10

Confirmatory Factor Analysis and Structural Equation Modeling

In this chapter, we discuss structural equation models in R. We show how R can be used for both covariance-based and partial least squares modeling, and present basic guidelines for model assessment. We also demonstrate the power of R to simulate data and to use simulation to inform our expectations.

Structural models are helpful when your modeling needs meet any of these conditions: you need to evaluate interconnections of multiple data points that do not map neatly to the division between predictors and an outcome variable (as would be the case in linear modeling); you wish to include unobserved latent variables such as attitudes and estimate their relationships to one another or to observed data; or you wish to estimate the overall fit between observed data and a proposed model with latent variables or complex connections. From this point of view, structural models are closely related to both linear modeling because they estimate associations and model fit, and to factor analysis because they use latent variables.

The uses for structural models in marketing follow from those needs. For example, the models can be used to determine whether concepts on a survey match assumptions, for instance to assess whether items are in fact related to an underlying construct as one hopes; this is an extension of factor analysis (see Chap. 8). With regard to latent variables, the models can be used to estimate the association between outcomes such as purchase behavior and underlying attitudes that influence those, such as satisfaction and brand perception. An even more complex model would be one where several latent variables are simultaneously associated with one another in multiple ways. For example, brand perception, purchase intent, willingness to pay, and satisfaction all relate to one another as latent constructs, and also relate in multiple ways to observed consumer behaviors such as purchases.

We assume in this chapter that the reader is familiar with structural models and primarily wishes to learn the R approach to them. The topic is too complex for a single chapter although we attempt to present an overview that is understandable

C. Chapman, E.M. Feit, *R for Marketing Research and Analytics*, Use R!,
DOI 10.1007/978-3-319-14436-8_10

for any analyst. Section 10.1 provides a conceptual introduction for readers new to the area; experienced analysts may wish to skip to Sect. 10.1.1.

10.1 The Motivation for Structural Models

The real world rarely divides into nicely controlled experiments and marketers are often interested to test complex models. Consider a consumer's likelihood to purchase a new product. The likelihood will be influenced by many factors such as prior product experience, perception of brand and features, price sensitivity, promotional effects, and so forth.

Imagine that we are brand managers interested in the impact of *brand perception* on *likelihood to purchase*. One approach to assess this might be to collect survey data on stated likelihood to purchase the product and attitudes about the brand. In schematic terms, we might model this as a linear relationship (Chap. 7): *purchase* $\sim$ *perception*. Yet whether we find an effect or not, our model is open to the challenge that there are many other possible variables that we didn't assess. Perhaps an effect we thought we found was due to prior experience and not to brand; or perhaps we didn't find an effect because we failed to account for a promotional campaign that influences the relationship.

Even imperfect assessment of those additional influences can improve our understanding. In a statistical model, any unbiased—even if incomplete—capture of variance will improve other parts of the model. For instance, we might only care about the relationship between *brand perception* on *likelihood to purchase*; yet if our model also includes *promotion* and *prior brand experience*, it will capture some of the variance due to those factors and give us a better, more realistic estimate for the relationship between brand and purchase. Including those influences will make us and our stakeholders more confident.

A common way to test complex models of this kind in marketing is *structural equation modeling* (SEM). It is impossible to model every possible influence in a market, and we don't recommend trying. Yet with SEM, it is feasible to do several things that improve our models: to include multiple influences, to posit unobserved concepts that underlie the observed indicators (i.e., constructs such as *brand preference*, *likelihood to purchase*, and *satisfaction*), to specify how those concepts influence one another, to assess the model's overall congruence to the data, and to determine whether the model fits the data better than alternative models.

As we will show, this is done by creating a graphical *path diagram* of influences and then estimating the strength of relationship for each path in the model. Such paths often concern two kinds of variables: *manifest* variables that are observed, i.e., that have data points, and *latent* variables that are conceived to underlie the observed data. For example, in the first model we examine, *product involvement* is conceived as a latent factor that underlies several other latent factors such as *image*

involvement, and those factors in turn are observed as manifest variables on survey items. The set of relationships among the latent variables is called the *structural model*, while the linkage between those elements and the observed, manifest variables is the *measurement model*.

Structural models pose many potential pitfalls and have a great deal of specialized jargon. We attempt to use a minimum of technical jargon in this chapter, yet we urge you not to use this chapter as your only guide to such models. Despite that warning, we believe that the chapter demonstrates the power and importance of such models and will prepare you to learn more about them.

Structural equation models are similar to linear regression models (Chap. 7) but differ in three regards. First, they assess the relationships among many variables, with models that may be more complex than simply predictors and outcomes. Second, those relationships allow for latent variables that represent underlying constructs that are thought to be manifested imperfectly in the observed data. Third, the models allow relationships to have multiple "downstream" effects. For example, experience with a product (a stated variable on a survey) might relate to brand perception (a latent construct expressed in several survey items) which then relates to willingness to pay (a latent construct) which relates observed behavior to purchase or not at a particular price point (perhaps in transaction data or as a stated choice on a survey item).

Finally, we close this introduction with a warning: with this potential for such connections among latent variables, it is tempting to interpret structural models as being about *causation* and many analysts, stakeholders, and even authors of academic papers do this. We believe that it is possible to use these models as part of causal modeling but to do so requires attention to issues and models that are well outside the scope of this book. In general, however, we recommend that you consciously avoid all discussion of causation, and instead talk about *relationships* or *association among the latent variables*.

10.1.1 Structural Models in This Chapter

In examining R's capabilities to specify, test, and visualize structural equation models (SEM), we present two examples: a confirmatory factor analysis (CFA) model that evaluates an assessment scale for product involvement, and a more general SEM that models the likelihood to repurchase a product, as related to quality, value, price, and customer satisfaction of a prior purchase. In each case, we demonstrate how to simulate data for test purposes, how to specify and fit the proposed model, and how to assess the proposed model.

We also show two different SEM approaches: the most common but more demanding *covariance based* (CB-SEM) approach, and the more flexible *partial least squares* (PLS-SEM) approach. We start with CB-SEM because it is virtually synonymous with "SEM" in the literature, especially outside marketing.

Nevertheless, PLS-SEM is often able to fit models in situations where CB-SEM fails and has become popular for marketing applications in the past decade, so we demonstrate it as well.

Several R packages are able to fit SEMs. In this chapter, we demonstrate CB-SEM using the `lavaan` package [135] (where *lavaan* abbreviates "*la*tent *va*riable *an*alysis") because of its simplicity for model specification and its rich set of available tools for data simulation, model comparison, and visualization. Then we demonstrate PLS-SEM with the `semPLS` package.

10.2 Scale Assessment: CFA

We start by considering a survey scale that assesses *product involvement*, using the survey items shown in Table 10.2 [26]. This survey scale reflects a model in which product involvement is a hierarchical construct comprising three factors: *general* involvement with a product category, involvement with the *choices and features* of the product, and involvement with the category in terms of personal *image*.

On the survey, three subscales reflect those factors and could lead to higher or lower scores depending on how consumers view a product. For instance, as marketers we would expect digital cameras to engage consumers in terms of their technical features and thus to score high on *feature involvement*. By contrast, clothing is a key component of personal image, and could be expected to score high on *image involvement*. Either category might be high or low on *general involvement* according to the interests of a specific respondent. To consider other categories, a generic good such as paper might show low consumer involvement on all three factors, while automobiles might be relatively high on all three. This model was proposed as an alternative to a single factor model of product involvement, where involvement is simply high or low overall with no differentiation between factors such as feature or image involvement.

The three-factor model here was named *PIES* as an abbreviation of the "*P*roduct *I*nvolvement and *E*nthusiasm *S*cale" [26]. It could be used in many marketing situations. For instance, if we assess that our product category is high on feature involvement, we might develop communication and positioning strategies that emphasize technical specifications. It may also be used to inform targeting: if we determine that a given demographic group views our category as important to their personal image, then we might target them with campaigns that highlight our product in terms of personal image.

The PIES structural model proposes four latent (unobserved) constructs that underlie product involvement: a *general* involvement factor (here abbreviated as "Gnr"), a *choice/feature* factor (hereafter "Feature" or "Ftr"), an *image* ("Img") factor, and a higher-order *PIE* factor (product involvement and engagement) that is conceived as the underlying level of interest underlying the other three factors. This hierarchical

factor model is shown in Fig. 10.1. The relationships among these are linear relationships of unobserved, latent variables that match a particular theory about product involvement (and whose relationship we will specify and test below).

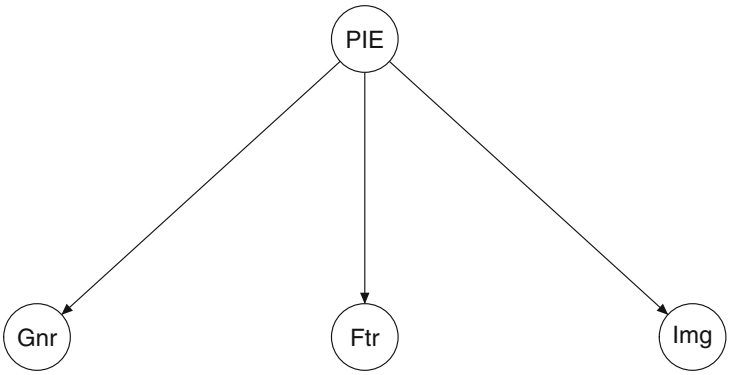

Fig. 10.1. PIES latent construct (factor) model, showing three factors of product involvement (General involvement = Gnr, Feature = Ftr, and Image = Img). These three latent factors relate to a higher-order, overall latent construct for involvement, PIE. None of these latent constructs is directly observed; for the observational model, see Fig. 10.2.

