SEM Tables with the semTable package for R

Paul E. Johnson, CRMDA, University of Kansas,
pauljohn@ku.edu
Benjamin Kite, CRMDA and H&R Block,
bakite@ku.edu

Guide No: 50

Keywords: Structural Equation Models, LaTeX Tables, HTML Tables
See https://crmda.ku.edu/guides for updates.

Feb. 28, 2019

Abstract

The `semTable` package features functions for preparing report tables for estimates of structural equation models fitted with `lavaan` (Rosseel, 2012). This essay discusses the usage of the `semTable` function and provides a profuse collection of output examples.

Contents

1425 Jayhawk Blvd.
Watson Library, Suite 470
Lawrence, KS 66045-7594

Web: https://crmda.ku.edu
Email: crmda@ku.edu
Phone: 785-864-3353

1 semTable

The `semTable` function is the featured offering of the `semTable` package for R (R Core Team, 2019). This function allows authors who estimate structural equation models (SEM) with the lavaan package for R to, more-or-less automatically, generate acceptable tabular summaries for reports and presentations. `semTable` has been offered as a part of the `kutils` package in the past, but we believe it is sufficiently well developed that it can stand on its own. As soon as `semTable` is accepted on CRAN, this function will be removed from `kutils`. The `semTable` package also includes a function called `modelComparison`, which can assemble a standard format for a "diff" test result for structural equation models. That function is considerably less elaborate than `semTable`.

.

2 Important Parameters

object: A fitted lavaan model object or a list with one or more SEM fitted lavaan model objects. We suggest providing "pretty names" for inclusion in the final report, as in `list("Model A" = obj1, "Model B" = obj2)`. Results will be side-by-side. Hint: take care to adjust the columns parameter to make the results fit within the space allowed.

paramSets: Each fitted SEM may have many different parameter sections. This argument can include any of the following:

`c("loadings", "slopes", "intercepts", "residualvariances", "residualcovariances", "latentmeans", "latentvariances", "latentcovariances", "thresholds", "constructed", "fits")` The default is `"all"`, and the setting applies to all of the models in the fitted object list.

paramSetLabels: This should be a named vector of "pretty-printable" labels for the parameter sets. It is not necessary to provide improved labels for all `paramSets` items, but it is allowed to do so.

columns: choose the desired columns in the output. In the default table in `lavaan` summaries, the columns are ["Estimate", "Std. Error", "Z", and "p"]. In our opinion, inclusion of both the standard error and the Z is redundant and the p value may not be necessary if significance stars are used. Hence we provide alternatives to "tighten up" the result table. To specify columns, the legal parameter values may be drawn from `c("est", "se", "z", "p", "rsquare", "estse", "eststars", "estsestars")`. The four new values are the R-square for the item, a compact combined estimate and standard errors (estse: "1.2(0.02)"), with estimates with stars and no standard errors (eststars: "1.2**") or estimates with standard errors in parentheses with "significance stars" (estsestars: "1.2(0.02)**"). Columns can differ between models, so a named list of vectors will be accepted: `list("Model A" = c("est", "se"), "Model B" = c("estse", "p"))`.

columnLabels: names list of "presentable" titles for columns that authors prefer. Provide these names by parameter, not by model, For example, any of these might be suitable: `est = "Estimate"`, `se = "Std. Err."`, `z = "Z"`, `p = "p"`, `rsquare = "R Square"`, `estse = "Estimate(Std.Err.)"`.

fits: selection of fit indices to include. The help page has a partial list of these, but the lavaan estimator can generate additional fit indicators for models of different types. This is the one part of the semTable output that is likely to require some hand-editing in order to be truly publication ready.

varLabels: Rather than R variable names, a truly polished table needs presentable variable names. Provide a named vector of names in the format “orig.R.name” = “new.pretty.name”. See examples below.

type: Can be ”latex”, ”html” or ”csv”. If a table object is created with one format in mind, the object can be re-designated to a new format. See examples in “?semTable”.

LaTeX specific arguments:

table.float: is an enclosing float table object required, or should we simply make a tabular (or longtable) object.

longtable If longtable is true (probably preferred), use the LaTeX class longtable. Otherwise tabular.

caption: title for the floating table, if table.float=TRUE.

label: label to be used for internal cross-referencing.

centering: One of these `c("siunitx", "none")`.

alpha: The levels of statistical significance to be used for assigning starts (only needed if columns includes eststars or estsestars).

file: This is a convenience function to save the results in a separate file. It will add suffix `".tex"`, `".html"`, or `".csv"`, depending on the value of type. This is the same as fitting a model with `print.results = FALSE`, then `semTable` to produce `object.tbl`, and then running `cat(object.tbl, file= "fn")` to save the result object in a file.

print.results: If FALSE, the results are not immediately displayed. This is advantageous if an author wants to revise the marked-up table results or control the point at which they are delivered in the documents.

3 Confirmatory Factor Analysis

```
require(lavaan)
library(semTable)
tempdir <- "tmpout"
if(!dir.exists(tempdir)) dir.create(tempdir)
```

```
## The example from lavaan's docs
HS.model <- ' visual   =~ x1 + x2 + x3
              textual =~ x4 + x5 + x6
              speed   =~ x7 + x8 + x9
 '
```

Because the SEM estimation can be time consuming, we will save a file with the fitted model and re-use it if it is available when the vignette is recompiled.

```
if(file.exists("fit1.rds")){
   fit1 <- readRDS("fit1.rds")
 } else {
     fit1 <- cfa(HS.model, data = HolzingerSwineford1939,
             std.lv = TRUE, meanstructure = TRUE)
     saveRDS(fit1, "fit1.rds")
 }
```

The “raw” output from the `lavaan` `summary` function is as follows:

```
summary(fit1)
```

```
lavaan 0.6-3 ended normally after 20 iterations

  Optimization method                           NLMINB
  Number of free parameters                         30

  Number of observations                           301

  Estimator                                         ML
  Model Fit Test Statistic                      85.306
  Degrees of freedom                                24
  P-value (Chi-square)                           0.000

Parameter Estimates:

  Information                                 Expected
  Information saturated (h1) model          Structured
  Standard Errors                             Standard

Latent Variables:
                   Estimate  Std.Err  z-value  P(>|z|)
  visual =~
    x1                0.900    0.081   11.128    0.000
    x2                0.498    0.077    6.429    0.000
    x3                0.656    0.074    8.817    0.000
  textual =~
    x4                0.990    0.057   17.474    0.000
    x5                1.102    0.063   17.576    0.000
    x6                0.917    0.054   17.082    0.000
  speed =~
    x7                0.619    0.070    8.903    0.000
    x8                0.731    0.066   11.090    0.000
    x9                0.670    0.065   10.305    0.000

Covariances:
                   Estimate  Std.Err  z-value  P(>|z|)
  visual ~~
    textual           0.459    0.064    7.189    0.000
    speed             0.471    0.073    6.461    0.000
  textual ~~
    speed             0.283    0.069    4.117    0.000

Intercepts:
                   Estimate  Std.Err  z-value  P(>|z|)
   .x1                4.936    0.067   73.473    0.000
   .x2                6.088    0.068   89.855    0.000
   .x3                2.250    0.065   34.579    0.000
   .x4                3.061    0.067   45.694    0.000
   .x5                4.341    0.074   58.452    0.000
   .x6                2.186    0.063   34.667    0.000
```

```
   .x7                 4.186    0.063   66.766    0.000
   .x8                 5.527    0.058   94.854    0.000
   .x9                 5.374    0.058   92.546    0.000
    visual             0.000
    textual            0.000
    speed              0.000

Variances:
                   Estimate  Std.Err  z-value  P(>|z|)
   .x1                 0.549    0.114    4.833    0.000
   .x2                 1.134    0.102   11.146    0.000
   .x3                 0.844    0.091    9.317    0.000
   .x4                 0.371    0.048    7.779    0.000
   .x5                 0.446    0.058    7.642    0.000
   .x6                 0.356    0.043    8.277    0.000
   .x7                 0.799    0.081    9.823    0.000
   .x8                 0.488    0.074    6.573    0.000
   .x9                 0.566    0.071    8.003    0.000
    visual             1.000
    textual            1.000
    speed              1.000
```

