Practical Assessment, Research & Evaluation

A peer-reviewed electronic journal.

Volume 18, Number 4, February 2013 ISSN 1531-7714

Factor Analysis using R

A. Alexander Beaujean, *Baylor University*

R (R Development Core Team, 2011) is a very powerful tool to analyze data, that is gaining in popularity due to its costs (its free) and flexibility (its open-source). This article gives a general introduction to using **R** (i.e., loading the program, using functions, importing data). Then, using data from Canivez, Konold, Collins, and Wilson (2009), this article walks the user through how to use the program to conduct factor analysis, from both an exploratory and confirmatory approach.

R (R Development Core Team, 2011) is an open-source statistical environment (language). It is a very flexible and powerful language that many data analysts are now using (e.g., Kelley, Lai, & Wu, 2008; Vance, 2009). This article will focus on using **R** in conducting factor analysis, but before doing do will first give a general introduction to the programming language.

Why Use R?

One question that readers may have is, *why use R when I am already familiar with data analysis package X?* One reason is that it is a very powerful programming language that is able to conduct a wide range of analysis. Thus, for many projects, all analysis can be conducted within the same program. This is particularly useful for factor analytic work, as one can examine the data's properties, check any model assumptions, conduct exploratory factor analysis, and then follow up with a confirmatory factor analysis without ever having to export data or write syntax in another program's language. Once the data has been entered into **R**, it will be available for use until the **R** program is closed. If the user saves the initial syntax used to import data, then importing the data for the second, third, forth, etc. data analysis session is only a matter of copying-and-pasting syntax written for the first session.

A second reason to use **R**, closely related to the first, is that it allows for users to submit their own packages to the **R** server for anyone to use. This has resulted in a large variety of packages (over 4000 at the time of writing this article) that can do everything from spatial statistics to text mining.

A third reason for using **R** is that it is a free open source programming language, available for most operating systems (e.g., Windows, Mac, Linux), and the vast majority of syntax and packages are transportable from one system to another. This can aid in both research collaboration and making one's research replicable (e.g., Pashler & Wagenmakers, 2012), as colleagues can reproduce results by copying and pasting the syntax. Moreover, these same features also allow **R** to be strong tool for teaching statistics, as it not only allows for students to reproduce the instructor's examples, but it allows them to examine the underlying processes behind complex topics such as Bayesian estimation and finding function extrema (e.g., Horton, Brown, & Qian, 2004).

Using R

R is currently maintained by the core-development team and the ***R** Project* 's web site (also known as Comprehensive R Archive Network [CRAN]) is http://www.r-project.org. This is the main site for **R** information, directions for obtaining the software, accompanying packages, and some user documentation.

Modes for Using R.

There are three different modes the user can select when using **R**. The first is through batch mode, whereby the user writes a document of **R** syntax, sends the entire file to **R** through the terminal (Mac or Linux) or command prompt (Windows), and then receives an output file with all the results. This mode is most useful when using **R** that is installed on a "supercomputer" for jobs that require massive computing power, such as Monte Carlo studies or Bayesian analysis of complex models. For more information about this mode, see Appendix B of Venables, Smith, and R Development Core Team (2012).

A second mode to run R is by using a graphical user interface (GUI). GUI mode allows the user to conduct the desired analyses through a series of point-and-clicks. There is not an "official" GUI for **R**, so many 3rd parties have developed GUIs for different niches of users, a list of which can be found at `http://www.sciviews.org/_rgui`. Because 3rd parties develop the GUI, there is not necessarily a universality to them. That is, some will only work on a single operating system, some are designed to do only a certain subset of analyses, and some are designed to interface with other software (e.g., the RStudio GUI can interface with LaTeX). While the specialization of the GUI can be useful for a given user, the lack of universality makes it hard to explain how to conduct analyses as it will be different from GUI to GUI. Moreover, like most statistical program GUIs, they tend to focus on certain subsets of analyses and neglect others. For example, one popular GUI is *R Commander* (Fox, 2005), whose menu-driven options are similar to those from SPSS. Thus, it has some basic options for general analysis, but if the user wants to conduct a "non-traditional" analysis [e.g., rotate factors using McCammon's (1966) entropy criterion], the GUI is not of much use.

The third, and arguably most popular mode for conducting analyses in **R** is through interactive mode, which basically means writing syntax and passing as much of it to *R* as needed. It is called interactive mode because when the user submits the syntax, **R** will parse the syntax immediately and deliver the output the user requested. One can use interactive in one of two ways. First, write the syntax in a syntax editor and have the editor pass the syntax into **R**. This method can be very useful as whatever syntax the user writes can be saved and then used again at a later time or by another user. **R** comes with a native syntax editor, but its functionality differs depending on the operating system used. The second way is to type (or paste from an external document) the syntax into **R** directly. This method has the largest universality, as the procedures for conducting a given analysis are usually identical across operating systems. Thus, this is the method this article will use in explaining **R** usage.