The three involvement factors and the higher-order PIE factor are modeled as *latent variables* that are not directly observed but are instead conceived to influence the survey items that *manifest* them. On the survey, each factor is represented by a subscale comprising several items, as shown in Table 10.1.

In the hierarchical model, the overall PIE factor does not directly influence any items on the scale. Rather, it influences the other three factors as a higher order latent variable. The complete structural model, showing the hierarchical relation of the latent constructs and the manifest scale items that are observed for each construct, is shown in Fig. 10.2.

An analyst's question with PIES—and the question for the PIES authors in the cited paper—might be this: *Is the PIES scheme a good model for some set of survey responses for the items in Table 10.2?* If we confirm that PIES is a good model, we will be much more confident in using this survey data to draw inferences about product involvement than if we had not assessed the model. We show how SEM in R can address that question.

To do this, we use a particular application of SEM known as CFA. In CFA, one specifies the factor structure and asks, "How well does the proposed model agree with the structure of the data?" We also address a closely related question, "Is that model better than some other specified model?"

Table 10.1. The hierarchical product involvement (PIES) scale, showing the subscales (factors) and items

Item	Text	Reversed?
General scale		
i1	____ are not very important to me.	Yes
i2	I never think about ____.	Yes
i3	I am very interested in ____.	
Feature scale		
i4	In choosing a ____ I would look for some specific features or options.	
i5	If I chose a new ____ I would investigate the available choices in depth.	
i6	Some ____ are clearly better than others.	
i7	If I were choosing a ____, I would wish to learn about the available options in detail.	
Image scale		
i8	When people see someone's ____, they form an opinion of that person.	
i9	A ____ expresses a lot about the person who owns it.	
i10	You can learn a lot about a person by seeing the person's ____.	
i11	It is important to choose a ____ that matches one's image.	

The survey would be given for a specific product category, filling in the blanks with a descriptive phrase such as "digital cameras" or "diet soda." From [26]

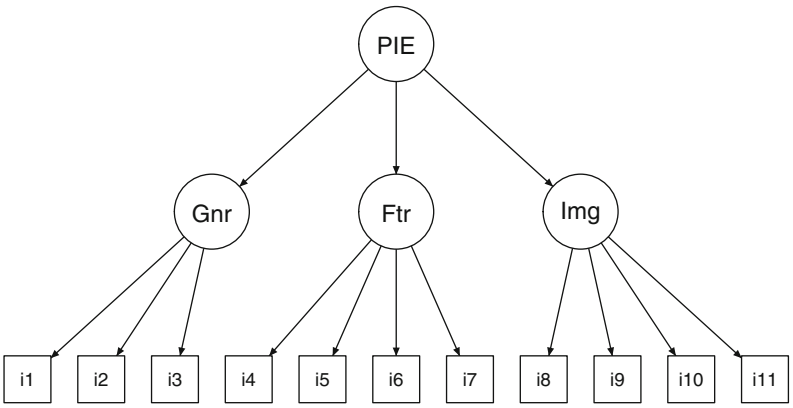

Fig. 10.2. The complete PIES model with latent factors and manifest scale items.

10.2.1 Simulating PIES CFA Data

To demonstrate CFA, we create a simulated data set with known factor structure that corresponds to the PIES model in Table 10.2. We use this data to demonstrate how to assess a CFA model (which ordinarily would be done with data collected from respondents). Then we evaluate alternative models and discuss the importance of model comparison in CFA.

If you prefer to download the data for the CFA example:

```
> piesSimData <- read.csv("http://goo.gl/yT0XwJ")
> summary(piesSimData)
       i1              i2              i3              i4              i5
 Min.   :1.000   Min.   :1.000   Min.   :1.00   Min.   :1.000   Min.   :1.000
 1st Qu.:4.000   1st Qu.:3.000   1st Qu.:3.00   1st Qu.:3.000   1st Qu.:3.000
 Median :4.000   Median :4.000   Median :4.00   Median :4.000   Median :4.000
 Mean   :4.339   Mean   :4.104   Mean   :4.11   Mean   :4.039   Mean   :3.999
 ...
```

Once you have the data, you may proceed to Sect. 10.2.2. Otherwise, continue here; data generation for CFA turns out to be rather easy.

We use the `lavaan` package for core SEM (and CFA) functionality including data simulation and model fitting [135], and extend its capabilities for model comparison and visualization using two other packages, `semTools` [124] and `semPlot` [43]. Our first step is to install those packages and make them available in R:

```
> install.packages(c("lavaan", "semTools", "semPlot"))
> library(lavaan)
> library(semTools)
> library(semPlot)
```

In `lavaan`, a structural model may be specified using syntax that is rather similar to R's linear model formulas (Sect. 7.3). We specify two models here: (1) a *structural* model that we fit to the data and whose structure we wish to assess, and (2) a *data* model that we use only to generate simulated data for test purposes. The structural model is specified according to the model as shown in Fig. 10.2, written as a simple string that `lavaan` will parse:

```
> piesModel <- " General =~ i1 + i2 + i3
+              Feature =~ i4 + i5 + i6  + i7
+              Image   =~ i8 + i9 + i10 + i11
+              PIES =~ General + Feature + Image "
```

In SEM code we read the "=∼" symbol as "is manifested by," which means that it is estimated to be a single variable that is a composite of the three items (with some degree of unreliability or error in each). Each line in this formula defines a new latent variable—`General`, `Feature`, and so forth—that does not appear in the data set but which `lavaan` will estimate for us based on the observed items `i1`, `i2`, etc. We can then use these latent variable in other parts of the formula to express additional relationships. For instance, in this code the latent variable `PIES` relates in turn to the other latent variables `General`, `Feature`, and `Image`. Such relationships of latent variables are a key differentiator between SEM and regular linear modeling.

The `piesModel` formulas say that PIES is manifested by three factors: General, Feature, and Image. Each of those is manifested by 3 or 4 of the items `i1` through `i11` as defined in Table 10.1.[1]

Next we simulate data similar to what might come from a PIES survey of consumers. (Of course if you only test a model against real data, then these data generation steps are not required.) We define our data simulation model using the same SEM syntax, but add *factor loading coefficients* for the items and subfactors in order to specify the structural relationships. We use factor loadings that approximate those reported by the PIES authors [26]:

```
> piesDataModel <- " General =~ 0.9*i1 + 0.7*i2 + 0.5*i3
+                  Feature =~ 0.3*i3 + 0.7*i4 + 0.9*i5 + 0.5*i6  + 0.9*i7
+                  Image   =~ 0.2*i3 + 0.8*i8 + 0.9*i9 + 0.8*i10 + 0.7*i11
+                  PIES =~ 0.7*General + 0.8*Feature + 0.8*Image"
```

We generate a data set with that factor structure by setting a random number seed and using `simulateData(MODEL, sample.nobs)`, where `sample.nobs` is the number of observations, or N. We choose $N = 3{,}600$ to approximate data reported in the PIES paper:

```
> set.seed(10001)     # another island Zip code
> piesSimData.norm <- simulateData(piesDataModel, sample.nobs=3600)
> print(head(piesSimData.norm), digits=2)
     i1    i2    i3     i4    i5    i6    i7     i8    i9    i10    i11
1 -0.16  2.07  1.14 -1.746 -1.68 -1.79 -1.46 -0.032  1.82  0.610 -0.032
2 -0.38  2.27  0.79  0.922  0.23  0.35  0.51  0.963 -1.11 -0.037  0.792
3 -0.65 -3.00  0.25 -0.077 -0.35  0.12 -1.63 -0.766 -0.22 -1.220  0.462
...
```

Each row here represents a set of hypothetical survey responses from one respondent. Note that the generated data is continuous (drawn from a normal distribution with decimal values), so it is not yet appropriate for PIES; as survey responses, PIES items are 1–7 Likert-type scores [26].

In order to convert the continuous data to discrete survey data, we use the function `cut(DATA, breaks=K)` that divides continuous data into K groups, expressed as K factor levels (see Sect. 12.4.1 for more on `cut()`). We could do this separately for each of the 11 columns of data, but it is more instructive to do it in a way that is generalizable. That involves a few conceptual steps.