It is not necessary to specify a set of "pretty" variable labels, but sometimes this makes for more pleasant tables. In this case, we create a vector of labels first, and then put it to use. To prevent the table from becoming too wide, we ask for two columns in the output, `"estse"` and `"p"`. The default `semTable` input would include a larger set of parameters, but we don't need all of them. The default table includes latent intercepts and latent variances, which are restricted to 0 and 1 by the model itself, so it is not necessary to include them. Hence, we ask for a smaller selection of parameter sets:

```
vlabs <- c("x1" = "Vis 1", "x2" = "Vis 2", x3 = "Vis 3", x4 = "Txt
   1", x5 = "Txt 2", x6 = "Txt 3", x7 = "Speed 1", x8 = "Speed 2",
   x9 = "Speed 3")
fit1.t1 <- semTable(fit1, columns = c("estse", "p"), paramSets =
   c("loadings", "intercepts",  "residualvariances", "
   latentcovariances"), fits = c("chisq", "rmsea"), file =
   file.path(tempdir, "fit1.t1"), varLabels = vlabs, type = "
   latex",  table.float = TRUE, caption = "Holzinger Swineford CFA
   in a longtable Float", label = "tab:HS10", longtable=TRUE)
```

Table 1: Holzinger Swineford CFA in a longtable Float

		Model	
		Estimate(Std.Err.)	p
		Factor Loadings	
visual			
	Vis 1	0.90(0.08)	0.000
	Vis 2	0.50(0.08)	0.000
	Vis 3	0.66(0.07)	0.000
textual			
	Txt 1	0.99(0.06)	0.000
	Txt 2	1.10(0.06)	0.000
	Txt 3	0.92(0.05)	0.000
speed			

Speed 1	0.62(0.07)	0.000
Speed 2	0.73(0.07)	0.000
Speed 3	0.67(0.07)	0.000
	Intercepts	
Vis 1	4.94(0.07)	0.000
Vis 2	6.09(0.07)	0.000
Vis 3	2.25(0.07)	0.000
Txt 1	3.06(0.07)	0.000
Txt 2	4.34(0.07)	0.000
Txt 3	2.19(0.06)	0.000
Speed 1	4.19(0.06)	0.000
Speed 2	5.53(0.06)	0.000
Speed 3	5.37(0.06)	0.000
	Residual Variances	
Vis 1	0.55(0.11)	0.000
Vis 2	1.13(0.10)	0.000
Vis 3	0.84(0.09)	0.000
Txt 1	0.37(0.05)	0.000
Txt 2	0.45(0.06)	0.000
Txt 3	0.36(0.04)	0.000
Speed 1	0.80(0.08)	0.000
Speed 2	0.49(0.07)	0.000
Speed 3	0.57(0.07)	0.000
	Latent Covariances	
visual w/textual	0.46(0.06)	0.000
visual w/speed	0.47(0.07)	0.000
textual w/speed	0.28(0.07)	0.000
	Fit Indices	
χ^2(df)	85.31(24)	0.000
RMSEA	0.09	

$^{+}$Fixed parameter

Because we have `longtable` set as TRUE, a floating table is created with the indicated caption in Table 1.

In the previous example, we allowed the LaTeX markup to be printed directly into the document. There might be times when we would rather delay the output. We might instead have set "`print.results = FALSE`" to prevent code display, and then at a time of our choosing we could export the fitted model into the document with the cat function.

```
cat(fit1.t1)
```

Using the cat function, we also can export the table into a file:

```
fn <- file.path(tempdir, "fit1.t12.tex")
cat(fit1.t1, file = fn)
```

This is the same result obtained by specifying file name in the `semTable` function itself. However, this latter approach leaves the door open for the user to inspect & edit the marked-up table.

Alternative Output Formats

This function was developed primarily for LaTeX output tables. However, we have redesigned so that the output may be requested in either Web code (HTML) or comma separated variable (CSV) files. If HTML output is desired, replace `type = "latex"` with `type = "html"`.

```
fit1.t1h <- semTable(fit1, columns = c("estse", "p"), paramSets =
   c("loadings", "intercepts",  "residualvariances", "
   latentcovariances"),
                     fits = c("chisq", "rmsea"), file =
                        file.path(tempdir, "fit1.t1h.html"),
                     varLabels = vlabs, type = "html",
                        print.results = FALSE)
```

In an interactive session, the output file can be inspected on the screen.

```
browseURL(file.path(tempdir, "fit1.t1.html"))
```

```
## Try CSV output next
fit1.t1c <- semTable(fit1, columns = c("estse", "p"),
                     fits = c("chisq", "rmsea"), file =
                        file.path(tempdir, "fit1.t1c"),
                     varLabels = vlabs, type = "csv",
                        print.results = FALSE)
```

```
## Go inspect this file with a spread sheet program:
attr(fit1.t1c, "file")
```

```
[1] "tmpout/fit1.t1c.csv"
```

It is possible to change the output format of an `semTable` object. This is done with the function `markupConvert`. Here we demonstrate how to re-channel a LaTeX fitted table to a csv object:

```
fit1.t1c2 <- markupConvert(attr(fit1.t1, "markedResults"), type = "
   csv")
```

4 To Float or Not to Float?

In an academic paper, we would seldom/never have a table that prints directly into the middle of the text. Instead, all tables are presented a numbered “floating” table objects. The author has a choice to ask `semTable` to create the floating table object (with indicated caption and label), or to simply create the tabular object that would be included inside the floated table in a following setup. The quick, easy method of specifying the floating table elements in the semTable command itself does not allow all of the flexibility that we might want in controlling table output, so we’ll illustrate both methods.

First, we have an ordinary table (not a `longtable`) that is created as a float by the the reference label “tab:hs1939”, as seen in Table 2.

Table 2: Table Floated (not a longtable)

	Model		
	Estimate	Std. Err.	p
	Factor Loadings		
visual			
Vis 1	0.90	0.08	0.000
Vis 2	0.50	0.08	0.000
Vis 3	0.66	0.07	0.000
textual			
Txt 1	0.99	0.06	0.000
Txt 2	1.10	0.06	0.000
Txt 3	0.92	0.05	0.000
speed			
Speed 1	0.62	0.07	0.000
Speed 2	0.73	0.07	0.000
Speed 3	0.67	0.07	0.000
	Fit Indices		
χ^2(df)	85.31(24)		0.000
RMSEA	0.09		

$^+$Fixed parameter

```
## floating table
fit1.t3 <- semTable(fit1, columns = c("est", "se", "p"), paramSets
   = c("loadings"), fits = c("chisq", "rmsea"), file =
   file.path(tempdir, "fit1.t3"),                        varLabels =
   vlabs, longtable=FALSE, table.float = TRUE, caption = "Table
   Floated (not a longtable)", label = "tab:fit1.t3")
```

In Table 3, we have a similar table produced with the `longtable` class. We have tested some alternative settings for the columns (just to keep this interesting for the reader). Of course, the benefit of a `longtable` is that a table that needs to "break" across pages will do so, while an ordinary `tabular` will run into the bottom margin. In this case, with a loadings-only display, the table stays on the page and the `longtable` is not strictly necessary. However, the `longtable` also does not appear to be harmful even for small tables. Hence, we use `longtable = TRUE` in most of our work.

```
## floating longtable
fit1.t4 <- semTable(fit1, columns = c("est", "estsestars"),
   paramSets = c("loadings"), fits = c("chisq", "rmsea"), file =
   file.path(tempdir, "fit1.t4"), varLabels = vlabs, longtable =
   TRUE, table.float=TRUE, caption = "Table Floated (longtable)",
   label = "tab:fit1.t4")
```

Table 3: Table Floated (longtable)

