A more efficient and consistent way of fitting PLS structural models: A better alternative to SEM than traditional PLS.

As published in Benchmarks Research Matters, February 2017

https://benchmarks.it.unt.edu/researchmatters

Jon Starkweather, PhD

Jon Starkweather, PhD
jonathan.starkweather@unt.edu
Consultant
Research and Statistical Support

https://www.unt.edu

http://it.unt.edu/research

R&SS hosts a number of "Short Courses".
A list of them is available at:
http://it.unt.edu/researchshortcourses

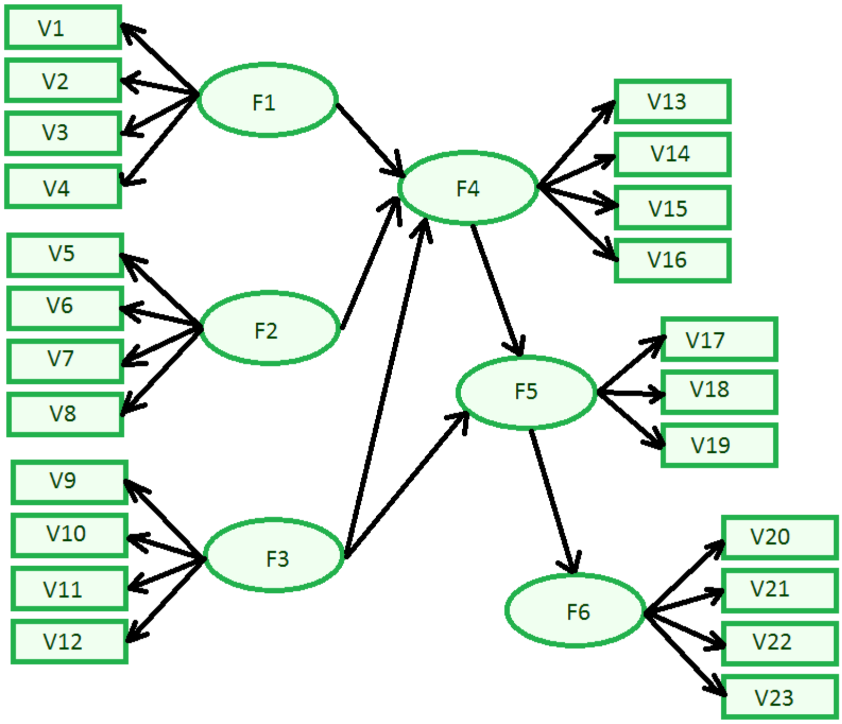
Those interested in learning more about R, or how to use it, can find information here:
`http://bayes.acs.unt.edu:8083/BayesContent/class/Jon/R_SC/`

A more efficient and consistent way of fitting PLS structural models: A better alternative to SEM than traditional PLS.

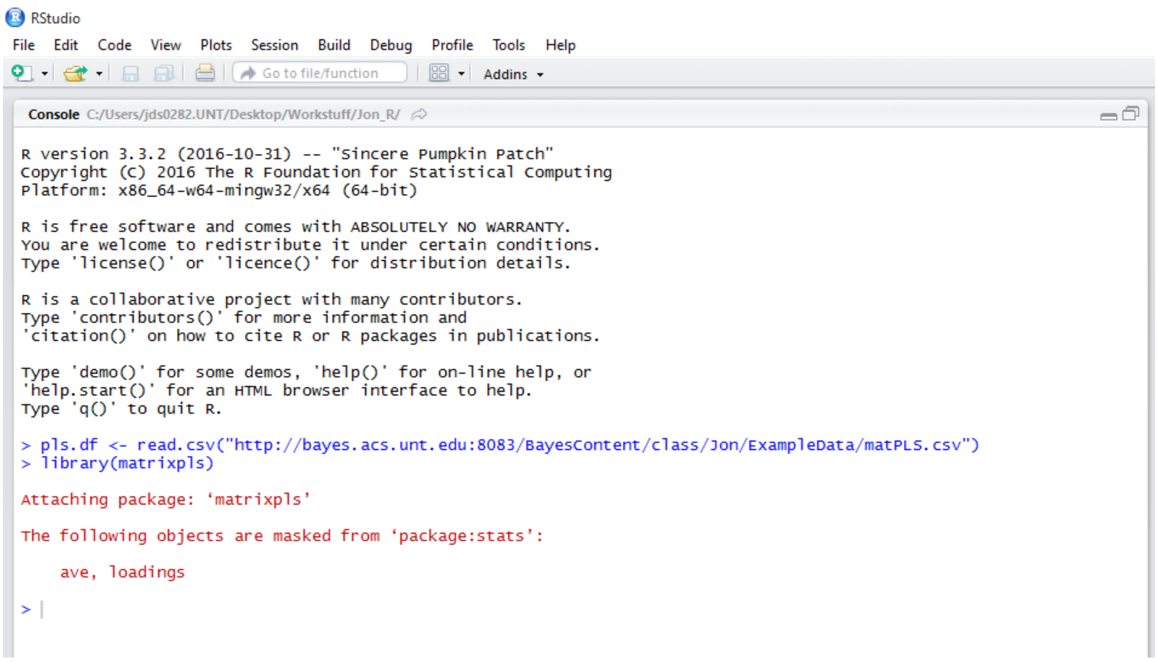
Partial Least Squares (PLS) modeling (Wold, 1965, 1966a, 1973) or ‘soft modeling’ (Wold, 1982; Faulk, & Miller, 1992) was discussed in this column several years back (Starkweather, 2011). PLS is an important alternative to traditional path modeling and / or structural equation modeling (SEM) when the data in hand does not conform to the assumptions of those modeling techniques. However, PLS modeling does have its drawbacks. Early on Dijkstra (1983) revealed a lack of consistency when PLS is used to estimate structural models. Other researchers (Wold, 1982; Fornell & Bookstein, 1982) have noted that “PLS does not solve a global optimization problem for parameter estimation, indicating that there exists no single criterion consistently minimized or maximized to determine model parameter estimates” (Hwang & Takane, 2004, p. 1). Hwang and Takane also point out that PLS offers no global goodness of fit statistic which would allow model comparisons. In response to the criticisms above, several researchers have proposed alternative methods for, or modifications to, the traditional PLS approach. As stated in the ‘matrixpls’ package vignette (Ronkko, 2016c):