We use `cut()` to convert a vector of continuous numeric data into seven factors, using `labels=FALSE` to keep the result as integers instead of labeled, nominal values. Then we enclose that in an anonymous function that can be used repeatedly by `apply()`. We `apply()` that anonymous recoding function to each of the columns of our data set using the *list* version of apply (`lapply()`), and assemble

[1] If you are experienced with other SEM software, you may wonder about details such as the need to fix a path for each factor and to specify error terms. Those are automatically handled by `lavaan` with defaults that are appropriate for many situations (for instance, having uncorrelated errors and fixing the first manifest variable path to 1.0).

the resulting set of discrete numeric vectors into a new `data.frame`. That comes together in amazingly compact R code (you should spend time deconstructing and tinkering with it to see how this works):

```
> piesSimData <- data.frame(lapply(piesSimData.norm,
+                           function(x) { cut(x, breaks=7, labels=FALSE) } ))
```

We now perform our usual data quality checks:

```
> library(car)
> some(piesSimData)
     i1 i2 i3 i4 i5 i6 i7 i8 i9 i10 i11
11    3  3  4  2  2  5  3  2  4   3   4
709   3  3  3  5  4  3  3  3  4   4   4
1392  4  3  3  3  4  4  3  4  4   5   4
...
> library(psych)
> describe(piesSimData)
    vars    n mean   sd median trimmed  mad min max range  skew kurtosis   se
i1     1 3600 4.34 1.00      4    4.32 1.48   1   7     6 -0.07    -0.01 0.02
i2     2 3600 4.10 1.05      4    4.09 1.48   1   7     6 -0.01    -0.07 0.02
i3     3 3600 4.11 1.02      4    4.10 1.48   1   7     6 -0.01    -0.13 0.02
...
```

The data set now comprises discrete values from 1–7, averaging about 4, with good distribution properties (no skew, sd around 1, and so forth). We visualize the relationships among the items using `scatterplotMatrix()` from the `car` package [51], selecting a subset of the items—two items from each factor—to make inspection easier:

```
> library(car)
> library(RColorBrewer)
> scatterplotMatrix(piesSimData[, c(1, 2, 4, 5, 8, 9)], diag="histogram",
+                   col=brewer.pal(3, "Paired"), ellipse=TRUE )
```

The result is shown in Fig. 10.3. In looking at the scatterplots, we see the situation as expected: items are discrete (as shown in histograms on the diagonal), and items have higher correlation within a subscale (as in the off-diagonal plots for `i1` vs. `i2`) than they do across scales (such as `i1` vs. `i4`).

Because this data reflects a factor model, we may also do a quick inspection of the apparent factor structure. Although we use CFA later to do a strong test of factor structure, it is useful to perform a brief check using the `factanal()` command to perform an exploratory factor analysis (EFA, see Sect. 8.3):

```
> factanal(piesSimData, factors=3)
...
Loadings:
    Factor1 Factor2 Factor3
i1  0.138   0.119   0.675
i2                  0.614
i3  0.277   0.362   0.476
i4  0.151   0.608
i5  0.126   0.715   0.102
```

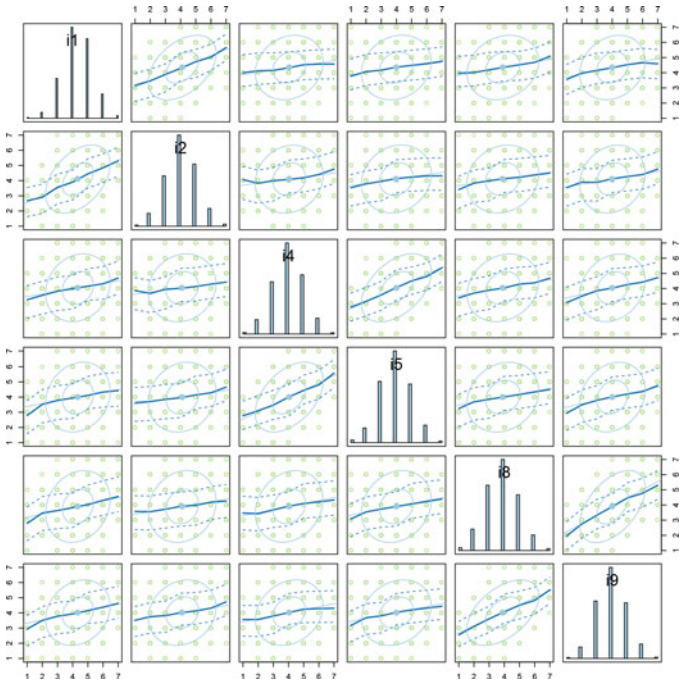

Fig. 10.3. Scatterplot matrix for selected items in the simulated PIES data. Individual items have discrete values that approximate a normal distribution (in the histograms on the diagonal). Items are all positively correlated. Items within a proposed factor, such as `i1` and `i2`, show stronger association than those in differing factors.

```
i6            0.519
i7   0.133    0.678    0.154
i8   0.665    0.137    0.128
i9   0.706    0.138    0.130
i10  0.655    0.117    0.145
i11  0.632    0.126
...
```

We see three plausible factors comprising items i8–i11, i4–i7, and i1–i3, respectively, as we would expect (the factor order is irrelevant). As a reminder, the EFA model does *not* test or confirm the PIES model; that is what CFA does. Instead, EFA reassures us that the data look reasonable before proceeding.

To recap, the simulated data—created using just 4 commands in R—have the kind of structure that might be expected from a consumer survey using the items in Table 10.2. We now proceed to the CFA.

10.2.2 Estimating the PIES CFA Model

CFA assessment begins by defining the model that we wish to evaluate. In this case, we model the three PIES factors (latent variables), `General`, `Feature`, and `Image` as manifest in items `i1`–`i11`. We then model the overall `PIES` latent variable as the composite of the other three factors (see Sect. 10.2.1 for an explanation of the formula syntax here):

```
> library(lavaan)
> piesModel <- " General =~ i1 + i2 + i3
+              Feature =~ i4 + i5 + i6  + i7
+              Image   =~ i8 + i9 + i10 + i11
+              PIES =~ General + Feature + Image "
```

We fit this model to data using `cfa(MODEL, data=DATA)` and inspect the result with `summary(FIT, fit.measures=TRUE)`. The output of `summary(FIT)` is lengthy in this case so we abbreviate it:

```
> pies.fit <- cfa(piesModel, data=piesSimData)
> summary(pies.fit, fit.measures=TRUE)
lavaan (0.5-17) converged normally after  41 iterations

  Number of observations                          3600
...
  Comparative Fit Index (CFI)                    0.975
  Tucker-Lewis Index (TLI)                       0.966
...
Root Mean Square Error of Approximation:
  RMSEA                                          0.041
  90 Percent Confidence Interval          0.036  0.045
  P-value RMSEA <= 0.05                          1.000

Standardized Root Mean Square Residual:
  SRMR                                           0.030

Parameter estimates:
                   Estimate  Std.err  Z-value  P(>|z|)
Latent variables:
  General =~
    i1                1.000
    i2                0.948    0.042   22.415    0.000
    i3                1.305    0.052   25.268    0.000
  Feature =~
    i4                1.000
    i5                1.168    0.037   31.168    0.000
    i6                0.822    0.033   25.211    0.000
    i7                1.119    0.036   31.022    0.000
  Image =~
    i8                1.000
```

```
    i9                0.963    0.028   34.657    0.000
    i10               0.908    0.027   33.146    0.000
    i11               0.850    0.027   31.786    0.000
  PIES =~
    General           1.000
    Feature           0.875    0.057   15.355    0.000
    Image             0.932    0.060   15.628    0.000
...
```

The CFA output establishes that the three-factor hierarchical model fits the data well. In the upper portion of the summary, we see that fit indices are strong (e.g., CFI = 0.975) and residuals are low (e.g., RMSEA = 0.041). The lower part of the summary shows that model parameters for the paths of latent variables to items, and for the upper-level PIES factor to the three subfactors, are all significant ("$P(>|z|)$" = 0), are similar to one another in magnitude (ranging 0.822–1.305), and are not far from 1.0 (a good thing because the items are intended to be used in simple additive subscales).

If these were real data, the CFA would establish both that the PIES hierarchical model fits well and—because the factor-item loadings are around 1.0—that it is reasonable to add up the items as a simple sum to form subscale scores, as is common for such surveys (instead of computing weighted factor scores).

The final model with fitted parameter estimates is plotted with the `semPaths()` command from the `semPlot` package. We use the argument `edge.label.cex` to scale the parameter font to be smaller and more readable. Many R packages use "cex" (*c*haracter *ex*pansion) to rescale the font for some element of the plot (in this case, *edge labels*, i.e., parameter estimates). Setting cex $>$ 1.0 enlarges a font; cex $<$ 1.0 shrinks it. If you're looking for a way to rescale a font, try searching for "cex" in a plot routine's help file. The model is drawn as follows:

```
> library(semPlot)
> semPaths(pies.fit, what="est", fade=FALSE, residuals=FALSE,
+          edge.label.cex=0.75)
```

The result is Fig. 10.4. This figure expresses some of the crucial information from the longer CFA text output above, in a more readable way. The graphical version makes it easy to see the relationships between the latent and manifest variables and to browse the coefficient values.

10.2.3 Assessing the PIES CFA Model

The PIES model fits the data extremely well. If this were real data, we'd be done, right?