Model

	Estimate	Estimate(Std.Err.)
		Factor Loadings
visual		
Vis 1	0.90	0.90(0.08)***
Vis 2	0.50	0.50(0.08)***
Vis 3	0.66	0.66(0.07)***
textual		
Txt 1	0.99	0.99(0.06)***
Txt 2	1.10	1.10(0.06)***
Txt 3	0.92	0.92(0.05)***
speed		
Speed 1	0.62	0.62(0.07)***
Speed 2	0.73	0.73(0.07)***
Speed 3	0.67	0.67(0.07)***
		Fit Indices
χ^2(df)	85.31(24)***	
RMSEA	0.09	

$^+$Fixed parameter
* p<0.05, ** p<0.01, ***p<0.001

As an alternative, in the following code we do not request a float to be created. After creating the table in the file named "**tmpout/fit1.t5.tex**", we use LaTeX commands to manually create the float that appears in Table 4. After creating the floating table, inside it we simply use the LaTeX code

```
\input{tmpout/fit1.t5.tex}.
```

```
##columnLabels
fit1.t5 <- semTable(fit1, fits = c("chisq", "rmsea"), paramSets =
   c("loadings"),  columns = c("est", "se", "p"), columnLabels =
   c(se = "S.E."), file = file.path(tempdir, "fit1.t5"),
   print.results = FALSE)
```

5 Fine tuning titles

We've already emphasized the ability to customize variable labels. Now we focus on column names as well as parameter sets. The ability to adjust both the column names and parameter section names is emphasized in Table 5.

```
##columnLabels
fit1.t6 <- semTable(list("A Fancy Fitted Model" = fit1), fits =
   c("chisq", "rmsea"), paramSets = c("loadings"), paramSetLabels
   = c("loadings" = "Loading Estimates(ML robust)"), columns =
   c("estsestars"), columnLabels = c("estsestars" = "
   Estimates(Std.Errors)"), file = file.path(tempdir, "fit1.t6"),
   table.float=TRUE, caption="Demonstrate Flexibility with Column
   and Parameter Set Labels", label = "tab:fit1.t6")
```

Table 4: A Manually Created Floating Table

	Model		
	Estimate	S.E.	p
		Factor Loadings	
visual			
x1	0.90	0.08	0.000
x2	0.50	0.08	0.000
x3	0.66	0.07	0.000
textual			
x4	0.99	0.06	0.000
x5	1.10	0.06	0.000
x6	0.92	0.05	0.000
speed			
x7	0.62	0.07	0.000
x8	0.73	0.07	0.000
x9	0.67	0.07	0.000
		Fit Indices	
χ^2(df)	85.31(24)		0.000
RMSEA	0.09		

$^+$Fixed parameter

Table 5: Demonstrate Flexibility with Column and Parameter Set Labels

	A Fancy Fitted Model
	Estimates(Std.Errors)
	Loading Estimates(ML robust)
visual	
x1	0.90(0.08)***
x2	0.50(0.08)***
x3	0.66(0.07)***
textual	
x4	0.99(0.06)***
x5	1.10(0.06)***
x6	0.92(0.05)***
speed	
x7	0.62(0.07)***
x8	0.73(0.07)***
x9	0.67(0.07)***
	Fit Indices
χ^2(df)	85.31(24)***
RMSEA	0.09

$^+$Fixed parameter

* $p<0.05$, ** $p<0.01$, $^{***}p<0.001$

The names of the latent variables will default to the names used in the lavaan model file. Those names, however, can be replaced in the variable label vector. See Table 6.

```
## Test alternative latent variable labels
vl <- c(vlabs, visual = "Seeing", textual = "Thumb Texting", speed
   = "Speed")
fit1.t7 <- semTable(fit1, fits = c("chisq", "rmsea"), paramSets =
   c("loadings", "intercepts"),  columns = c("eststars", "p"),
   columnLabels = c("eststars" = "Est(SE)"), file =
   file.path(tempdir, "fit1.t7"), varLabels = vl, longtable =
   TRUE, type = "latex", table.float=TRUE, caption="Variable
   Labels can include parameter sections", label = "tab:fit1.t7")
```

Table 6: Variable Labels can include parameter sections

		Model	
		Est(SE)	p
		Factor Loadings	
Seeing			
	Vis 1	0.90***	0.000
	Vis 2	0.50***	0.000
	Vis 3	0.66***	0.000
Thumb Texting			
	Txt 1	0.99***	0.000
	Txt 2	1.10***	0.000
	Txt 3	0.92***	0.000
Speed			
	Speed 1	0.62***	0.000
	Speed 2	0.73***	0.000
	Speed 3	0.67***	0.000
		Intercepts	
	Vis 1	4.94***	0.000
	Vis 2	6.09***	0.000
	Vis 3	2.25***	0.000
	Txt 1	3.06***	0.000
	Txt 2	4.34***	0.000
	Txt 3	2.19***	0.000
	Speed 1	4.19***	0.000
	Speed 2	5.53***	0.000
	Speed 3	5.37***	0.000
		Fit Indices	
	χ^2(df)	85.31(24)***	0.000
	RMSEA	0.09	

$^+$Fixed parameter
* $p<0.05$, ** $p<0.01$, $^{***}p<0.001$

The ability to fine-tune the selection and labels for fit values is demonstrated in Table 7.

Table 7: Customized Fits and Labels

		Model			
		Estimate	Std. Err.	z	p
			Factor Loadings		
visual					
	x1	0.90	0.08	11.13	0.000
	x2	0.50	0.08	6.43	0.000
	x3	0.66	0.07	8.82	0.000
textual					
	x4	0.99	0.06	17.47	0.000
	x5	1.10	0.06	17.58	0.000
	x6	0.92	0.05	17.08	0.000
speed					
	x7	0.62	0.07	8.90	0.000
	x8	0.73	0.07	11.09	0.000
	x9	0.67	0.07	10.30	0.000
			Fit Indices		
Root M.SQ.E.A		0.09			
CompFitIdx		0.93			
χ^2(df)		85.31(24)			0.000

$^{+}$Fixed parameter

```
fit4.t2 <- semTable(fit1, paramSets = c("loadings"), fits =
   c("rmsea", "cfi", "chisq"), fitLabels = c(rmsea = "Root
   M.SQ.E.A", cfi = "CompFitIdx", chisq = "chisq"), type = "
   latex", table.float = TRUE, caption = "Customized Fits and
   Labels", label = "tab:fit1.t8")
```

6 Two/Multi Group Models

A model that estimates parameters for a two group model, using school as the grouping model, is obtained with lavaan as follows:

```
if(file.exists("fit1.g.rds")){
   fit1.g <- readRDS("fit1.g.rds")
 } else {
     fit1.g <- cfa(HS.model, data = HolzingerSwineford1939, std.lv
        = TRUE, group = "school", estimator = "MLR")
     saveRDS(fit1.g, "fit1.g.rds")
 }
```

A table that displays both groups can be obtained, as illustrated in Table 8. This table runs into the margins unless we specify `longtable = TRUE`. Also it is worth noting that the table does not fit within the allowed horizontal space if we try to print the 4 standard columns individually. Hence, we use the more compact “Est(Std.Err.)” format.

```
## 2 groups table
fit1.gt1 <- semTable(fit1.g, columns = c("estsestars", "p"),
   columnLabels = c(estsestars = "Est(Std.Err.)", p = "p-value"),
   file = file.path(tempdir, "fit1.gt1"), table.float = TRUE,
   caption = "A Two Group Model", label = "tab:fit1.gt1",
   longtable=TRUE)
```

