do not recommend this method to be used for copying and pasting results to research reports or articles. Instead, we recommend exporting tables and matrices to .csv files, which can be imported into documents or presentations, and that figures are exported to .pdf files to ensure the best print quality. In this section, we demonstrate how to best export results for high print quality and readability.

The `write.csv()` function takes an object from the global environment and writes it into a .csv file in the working directory of the project. This function applies two arguments: `x` is the name of the object to be written to file, and `file` is the name of the file to be created and written to. Thus, if we wish to report the bootstrapped paths from the previously discussed simple model, we would use the `write.csv()` function with argument `x = sum_boot_simple_corp_rep$bootstrapped_loadings` and `file = "boot_loadings.csv"`.

```

~/seminr/
> # Bootstrap the model
> boot_simple_corp_rep <- bootstrap_model(seminr_model = corp_rep_simple_model,
+   nboot = 1000,
+   cores = NULL,
+   seed = 123)
Bootstrapping model using seminr...
SEMinR Model successfully bootstrapped
> # Store the summary of the bootstrapped model
> sum_boot_simple_corp_rep <- summary(boot_simple_corp_rep)
> # Inspect the bootstrapped structural paths
> sum_boot_simple_corp_rep$bootstrapped_paths
      Original Est. Bootstrap Mean Bootstrap SD T Stat. 2.5% CI 97.5% CI
COMP -> CUSA      0.162      0.166      0.068  2.374  0.038  0.298
COMP -> CUSL      0.009      0.011      0.056  0.165 -0.098  0.126
LIKE -> CUSA      0.424      0.422      0.062  6.858  0.299  0.542
LIKE -> CUSL      0.342      0.340      0.056  6.059  0.227  0.450
CUSA -> CUSL      0.504      0.504      0.042 11.978  0.419  0.585
> # Inspect the bootstrapped indicator loadings
> sum_boot_simple_corp_rep$bootstrapped_loadings
      Original Est. Bootstrap Mean Bootstrap SD T Stat. 2.5% CI 97.5% CI
comp_1 -> COMP      0.858      0.858      0.021 41.366  0.813  0.895
comp_2 -> COMP      0.798      0.797      0.029 27.709  0.730  0.843
comp_3 -> COMP      0.818      0.814      0.031 26.246  0.746  0.866
like_1 -> LIKE      0.879      0.880      0.017 51.088  0.843  0.910
like_2 -> LIKE      0.870      0.869      0.018 47.428  0.830  0.900
like_3 -> LIKE      0.843      0.842      0.020 41.417  0.799  0.879
cusa -> CUSA      1.000      1.000      0.000  .      1.000  1.000
cysl_1 -> CUSL      0.833      0.832      0.024 35.331  0.780  0.874
cysl_2 -> CUSL      0.917      0.917      0.010 88.874  0.894  0.935
cysl_3 -> CUSL      0.843      0.842      0.023 37.134  0.793  0.881
>

```

■ **Fig. 3.8** Bootstrapped structural paths and indicator loadings. (Source: authors' screenshot from RStudio)

```

# Write the bootstrapped paths object to csv file
write.csv(x = sum_boot_simple_corp_rep$bootstrapped_loadings,
  file = "boot_loadings.csv")

```

Once the `boot_loadings.csv` file has been saved into the working directory, we can open it with Apache OpenOffice Calc, Microsoft Excel, or other spreadsheet software. These spreadsheet software applications enable formatting and editing the table to produce high-quality tables in reports. We followed this procedure to create ■ **Table 3.8** of bootstrapped indicator loadings for the simple corporate reputation model.

Next, we discuss how to generate high-quality figures from the SEMinR results. First, we generate a sample plot for export from RStudio. To do this, we use a sub-