No! A common error with SEM is to propose a model, fit the data, and then assert on the basis of fit indices that the model is "good." The problem is that some other,

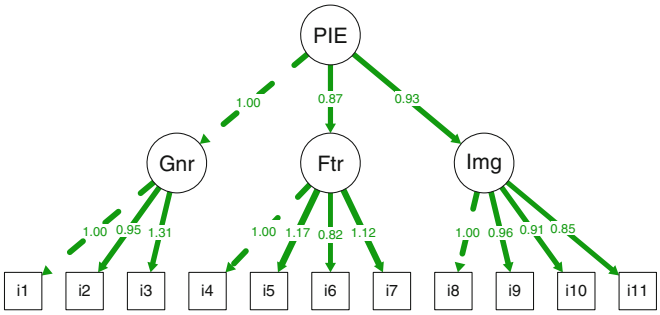

Fig. 10.4. Coefficients for the PIES structural model, using simulated consumer survey responses. Dotted lines represent coefficients automatically fixed to 1.00.

and perhaps more reasonable, model might be just as good or even better. Thus, there is an important second stage: establish that the proposed model fits *better* than a reasonable alternative model.

We test the PIES hierarchical model ("PIES $3+1$") against two alternatives. The first is a single factor alternative where one underlying involvement factor manifests in all items (as in Fig. 10.5), which we call "PIES 1." PIES 1 is a simpler model that proposes product involvement to be a single latent factor; if it fits the data as well as PIES $3+1$, then we could reject the more complex model and use this simple one instead. It is a good alternative to the hierarchical model both because it is simpler and because it focuses on the top level of the hierarchy, assessing whether it is advantageous to add the complications of the subfactors in PIES $3+1$.

The second alternative we consider is an uncorrelated three-factor model, where three independent factors are manifest in the three respective sets of items (shown in Fig. 10.6), or "PIES 3." This omits the top level, overall factor from the hierarchy and focuses on the three subfactors, asking whether they are better conceived as being separate instead of relating to a hierarchical model. If the PIES 3 model fits as well as PIES $3+1$, we could reject the complication of the hierarchical model and consider using the subscales as independent, largely unrelated assessment measures.

We specify and fit a one-factor model for PIES 1 using `lavaan` as follows:

```
> piesModelNH1 <- " PIES =~ i1 + i2 + i3 + i4 + i5 + i6  +
+                           i7 + i8 + i9 + i10 + i11 "
> pies.fit.NH1 <- cfa(piesModelNH1, data=piesSimData)
```

There is a complication in asserting the PIES 3 model. We can see in Fig. 10.6 that the number of paths and manifest variables in PIES 3 is the same as in the baseline $3+1$ hierarchical model (Fig. 10.2) because we allow the factors to be associated with one another. Because it estimates the same number of paths among all the

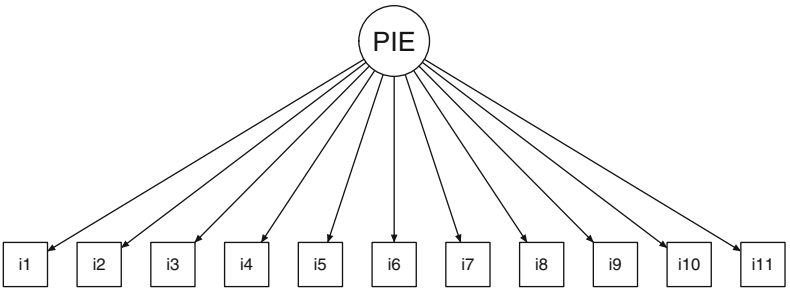

Fig. 10.5. A one-factor alternative model, PIES 1, in which a single latent factor of product involvement is manifest in all of the items, with no subfactors. We use this to test a simpler model than PIES 3+1 and determine whether it fits the data just as well.

variables, the global fit index for PIES 3 would be identical to that of PIES $3+1$. To differentiate the models, it is necessary to constrain the PIES 3 model in some other way.

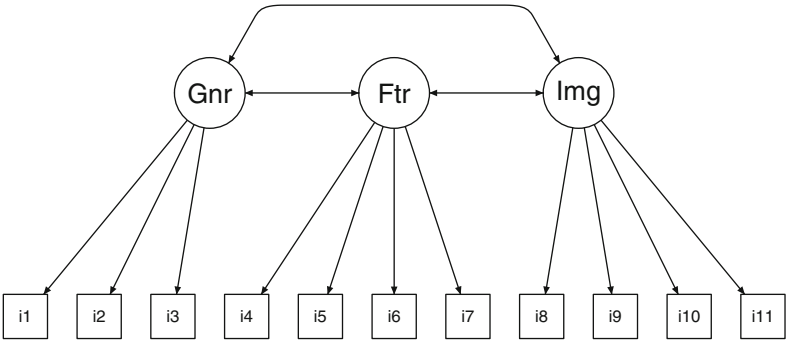

Fig. 10.6. A three-factor alternative, PIES 3. To differentiate this from the PIES $3+1$ model, the latent factor correlations here are constrained to show weak association among the factors. This allows us to test a model where the factors express largely separate constructs as opposed to closely related ones.

How should we constrain PIES 3? Because PIES 3 asserts that the 3 factors are largely independent and not part of a larger hierarchy, it implies that their intercorrelations should be relatively low. Thus, we could constrain the latent variable correlations to a small value such that they are not reasonably part of a hierarchy. A correlation of zero is unreasonable as it implies absolutely no relationship.[2] Instead of zero, we fix the non-hierarchical model to have correlation of 0.1 between

[2] Always be wary of models that assert or test independence; a well-known phenomenon in human research is that within a given domain, "everything correlates with everything else." Paul Meehl referred to this as the "crud" factor in research, and showed that it leads to research that finds "significant" associations everywhere [112].

the latent variables; this reflects an expectation of modest association that is too weak to justify a hierarchical model.

In `lavaan` we add the fixed relationships to the PIES 3 model syntax as additional lines and fit the model to the simulated data:

```
> piesModelNH3 <- " General =~ i1 + i2 + i3
+                   Feature =~ i4 + i5 + i6  + i7
+                   Image   =~ i8 + i9 + i10 + i11
+                   General ~~ 0.1*Feature
+                   General ~~ 0.1*Image
+                   Feature ~~ 0.1*Image "
> pies.fit.NH3 <- cfa(piesModelNH3, data=piesSimData)
```

In this model specification, the "~~" operator specifies a correlation between variables. By using a fixed value `0.1`, we specify that the value of the correlation should not be estimated but should be constrained to 0.1. The PIES 3 model requires that correlation be small among the latent factors, so we set the three possible correlations (General~Feature, General~Image, and Feature~Image) to our chosen value of 0.1.

The `semTools` package provides a command to compare CB-SEM (and therefore CFA) models: `compareFit(MODEL1, MODEL2, ...)`. This reports individual fit measures for each model along with pairwise model comparisons. Our PIES models are *nested*, meaning that one might start with the hierarchical model and then fix some of the paths coefficient to derive the three-factor model (specifically, constraining the factor correlations to 0.1), and again could fix some paths to derive the single factor model (specifically, setting the factor correlations to 1.0 so that they are identical and thus a single factor).

Here is the comparison of all three models:

```
> library(semTools)
> compareFit(pies.fit.NH1, pies.fit.NH3, pies.fit)
################## Nested Model Comparison #########################
                              chi df      p delta.cfi
pies.fit - pies.fit.NH3      222.43  3  <.001    0.0222
pies.fit.NH3 - pies.fit.NH1 2774.50  0  <.001    0.2812

################### Fit Indices Summaries ##########################
               chisq df pvalue   cfi   tli         aic         bic rmsea  srmr
pies.fit.NH1 3284.581 44  .000† .672  .589  108812.709  108948.860  .143  .102
pies.fit.NH3  510.078 44  .000† .953  .941  106038.205  106174.356  .054  .078
pies.fit      287.649 41  .000† .975† .966† 105821.776† 105976.494† .041† .030†
```

We start by inspecting the second half of the report, the "Fit Indices Summaries." For the non-hierarchical three-factor model PIES 3 (`pies.fit.NH3`), the fit was strong (e.g., CFI = 0.953, RMSEA = 0.054). If that were the only model that we tested, we would have concluded that it was an excellent fit. Yet when we

compare the PIES $3+1$ model, `pies.fit`, the fit indices are stronger (CFI $=$ 0.975, RMSEA $=$ 0.041). The stronger fit indices are indicated by the dagger symbol ("†").

Is PIES 3+1 stronger than PIES 3? We turn to the upper portion of the report to examine the model comparison. The first line of output ("pies.fit - pies.fit.NH3") reports the Chi-square test of the difference between the two models: Chisq $= 222.43$, $df = 3$. This is a strong and statistically significant difference, $p < 0.001$. We also see in the results that the one-factor model PIES 1 was a poor fit (CFI $= 0.672$ in the fit index summary) and much worse in comparison with PIES $3+1$ than even the non-hierarchical PIES 3 model. (In Sect. 11.3.5 we will also see how to interpret the `bic` values for model comparison.)

What does this tell us? For our data—which of course were simulated to fit the $3+1$ model—the three-factor hierarchical model was an excellent fit in itself and was better than two reasonable alternative models. If this were the case in real data (as claimed in [26]), it would establish a strong argument for the model.

What does this mean for us as marketers? It means that, if we saw such results in a product category of interest to us, we would not assume that product involvement is a unitary, single factor. Instead, we would wish to use the somewhat more informative and differentiated hierarchical model that assesses overall product interest alongside measures of feature and image involvement. Additionally, because the overall model fits the data well, it tells us that the $3+1$ model is a good representation of associations in the data (relative to plausible alternatives). This enhances our confidence that the survey items really do assess what we intend.