Table 8: A Two Group Model

		Pasteur		Grant-White	
		Est(Std.Err.)	p-value	Est(Std.Err.)	p-value
			Factor Loadings		
visual					
	x1	1.05(0.18)***	0.000	0.78(0.13)***	0.000
	x2	0.41(0.16)**	0.008	0.57(0.10)***	0.000
	x3	0.60(0.13)***	0.000	0.72(0.10)***	0.000
textual					
	x4	0.95(0.08)***	0.000	0.97(0.08)***	0.000
	x5	1.12(0.07)***	0.000	0.96(0.08)***	0.000
	x6	0.83(0.08)***	0.000	0.93(0.08)***	0.000
speed					
	x7	0.59(0.12)***	0.000	0.68(0.09)***	0.000
	x8	0.67(0.10)***	0.000	0.83(0.11)***	0.000
	x9	0.55(0.11)***	0.000	0.72(0.13)***	0.000
			Intercepts		
	x1	4.94(0.09)***	0.000	4.93(0.10)***	0.000
	x2	5.98(0.10)***	0.000	6.20(0.09)***	0.000
	x3	2.49(0.09)***	0.000	2.00(0.09)***	0.000
	x4	2.82(0.09)***	0.000	3.32(0.09)***	0.000
	x5	4.00(0.10)***	0.000	4.71(0.10)***	0.000
	x6	1.92(0.08)***	0.000	2.47(0.09)***	0.000
	x7	4.43(0.09)***	0.000	3.92(0.09)***	0.000
	x8	5.56(0.08)***	0.000	5.49(0.09)***	0.000
	x9	5.42(0.08)***	0.000	5.33(0.09)***	0.000
			Residual Variances		
	x1	0.30(0.34)	0.378	0.71(0.18)***	0.000
	x2	1.33(0.18)***	0.000	0.90(0.14)***	0.000
	x3	0.99(0.15)***	0.000	0.56(0.12)***	0.000
	x4	0.43(0.07)***	0.000	0.32(0.07)***	0.000
	x5	0.46(0.09)***	0.000	0.42(0.07)***	0.000
	x6	0.29(0.06)***	0.000	0.41(0.08)***	0.000
	x7	0.82(0.13)***	0.000	0.60(0.10)***	0.000
	x8	0.51(0.10)***	0.000	0.40(0.16)*	0.012
	x9	0.68(0.13)***	0.000	0.53(0.14)***	0.000
			Latent Intercepts		
	visual	0.00^+		0.00^+	
	textual	0.00^+		0.00^+	

speed	0.00^{+}		0.00^{+}	
		Latent Variances		
visual	1.00^{+}		1.00^{+}	
textual	1.00^{+}		1.00^{+}	
speed	1.00^{+}		1.00^{+}	
		Latent Covariances		
visual w/textual	$0.48(0.09)^{***}$	0.000	$0.54(0.10)^{***}$	0.000
visual w/speed	$0.30(0.14)^{*}$	0.031	$0.52(0.15)^{***}$	0.000
textual w/speed	$0.33(0.10)^{**}$	0.001	$0.34(0.14)^{*}$	0.019
		Fit Indices		
Scaled χ^2(df)	$121.74(48)^{***}$	0.000		
CFI	0.92			
TLI	0.89			
RMSEA	0.10			

$^{+}$Fixed parameter
* $p<0.05$, ** $p<0.01$, $^{***}p<0.001$

It is not necessary to display all of the groups in the table. It is possible to select groups by name, as we see in Table 9.

```
## Now name particular group by name
fit1.gt2 <- semTable(fit1.g, columns = c("estsestars", "p"),
   paramSets = c("loadings", "intercepts", "residualvariances"),
   columnLabels = c(estsestars = "Est w/stars", p = "p-value"),
   file = file.path(tempdir, "fit1.gt2"), groups = "Pasteur",
   table.float = TRUE, caption = "Group 'Pasteur' Group from the 2
   Model", label = "tab:fit1.gt2")
```

It is also possible to select groups by integer numbers in the group list, rather than by name. The results for group 1 are offered in Table 10.

```
## Name particular group by number
fit1.gt3 <- semTable(fit1.g, columns = c("estsestars", "p"),
   paramSets = c("loadings"), columnLabels = c(estsestars = "Est
   w/stars", p = "p-value"), file = file.path(tempdir, "
   fit1.gt3"), groups = 2, table.float = TRUE, caption = "Group
   '2' from the 2 Model is 'Grant-White'", label = "tab:fit1.gt3")
```

7 Two Models Side-by-Side

One might wonder if the "standardized" SEM estimates are substantively different from the original estimates. With lavaan, we can refit the original CFA model and specify that we want standardized latent and observed variables, along with estimates of a mean structure.

```
if(file.exists("fit1.std.rds")){
   fit1.std <- readRDS("fit1.std.rds")
 } else {
```

Table 9: Group 'Pasteur' Group from the 2 Model

		Pasteur	
		Est w/stars	p-value
		Factor Loadings	
visual			
	x1	1.05(0.18)***	0.000
	x2	0.41(0.16)**	0.008
	x3	0.60(0.13)***	0.000
textual			
	x4	0.95(0.08)***	0.000
	x5	1.12(0.07)***	0.000
	x6	0.83(0.08)***	0.000
speed			
	x7	0.59(0.12)***	0.000
	x8	0.67(0.10)***	0.000
	x9	0.55(0.11)***	0.000
		Intercepts	
	x1	4.94(0.09)***	0.000
	x2	5.98(0.10)***	0.000
	x3	2.49(0.09)***	0.000
	x4	2.82(0.09)***	0.000
	x5	4.00(0.10)***	0.000
	x6	1.92(0.08)***	0.000
	x7	4.43(0.09)***	0.000
	x8	5.56(0.08)***	0.000
	x9	5.42(0.08)***	0.000
		Residual Variances	
	x1	0.30(0.34)	0.378
	x2	1.33(0.18)***	0.000
	x3	0.99(0.15)***	0.000
	x4	0.43(0.07)***	0.000
	x5	0.46(0.09)***	0.000
	x6	0.29(0.06)***	0.000
	x7	0.82(0.13)***	0.000
	x8	0.51(0.10)***	0.000
	x9	0.68(0.13)***	0.000
		Fit Indices	
Scaled χ^2(df)		121.74(48)***	0.000
CFI		0.92	
TLI		0.89	
RMSEA		0.10	

$^{+}$Fixed parameter

* p<0.05, ** p<0.01, ***p<0.001

Table 10: Group '2' from the 2 Model is 'Grant-White'

		Grant-White	
		Est w/stars	p-value
		Factor Loadings	
visual			
	x1	0.78(0.13)***	0.000
	x2	0.57(0.10)***	0.000
	x3	0.72(0.10)***	0.000
textual			
	x4	0.97(0.08)***	0.000
	x5	0.96(0.08)***	0.000
	x6	0.93(0.08)***	0.000
speed			
	x7	0.68(0.09)***	0.000
	x8	0.83(0.11)***	0.000
	x9	0.72(0.13)***	0.000
		Fit Indices	
Scaled χ^2(df)		121.74(48)***	0.000
CFI		0.92	
TLI		0.89	
RMSEA		0.10	

$^+$Fixed parameter
* p<0.05, ** p<0.01, ***p<0.001

```
    fit1.std <- update(fit1, std.lv = TRUE, std.ov = TRUE,
       meanstructure = TRUE)
    saveRDS(fit1.std, "fit1.std.rds")
}
```

The two models are presented in Table 11. We combine the estimates and standard errors into one column in order to improve the chances that the table stays within the margins. Nevertheless, the right side is well into the margin.

```
# include 2 models in table request
fit1.t2 <- semTable(list("Ordinary" = fit1, "Standardized" =
   fit1.std), columns=c("estse", "z", "p"), file =
   file.path(tempdir, "fit1.2.1"), table.float = TRUE, longtable =
   TRUE, caption = "Ordinary and Standardized CFA Estimates",
   label = "tab:fit1.t2")
```