> “Hwang and Takane (2014; 2004) proposed generalized structured component analysis (GSCA) arguing that it is superior over PLS because it has an explicit optimization criterion, which the PLS algorithm lacks. Dijkstra (2011; Dijkstra and Henseler 2015b; Dijkstra and Henseler 2015a) proposed that PLS can be made consistent by applying disattenuation, referring to this estimator as PLSc. Huang (2013; Bentler and Huang 2014) proposed two additional estimators that parameterize LISREL estimators based on Dijkstra’s PLSc estimator. These estimators, referred to as PLSe1 an PLSe2 are argued to be more efficient than the consistent PLSc estimator” (p. 2).

Naturally, the focus of the current article is the ‘matrixpls’ package (Ronkko, 2016a) and its capabilities to use the new methods mentioned in the preceding paragraph. Below a simulated dataset is used to demonstrate the ‘matrixpls’ function of the ‘matrixpls’ package fitting the following model.



First, import the simulated data from the Research and Statistical Support (R&SS) server[1] and load the 'matrixpls' package (Ronkko, 2016a).



Next, we specify the structural model using the 3 matrices style (i.e. inner, outer or reflective, and formative). We begin by creating the inner matrix (i.e. a matrix specifying the unobserved variable relationships). Keep in mind, the 'matrixpls' function can recognize multiple methods for specifying a model (e.g., using 'lavaan' package syntax; Yves, 2012a & 2012b, as well as using the 'semPLS'

[1] `http://it.unt.edu/research`

package syntax; Monecke, & Leisch, 2012); the example below shows how the inner matrix would be specified for the 'plspm' package (Sanchez, Trinchera, & Russolillo, 2016) and function, which is also accepted by the 'matrixpls' function.

```
> inner.matrix <- matrix(c(0, 0, 0, 0, 0, 0,
+                          0, 0, 0, 0, 0, 0,
+                          0, 0, 0, 0, 0, 0,
+                          1, 1, 1, 0, 0, 0,
+                          0, 0, 1, 1, 0, 0,
+                          0, 0, 0, 0, 1, 0), 6, 6, byrow = TRUE)
> dimnames(inner.matrix) <- list(c("f1", "f2", "f3", "f4", "f5", "f6"),
+                              c("f1", "f2", "f3", "f4", "f5", "f6"))
> inner.matrix
   f1 f2 f3 f4 f5 f6
f1  0  0  0  0  0  0
f2  0  0  0  0  0  0
f3  0  0  0  0  0  0
f4  1  1  1  0  0  0
f5  0  0  1  1  0  0
f6  0  0  0  0  1  0
>
```

Next, we create the outer matrix (i.e. specify the relationships between observed variables and the unobserved variables). Later, this matrix will be referred to as the 'reflective' matrix because it specifies those types of relationships.

```
> outer.matrix <- matrix(rep(0, 23*6), ncol = 6)
> outer.matrix[1:4, 1] <- 1
> outer.matrix[5:8, 2] <- 1
> outer.matrix[9:12, 3] <- 1
> outer.matrix[13:16, 4] <- 1
> outer.matrix[17:19, 5] <- 1
> outer.matrix[20:23, 6] <- 1
> dimnames(outer.matrix) <- list(names(pls.df), c("f1", "f2", "f3", "f4", "f5", "f6"))
> outer.matrix
    f1 f2 f3 f4 f5 f6
v1   1  0  0  0  0  0
v2   1  0  0  0  0  0
v3   1  0  0  0  0  0
v4   1  0  0  0  0  0
v5   0  1  0  0  0  0
v6   0  1  0  0  0  0
v7   0  1  0  0  0  0
v8   0  1  0  0  0  0
v9   0  0  1  0  0  0
v10  0  0  1  0  0  0
v11  0  0  1  0  0  0
v12  0  0  1  0  0  0
v13  0  0  0  1  0  0
v14  0  0  0  1  0  0
v15  0  0  0  1  0  0
v16  0  0  0  1  0  0
v17  0  0  0  0  1  0
v18  0  0  0  0  1  0
v19  0  0  0  0  1  0
v20  0  0  0  0  0  1
v21  0  0  0  0  0  1
v22  0  0  0  0  0  1
v23  0  0  0  0  0  1
>
```

Next, we create the formative matrix, which in this example is all zeros because all our observed variables are reflective (i.e. not formative). Recall, reflective means that the unobserved variables are theorized to *cause* the observed variables (scores); whereas formative means the unobserved variables are *caused by* the observed variables.

```
> form.matrix <- matrix(rep(0, 23*6), nrow = 6)
> dimnames(form.matrix) <- list(c("f1", "f2", "f3", "f4", "f5", "f6"), names(pls.df))
> form.matrix
   v1 v2 v3 v4 v5 v6 v7 v8 v9 v10 v11 v12 v13 v14 v15 v16 v17 v18 v19 v20 v21 v22 v23
f1  0  0  0  0  0  0  0  0  0   0   0   0   0   0   0   0   0   0   0   0   0   0   0
f2  0  0  0  0  0  0  0  0  0   0   0   0   0   0   0   0   0   0   0   0   0   0   0
f3  0  0  0  0  0  0  0  0  0   0   0   0   0   0   0   0   0   0   0   0   0   0   0
f4  0  0  0  0  0  0  0  0  0   0   0   0   0   0   0   0   0   0   0   0   0   0   0
f5  0  0  0  0  0  0  0  0  0   0   0   0   0   0   0   0   0   0   0   0   0   0   0
f6  0  0  0  0  0  0  0  0  0   0   0   0   0   0   0   0   0   0   0   0   0   0   0
> 
```