After downloading and starting **R** in interactive mode, the user will see a >, which is called the prompt. It is not typed (if typed, **R** will assume the user means "greater than"). Instead, it is used to indicate where to type syntax. If a command is too long to fit on a line, a + is used for the continuation prompt. Another symbol that most users will use frequently is the left arrow, `<-`, which is **R**'s standard assignment operator (another option is using =, but it is better to reserve using = for defining argument values within functions). The <- is **R**'s way of assigning whatever is on the right of the arrow to the object on the left of the arrow.

Functions and Packages

R stores data and analysis results in the computer's active memory in the form of named *objects*. The user can then do actions on these objects with *functions* (which are themselves objects). To use functions: (*a*) give the function's name followed by parentheses; and (*b*) in the parentheses, give the necessary values for the function's argument(s). Table 1 contains some widely used functions.

Table 1: Commonly Used **R** Functions

Function	Use
#	Comment. **R** ignores all text after the #.
c()	Concatenate. Concatenate the arguments included in the function.
Help() or ?	Show information about, and e xample uses of, a function.

With the initial download of **R**, it will come with some pre-defined functions (e.g., `mean()`, `var()`), stored in packages, most of which automatically load when starting **R**. These functions, however, may not conduct the needed analysis. Thus, the user can search CRAN to see if a contributed package can do the required analysis. These contributed packages usually consist of data and functions that were written in the **R** language.

To install an **R** package, use the `install.packages()` function, naming the package to install in quotation marks. One argument for `install.packages()` is `dep`, which stands for dependencies. If the user specifies `dep = TRUE`, **R** will concurrently download any other package upon which the package of interest is dependent, which saves the user from having to download each required package separately. For example, to install the *BaylorEdPsych* (Beaujean, 2012) package, use the following syntax.

```
install.packages("BaylorEdPsych", dep =
    TRUE)
```

The user only needs to install an **R** package to the hard drive one time, but will need to load it into memory every time the user starts a new **R** session and needs to use one of the package's functions. To load a package into **R**'s memory, use the `library()` function, without any quotation marks around the package name.

Importing Data

Users will usually want to store the data in an external file and then have **R** load the data. There are multiple ways to import data into **R**, depending on the format of the data.

The easiest way to import data into **R** is to save the file as a tab- or comma-delimited text file. Most spreadsheet and database programs can save data this way. If it is possible to store (or export) the data in this format, then the `read.table()` or `read.csv` functions, both of which are native to **R**, can read the data. Before importing it though, the user is advised to do two things. First, replace all missing value indicators to NA, which is **R**'s default missing value indicator. Second, either remove any spaces in the variable names or replace it with a period (e.g., first.name).

If it is not possible to store/export the data as a text file, there are **R** packages that can read data from other file formats. Table 2 lists some packages and the data type they will open.

Table 2: Different **R** Packages to Import Various Data Formats

Package	Filetypes
xlsReadWrite, gdata	Excel (.xls)
xlsx	Excel (.xlsx)
RODBC	databases that are ODBC compliant, e.g., MS SQLServer, MS Access, Oracle
RMySQL	MySQLRJDBC JDBC compliant databases
RSQLite	SQLite
foreign	Minitab, S3, SAS, SPSS, Stata,Systat, dBase, ARFF, DBF, REC, Octave

In order to read the data file, point **R** to the directory where it is located. **R** uses forward slashes for reading directory structures (i.e., how UNIX-type systems store files) or double backslash, e.g., `C:\\Regression\\Data.csv`. As an example, say a dataset with variable names in the first row is stored in a file named `data.csv` that is located in a folder called *name*, which is in a folder called *file*. **R** can read the file using the following syntaxes:

```
#Windows
new.data<-
    read.csv("C:\\file\\name\\data.csv",
    header=TRUE)
#Windows, Mac, and Unix systems
```

```
new.data<-
   read.csv("C:/file/name/data.csv",
   header=TRUE)
```

Inputting a covariance matrix.

In some situations, the user does not have access to raw data, but does have access to a covariance matrix. One option in such cases is to type the entire matrix into **R** using the `matrix()` function, but the user can make use of the fact that covariance matrices are symmetric and use the `diag()`, `upper.tri()`, `lower.tri()` and `t()` (transpose) functions. After entering the matrix data, it is useful to name the variables, which can be done using the `rownames()` and `colnames()` functions. The following syntax creates a correlation matrix titled `CorM` that consists of the correlations among four variables (the default option in **R** is to enter matrix data by columns), and names the columns and rows of the matrix.

```
CovM<-diag(4) #Create a 4 x 4 diagonal
   matrix, with ones on the diagonal
CovM[lower.tri(CovM, diag=FALSE)]<-c(.85,
   .84, .68, .61, .59, .41) #input the
   lower triangle of the matrix, by
   columns
CovM[upper.tri(CorM, diag=FALSE)] <-
   t(CorM)[upper.tri(CorM)] #make matrix
   full
> rownames(CovM)<-colnames(CovM)<-
   c("Var1", "Var2", "Var3", "Var4") #
   Name rows and columns
```

For more information on importing data or using other **R** functions, CRAN has manuals (http://cran.r-project.org/manuals.html) and other documents available (e.g., Paradis, 2005). In addition, there is a growing number of textbooks, both introductory and advanced, devoted to using **R** (e.g., Crawley, 2007; Field, Miles, & Field, 2012), including a whole series of books published by Springer titled *Use R!* (http://www.springer.com/series/6991).