We note two important lessons. First, the simulated data is useful to examine the likelihood of being able to support a model. Simulated data showed us that the non-hierarchical PIES 3 model could fit the data well—if interpreted on its own—even when the PIES $3+1$ model was "true" given the data simulation process. Such tests with simulated data inform us about the *power* needed for model comparison.

Second, we see that simply establishing strong fit for a model is not enough; we also need to establish superiority over alternative models. If we only tested the non-hierarchical three-factor PIES 3 model, we might have concluded that it was an excellent model. Yet when we compare it to the PIES $3+1$ hierarchical model, we find the latter is a better fit to the data. We will encounter this again when we consider more general SEM models.

For marketers, there is another implication: when we devise a survey scale, we should test the assumed factor model to ensure that it meets our expectation. Imagine that we write a survey that asks about product preferences in four areas: performance, price, appearance, and quality. If each area has a few survey items and we add them together—as is common with surveys—then we are implicitly asserting a four-factor model for our survey. Before we use those added-up scores, we should

check our assumption about factors. Does our model match the data as we *believe* it should? If not, we might draw very misleading conclusions from the data. Test the model! R and `lavaan` make it easy to do this in just a few lines of code.

10.3 General Models: Structural Equation Models

We now consider a more general form of structural models, where latent constructs may influence one another in more complex ways. We consider an example from Iacobucci [83] concerning customer satisfaction ratings and their effect on stated intention to repurchase HP printers. The data consisted of responses to 15 satisfaction items, where there were three items each for factors of Quality, Cost (fair pricing), Value, Customer Satisfaction (CSat), and Repeat purchase intention.

The survey items, the variable names we use for them, and the higher-order latent factors (Quality, Cost, and so forth) are shown in Table 10.2.

The proposed structural model for the associations among the latent variables is shown in Fig. 10.7. For brevity, we omit consideration of the measurement model and the individual items for each factor.

As marketers, if we had collected consumer data from a survey such as this, we would have two goals. First, as we did with CFA above, we would wish to ascertain whether our proposed model of influence—for example, that perception of cost is associated with both perception of value and intent to repurchase, as shown in Fig. 10.7—is an adequate model for the data we have collected. Second, if the model fits the data well, we would answer questions about the relationships: how much does perception of quality relate to satisfaction? Is quality more important than perceived value? What is the largest determinant of stated intent to purchase again? And so forth.

To explore how to do this, we follow the same process as with CFA above. Specifically, we use a covariance-based SEM with four steps:

1. Define the structural model to be tested
2. Create simulated data that we use for illustration and debugging
3. Fit the model to the data
4. Compare the model to a simpler, alternative model

As always, simulating the data in Step 2 is illustrative here; you would use your own data instead, although we believe additional simulation is useful.

Table 10.2. A 15-item survey of purchase satisfaction, perceived value, and repurchase intent

Item	Text
Quality	
q1	The quality of the HP printer I bought is excellent
q2	HP printers are known to be highly reliable
q3	I'm sure my HP printer will last a long time
Cost	
c1	The HP printer was reasonably priced
c2	HP sets fair prices for its products
c3	The HP printers are no more expensive than others
Value	
v1	I feel like I got good value for this purchase
v2	The quality of the printer is worth its cost
v3	I could tell my boss this purchase was good value
CSat	
cs1	I am very satisfied with my newly purchase HP printer
cs2	My printer is better than I expected it would be
cs3	I have no regrets about having bought this printer
Repeat	
r1	I would buy another HP if I had to buy another printer
r2	I would buy other HP products
r3	I would tell my friends and coworkers to buy HPs

Item (variable) names are listed in the first column. Each division (Quality, Cost, Value, etc.) represents a latent factor manifest in the three following items. From Iacobucci [83]

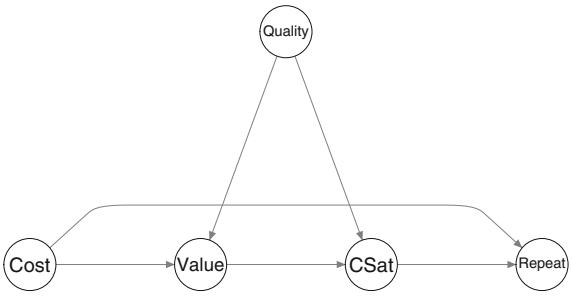

Fig. 10.7. A model of repeat purchase intent. In this model, the cost of a product is associated with both perception of value and intent to repurchase, while perception of quality relates to both perceived value and satisfaction, which is then associated with repurchase. Adapted from Iacobucci [83].

10.3.1 The Repeat Purchase Model in R

We begin by specifying the structural model that we wish to assess. This consists of a left-hand name of each latent factor, followed by the "is manifested by" symbol, "=∼" with the latent variables that it influences and its observed manifest variables (in this case, the 15 items from the customer survey). For convenience, we write the latent variables with capitalized names, and manifest items in lowercase. In `lavaan` this is:

```
> satModel <- " Quality =~ CSat + Value + q1 + q2 + q3  + 0*Cost
+              Cost    =~ Value + Repeat + c1 + c2 + c3
+              Value   =~ CSat + v1 + v2 + v3
+              CSat    =~ Repeat + cs1 + cs2 + cs3
+              Repeat  =~ r1 + r2 + r3 "
```

We read the first line as saying, "Quality influences CSat and Value, and is manifested as items q1, q2, and q3." Notice that we specify a fixed loading of zero between *Cost* and *Quality*. That reflects Iacobucci's report that those factors had near zero relationship (specifically, correlation of -0.03, [83] p. 676). Also, we are not interested in their relationship in this model and constraining the relationship may prevent spurious model effects.[3] Continuing with the model, we read "Cost influences Value and Repeat purchase intention, and is manifested on items c1, c2, and c3," and similarly for the other lines.

Next we obtain simulated data to use. If you prefer to load the data instead of simulating it, you may download it as follows:

```
> satSimData <- read.csv("http://goo.gl/MhghRq")
> summary(satSimData)
       q1              q2              q3              c1              c2
 Min.   :1.00   Min.   :1.000   Min.   :1.000   Min.   :1.00   Min.   :1.000
 1st Qu.:3.00   1st Qu.:3.000   1st Qu.:3.000   1st Qu.:3.00   1st Qu.:3.000
 Median :4.00   Median :3.000   Median :4.000   Median :4.00   Median :4.000
 Mean   :3.95   Mean   :3.535   Mean   :3.805   Mean   :4.34   Mean   :4.185
 ...
```

Once you have the data, you may proceed to Sect. 10.3.2. Otherwise, continue with the following; once again, data simulation is surprisingly straightforward.

Using the approximate loadings reported by Iacobucci [83, p. 677], we write the data model as:

```
> satDataModel <- " Quality =~  0.59*CSat + 0.56*Value +
+                               0.9*q1 + 0.9*q2 + 0.9*q3 + 0*Cost
+                  Cost    =~ -0.5*Value + -0.29*Repeat +
+                               0.9*c1 + 0.9*c2 + 0.9*c3
+                  Value   =~  0.06*CSat + 0.9*v1 + 0.9*v2 + 0.9*v3
+                  CSat    =~  0.48*Repeat + 0.9*cs1 + 0.9*cs2 + 0.9*cs3
+                  Repeat  =~  0.9*r1 + 0.9*r2 + 0.9*r3 "
```

Then we simulate the data for $N = 200$ respondents and convert to Likert type scaled values using the same approach as in Sect. 10.2.1:

```
> set.seed(33706)  # continuing the island tour
> satData.norm <- simulateData(satDataModel, sample.nobs=200)
> satSimData <- data.frame(lapply(satData.norm,
+                          function(x) { as.numeric(cut(x, breaks=7)) } ))
```

We omit here the data quality checks (see Sect. 10.2.1), but it is a good idea for you to inspect those.

[3] In general, fixing parameters is not recommended; the whole point of SEM is to estimate parameters. However, in some cases, especially with smaller samples as we consider here, it may help to focus a model on the influences under consideration if one constrains factors. It is possible with `lavaan` to constrain to any value, not just 0.

10.3.2 Assessing the Repeat Purchase Model

To fit the model, we use `sem(MODEL, DATA)` and add an argument, `std.lv=TRUE`, to standardize the latent variables because we are interested to compare relative influence strength (the alternative is to treat them in terms of the unit scales of the observed items, which might be of interest for CFA). In the abbreviated output, we see a strong model fit (CFI = 0.998 and low residuals):

```
> sat.fit <- sem(satModel, data= satSimData, std.lv=TRUE)
> summary(sat.fit, fit.measures=TRUE)
lavaan (0.5-17) converged normally after  24 iterations

  Number of observations                           200

  Estimator                                         ML
  Minimum Function Test Statistic               85.454
  Degrees of freedom                                84
  P-value (Chi-square)                           0.435
...
User model versus baseline model:
  Comparative Fit Index (CFI)                    0.998
...
Root Mean Square Error of Approximation:
  RMSEA                                          0.009
  90 Percent Confidence Interval          0.000  0.040
  P-value RMSEA <= 0.05                          0.993

Standardized Root Mean Square Residual:
  SRMR                                           0.052
...
```

We plot the resulting structural coefficients for the proposed model with arguments for `structural=TRUE` to suppress the loadings for the manifest items and `nCharNodes=7` to put the full factor names in the latent variable circles:

```
> semPaths(sat.fit, what="est", fade=FALSE, residuals=FALSE,
+          layout="tree", structural=TRUE, nCharNodes=7, edge.label.cex=1)
```

The result is Fig. 10.8. Not surprisingly, the simulated data show effects close to what we specified (but not exactly the same, which demonstrates that model recovery is not perfect).