Now we can combine all three matrices into a list object which specifies the structural model.

```
> s.model <- list(inner = inner.matrix,
+                 reflective = outer.matrix,
+                 formative = form.matrix)
> s.model
$inner
   f1 f2 f3 f4 f5 f6
f1  0  0  0  0  0  0
f2  0  0  0  0  0  0
f3  0  0  0  0  0  0
f4  1  1  1  0  0  0
f5  0  0  1  1  0  0
f6  0  0  0  0  1  0

$reflective
    f1 f2 f3 f4 f5 f6
v1   1  0  0  0  0  0
v2   1  0  0  0  0  0
v3   1  0  0  0  0  0
v4   1  0  0  0  0  0
v5   0  1  0  0  0  0
v6   0  1  0  0  0  0
v7   0  1  0  0  0  0
v8   0  1  0  0  0  0
v9   0  0  1  0  0  0
v10  0  0  1  0  0  0
v11  0  0  1  0  0  0
v12  0  0  1  0  0  0
v13  0  0  0  1  0  0
v14  0  0  0  1  0  0
v15  0  0  0  1  0  0
v16  0  0  0  1  0  0
v17  0  0  0  0  1  0
v18  0  0  0  0  1  0
v19  0  0  0  0  1  0
v20  0  0  0  0  0  1
v21  0  0  0  0  0  1
v22  0  0  0  0  0  1
v23  0  0  0  0  0  1

$formative
   v1 v2 v3 v4 v5 v6 v7 v8 v9 v10 v11 v12 v13 v14 v15 v16 v17 v18 v19 v20 v21 v22 v23
f1  0  0  0  0  0  0  0  0  0   0   0   0   0   0   0   0   0   0   0   0   0   0   0
f2  0  0  0  0  0  0  0  0  0   0   0   0   0   0   0   0   0   0   0   0   0   0   0
f3  0  0  0  0  0  0  0  0  0   0   0   0   0   0   0   0   0   0   0   0   0   0   0
f4  0  0  0  0  0  0  0  0  0   0   0   0   0   0   0   0   0   0   0   0   0   0   0
f5  0  0  0  0  0  0  0  0  0   0   0   0   0   0   0   0   0   0   0   0   0   0   0
f6  0  0  0  0  0  0  0  0  0   0   0   0   0   0   0   0   0   0   0   0   0   0   0

> 
```

The only other thing we need is the variance-covariance matrix of the observed variables.