Table 11: Ordinary and Standardized CFA Estimates

		Ordinary			Standardized		
		Estimate(Std.Err.)	z	p	Estimate(Std.Err.)	z	p
				Factor Loadings			
visual							
	x1	0.90(0.08)	11.13	0.000	0.77(0.07)	11.13	0.000

x2	0.50(0.08)	6.43	0.000	0.42(0.07)	6.43	0.000
x3	0.66(0.07)	8.82	0.000	0.58(0.07)	8.82	0.000
textual						
x4	0.99(0.06)	17.47	0.000	0.85(0.05)	17.47	0.000
x5	1.10(0.06)	17.58	0.000	0.85(0.05)	17.58	0.000
x6	0.92(0.05)	17.08	0.000	0.84(0.05)	17.08	0.000
speed						
x7	0.62(0.07)	8.90	0.000	0.57(0.06)	8.90	0.000
x8	0.73(0.07)	11.09	0.000	0.72(0.07)	11.09	0.000
x9	0.67(0.07)	10.30	0.000	0.66(0.06)	10.30	0.000
			Intercepts			
x1	4.94(0.07)	73.47	0.000	0.00(0.06)	0.00	1.000
x2	6.09(0.07)	89.85	0.000	0.00(0.06)	0.00	1.000
x3	2.25(0.07)	34.58	0.000	0.00(0.06)	0.00	1.000
x4	3.06(0.07)	45.69	0.000	0.00(0.06)	0.00	1.000
x5	4.34(0.07)	58.45	0.000	0.00(0.06)	0.00	1.000
x6	2.19(0.06)	34.67	0.000	0.00(0.06)	0.00	1.000
x7	4.19(0.06)	66.77	0.000	0.00(0.06)	0.00	1.000
x8	5.53(0.06)	94.85	0.000	0.00(0.06)	0.00	1.000
x9	5.37(0.06)	92.55	0.000	0.00(0.06)	0.00	1.000
			Residual Variances			
x1	0.55(0.11)	4.83	0.000	0.40(0.08)	4.83	0.000
x2	1.13(0.10)	11.15	0.000	0.82(0.07)	11.15	0.000
x3	0.84(0.09)	9.32	0.000	0.66(0.07)	9.32	0.000
x4	0.37(0.05)	7.78	0.000	0.27(0.04)	7.78	0.000
x5	0.45(0.06)	7.64	0.000	0.27(0.04)	7.64	0.000
x6	0.36(0.04)	8.28	0.000	0.30(0.04)	8.28	0.000
x7	0.80(0.08)	9.82	0.000	0.67(0.07)	9.82	0.000
x8	0.49(0.07)	6.57	0.000	0.48(0.07)	6.57	0.000
x9	0.57(0.07)	8.00	0.000	0.56(0.07)	8.00	0.000
			Latent Intercepts			
visual	0.00^+			0.00^+		
textual	0.00^+			0.00^+		
speed	0.00^+			0.00^+		
			Latent Variances			
visual	1.00^+			1.00^+		
textual	1.00^+			1.00^+		
speed	1.00^+			1.00^+		
			Latent Covariances			
visual w/textual	0.46(0.06)	7.19	0.000	0.46(0.06)	7.19	0.000
visual w/speed	0.47(0.07)	6.46	0.000	0.47(0.07)	6.46	0.000
textual w/speed	0.28(0.07)	4.12	0.000	0.28(0.07)	4.12	0.000
			Fit Indices			
χ^2(df)	85.31(24)		0.000	85.31(24)		0.000
CFI	0.93			0.93		
TLI	0.90			0.90		

RMSEA	0.09	0.09

+Fixed parameter

In the present case, it is perhaps not needed to display p values on the standardized model estimates, so we might economize on horizontal space by keeping just the estimates from the standardized model. To demonstrate the fact that it is possible to select different columns for the two models, we offer Table 12.

```
fit1.t2.2 <- semTable(list("Ordinary" = fit1, "Standardized" =
   fit1.std), columns = list("Ordinary" = c("estse", "z", "p"), "
   Standardized" = c("estse")), columnLabels = c(estse = "
   Est(S.E.)", z = "Z", se = "SE"), file = file.path(tempdir, "
   fit1.t2.2"), table.float = TRUE, longtable = TRUE, caption = "
   Customizing Column Selections by Model", label = "
   tab:fit1.t2.2" )
```

Table 12: Customizing Column Selections by Model

		Ordinary			Standardized
		Est(S.E.)	Z	p	Est(S.E.)
			Factor Loadings		
visual					
	x1	0.90(0.08)	11.13	0.000	0.77(0.07)
	x2	0.50(0.08)	6.43	0.000	0.42(0.07)
	x3	0.66(0.07)	8.82	0.000	0.58(0.07)
textual					
	x4	0.99(0.06)	17.47	0.000	0.85(0.05)
	x5	1.10(0.06)	17.58	0.000	0.85(0.05)
	x6	0.92(0.05)	17.08	0.000	0.84(0.05)
speed					
	x7	0.62(0.07)	8.90	0.000	0.57(0.06)
	x8	0.73(0.07)	11.09	0.000	0.72(0.07)
	x9	0.67(0.07)	10.30	0.000	0.66(0.06)
			Intercepts		
	x1	4.94(0.07)	73.47	0.000	0.00(0.06)
	x2	6.09(0.07)	89.85	0.000	0.00(0.06)
	x3	2.25(0.07)	34.58	0.000	0.00(0.06)
	x4	3.06(0.07)	45.69	0.000	0.00(0.06)
	x5	4.34(0.07)	58.45	0.000	0.00(0.06)
	x6	2.19(0.06)	34.67	0.000	0.00(0.06)
	x7	4.19(0.06)	66.77	0.000	0.00(0.06)
	x8	5.53(0.06)	94.85	0.000	0.00(0.06)
	x9	5.37(0.06)	92.55	0.000	0.00(0.06)
			Residual Variances		
	x1	0.55(0.11)	4.83	0.000	0.40(0.08)
	x2	1.13(0.10)	11.15	0.000	0.82(0.07)
	x3	0.84(0.09)	9.32	0.000	0.66(0.07)

x4	0.37(0.05)	7.78	0.000	0.27(0.04)
x5	0.45(0.06)	7.64	0.000	0.27(0.04)
x6	0.36(0.04)	8.28	0.000	0.30(0.04)
x7	0.80(0.08)	9.82	0.000	0.67(0.07)
x8	0.49(0.07)	6.57	0.000	0.48(0.07)
x9	0.57(0.07)	8.00	0.000	0.56(0.07)
		Latent Intercepts		
visual	0.00^{+}			0.00^{+}
textual	0.00^{+}			0.00^{+}
speed	0.00^{+}			0.00^{+}
		Latent Variances		
visual	1.00^{+}			1.00^{+}
textual	1.00^{+}			1.00^{+}
speed	1.00^{+}			1.00^{+}
		Latent Covariances		
visual w/textual	0.46(0.06)	7.19	0.000	0.46(0.06)
visual w/speed	0.47(0.07)	6.46	0.000	0.47(0.07)
textual w/speed	0.28(0.07)	4.12	0.000	0.28(0.07)
		Fit Indices		
χ^2(df)	85.31(24)		0.000	85.31(24)
CFI	0.93			0.93
TLI	0.90			0.90
RMSEA	0.09			0.09

$^{+}$Fixed parameter

8 Larger Models

The structural equation model (SEM) introduces a regression relationship between the latent variables. In `lavaan`, the regression relationships are introduced by the same notation as regression in linear models.

```
regmodel1 <- 'visual  =~ x1 + x2 + x3
              textual =~ x4 + x5 + x6
              speed   =~ x7 + x8 + x9
              visual ~ textual + speed
'
```

```
if(file.exists("fit2.rds")){
   fit2 <- readRDS("fit2.rds")
} else {
     fit2 <- sem(regmodel1, data = HolzingerSwineford1939, std.lv
        = FALSE, meanstructure = TRUE)
     saveRDS(fit2, "fit2.rds")
}
```

```
if(file.exists("fit2.std.rds")){
   fit2.std <- readRDS("fit2.std.rds")
```

```
} else {
    fit2.std <- update(fit2, std.lv = TRUE, std.ov = TRUE,
       meanstructure = TRUE)
    saveRDS(fit2.std, "fit2.std.rds")
}
```