As we have already seen, a great fit in CB-SEM is not enough! We still need to compare our proposed model to one or more plausible alternative models, in order to demonstrate that our proposal is superior to other reasonable models.

How do we define an alternative model? It depends on your goal and theory. In some cases, you might wish to compare to a simpler model, in order to show that relationships are more complex or to fit a more precise model. In other cases, you could

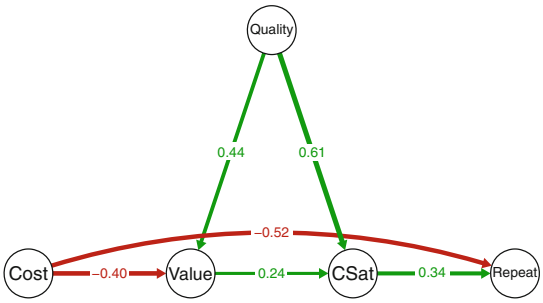

Fig. 10.8. Coefficient estimates for the repeat purchase model, using simulated data.

compare to an existing model from the literature or previous research. In still others, you might show that a proposed complex model is too complex, and that a simpler model is more effective. As a general principle, we prefer to show that a model is better than a simpler model with fewer paths, and just as good as (i.e., not significantly worse than) a more complex model with a larger number of paths.

In the present case, our full model proposes that `Quality` and `Cost` do not have a simple relationship with single variables but are associated with multiple other variables. For instance, `Cost` influences not only perception of `Value` but also the likelihood of `Repeat` purchase. An alternative is a simpler, more obvious model, where each is associated with only a single other variable, such as `Cost` affecting `Value` but not directly influencing `Repeat`. To support our more complex model, we wish to show that the simpler model is inadequate. Thus, we define an alternative model where each latent variable only influences one other variable, giving us a model with 2 fewer paths as shown in Fig. 10.9. The alternative model specification in `lavaan` is:

```
satAltModel <- " Quality =~ CSat  + q1 + q2 + q3 + 0*Cost
                 Cost    =~ Value + c1 + c2 + c3
                 Value   =~ CSat  + v1 + v2 + v3
                 CSat    =~ Repeat + cs1 + cs2 + cs3
                 Repeat  =~ r1 + r2 + r3 "
```

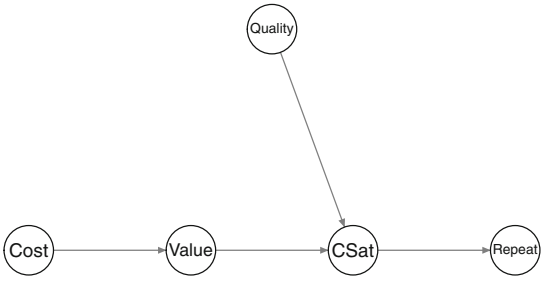

Fig. 10.9. An alternative structural model for repeat purchase influence, omitting the direct associations of cost with repeat purchase and of perceived quality with value.

We fit the alternative model to the simulated data with `sem()` and compare that fit to the proposed model with `compareFit()`:

```
> satAlt.fit <- sem(satAltModel, data=satSimData, std.lv=TRUE)
> compareFit(sat.fit, satAlt.fit, nested=TRUE)
################## Nested Model Comparison #########################
                      chi df      p delta.cfi
sat.fit - satAlt.fit 37.51  2  <.001    0.0495

#################### Fit Indices Summaries ##########################
              chisq df pvalue   cfi   tli       aic       bic rmsea  srmr
sat.fit     85.454 84   .435† .998† .997† 9174.942† 9293.681† .009† .052†
satAlt.fit 122.962 86   .006  .949  .937  9208.449  9320.592  .046  .095
```

Once again we see that—taken on its own—the alternative model appears to be a good fit to the data (e.g., CFI = 0.95, RMSEA = 0.046) yet the proposed model is significantly better, showing Chi-square ($df = 2$) = 37.51 for the model difference, $p < 0.001$, and stronger fit indices with lower residuals.

If these were results from real data, we could draw a few conclusions. First, the model shows good fit to the observed (in this case, simulated) data, so we are able to interpret the results. Second, it is better than a simpler alternative model, which argues that our model is not an arbitrarily good fit but is preferable to a plausible alternative. Most importantly, we would use the coefficient estimates in the model to answer our substantive questions about the associations of the latent factors with the outcomes of interest to us as marketers. However, we omit this step here because we've done this several times for other models and it does not further advance our knowledge of R; see Iacobucci [83] for conclusions in this case.

10.4 The Partial Least Squares (PLS) Alternative

The first two models we have considered in this chapter exemplify *covariance-based* structural equation modeling (CB-SEM). Such models attempt to account for as much of the total covariance in the data as possible, among all observed and latent variables. CB-SEM requires that a data set complies with relatively strict assumptions about data distributions (continuous data, normally distributed residuals), the number of indicators per factor (generally three or more), reliability of indicators, and sample size (some authorities recommend several hundred, although it is possible that samples of $N = 100$ or even $N = 50$ may be adequate when measures are very reliable; see Iacobucci [84]). When such assumptions are met, CB-SEM is a powerful tool that tests a model rigorously, assesses overall strength of the model, and allows for model comparison.

When data do not comply with the assumptions of CB-SEM or come from a modest sample size with potentially less reliable indicators, an alternative is *partial least squares* structural equation modeling (PLS-SEM). PLS-SEM is often able to yield

estimates of path coefficients in models where CB-SEM would fail. However, PLS-SEM does not allow one to say much about model fit or comparative strength; there is no accepted measure of global "goodness of fit" that is comparable across models. Thus, we recommend CB-SEM when possible; but when CB-SEM fails, PLS-SEM may still give useful estimates of model coefficients.

In this section, we demonstrate PLS-SEM for the repeat purchase model that we examined above. We will see that PLS-SEM can estimate parameters with a sample where CB-SEM fails, but with greater uncertainty about the model's results.

10.4.1 PLS-SEM for Repeat Purchase

We conduct PLS with the `semPLS` package; install that for the following examples. We continue with the example from Iacobucci [83] of the influence of customer satisfaction and perceived value on intended repeat purchase of a computer printer. You may review that model—and find steps to create simulated data that we use here—in Sect. 10.3 above.

Let's see why PLS-SEM can be useful. Our simulated data set (`satSimData`) has $N = 200$ observations, which was modeled successfully with CB-SEM. What if the sample were smaller? Let's take $N = 50$ rows and try to fit the CB-SEM model to those:

```
> set.seed(90704)
> satSimData2 <- satSimData[sample(nrow(satSimData), 50), ]
> describe(satSimData2)
    vars  n mean   sd median trimmed  mad min max range  skew kurtosis   se
q1     1 50 3.80 1.39      4    3.75 1.48   1   7     6  0.26    -0.23 0.20
...

> sat.fit2 <- sem(satModel, data= satSimData2, std.lv=TRUE)
Warning messages:
1: In lav_model_vcov(lavmodel = lavmodel, lavsamplestats = lavsamplestats,  :
  lavaan WARNING: could not compute standard errors!
...
```

Model estimation with `lavaan` fails because we do not have enough data.[4] If you inspect the model object, you will see extreme and nonsensical values:

```
> summary(sat.fit2, fit.measures=TRUE)
lavaan (0.5-17) converged normally after 9043 iterations
...
                   Estimate  Std.err  Z-value  P(>|z|)
Latent variables:
...
```

[4] The small sample exacerbates another reason for estimation difficulty: our data is highly collinear due to the factor structure imposed when we simulated it to match the report by Iacobucci [83].

```
  Cost =~
    Value             -0.003
    Repeat            -0.011
    c1                 0.014
    c2                57.293
...
Variances:
    q1                 1.540
...
    c1                 1.515
    c2             -3280.640
...
```

It is unreasonable to think that one survey item about cost (c1) has nearly zero relationship while another (c2) is thousands of times more strongly related (57.29/0.014), or that one has thousands of times as much variance as another. This indicates model instability as the message from `lavaan` warned us.

We will try PLS-SEM instead. The first step is to define a *measurement* model that links underlying latent variables to their observed manifest variables such as survey items, and then to define a *structural* model that links latent variables to one another. In `lavaan` these were combined into a single step (cf. Sect. 10.3.1) but with the `semPLS` package they are separate.

Whereas `lavaan` uses a formula syntax to define relationships among variables, `semPLS` uses a matrix format. In this format, each row represents one "arrow" in a model. The first column of the row represents the *from* variable while the second column represents the *to* variable. Thus, an arrow from *Quality* to the manifest variable *q1* would be represented as a matrix line `("Quality", "q1")`.