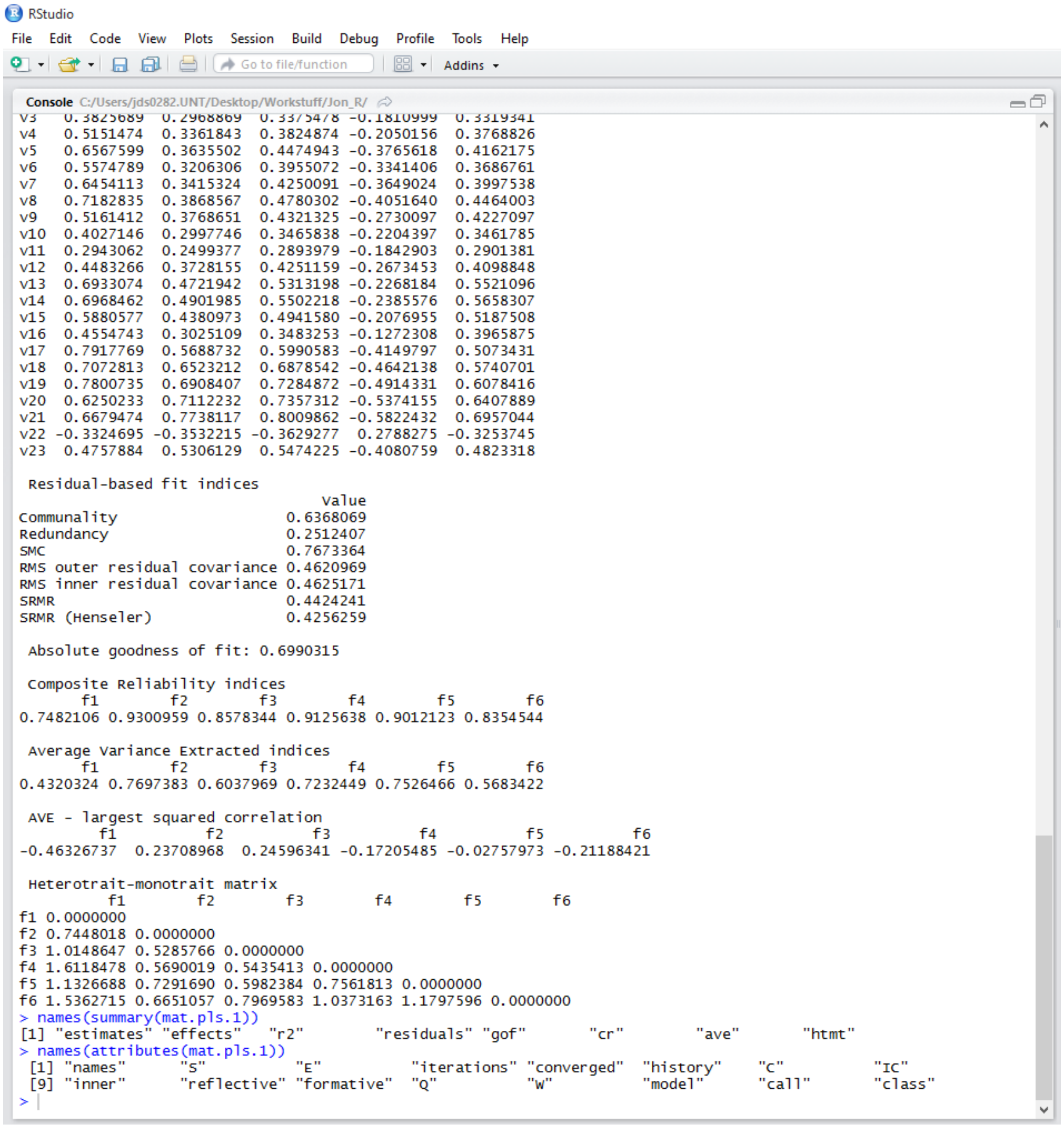
```
> cov.mat <- cov(pls.df)
> 
```

Now we can apply the ‘matrixpls’ function using OLS Regression estimation as would be done with traditional PLS (e.g., the ‘plspm’ package).

```
RStudio
File  Edit  Code  View  Plots  Session  Build  Debug  Profile  Tools  Help
Go to file/function    Addins

Console C:/Users/jds0282.UNT/Desktop/Workstuff/Jon_R/
> mat.pls.1 <- matrixpls(S = cov.mat,
+                        model = s.model,
+                        disattenuate = TRUE,
+                        innerEstim = innerEstim.centroid,
+                        outerEstim = outerEstim.modeA,  # Simple correlation coefficients.
+                        parametersInner = estimator.ols,
+                        parametersReflective = estimator.cfaLoadings,
+                        validateInput = TRUE,
+                        standardize = TRUE)
> 
```

```
RStudio
File  Edit  Code  View  Plots  Session  Build  Debug  Profile  Tools  Help
Go to file/function    Addins

Console C:/Users/jds0282.UNT/Desktop/Workstuff/Jon_R/
> mat.pls.1  # <-- just the parameter estimates and weights.

 matrixpls parameter estimates
                Est.
f4~f1    0.959585619
f4~f2   -0.005789733
f4~f3   -0.016649558
f5~f3    0.259157126
f5~f4    0.619161402
f6~f5    0.883304239
f1=~v1   0.774814345
f1=~v2  -0.506019294
f1=~v3   0.614495588
f1=~v4   0.702945190
f2=~v5   0.874309274
f2=~v6   0.783034469
f2=~v7   0.886793723
f2=~v8   0.956551212
f3=~v9   0.871140380
f3=~v10  0.769987877
f3=~v11  0.653695152
f3=~v12  0.797560788
f4=~v13  0.879640677
f4=~v14  0.873682892
f4=~v15  0.851192387
f4=~v16  0.794582707
f5=~v17  0.881930823
f5=~v18  0.836698553
f5=~v19  0.883217693
f6=~v20  0.843340478
f6=~v21  0.894978351
f6=~v22 -0.528041185
f6=~v23  0.694501106

 matrixpls weights
          v1         v2        v3        v4        v5        v6        v7        v8        v9       v10
f1 0.3928233 -0.2425821 0.3121514 0.3656424 0.0000000 0.0000000 0.0000000 0.0000000 0.0000000 0.000000
f2 0.0000000  0.0000000 0.0000000 0.0000000 0.2615038 0.2584804 0.2945904 0.2862354 0.0000000 0.000000
f3 0.0000000  0.0000000 0.0000000 0.0000000 0.0000000 0.0000000 0.0000000 0.0000000 0.3023735 0.315368
f4 0.0000000  0.0000000 0.0000000 0.0000000 0.0000000 0.0000000 0.0000000 0.0000000 0.0000000 0.000000
f5 0.0000000  0.0000000 0.0000000 0.0000000 0.0000000 0.0000000 0.0000000 0.0000000 0.0000000 0.000000
f6 0.0000000  0.0000000 0.0000000 0.0000000 0.0000000 0.0000000 0.0000000 0.0000000 0.0000000 0.000000
         v11       v12       v13      v14       v15       v16      v17       v18       v19       v20      v21
f1 0.0000000 0.0000000 0.0000000 0.000000 0.0000000 0.0000000 0.000000 0.0000000 0.0000000 0.0000000 0.000000
f2 0.0000000 0.0000000 0.0000000 0.000000 0.0000000 0.0000000 0.000000 0.0000000 0.0000000 0.0000000 0.000000
f3 0.2974447 0.2829879 0.0000000 0.000000 0.0000000 0.0000000 0.000000 0.0000000 0.0000000 0.0000000 0.000000
f4 0.0000000 0.0000000 0.2630766 0.276333 0.2737062 0.3150948 0.000000 0.0000000 0.0000000 0.0000000 0.000000
f5 0.0000000 0.0000000 0.0000000 0.000000 0.0000000 0.0000000 0.369771 0.3515718 0.3717692 0.0000000 0.000000
f6 0.0000000 0.0000000 0.0000000 0.000000 0.0000000 0.0000000 0.000000 0.0000000 0.0000000 0.3337904 0.351649
          v22       v23
f1  0.0000000 0.0000000
f2  0.0000000 0.0000000
f3  0.0000000 0.0000000
f4  0.0000000 0.0000000
f5  0.0000000 0.0000000
f6 -0.2416281 0.2964546

Weight algorithm converged in 4 iterations.
> 
```