■ **Table 3.8** Export of bootstrapped indicator loadings from SEMinR

Paths	Original est.	Bootstrap mean	Bootstrap SD	T stat.	2.5% CI	97.5% CI
comp_1 -> COMP	0.858	0.858	0.021	41.366	0.813	0.895
comp_2 -> COMP	0.798	0.797	0.029	27.709	0.730	0.843
comp_3 -> COMP	0.818	0.814	0.031	26.246	0.746	0.866
like_1 -> LIKE	0.879	0.880	0.017	51.088	0.843	0.910
like_2 -> LIKE	0.870	0.869	0.018	47.428	0.830	0.900
like_3 -> LIKE	0.843	0.842	0.020	41.417	0.799	0.879
cusa -> CUSA	1	1	0	NA	1	1
cusl_1 -> CUSL	0.833	0.832	0.024	35.331	0.780	0.874
cusl_2 -> CUSL	0.917	0.917	0.010	88.874	0.894	0.935
cusl_3 -> CUSL	0.843	0.842	0.023	37.134	0.793	0.881

Source: authors' own table

object of `summary_simple_corp_rep` and plot the constructs' internal consistency reliabilities (i.e., Cronbach's α , ρ_{Λ} , and ρ_{ζ}) with `plot(summary_simple_corp_rep$reliability)`. Once this plot displays in the plots tab in RStudio (■ Fig. 3.9), click the **Export** dropdown list and select **Save as PDF** to bring up the save plot as a .pdf window. Select the size and output name and save the document. Note that the size will affect the rendering of the plot and might need to be adjusted several times before the ideal format is found. These .pdf images can be imported directly into documents and reports at very high print quality. Alternatively, we can save the plot as an image file in .png, .jpeg, .eps, and many other formats. To do so, click the **Export** dropdown list in the **plots** tab and select **Save as Image**.

■ **Fig. 3.9** Exporting the plot from RStudio using Save as PDF. (Source: authors' screenshot from RStudio)

Summary

In this chapter, we introduced the SEMinR syntax necessary for loading data, specifying and estimating a PLS path model, and reporting the results. Unlike popular graphical user interface software that uses menus and buttons, using a programming language, such as R, creates many opportunities for errors and bugs to be introduced. It is crucial that you are well versed in the SEMinR syntax, functions, and arguments before you proceed to the next few chapters. For this reason, we strongly recommend reviewing this chapter several times and attempting to complete the exercises before moving onto subsequent chapters. The upside of the programming approach is that every step and parameter of your analysis are explicitly defined for others to repeat

or replicate. In addition, more experienced users can draw on a large number of supplementary R packages that extend the analyses supported by SEMinR.

The SEMinR syntax for PLS-SEM is broadly divided into four stages: (1) loading and cleaning the data, (2) specifying the measurement models, (3) specifying the structural model, and (4) estimating, bootstrapping, and summarizing the model. When loading data, it is important that the format of the file to be imported is well understood to prevent later errors. Special attention should be paid to the column headers, the separator and decimal characters used, and the missing value indicator. The raw data file can be inspected using a text editor prior to importing it into the RStudio environment. The imported data should also be compared to the raw data file to ensure no errors occurred in the process.

The measurement model is specified using the SEMinR functions `constructs()`, `composite()`, `interaction_term()`, and `multi_items()` or `single_item()`. The measurement model can be specified and reused across different datasets and structural model configurations. The structural model is specified using `relationships()`, `paths()`, and intuitive arguments `from` and `to` for specific paths. The PLS path model is estimated using the `estimate_pls()` function, which allows for specification of the inner model weighting scheme, as path weighting or factorial. The `bootstrap_model()` function is used to bootstrap a previously estimated SEMinR model. Reports are generated using the `summarize()`, `plot()`, and `print()` functions, and high-quality figures and tables can be exported to reports and presentations.

? Exercise

The SEMinR package comes bundled with a model (i.e., the influencer model), which analyzes if consumers are likely to follow social media influencers' purchase recommendations and whether they feel connected with the influencer. Specifically, the model examines the impact of self-influencer connection (i.e., the level of own identification with the influencer presenting a specific product, SIC) on product liking (*PL*), perceived quality (*PQ*), and purchase intention (*PI*). That is, product liking and perceived quality act as potential mediators in the relationship between self-influencer connection and purchase intention. Finally, the model hypothesizes a direct effect from purchase intention to willingness to pay (*WTP*). Pick (2020) provides the theoretical background on a similar influencer model.