The matrix definition is not as difficult as it may sound; we need only list the *from* and *to* entries in a simple format. Referring to the model in Sect. 10.3.1, we define the measurement model (latent to observed variables) model as:

```
> satPLSmm <- matrix(c(
+   "Quality", "q1",
+   "Quality", "q2",
+   "Quality", "q3",
+   "Cost",    "c1",
+   "Cost",    "c2",
+   "Cost",    "c3",
+   "Value",   "v1",
+   "Value",   "v2",
+   "Value",   "v3",
+   "CSat",    "cs1",
+   "CSat",    "cs2",
+   "CSat",    "cs3",
+   "Repeat",  "r1",
+   "Repeat",  "r2",
+   "Repeat",  "r3" ), ncol=2, byrow=TRUE)
```

The structural model presents the latent variable relationships using the same kind of matrix format. Referring to the model shown in Fig. 10.7 we write:

```
> satPLSsm <- matrix(c(
+   "Quality", "CSat",
+   "Quality", "Value",
+   "Cost",    "Value",
+   "Cost",    "Repeat",
+   "Value",   "CSat",
+   "CSat",    "Repeat" ), ncol=2, byrow=TRUE)
```

We now fit the PLS model using the simulated 50-respondent data set. We use `plsm(data, strucmod, measuremod)` from `semPLS` to create a PLS model using the structural and measurement model matrices that we defined. Then we use `sempls(model, data, wscheme)` to estimate the model parameters:

```
> library(semPLS)

> satPLS.mod <- plsm(data=satSimData2, strucmod=satPLSsm, measuremod=satPLSmm)
> satPLS.fit <- sempls(model=satPLS.mod, data=satSimData2)
All 50 observations are valid.
Converged after 14 iterations.
Tolerance: 1e-07
Scheme: centroid
```

We can now inspect the results. To begin, we examine the fit between the latent variables and the manifest observations (items), i.e., the estimated factor structure, using `plsLoadings(MODEL)`. We see that the items have positive and moderate to high loadings, and are similar in magnitude within each latent variable:

```
> plsLoadings(satPLS.fit)
    Cost Quality Value CSat Repeat
c1  0.39       .     .    .      .
c2  0.82       .     .    .      .
c3  0.78       .     .    .      .
q1     .    0.54     .    .      .
...
```

Each latent variables has a moderate to strong loading with its manifest variables, so we are reassured that the model reflects those relationships. If a latent variable failed to load significantly—for example, with factor loadings below 0.3 for any manifest variable, or below 0.5 for all of its manifest variables—then we would be concerned about the model, sample size, or reliability of the measures and would conduct further investigation (or, at a minimum, replicate the results, as in Sect. 10.4.3 below).

We now use `pathCoeff(MODEL)` to examine the structural coefficients between latent variables, which is what we most care about:

```
> pathCoeff(satPLS.fit)
        Cost Quality  Value   CSat Repeat
Cost       .       . -0.196      . -0.393
Quality    .       .  0.323  0.400      .
Value      .       .      .  0.062      .
CSat       .       .      .      .  0.231
Repeat     .       .      .      .      .
```

We see that cost has a negative influence on perceived value and likelihood to repeat purchase, while customer satisfaction has a positive influence on repeat purchase.

10.4.2 Visualizing the Fitted PLS Model*

This section is optional because it detours into modest additional requirements and file handling.

As we have seen, it is convenient to plot the results of structural models and interpret coefficients and models visually. For the PLS model, we can plot the structural coefficients using `pathDiagram(MODEL, FILE, full=FALSE, ...)` but this does not immediately create a plot within R. Instead, it outputs a `.dot` file that is then processed by the freely available Graphviz software package to produce a corresponding image as a PDF file [59]. Graphviz is available at http://www.graphviz.org.

Once Graphviz is installed, a PDF output file with the paths and coefficients for a fitted PLS model objected may be created with `pathDiagram`:

```
>   pathDiagram(satPLS.fit, file = "satPLSstruc", full = FALSE, digits = 2,
+     edge.labels = "values", output.type = "graphics", graphics.fmt = "pdf")
```

The result is shown in Fig. 10.10. Comparing the values to those obtained from the full sample with CB-SEM (Fig. 10.8) we see that the coefficients are identical in direction (positive or negative) and similar in relative magnitude.

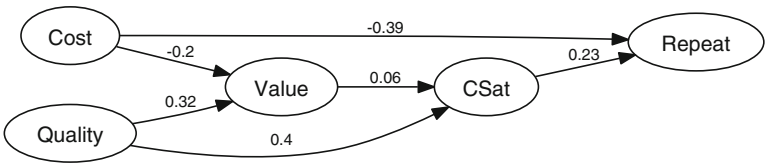

Fig. 10.10. PLS estimate for the repeat purchase model (with $N = 50$).

Because PLS models do not assess global model fit, there is not a general way to compare CB-SEM and PLS-SEM results apart from interpreting the models and their implications, so it is not advisable to compare the coefficients directly between Figs. 10.10 and 10.8.

10.4.3 Assessing the PLS-SEM Model

Unlike CB-SEM, PLS-SEM models do not have a summary metric that allows global model assessment and comparison [73]. Instead, at a minimum we recommend three steps:

- Examine the model's coefficients for intelligibility, as we did in Sect. 10.4.1
- Examine the overall R^2 for the model, and determine (largely subjectively) whether sufficient variance is explained to be useful
- Bootstrap the model to examine coefficient stability

It is easy to find R^2 for each of the latent variables using the `rSquared(MODEL)` function:

```
> rSquared(satPLS.fit)
        R-squared
Cost            .
Quality         .
Value        0.18
CSat         0.18
Repeat       0.26
```

A problem with R^2 is that there is no general standard for whether the values are adequate. R^2 is a measure of overall variance explained within each part of the model, but its interpretation is dependent on what you might expect for a given type of data (in other words, it depends on your experience with models in a domain) and it can be increased simply by adding variables (i.e., overfitting). There are various rules of thumb for interpreting R^2, but they are domain specific. If we use the standards for interpreting correlation coefficients in behavioral data, where $r = 0.3$ indicates a moderately strong correlation (Sect. 4.5), then $R^2 > 0.09$ could be a reasonable goal for a moderately strong association in the model, assuming that you have been parsimonious in selecting the number of associated variables.

We recommend a more general approach that does not rely on R^2 and instead uses a bootstrap process to assess coefficient stability. In `semPLS`, this may be done with the `bootsempls()` command. We fit the PLS model object to 500 resampled sets of observations:

```
> set.seed(04460)  #note:some semPLS versions give different results
> satPLS.boot <- bootsempls(satPLS.fit, nboot=500, start="ones")
Resample: 500 Done.
Warning message:
In bootsempls(satPLS.fit, nboot = 500, start = "ones") :
  There were 409 apparent convergence failures;
  these are discarded from the 500 bootstrap replications returned.
> summary(satPLS.boot, type = "bca", level = 0.9)
Call: bootsempls(object = satPLS.fit, nboot = 500, start = "ones")

Lower and upper limits are for the 90 percent bca confidence interval

         Estimate     Bias Std.Error   Lower  Upper
lam_1_1    0.3953  0.00452    0.2268 -0.0796  0.671
lam_1_2    0.8167 -0.00667    0.0926  0.5314  0.897
...
beta_1_3  -0.2061 -0.06158    0.1242 -0.3370  0.271
beta_2_3   0.3159 -0.01384    0.1431  0.1195  0.540
beta_2_4   0.4013  0.04927    0.1036  0.1577  0.514
beta_3_4   0.0627  0.00689    0.1706 -0.3480  0.251
beta_1_5  -0.3873 -0.00942    0.1406 -0.5515 -0.139
beta_4_5   0.2426  0.00496    0.1533 -0.2289  0.442
```

In examining the results, we see two indications of problems: a warning that approximately 80 % of the PLS iterations failed to converge, and several model coefficients (such as the beta values that reflect the structural model) whose upper and lower bounds include 0, and for which we therefore do not have even *directional* confidence.

We can see the problems visually using a parallel plot. This plots all bootstrap estimates of the structural coefficients so we can see the spread in estimates; we use `reflinesAt=0` to add a reference line at 0 in order to see direction, and include `varnames` to label the *Y* axis with friendly names:

```
> parallelplot(satPLS.boot, reflinesAt = 0, alpha=0.8,
+   varnames=attr(satPLS.boot$t, "path")[16:21],
+   main="Path coefficients in 500 PLS bootstrap iterations (N=50)")
```

The resulting plot is shown in Fig. 10.11, where the gray lines represent individual bootstrap estimates and the red lines show median (solid line) and outer 95 % observed intervals (dotted lines). The estimates fluctuate widely for most of the coefficients. We read this by looking at the spread of estimates along each of the horizontal grid lines representing one model coefficient. For example, the influence of `Cost` on `Repeat` purchase is generally estimated to be strongly negative, but several of the estimates hold the relationship to be strongly positive. Additionally, 2 of the 6 coefficient ranges straddle the zero line and thus are not significantly different from zero.