A much more thorough presentation of output can be seen by using either the ‘summary’ function or the ‘attributes’ function; however, the output of each is only partially presented here due to their size. The image below shows what is returned when applying the ‘names’ function to the ‘summary’, as well as the ‘attributes’, of the ‘matrixpls’ object. Using the various returned names one can then extract relevant elements of the ‘matrixpls’ object (e.g., `summary(mat.pls.1)$gof` can be used to extract the absolute goodness of fit).



```
v3   0.3825689  0.2968869  0.3375478 -0.1810999  0.3319341
v4   0.5151474  0.3361843  0.3824874 -0.2050156  0.3768826
v5   0.6567599  0.3635502  0.4474943 -0.3765618  0.4162175
v6   0.5574789  0.3206306  0.3955072 -0.3341406  0.3686761
v7   0.6454113  0.3415324  0.4250091 -0.3649024  0.3997538
v8   0.7182835  0.3868567  0.4780302 -0.4051640  0.4464003
v9   0.5161412  0.3768651  0.4321325 -0.2730097  0.4227097
v10  0.4027146  0.2997746  0.3465838 -0.2204397  0.3461785
v11  0.2943062  0.2499377  0.2893979 -0.1842903  0.2901381
v12  0.4483266  0.3728155  0.4251159 -0.2673453  0.4098848
v13  0.6933074  0.4721942  0.5313198 -0.2268184  0.5521096
v14  0.6968462  0.4901985  0.5502218 -0.2385576  0.5658307
v15  0.5880577  0.4380973  0.4941580 -0.2076955  0.5187508
v16  0.4554743  0.3025109  0.3483253 -0.1272308  0.3965875
v17  0.7917769  0.5688732  0.5990583 -0.4149797  0.5073431
v18  0.7072813  0.6523212  0.6878542 -0.4642138  0.5740701
v19  0.7800735  0.6908407  0.7284872 -0.4914331  0.6078416
v20  0.6250233  0.7112232  0.7357312 -0.5374155  0.6407889
v21  0.6679474  0.7738117  0.8009862 -0.5822432  0.6957044
v22 -0.3324695 -0.3532215 -0.3629277  0.2788275 -0.3253745
v23  0.4757884  0.5306129  0.5474225 -0.4080759  0.4823318

 Residual-based fit indices
                                  Value
Communality                   0.6368069
Redundancy                    0.2512407
SMC                           0.7673364
RMS outer residual covariance 0.4620969
RMS inner residual covariance 0.4625171
SRMR                          0.4424241
SRMR (Henseler)               0.4256259

 Absolute goodness of fit: 0.6990315

 Composite Reliability indices
       f1        f2        f3        f4        f5        f6
0.7482106 0.9300959 0.8578344 0.9125638 0.9012123 0.8354544

 Average Variance Extracted indices
       f1        f2        f3        f4        f5        f6
0.4320324 0.7697383 0.6037969 0.7232449 0.7526466 0.5683422

 AVE - largest squared correlation
         f1          f2          f3          f4          f5          f6
-0.46326737  0.23708968  0.24596341 -0.17205485 -0.02757973 -0.21188421

 Heterotrait-monotrait matrix
          f1        f2        f3        f4        f5        f6
f1 0.0000000
f2 0.7448018 0.0000000
f3 1.0148647 0.5285766 0.0000000
f4 1.6118478 0.5690019 0.5435413 0.0000000
f5 1.1326688 0.7291690 0.5982384 0.7561813 0.0000000
f6 1.5362715 0.6651057 0.7969583 1.0373163 1.1797596 0.0000000
> names(summary(mat.pls.1))
[1] "estimates" "effects"   "r2"        "residuals" "gof"       "cr"        "ave"       "htmt"
> names(attributes(mat.pls.1))
 [1] "names"      "S"          "E"          "iterations" "converged"  "history"    "C"          "IC"
 [9] "inner"      "reflective" "formative"  "Q"          "W"          "model"      "call"       "class"
>
```

Next, we re-specify the same model using the ‘lavaan’ package style of model specification syntax. This style of model specification syntax is much more intuitive and requires fewer lines (of code); as well as fewer objects in the workspace (i.e. not 3 matrices).

RStudio

File Edit Code View Plots Session Build Debug Profile Tools Help

Go to file/function | Addins

Console C:/Users/jds0282.UNT/Desktop/Workstuff/Jon_R/

```
> lavaan.s.model <- '
+   F4 ~ F1 + F2 + F3
+   F5 ~ F3 + F4
+   F6 ~ F5
+   F1 =~ v1 + v2 + v3 + v4
+   F2 =~ v5 + v6 + v7 + v8
+   F3 =~ v9 + v10 + v11 + v12
+   F4 =~ v13 + v14 + v15 + v16
+   F5 =~ v17 + v18 + v19
+   F6 =~ v20 + v21 + v22 + v23
+   '
> 
```

Next, we will fit the same model, as re-specified using ‘lavaan’ syntax, but we will use GSCAc estimation and 2-stage least squares. Note, only partial output is displayed from the ‘summary’ function.