A table comparing the standardized with the non-standardized models is offered in Table 13.

```
fit2.t <- semTable(list("Ordinary" = fit2, "Standardized" =
   fit2.std), fits = "rmsea", columns = list("Ordinary" = c("est",
   "se", "p"), "Standardized" = c("estsestars")), columnLabels =
   c("est" = "Est", "se" = "Std.Err.", "p" = "p", "estsestars" = "
   Standardized Est."), paramSets = c("loadings", "intercepts", "
   slopes", "latentcovariances"), file = file.path(tempdir, "
   fit2.t1"), type = "latex", table.float = TRUE, longtable =
   TRUE, caption = "SEM, Standardized or Not", label = "
   tab:fit2.t")
```

Table 13: SEM, Standardized or Not

		Ordinary			Standardized
		Est	Std.Err.	p	Standardized Est.
			Factor Loadings		
visual					
	x1	1.00^{+}			1.00^{+}
	x2	0.55	0.10	0.000	$0.25(0.05)^{***}$
	x3	0.73	0.11	0.000	$0.39(0.05)^{***}$
textual					
	x4	1.00^{+}			1.00^{+}
	x5	1.11	0.07	0.000	$0.94(0.04)^{***}$
	x6	0.93	0.06	0.000	$0.92(0.04)^{***}$
speed					
	x7	1.00^{+}			1.00^{+}
	x8	1.18	0.16	0.000	$0.83(0.06)^{***}$
	x9	1.08	0.15	0.000	$0.68(0.06)^{***}$
			Regression Slopes		
visual					
	textual	0.32	0.07	0.000	$0.38(0.06)^{***}$
	speed	0.54	0.13	0.000	$0.22(0.07)^{**}$
			Intercepts		
	x1	4.94	0.07	0.000	0.00(0.06)
	x2	6.09	0.07	0.000	0.00(0.06)
	x3	2.25	0.07	0.000	0.00(0.06)
	x4	3.06	0.07	0.000	0.00(0.06)
	x5	4.34	0.07	0.000	0.00(0.06)
	x6	2.19	0.06	0.000	0.00(0.06)
	x7	4.19	0.06	0.000	0.00(0.07)
	x8	5.53	0.06	0.000	0.00(0.06)

x9	5.37	0.06	0.000	0.00(0.06)
		Latent Covariances		
textual w/speed	0.17	0.05	0.000	0.35(0.06)***
		Fit Indices		
RMSEA	0.09			0.12

+Fixed parameter

* p<0.05, ** p<0.01, ***p<0.001

```
# Change output format to csv
cat(markupConvert(attr(fit2.t, "markedResults"), type = "csv"),
   file = file.path(tempdir, "fit2.t.converted.csv"))
cat(markupConvert(attr(fit2.t, "markedResults"), type = "html"),
   file = file.path(tempdir, "fit2.t.converted.html"))
```

It is not necessary to have all of the same relationships estimated in every model. To demonstrate that, we estimate a second structural equation with some indicators removed and also we have a different regression relationship.

```
regmodel2 <- 'visual  =~ x1 + x2 + x3
               textual =~ x4 +  x6
               speed   =~  x8 + x9
               visual ~ speed
               textual ~ speed
 '
```

```
if(file.exists("fit3.rds")){
   fit3 <- readRDS("fit3.rds")
 } else {
     fit3 <- sem(regmodel2, data = HolzingerSwineford1939, std.lv
        = TRUE,  meanstructure = TRUE)
     saveRDS(fit3, "fit3.rds")
 }
if(file.exists("fit3.std.rds")){
   fit3.std <- readRDS("fit3.std.rds")
 } else {
     fit3.std <- update(fit3, std.lv = TRUE, std.ov = TRUE)
     saveRDS(fit3.std, "fit3.std.rds")
 }
```

See Table 14 for output from the following, which combines four model columns. Note that the result table handles the problem that some estimates are not included in each model.

```
fit3.std.t1 <- semTable(list("Mod 1" = fit2, "Mod 1 std" =
   fit2.std, "Mod 2" = fit3, "Mod 3 std" = fit3.std), paramSets =
   c("loadings", "slopes", "intercepts", "residualvariances", "
   latentvariances", "latentcovariances"),  columns =
   c("estsestars"), type = "latex", file = file.path(tempdir, "
   fit3.std.t1"), table.float = TRUE, longtable = TRUE, caption = "
   Several SEMs with Differing Parameters", label = "
   tab:fit3.std.t1")
```

Table 14: Several SEMs with Differing Parameters

	Mod 1	Mod 1 std	Mod 2	Mod 3 std
	Estimate(Std.Err.)	Estimate(Std.Err.)	Estimate(Std.Err.)	Estimate(Std.Err.)
			Factor Loadings	
visual				
x1	1.00^{+}	1.00^{+}	$0.72(0.08)^{***}$	$0.61(0.07)^{***}$
x2	$0.55(0.10)^{***}$	$0.25(0.05)^{***}$	$0.41(0.07)^{***}$	$0.35(0.06)^{***}$
x3	$0.73(0.11)^{***}$	$0.39(0.05)^{***}$	$0.54(0.07)^{***}$	$0.48(0.06)^{***}$
textual				
x4	1.00^{+}	1.00^{+}	$0.92(0.08)^{***}$	$0.79(0.06)^{***}$
x5	$1.11(0.07)^{***}$	$0.94(0.04)^{***}$		
x6	$0.93(0.06)^{***}$	$0.92(0.04)^{***}$	$0.90(0.07)^{***}$	$0.82(0.07)^{***}$
speed				
x7	1.00^{+}	1.00^{+}		
x8	$1.18(0.16)^{***}$	$0.83(0.06)^{***}$	$0.51(0.07)^{***}$	$0.50(0.07)^{***}$
x9	$1.08(0.15)^{***}$	$0.68(0.06)^{***}$	$0.90(0.10)^{***}$	$0.89(0.10)^{***}$
			Regression Slopes	
visual				
textual	$0.32(0.07)^{***}$	$0.38(0.06)^{***}$		
speed	$0.54(0.13)^{***}$	$0.22(0.07)^{**}$	$0.72(0.14)^{***}$	$0.72(0.14)^{***}$
textual				
speed			$0.30(0.08)^{***}$	$0.30(0.08)^{***}$
			Intercepts	
x1	$4.94(0.07)^{***}$	0.00(0.06)	$4.94(0.07)^{***}$	0.00(0.06)
x2	$6.09(0.07)^{***}$	0.00(0.06)	$6.09(0.07)^{***}$	0.00(0.06)
x3	$2.25(0.07)^{***}$	0.00(0.06)	$2.25(0.07)^{***}$	0.00(0.06)
x4	$3.06(0.07)^{***}$	0.00(0.06)	$3.06(0.07)^{***}$	0.00(0.06)
x5	$4.34(0.07)^{***}$	0.00(0.06)		
x6	$2.19(0.06)^{***}$	0.00(0.06)	$2.19(0.06)^{***}$	0.00(0.06)
x7	$4.19(0.06)^{***}$	0.00(0.07)		
x8	$5.53(0.06)^{***}$	0.00(0.06)	$5.53(0.06)^{***}$	0.00(0.06)
x9	$5.37(0.06)^{***}$	0.00(0.06)	$5.37(0.06)^{***}$	0.00(0.06)
			Residual Variances	
x1	$0.55(0.11)^{***}$	$-0.12(0.07)$	$0.58(0.10)^{***}$	$0.42(0.07)^{***}$
x2	$1.13(0.10)^{***}$	$0.93(0.07)^{***}$	$1.12(0.10)^{***}$	$0.81(0.07)^{***}$
x3	$0.84(0.09)^{***}$	$0.83(0.07)^{***}$	$0.83(0.09)^{***}$	$0.65(0.07)^{***}$
x4	$0.37(0.05)^{***}$	$0.25(0.03)^{***}$	$0.43(0.11)^{***}$	$0.32(0.08)^{***}$
x5	$0.45(0.06)^{***}$	$0.27(0.03)^{***}$		
x6	$0.36(0.04)^{***}$	$0.31(0.04)^{***}$	$0.32(0.10)^{**}$	$0.27(0.09)^{**}$
x7	$0.80(0.08)^{***}$	$0.53(0.08)^{***}$		
x8	$0.49(0.07)^{***}$	$0.49(0.06)^{***}$	$0.76(0.08)^{***}$	$0.74(0.08)^{***}$
x9	$0.57(0.07)^{***}$	$0.66(0.07)^{***}$	0.21(0.16)	0.20(0.15)
			Latent Variances	
visual	$0.54(0.12)^{***}$	1.00^{+}	1.00^{+}	1.00^{+}
textual	$0.98(0.11)^{***}$	1.00^{+}	1.00^{+}	1.00^{+}