The data were collected as part of a larger study on social media influencer marketing (Pick, 2020) via an online survey between January and April 2019. The final dataset consists of $N = 222$ observations. The dataset is bundled with the SEMinR package and is named `influencer_data`. Participants saw either a “real” influencer called Franklin who presented a fitness shake ($N = 100$) or a “fake” influencer called Emma who presented a hand blender ($N = 122$) (indicator, `influencer_group`). After seeing the real or fake influencer, participants provided information about their self-

influencer connection on a 7-point Likert scale (1 = completely disagree, 7 = completely agree). Different from Pick (2020), we consider self-influencer connection as a formative measure using the set of items shown in ■ Table 3.9. To assess the formative construct’s convergent validity, a single item was included in the survey (indicator, *sic_global*), which serves as the criterion measure for a redundancy analysis.

In the next step, respondents stated their perceived influencer competence (used in ► Chap. 8), perceived quality, product liking, and purchase intention, all of which are measured reflectively on a 7-point Likert scale (1 = completely disagree, 7 = completely agree). Finally, willingness to pay is measured using a single question, asking respondents for their willingness to pay (in Euro) for the presented product. See ■ Table 3.10 for a complete list of item wordings. The table also includes items for an additional construct (perceived influencer competence), which we will introduce in ► Chap. 8.

The influencer model is illustrated in ■ Fig. 3.10. If you need help or hints, consult the SEMinR demo topic file for the influencer model:

```
# Access the demo file for the ECSI dataset
demo(topic = "seminr-pls-influencer", package = "seminr")
```

- 1. Reproduce the influencer measurement models in SEMinR syntax.
- 2. Reproduce the influencer structural model in SEMinR syntax.
- 3. Estimate the influencer model using the standard settings. Remember to specify the **influencer_data** dataset.

■ Table 3.9 Indicators for the formatively measured construct of the influencer model

Self-influencer connection (SIC)

<i>sic_1</i>	The influencer reflects who I am
<i>sic_2</i>	I can identify with the influencer
<i>sic_3</i>	I feel a personal connection to the influencer
<i>sic_4</i>	I (can) use the influencer to communicate who I am to other people
<i>sic_5</i>	I think the influencer helps (could help) me become the type of person I want to be
<i>sic_6</i>	I consider the influencer to be “me”
<i>sic_7</i>	The influencer suits me well
<i>sic_global</i>	My personality and the personality of the influencer relate to one another

Source: authors’ own table

Table 3.10 Indicators for the reflectively measured construct of the influencer model

Perceived Influencer Competence (PIC)

<i>pic_1</i>	The influencer is qualified
<i>pic_2</i>	The influencer is competent
<i>pic_3</i>	The influencer is an expert
<i>pic_4</i>	The influencer is experienced
<i>pic_5</i>	The influencer is knowledgeable

Perceived Quality (PQ)

<i>pq_1</i>	The product has excellent quality
<i>pq_2</i>	The product looks reliable and durable
<i>pq_3</i>	The product will have fewer problems
<i>pq_4</i>	The product has excellent quality features

Product Liking (PL)

<i>pl_1</i>	I dislike the product
<i>pl_2</i>	The product is appealing to me
<i>pl_3</i>	The presented product raises a positive feeling in me
<i>pl_4</i>	The product is interesting to me

Purchase Intention (PI)

<i>pi_1</i>	It is very likely that I will purchase this product
<i>pi_2</i>	I will purchase this product the next time I need it
<i>pi_3</i>	I would definitely try out the products
<i>pi_4</i>	I would recommend this product to my friends
<i>pi_5</i>	I am willing to purchase this product

Willingness to Pay (WTP)

<i>wtp</i>	Please state your willingness to pay (in Euro) for the presented product
------------	--

Source: authors' own table

■ Fig. 3.10 Influencer model. (Source: authors' own figure)