RStudio

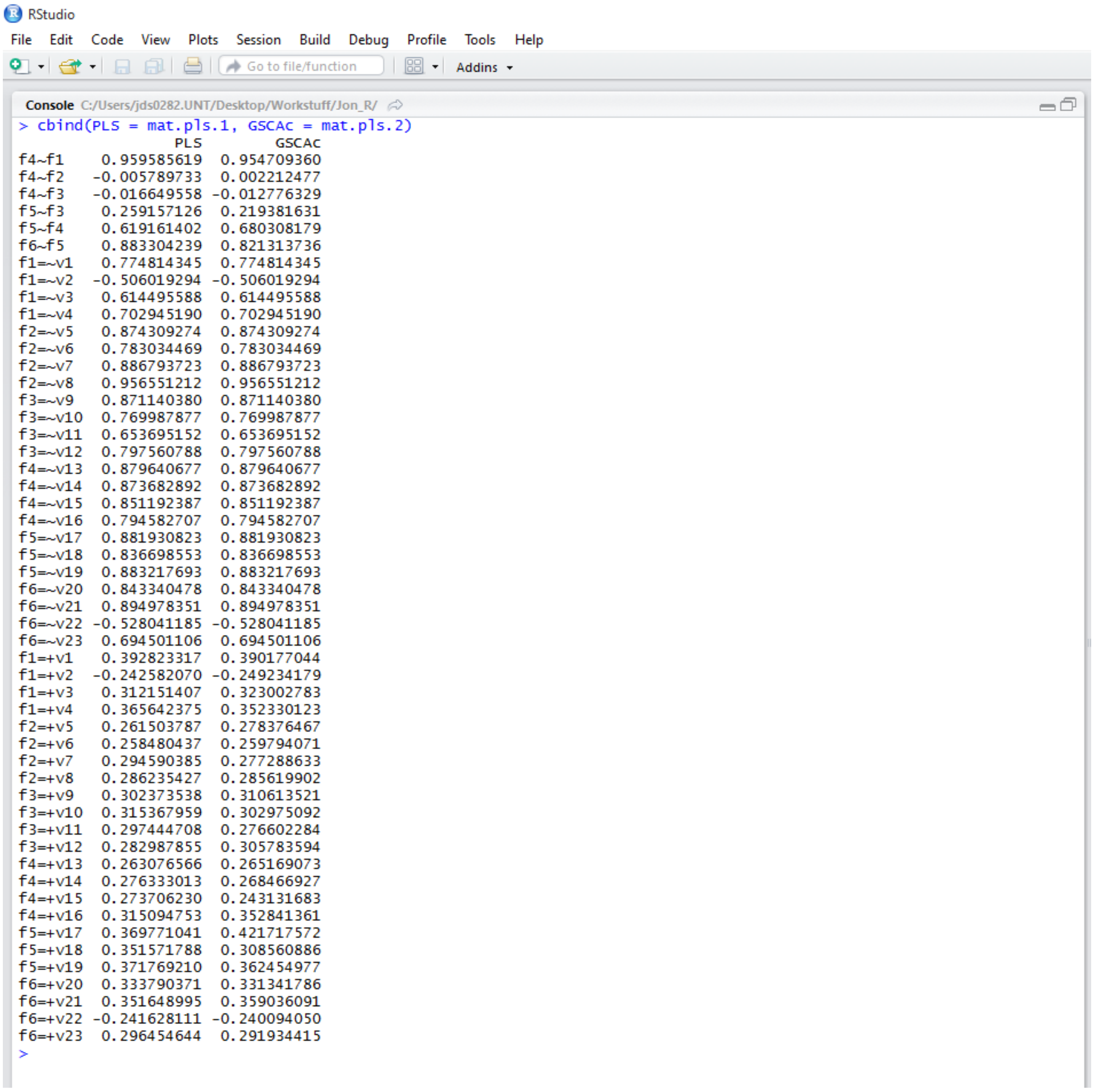
File Edit Code View Plots Session Build Debug Profile Tools Help

Go to file/function | Addins

Console C:/Users/jds0282.UNT/Desktop/Workstuff/Jon_R/

```
> mat.pls.2 <- matrixpls(S = cov.mat,
+                        model = lavaan.s.model,
+                        disattenuate = TRUE,
+                        innerEstim = innerEstim.gsca,
+                        outerEstim = outerEstim.gsca,
+                        parametersInner = estimator.tsls,
+                        parametersReflective = estimator.cfaLoadings,
+                        validateInput = TRUE,
+                        standardize = TRUE)
> summary(mat.pls.2)

 matrixpls parameter estimates
                 Est.
F4~F1     0.954709360
F4~F2     0.002212477
F4~F3    -0.012776329
F5~F3     0.219381631
F5~F4     0.680308179
F6~F5     0.821313736
F1=~v1    0.774814345
F1=~v2   -0.506019294
F1=~v3    0.614495588
F1=~v4    0.702945190
F2=~v5    0.874309274
F2=~v6    0.783034469
F2=~v7    0.886793723
F2=~v8    0.956551212
F3=~v9    0.871140380
F3=~v10   0.769987877
F3=~v11   0.653695152
F3=~v12   0.797560788
F4=~v13   0.879640677
F4=~v14   0.873682892
F4=~v15   0.851192387
F4=~v16   0.794582707
F5=~v17   0.881930823
F5=~v18   0.836698553
F5=~v19   0.883217693
F6=~v20   0.843340478
F6=~v21   0.894978351
F6=~v22  -0.528041185
F6=~v23   0.694501106

 matrixpls weights
         v1         v2        v3        v4        v5        v6        v7        v8        v9       v10
F1 0.390177 -0.2492342 0.3230028 0.3523301 0.0000000 0.0000000 0.0000000 0.0000000 0.0000000 0.0000000
F2 0.000000  0.0000000 0.0000000 0.0000000 0.2783765 0.2597941 0.2772886 0.2856199 0.0000000 0.0000000
F3 0.000000  0.0000000 0.0000000 0.0000000 0.0000000 0.0000000 0.0000000 0.0000000 0.3106135 0.3029751
F4 0.000000  0.0000000 0.0000000 0.0000000 0.0000000 0.0000000 0.0000000 0.0000000 0.0000000 0.0000000
F5 0.000000  0.0000000 0.0000000 0.0000000 0.0000000 0.0000000 0.0000000 0.0000000 0.0000000 0.0000000
F6 0.000000  0.0000000 0.0000000 0.0000000 0.0000000 0.0000000 0.0000000 0.0000000 0.0000000 0.0000000
         v11       v12       v13       v14       v15       v16       v17       v18      v19       v20
F1 0.0000000 0.0000000 0.0000000 0.0000000 0.0000000 0.0000000 0.0000000 0.0000000 0.000000 0.0000000
F2 0.0000000 0.0000000 0.0000000 0.0000000 0.0000000 0.0000000 0.0000000 0.0000000 0.000000 0.0000000
F3 0.2766023 0.3057836 0.0000000 0.0000000 0.0000000 0.0000000 0.0000000 0.0000000 0.000000 0.0000000
F4 0.0000000 0.0000000 0.2651691 0.2684669 0.2431317 0.3528414 0.0000000 0.0000000 0.000000 0.0000000
F5 0.0000000 0.0000000 0.0000000 0.0000000 0.0000000 0.0000000 0.4217176 0.3085609 0.362455 0.0000000
F6 0.0000000 0.0000000 0.0000000 0.0000000 0.0000000 0.0000000 0.0000000 0.0000000 0.000000 0.3313418
         v21       v22       v23
F1 0.0000000  0.000000 0.0000000
```