The convergence problems and bootstrap ranges demonstrate that estimates in our PLS model with $N = 50$ are unstable. However, whether they are useful in a given situation is a judgment call. Depending on the question at hand and the risks involved, an analyst might conclude that the estimates are not adequately reliable—or alternatively might conclude that, although the instability is not ideal, the estimates are still useful because they are more informative than nothing.

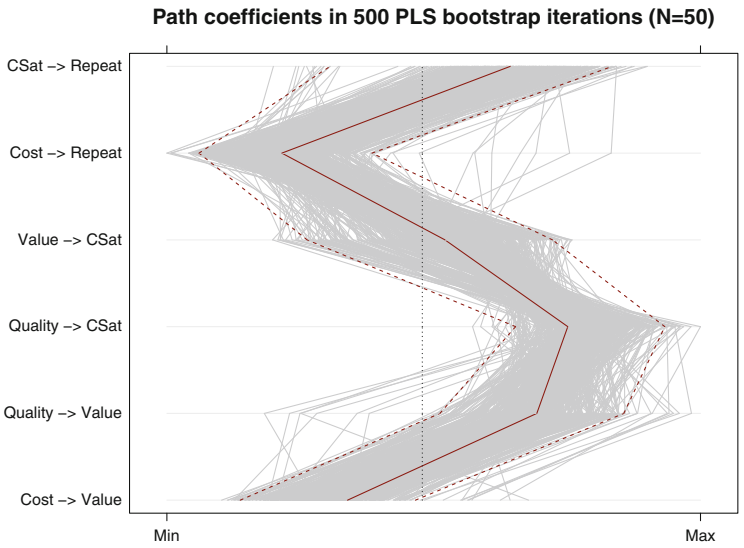

Fig. 10.11. Bootstrapped coefficients for the PLS model, showing divergent estimates for $N = 50$ observations. Each line plots the six estimated coefficients for one complete bootstrap iteration. The model is unstable with the small sample and 409 of 500 bootstrap iterations failed to converge, so these results come from the other 91 iterations.

10.4.4 PLS-SEM with the Larger Sample

Is PLS-SEM more stable with the larger sample? We can examine that quickly for the full data set from Sect. 10.3.1 with no need to respecify the model. The analysis is identical, except for using the full data (`satSimData`) in the modeling commands:

```
> satPLS.modF <- plsm(data=satSimData, strucmod=satPLSsm, measuremod=satPLSmm)
> satPLS.fitF <- semPls(model=satPLS.mod, data=satSimData)
All 200 observations are valid.
Converged after 7 iterations.
Tolerance: 1e-07
Scheme: centroid
```

We see that the path coefficients for the $N = 200$ data are similar to the $N = 50$ estimates, but the magnitudes are somewhat different:

```
> pathCoeff(satPLS.fitF)
         Cost Quality Value  CSat Repeat
Cost        .       . -0.27     .  -0.32
Quality     .       .  0.30  0.34      .
Value       .       .     .  0.22      .
CSat        .       .     .     .   0.29
Repeat      .       .     .     .      .
```

As before, we check PLS-SEM stability with a bootstrap. We repeat the procedure from Sect. 10.4.3, this time with the model for the full data set:

```
> set.seed(04460)
> satPLS.bootF <- bootsempls(satPLS.fitF, nboot=500, start="ones")
Resample: 500 Done.
> parallelplot(satPLS.bootF, reflinesAt = 0, alpha=0.8,
+   varnames=attr(satPLS.bootF$t, "path")[16:21],
+   main="Path coefficients in 500 PLS bootstrap iterations (N=200)")
```

The bootstrap with full $N = 200$ data converged on all 500 iterations (as opposed to 80 % failures to converge with $N = 50$ above). The parallel plot of estimates in Fig. 10.12 shows bootstrap coefficient values that are grouped much more tightly (the gray lines) and confidence intervals that do not cross zero (red lines).

These are results that we could use more confidently than we obtained for $N = 50$. Either way, PLS-SEM opens up the opportunity for that decision. With the bootstrap we were able to find instability with the smaller sample but stability for the larger. With such capability we can make an informed choice about whether to use imperfect results.

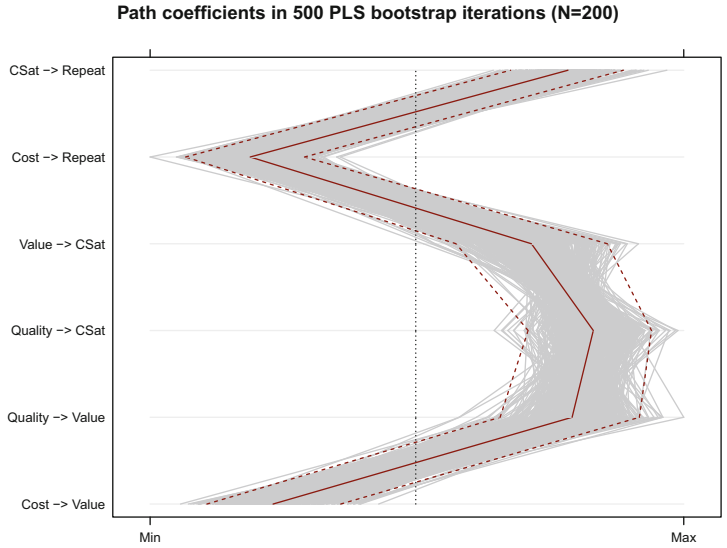

Fig. 10.12. Bootstrapped coefficients for the PLS model with a larger sample, showing tighter estimates with $N = 200$ observations.

10.5 Learning More*

Structural models are a complex topic, and this chapter is intended primarily to demonstrate R's capability for experienced SEM users while inspiring others to learn more. To use SEM well, you will need substantial background in addition to this overview. For CB-SEM, an excellent text is Kline's *Principles and Practice of Structural Equation Modeling* [92]. Kline presents a social science perspective that is similar to many marketing applications, especially for application to survey data.

A guide to SEM models in R using the `lavaan` package is Beaujean [9]. The combination of Kline's *Principles* [92] for general concepts along with Beaujean's guide to implementation in R would provide a thorough grounding in SEM with R.

As you learn more about structural models, you will encounter SEM traditions that reflect a diversity of statistical foundations and applications. One difference involves model specification. The *Lisrel* tradition—named after one of the first SEM software programs [88, 89]—is exemplified in Iacobucci's article [83] and presents models in terms of matrix algebra and Greek lettering. This is a very precise way to specify models but is difficult for non-specialists. An alternative is the *Mplus* tradition—also named after a software program—which uses simpler equation style specifications. We generally recommend marketers to start with the latter kind of model specification, as we did in the present chapter.

PLS-SEM is popular for marketing applications but, unlike the case with CB-SEM, to date there have been few sources to learn about it. A paper from Hair et al describes how to do PLS-SEM appropriately [73]. At the time of writing, other general references on PLS-SEM included a textbook [72] and a paper presenting an overview of marketing applications [76].

In R, there are several packages available for SEM. In this chapter we used the `lavaan` package [135] for CB-SEM and the `semPLS` package [115] for PLS-SEM. One of the earliest and widely used packages for SEM is the `sem` package [52], which has many examples available online for various models and situations (e.g., [50]). The OpenMx Project provides a powerful system for SEM in the `OpenMx` package [13].

10.6 Key Points

We have seen two examples of complex models in marketing applications: to examine whether a survey instrument has good factor structure, and to estimate the relationship in survey data between customer satisfaction and intent to repurchase a product. Additionally, we saw how such models may be estimated in both covariance and partial least squares approaches.

The following suggestions will help you to succeed at this kind of modeling:

- Learn about structural models and their assumptions; do not fit them blindly. If you become discouraged by mathematical treatments, keep looking; the concepts can be challenging but there are excellent and readable expositions such as Kline [92].
- A structural equation model (SEM) relates observed manifest variables—such as data points or survey responses—to underlying latent variables. It estimates the strength of associations in a proposed model, as well as the degree to which the model fits the observed data.
- SEM may be used to check the factor structure of survey items and their relationships to proposed latent variables; this is known as CFA. A good practice in survey research is to assess those relationships; do not simply assume that survey items relate to one another or to latent constructs as expected (Sect. 10.2).
- SEM can also be used to test more complex models, in which latent variables affect one another and are related to multiple sets of manifest variables.
- Two general approaches to SEM are the covariance-based approach (CB-SEM), which attempts to model the relationships among the variables at once and thus is a strong test of the model, and the partial least squares approach (PLS-SEM), which fits parts of the data sequentially and has less stringent requirements.
- After you specify a CB-SEM model, simulate a data set using `simulateData()` from `lavaan` with reasonable guesses as to variable loadings. Use the simulated data to determine whether your model is likely to converge for the sample size you expect.
- Plot your specified model graphically and inspect it carefully to check that it is the model you intended to estimate.
- Whenever possible, specify one or two alternative models and check those in addition to your model. Before accepting a CB-SEM model, use `compareFit()` to demonstrate that your model fits the data better than alternatives.
- If you have data of varying quality, nominal categories, small sample, or problems converging a CB-SEM model, consider partial least squares SEM (PLS-SEM).
- For PLS-SEM, use a bootstrap procedure (such as `bootsempls()` in the `semPLS` package) to examine the stability of coefficients.