References

- Chin, W. W. (1998). The partial least squares approach to structural equation modeling. In G. A. Marcoulides (Ed.), *Modern methods for business research* (pp. 295–358). Mahwah, NJ: Erlbaum.
- Eberl, M. (2010). An application of PLS in multi-group analysis: The need for differentiated corporate-level marketing in the mobile communications industry. In V. Esposito Vinzi, W. W. Chin, J. Henseler, & H. Wang (Eds.), *Handbook of partial least squares: Concepts, methods and applications in marketing and related fields* (Springer Handbooks of Computational Statistics Series) (Vol. II, pp. 487–514). Berlin: Springer.
- Hair, J. F., Hult, G. T. M., Ringle, C. M., & Sarstedt, M. (2022). *A primer on partial least squares structural equation modeling (PLS-SEM)* (3rd ed.). Thousand Oaks, CA: Sage.
- Hair, J. F., Risher, J. J., Sarstedt, M., & Ringle, C. M. (2019). When to use and how to report the results of PLS-SEM. *European Business Review*, 31(1), 2–24.
- Helm, S., Eggert, A., & Garnefeld, I. (2010). Modelling the impact of corporate reputation on customer satisfaction and loyalty using PLS. In V. Esposito Vinzi, W. W. Chin, J. Henseler, & H. Wang (Eds.), *Handbook of partial least squares: Concepts, methods and applications in marketing and related fields* (Springer Handbooks of Computational Statistics Series) (Vol. II, pp. 515–534). Berlin: Springer.
- Lohmöller, J.-B. (1989). *Latent variable path modeling with partial least squares*. Heidelberg: Physica.
- Memon, M. A., Cheah, J.-H., Ramayah, T., Ting, H., Chuah, F., & Cham, T. H. (2019). Moderation analysis: Issues and guidelines. *Journal of Applied Structural Equation Modeling*, 3(1), i–ix.
- Pick, M. (2020). Psychological ownership in social media influencer marketing. *European Business Review*, 33(1), 9–30.

- R Core Team. (2021). *R: A language and environment for statistical computing [computer software]*. Vienna, Austria: R Foundation for Statistical Computing. Retrieved from <https://www.R-project.org/>
- Rigdon, E. E. (2012). Rethinking partial least squares path modeling: In praise of simple methods. *Long Range Planning*, 45(5–6), 341–358.
- 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.
- Sarstedt, M., Wilczynski, P., & Melewar, T. C. (2013). Measuring reputation in global markets—A comparison of reputation measures' convergent and criterion validities. *Journal of World Business*, 48(3), 329–339.
- Sarstedt, M., Hair, J. F., Cheah, J. H., Becker, J. M., & Ringle, C. M. (2019). How to specify, estimate, and validate higher-order constructs in PLS-SEM. *Australasian Marketing Journal*, 27(3), 197–211.
- Streukens, S., & Leroi-Werelds, S. (2016). Bootstrapping and PLS-SEM: A step-by-step guide to get more out of your bootstrapping results. *European Management Journal*, 34(6), 618–632.
- Tenenhaus, M., Esposito Vinzi, V., Chatelin, Y.-M., & Lauro, C. (2005). PLS path modeling. *Computational Statistics & Data Analysis*, 48(1), 159–205.

Suggested Reading

- Chin, W. W. (1998). The partial least squares approach to structural equation modeling. In G. A. Marcoulides (Ed.), *Modern methods for business research* (pp. 295–358). Mahwah, NJ: Erlbaum.
- Hair, J. F., Hult, G. T. M., Ringle, C. M., & Sarstedt, M. (2022). *A primer on partial least squares structural equation modeling (PLS-SEM)* (3rd ed.). Thousand Oaks, CA: Sage.
- Hair, J. F., Risher, J. J., Sarstedt, M., & Ringle, C. M. (2019). When to use and how to report the results of PLS-SEM. *European Business Review*, 31(1), 2–24.
- Ray, S., & Danks, N.P. (2020). SEMinR Vignette. Retrieved from: <https://cran.r-project.org/web/packages/seminr/vignettes/SEMinR.html>
- Tenenhaus, M., Esposito Vinzi, V., Chatelin, Y.-M., & Lauro, C. (2005). PLS path modeling. *Computational Statistics & Data Analysis*, 48(1), 159–205.

Open Access This chapter is licensed under the terms of the Creative Commons Attribution 4.0 International License (<http://creativecommons.org/licenses/by/4.0/>), which permits use, sharing, adaptation, distribution and reproduction in any medium or format, as long as you give appropriate credit to the original author(s) and the source, provide a link to the Creative Commons license and indicate if changes were made.

The images or other third party material in this chapter are included in the chapter's Creative Commons license, unless indicated otherwise in a credit line to the material. If material is not included in the chapter's Creative Commons license and your intended use is not permitted by statutory regulation or exceeds the permitted use, you will need to obtain permission directly from the copyright holder.