speed	0.38(0.09)***	1.00^{+}	1.00^{+}	1.00^{+}
		Latent Covariances		
textual w/speed	0.17(0.05)***	0.35(0.06)***		
visual w/textual			0.44(0.08)***	0.44(0.08)***
		Fit Indices		
χ^2(df)	85.31(24)***	145.49(27)***	14.46(11)	14.46(11)
CFI	0.93	0.87	0.99	0.99
TLI	0.90	0.82	0.99	0.99
RMSEA	0.09	0.12	0.03	0.03

$^{+}$Fixed parameter

* p<0.05, ** p<0.01, ***p<0.001

SEM estimates can also be obtained for the two group model, as illustrated in Table 15:

```
if(file.exists("fit3.g2.rds")){
   fit3.g2 <- readRDS("fit3.g2.rds")
 } else {
     fit3.g2 <- sem(regmodel1, data = HolzingerSwineford1939,
        group = "school")
     saveRDS(fit3.g2, "fit3.g2.rds")
 }
#
```

```
fit3.g2.2 <- semTable(fit3.g2, paramSets = c("loadings", "slopes",
   "intercepts"), columns = c("estsestars"), fits = c("chisq", "
   rmsea", "cfi"), type = "latex", file = file.path(tempdir, "
   fit3.g2"), table.float=TRUE, longtable=TRUE, caption = "SEM
   with Two Groups", label = "tab:fit3.g2")
```

Table 15: SEM with Two Groups

	Pasteur	Grant-White
	Estimate(Std.Err.)	Estimate(Std.Err.)
	Factor Loadings	
visual		
x1	1.00^{+}	1.00^{+}
x2	0.39(0.12)**	0.74(0.15)***
x3	0.57(0.14)***	0.92(0.17)***
textual		
x4	1.00^{+}	1.00^{+}
x5	1.18(0.10)***	0.99(0.09)***
x6	0.87(0.08)***	0.96(0.08)***
speed		
x7	1.00^{+}	1.00^{+}
x8	1.12(0.28)***	1.23(0.19)***
x9	0.92(0.22)***	1.06(0.16)***
	Regression Slopes	

visual		
textual	0.48(0.11)***	0.33(0.09)***
speed	0.28(0.20)	0.44(0.14)**
	Intercepts	
x1	4.94(0.09)***	4.93(0.10)***
x2	5.98(0.10)***	6.20(0.09)***
x3	2.49(0.09)***	2.00(0.09)***
x4	2.82(0.09)***	3.32(0.09)***
x5	4.00(0.10)***	4.71(0.10)***
x6	1.92(0.08)***	2.47(0.09)***
x7	4.43(0.09)***	3.92(0.09)***
x8	5.56(0.08)***	5.49(0.09)***
x9	5.42(0.08)***	5.33(0.09)***
	Fit Indices	
χ^2(df)	115.85(48)***	
RMSEA	0.10	
CFI	0.92	

+Fixed parameter
* p<0.05, ** p<0.01, ***p<0.001

Mediation Model

There seems to be an unwritten rule in SEM that all tutorials must use, for at least one model, Bollen's famous data set about industrialization and democracy. We'll use that data for a model with mediation. We'll make this interesting by calculating one set of ML estimates for the standard errors and one bootstrapped set.

```
## Model 5 - Mediation model with equality constraints
model5 <-
      '
      # latent variable definitions
      ind60 =~ x1 + x2 + x3
      dem60 =~ y1 + e*y2 + d*y3 + y4
      dem65 =~ y5 + e*y6 + d*y7 + y8
      # regressions
      dem60 ~ a*ind60
      dem65 ~ c*ind60 + b*dem60
      # residual correlations
      y1 ~~ y5
      y2 ~~ y4 + y6
      y3 ~~ y7
      y4 ~~ y8
      y6 ~~ y8

      # indirect effect (a*b)
      ## := operator defines new parameters
      ab := a*b
```

```
    ## total effect
    total := c + (a*b)
    '
```

Because this model can take a while to estimate, we use the saved copies if they are available.

```
if(file.exists("fit5.rds")){
   fit5 <- readRDS("fit5.rds")
 } else {
     fit5 <- sem(model5, data=PoliticalDemocracy)
     saveRDS(fit5, "fit5.rds")
 }
if(file.exists("fit5boot.rds")){
   fit5boot <- readRDS("fit5boot.rds")
 } else {
     fit5boot <- sem(model5, data=PoliticalDemocracy, se = "
        bootstrap", bootstrap = 500)
     saveRDS(fit5boot, "fit5boot.rds")
 }
```

The ML and bootstrapped estimates are compared in Table 16.

```
fit5.t2 <- semTable(list("ML estimates" = fit5, "Bootstrapped SE"
   = fit5boot), columns = c("estsestars", "rsquare"), file =
   file.path(tempdir, "fit5.2"), type = "latex", longtable = TRUE,
   table.float = TRUE, caption = "Comparing ML and Bootstrapped
   Estimates", label = "tab:fit5t2")
```

Table 16: Comparing ML and Bootstrapped Estimates

	ML estimates		Bootstrapped SE	
	Estimate(Std.Err.)	R Square	Estimate(Std.Err.)	R Square
		Factor Loadings		
ind60				
x1	1.00$^+$	0.85	1.00$^+$	0.85
x2	2.18(0.14)***	0.95	2.18(0.15)***	0.95
x3	1.82(0.15)***	0.76	1.82(0.15)***	0.76
dem60				
y1	1.00$^+$	0.72	1.00$^+$	0.72
y2	1.19(0.14)***	0.47	1.19(0.15)***	0.47
y3	1.17(0.12)***	0.57	1.17(0.12)***	0.57
y4	1.26(0.13)***	0.70	1.26(0.15)***	0.70
dem65				
y5	1.00$^+$	0.67	1.00$^+$	0.67
y6	1.19(0.14)***	0.57	1.19(0.15)***	0.57
y7	1.17(0.12)***	0.64	1.17(0.12)***	0.64
y8	1.25(0.14)***	0.69	1.25(0.16)***	0.69
		Regression Slopes		

dem60				
ind60	1.47(0.39)***		1.47(0.38)***	
dem65				
ind60	0.60(0.23)**		0.60(0.25)*	
dem60	0.87(0.08)***		0.87(0.08)***	
		Residual Variances		
x1	0.08(0.02)***		0.08(0.02)***	
x2	0.12(0.07)		0.12(0.08)	
x3	0.47(0.09)***		0.47(0.08)***	
y1	1.86(0.43)***		1.86(0.45)***	
y2	7.59(1.37)***		7.59(1.25)***	
y3	4.96(0.96)***		4.96(1.08)***	
y4	3.22(0.73)***		3.22(0.80)***	
y5	2.31(0.48)***		2.31(0.58)***	
y6	4.97(0.92)***		4.97(0.94)***	
y7	3.56(0.71)***		3.56(0.62)***	
y8	3.31(0.71)***		3.31(0.92)***	
		Residual Covariances		
y1 w/y5	0.58(0.36)		0.58(0.46)	
y2 w/y4	1.44(0.69)*		1.44(0.72)*	
y2 w/y6	2.18(0.74)**		2.18(0.83)**	
y3 w/y7	0.71(0.61)		0.71(0.59)	
y4 w/y8	0.36(0.44)		0.36(0.45)	
y6 w/y8	1.37(0.58)*		1.37(0.78)	
		Latent Variances		
ind60	0.45(0.09)***		0.45(0.07)***	
dem60	3.87(0.87)***	0.20	3.87(0.81)***	0.20
dem65	0.16(0.23)	0.96	0.16(0.26)	0.96
		Constructed		
ab	1.27(0.36)***		1.27(0.35)***	
total	1.88(0.37)***		1.88(0.38)***	
		Fit Indices		
χ^2(df)	40.18(37)		40.18(37)	
CFI	1.00		1.00	
TLI	0.99		0.99	
RMSEA	0.03		0.03	