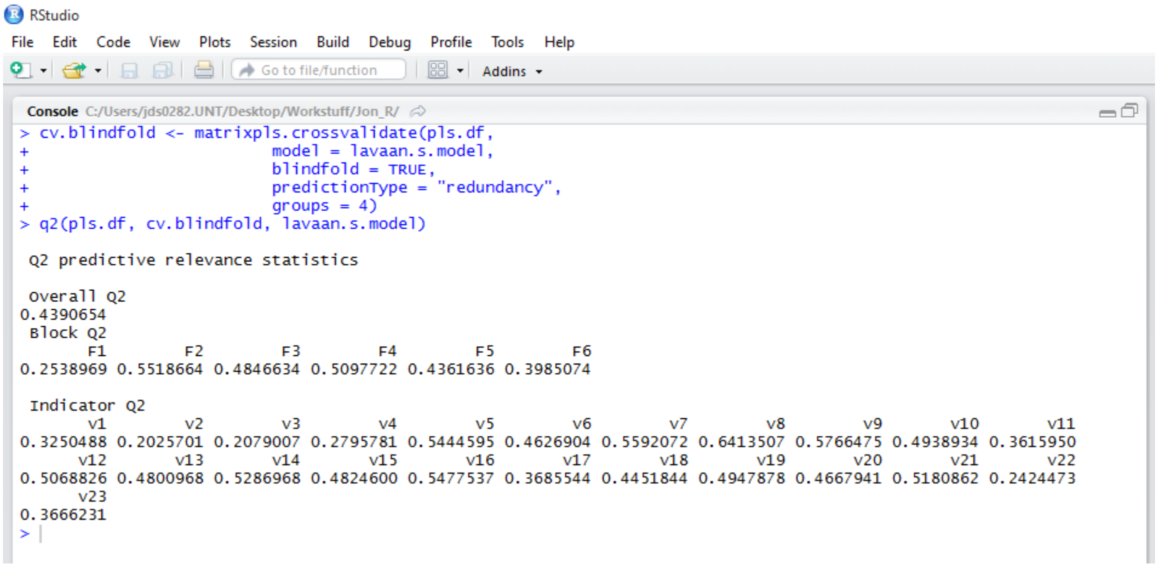
So, now we can do a comparison of the coefficients based on the two estimation techniques used on the same data and the same model — keep in mind with this simulated data there are very small differences in the statistics produced.




```
> cbind(PLS = mat.pls.1, GSCAC = mat.pls.2)
               PLS        GSCAC
f4~f1     0.959585619  0.954709360
f4~f2    -0.005789733  0.002212477
f4~f3    -0.016649558 -0.012776329
f5~f3     0.259157126  0.219381631
f5~f4     0.619161402  0.680308179
f6~f5     0.883304239  0.821313736
f1=~v1    0.774814345  0.774814345
f1=~v2   -0.506019294 -0.506019294
f1=~v3    0.614495588  0.614495588
f1=~v4    0.702945190  0.702945190
f2=~v5    0.874309274  0.874309274
f2=~v6    0.783034469  0.783034469
f2=~v7    0.886793723  0.886793723
f2=~v8    0.956551212  0.956551212
f3=~v9    0.871140380  0.871140380
f3=~v10   0.769987877  0.769987877
f3=~v11   0.653695152  0.653695152
f3=~v12   0.797560788  0.797560788
f4=~v13   0.879640677  0.879640677
f4=~v14   0.873682892  0.873682892
f4=~v15   0.851192387  0.851192387
f4=~v16   0.794582707  0.794582707
f5=~v17   0.881930823  0.881930823
f5=~v18   0.836698553  0.836698553
f5=~v19   0.883217693  0.883217693
f6=~v20   0.843340478  0.843340478
f6=~v21   0.894978351  0.894978351
f6=~v22  -0.528041185 -0.528041185
f6=~v23   0.694501106  0.694501106
f1=+v1    0.392823317  0.390177044
f1=+v2   -0.242582070 -0.249234179
f1=+v3    0.312151407  0.323002783
f1=+v4    0.365642375  0.352330123
f2=+v5    0.261503787  0.278376467
f2=+v6    0.258480437  0.259794071
f2=+v7    0.294590385  0.277288633
f2=+v8    0.286235427  0.285619902
f3=+v9    0.302373538  0.310613521
f3=+v10   0.315367959  0.302975092
f3=+v11   0.297444708  0.276602284
f3=+v12   0.282987855  0.305783594
f4=+v13   0.263076566  0.265169073
f4=+v14   0.276333013  0.268466927
f4=+v15   0.273706230  0.243131683
f4=+v16   0.315094753  0.352841361
f5=+v17   0.369771041  0.421717572
f5=+v18   0.351571788  0.308560886
f5=+v19   0.371769210  0.362454977
f6=+v20   0.333790371  0.331341786
f6=+v21   0.351648995  0.359036091
f6=+v22  -0.241628111 -0.240094050
f6=+v23   0.296454644  0.291934415
>
```

We can also compare the measurement model's composite reliability estimates (i.e. the "Q" statistic) from each method of estimation. Note, the "Q" statistic produced is analogous to coefficient alpha from traditional item evaluation.

If interested in obtaining the Q2 predictive relevance statistic(s) you first need to run the cross–validation function which mimics the blindfolding procedure used with the ‘semPLS’ function of the package by the same name. Then, apply the ‘q2’ function to the object returned by the cross-validation.



There are many benefits to using the ’matrixpls’ package (Ronkko, 2016a) rather than the ‘plspm’ package (Sanchez, Trinchera, & Russolillo, 2016) or the ‘semPLS’ package (Monecke, & Leisch, 2012) for fitting structural models when SEM cannot be used. Obviously, the greatest benefits of the ‘matrixpls’ package is the ability to use multiple new and more robust estimation methods; only one such combination of estimation techniques was used above. The ‘matrixpls’ function is also more computationally efficient (Ronkko, 2016c, p. 2) and offers more flexibility with respect to the types of models that can be fit. Consider the limitations on the specification of model matrices with traditional PLS packages. Those packages require the following matrix restrictions: “the inner must be a lower triangular matrix, reflective must have exactly one non-zero value on each row and must have at least one non-zero value on each column, and formative must only contain zeros” (Ronkko, 2016c, p. 4). The ‘matrixpls’ function has two restrictions; all matrices must be binary and the inner must have zeros on the diagonal (Ronkko, 2016c). Another benefit of the ‘matrixpls’ function is the ability to specify a model using the ‘lavaan’ package (Yves, 2012a) model specification syntax which is highly intuitive; interested readers should review the ‘lavaan’ package documentation available at CRAN (see Yves, 2012b). Lastly, another major benefit to using the ‘matrixpls’ package is a function for doing Monte Carlo simulations of a ‘matrixpls’ object.

A version of the R script used in this article can be found on the R&SS Do-It-Yourself Introduction to

R^2 website at the bottom of the Module 9 section.

Until next time; *everybody look what's going down*

References and Resources

Bentler, P. M., & Huang, W. (2014). On components, latent variables, PLS and simple methods: Reactions to Rigdon's rethinking of PLS. *Long Range Planning, 47*(3), 138 - 145.

Dijkstra, T. (1983). Some comments on maximum likelihood and partial least squares methods. *Journal of Econometrics, 22*, 67 - 90.

Dijkstra, T. K., & Henseler, J. (2015a). Consistent partial least squares path modeling. *MIS Quarterly, 39*(2), 297 - 316.

Dijkstra, T. K., & Henseler, J. (2015b). Consistent and asymptotically normal PLS estimators for linear structural equations. *Computational Statistics and Data Analysis, 81*, 10 - 23.

Faulk, R. R., & Miller, N. B. (1992). *A primer for soft modeling.* Akron, OH: The University of Akron.

Fornell, C., & Bookstein, F. L. (1982). Two structural equation models: LISREL and PLS applied to consumer exit-voice theory. *Journal of Marketing Research, 19*, 440-452.

Huang, W. (2013). "PLSe: Efficient Estimators and Tests for Partial Least Squares." Doctoral dissertation, Los Angeles: University of California.

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

Hwang, H., & Takane, Y. (2014). *Generalized structured component analysis: A component-based approach to structural equation modeling.* New York: CRC Press / Taylor & Francis Group.

Monecke, A., & Leisch, F. (2012). semPLS: Structural Equation Modeling Using Partial Least Squares. *Journal of Statistical Software, 48*(3), 1-32. URL `http://www.jstatsoft.org/v48/i03/`. Documentation available at CRAN: `https://cran.r-project.org/web/packages/semPLS/index.html`

Rigdon, E. E. (2012). Rethinking partial least squares path modeling: In praise of simple methods. *Long Range Planning, 45*, 341 - 358.

Ronkko, M. (2016a). matrixpls: Matrix-based Partial Least Squares Estimation. R package version 1.0.4. Documentation available at CRAN: `https://cran.r-project.org/web/packages/matrixpls/index.html`

Ronkko, M. (2016b). Package matrixpls manual. Available at CRAN: `https://cran.r-project.org/web/packages/matrixpls/matrixpls.pdf`

[2] `http://bayes.acs.unt.edu:8083/BayesContent/class/Jon/R_SC/`

Ronkko, M. (2016c). Package matrixpls vignette: Introduction to matrixpls. Available at CRAN: https://cran.r-project.org/web/packages/matrixpls/vignettes/matrixpls-intro

Sanchez, G. (2013). *PLS Path Modeling with R* (Trowchez Editions). Berkeley, 2013. Available at: http://gastonsanchez.com/PLS_Path_Modeling_with_R.pdf

Sanchez, G., Trinchera, L., & Russolillo, G. (2016). Package 'plspm' with documentation, including package manual, available at CRAN: https://cran.r-project.org/web/packages/plspm/index.html

Starkweather, J. (2011). An alternative modeling strategy: Partial least squares. As published in the *RSS Matters* (now called *Research Matters*) column of University Information Technology's *Benchmarks* online magazine. Available at: http://bayes.acs.unt.edu:8083/BayesContent/class/Jon/Benchmarks/PLS_JDS_Jul

Vinzi, V. E., Chin, W. W., Hensler, J., & Wang, H. (2010). *Handbook of partial least squares: Concepts, methods, and applications*. New York: Springer.

Wold, H. O. A. (1965). A fixed-point theorem with econometric background, I-II. *Arkiv for Matematik*, *6*, 209 - 240.

Wold, H. O. A. (1966a). Estimation of principal components and related methods by iterative least squares. In P. R. Krishnaiah (Ed.), *Multivariate Analysis* (pp. 391 - 420). New York: Academic Press.

Wold, H. O. A. (1966b). Nonlinear estimation by iterative least squares procedures, in: E. N. David (Ed.), *Research papers in statistics: Festschrift for J. Neyman* (pp. 411 - 444). New York: Wiley.

Wold, H. O. A. (1973). Nonlinear iterative partial least squares (NIPALS) modeling: Some current developments. In P. R. Krishnaiah (Ed.), *Multivariate Analysis* (pp. 383 - 487). New York: Academic Press.

Wold, H. O. A. (1982). Soft modeling: The basic design and some extension. In K. G. Joreskog and H. Wold (Eds.), *Systems under Indirect Observations II* (pp. 1 - 54). Amsterdam: North-Holland.

Yves, R. (2012a). lavaan: An R Package for Structural Equation Modeling. *Journal of Statistical Software, 48*(2), 1-36. URL http://www.jstatsoft.org/v48/i02/.

Yves, R. (2012b). Package 'lavaan' manual which is available at CRAN: https://cran.r-project.org/

This article was last updated on January 23, 2017.

This document was created using LaTeX