$^+$Fixed parameter

* $p<0.05$, ** $p<0.01$, $^{***}p<0.001$

Now we remove some indicators from the model and re-fit. The results are presented in Table 17.

```
##'
## Model 5b - Revision of Model 5s
model5b <-
    '
    # Cut some indicators from the measurement model
    ind60 =~ x1 + x2
    dem60 =~ y1 + e*y2 + d*y3 + y4
```

```
    dem65 =~ y5 + e*y6 + d*y7
    # regressions
    dem60 ~ a*ind60
    dem65 ~ c*ind60 + b*dem60
    # cut out the residual correlations
    # indirect effect (a*b)
    ## := operator defines new parameters
    ab := a*b
    ## total effect
    total := c + (a*b)
    '
```

Again, we use the saved fits, or if they are unavailable, we re-calculate.

```
if(file.exists("fit5b.rds")){
   fit5b <- readRDS("fit5.rds")
 } else {
     fit5b <- sem(model5b, data=PoliticalDemocracy, se = "
        bootstrap", bootstrap = 500)
     saveRDS(fit5b, "fit5b.rds")
 }
```

```
fit5.5b <- semTable(list("Model 5" = fit5boot, "Model 5b" =
   fit5b), columns = c("estsestars", "rsquare"), file =
   file.path(tempdir, "fit5.5"), type = "latex", longtable = TRUE,
   table.float = TRUE, caption = "Models 5 and 5b", label = "
   tab:fit5.5b")
```

Table 17: Models 5 and 5b

	Model 5		Model 5b	
	Estimate(Std.Err.)	R Square	Estimate(Std.Err.)	R Square
		Factor Loadings		
ind60				
x1	1.00^{+}	0.85	1.00^{+}	0.85
x2	$2.18(0.15)^{***}$	0.95	$2.18(0.14)^{***}$	0.95
x3	$1.82(0.15)^{***}$	0.76	$1.82(0.15)^{***}$	0.76
dem60				
y1	1.00^{+}	0.72	1.00^{+}	0.72
y2	$1.19(0.15)^{***}$	0.47	$1.19(0.14)^{***}$	0.47
y3	$1.17(0.12)^{***}$	0.57	$1.17(0.12)^{***}$	0.57
y4	$1.26(0.15)^{***}$	0.70	$1.26(0.13)^{***}$	0.70
dem65				
y5	1.00^{+}	0.67	1.00^{+}	0.67
y6	$1.19(0.15)^{***}$	0.57	$1.19(0.14)^{***}$	0.57
y7	$1.17(0.12)^{***}$	0.64	$1.17(0.12)^{***}$	0.64
y8	$1.25(0.16)^{***}$	0.69	$1.25(0.14)^{***}$	0.69
		Regression Slopes		

dem60				
ind60	1.47(0.38)***		1.47(0.39)***	
dem65				
ind60	0.60(0.25)*		0.60(0.23)**	
dem60	0.87(0.08)***		0.87(0.08)***	
Residual Variances				
x1	0.08(0.02)***		0.08(0.02)***	
x2	0.12(0.08)		0.12(0.07)	
x3	0.47(0.08)***		0.47(0.09)***	
y1	1.86(0.45)***		1.86(0.43)***	
y2	7.59(1.25)***		7.59(1.37)***	
y3	4.96(1.08)***		4.96(0.96)***	
y4	3.22(0.80)***		3.22(0.73)***	
y5	2.31(0.58)***		2.31(0.48)***	
y6	4.97(0.94)***		4.97(0.92)***	
y7	3.56(0.62)***		3.56(0.71)***	
y8	3.31(0.92)***		3.31(0.71)***	
Residual Covariances				
y1 w/y5	0.58(0.46)		0.58(0.36)	
y2 w/y4	1.44(0.72)*		1.44(0.69)*	
y2 w/y6	2.18(0.83)**		2.18(0.74)**	
y3 w/y7	0.71(0.59)		0.71(0.61)	
y4 w/y8	0.36(0.45)		0.36(0.44)	
y6 w/y8	1.37(0.78)		1.37(0.58)*	
Latent Variances				
ind60	0.45(0.07)***		0.45(0.09)***	
dem60	3.87(0.81)***	0.20	3.87(0.87)***	0.20
dem65	0.16(0.26)	0.96	0.16(0.23)	0.96
Constructed				
ab	1.27(0.35)***		1.27(0.36)***	
total	1.88(0.38)***		1.88(0.37)***	
Fit Indices				
χ^2(df)	40.18(37)		40.18(37)	
CFI	1.00		1.00	
TLI	0.99		0.99	
RMSEA	0.03		0.03	

+Fixed parameter

* $p<0.05$, ** $p<0.01$, *** $p<0.001$

9 Conclusion

The `semTable` function is, to our knowledge, the first effort successful effort to create a flexible function to present various kinds of `lavaan` estimates of confirmatory factor analyses and structural equation models. This version includes the ability of authors to select parameter sets, output columns, fit indices, as well as to customize many of the labels.

This document is not intended as a lesson in structural equation modeling. Instead, we offer it as evidence that the `semTable` function does work, as promised, for a wide variety of contexts.

References

R Core Team (2019). *R: A language and environment for statistical computing.* R Foundation for Statistical Computing, Vienna, Austria: R Foundation for Statistical Computing.

Rosseel, Y. (2012). lavaan: An R package for structural equation modeling. *Journal of Statistical Software*, 48(2), 1–36.

Replication Information

Please leave this next code chunk if you are producing a guide document.

```
R version 3.5.2 (2018-12-20)
Platform: x86_64-pc-linux-gnu (64-bit)
Running under: Ubuntu 18.10

Matrix products: default
BLAS: /usr/lib/x86_64-linux-gnu/blas/libblas.so.3.8.0
LAPACK: /usr/lib/x86_64-linux-gnu/lapack/liblapack.so.3.8.0

locale:
 [1] LC_CTYPE=en_US.UTF-8       LC_NUMERIC=C
 [3] LC_TIME=en_US.UTF-8        LC_COLLATE=en_US.UTF-8
 [5] LC_MONETARY=en_US.UTF-8    LC_MESSAGES=en_US.UTF-8
 [7] LC_PAPER=en_US.UTF-8       LC_NAME=C
 [9] LC_ADDRESS=C               LC_TELEPHONE=C
[11] LC_MEASUREMENT=en_US.UTF-8 LC_IDENTIFICATION=C

attached base packages:
[1] stats     graphics  grDevices utils     datasets  methods   base

other attached packages:
[1] semTable_1.0       lavaan_0.6-3       stationery_0.98.14

loaded via a namespace (and not attached):
 [1] Rcpp_1.0.0      digest_0.6.18   plyr_1.8.4      xtable_1.8-3
 [5] stats4_3.5.2    evaluate_0.13   zip_1.0.0       pbivnorm_0.6.0
 [9] openxlsx_4.1.0  rmarkdown_1.11  tools_3.5.2     foreign_0.8-71
[13] kutils_1.64     xfun_0.5        compiler_3.5.2  mnormt_1.5-5
[17] htmltools_0.3.6 knitr_1.21
```