Using R to Conduct a Factor Analysis

To facilitate the explanation of using **R** for factor analysis (FA), this analysis will use the data reported by Canivez, Konold, Collins, and Wilson (2009), who conducted both an exploratory FA (EFA) and confirmatory FA (CFA). They collected data on 152 individuals on two brief tests of cognitive ability: Wechsler Abbreviated Scale of Intelligence (WASI; The Psychological Corporation, 1999) and Wide Range Intelligence Test (WRIT; Glutting, Adams, & Sheslow, 2000). They were interested in examining if the data were better explained by a single factor (i.e., general intelligence, *g*; Jensen, 1998), or two factors (i.e., Fluid Reasoning and Comprehension-Knowledge; Newton & McGrew, 2010). Their summary statistics are reproduced in Appendix 1.

The following syntax will input the Canivez et al. (2009) data into **R**.

```
#make an empty matrix with 8 rows and
  columns
WASIWRIT.cor<-matrix(NA, 8,8)
#input ones on the diagonal elements of
  the matrix
diag(WASIWRIT.cor)<-1
#input the lower triangle of the
  correlation matrix
WASIWRIT.cor [lower.tri(WASIWRIT.cor)]<-
c(.57, .79, .62, .69, .83, .56, .51, .57,
  .65, .51, .54, .59, .66, .60, .70, .74,
  .58,
  .55, .53, .57, .71, .62, .71, .65, .51,
  .58, .53, .62)
#input the upper triangle of the
  correlation matrix
WASIWRIT.cor[upper.tri(WASIWRIT.cor)]<-
  t(WASIWRIT.cor)[upper.tri(WASIWRIT.cor)]
#Name the rows and columns of the
  correlation matrix
dimnames(WASIWRIT.cor)<-
  list(c(paste("WASI.", c("Voc", "BD",
  "Sim", "MR"), sep=""), paste ("WRIT.",
  c("VerbAn", "Voc", "Mat", "Dia"),
  sep="")), c(paste("WASI.", c("Voc",
  "BD", "Sim", "MR"), sep=""),
  paste("WRIT.", c("VerbAn", "Voc", "Mat",
  "Dia"), sep="")))
#Create a vector of means
WASIWRIT.mean<-c(97.75, 97.87, 103.81,
  99.81, 101.51, 100.63, 101.45, 100.64)
#Name the columns of the mean vector
names(WASIWRIT.mean)<-c(paste("WASI.",
  c("Voc", "BD", "Sim", "MR"), sep=""),
  paste("WRIT. ", c("VerbAn", "Voc",
  "Mat", "Dia"), sep=""))
#Create a vector of standard deviations
WASIWRIT.sd<-c(17.37, 14.49, 17.26,
  16.61, 14.77, 16.42, 16.17, 13.92)
```

```
19 #Name the columns of the standard
   deviaitons vector
20 names(WASIWRIT.sd)<-c(paste("WASI.",
    c("Voc", "BD", "Sim", "MR"), sep=""),
    paste("WRIT." , c("VerbAn", "Voc",
    "Mat", "Dia"), sep=""))
```

The paste() function concatenates string arguments, and is useful when there is repetition in strings.

Exploratory Factor Analysis

R has multiple functions that will do factor extraction. As part of **R**'s native packages, the factanal() function will do maximum likelihood extraction. The psych package (Revelle, 2012) has a function, fa(), that will do extraction using either principal axis/factor, maximum likelihood, minres (ordinary least squares), weighted least squares, or generalized weighted least squares methods. Below is syntax for a principal axis extraction of a single factor using the fa() function.

```
1> fa(WASIWRIT.cor, nfactors=1, n.obs=152,
   fm="pa")
2 Standardized loadings (pattern matrix)
   based upon correlation matrix
             3P  A  1h     2u        2
4 WASI.Voc        .84    .71      .29
5 WASI.BD         .73    .54      .46
6 WASI.Sim        .83    .70      .30
7 WASI.MR         .78    .60      .40
8 WRIT.VerbAn     .78    .62      .38
9 WRIT.Voc        .83    .69      .31
10 WRIT.Mat       .77    .60      .40
11 WRIT.Dia       .71    .51      .49
```

The nfactors argument tells the fa() function how many factors to extract, the n.obs arguments gives the sample size (only needed to calculate some fit statistics), and the fm arguments tells the type of extraction to conduct, with "pa" standing for principal axis.

To rotate the extracted factors with the fa() function, use the rotate argument. For example, to replicate Canivez et al.'s (2009) two-factor extraction with varimax rotation, use the following syntax:

```
1 > fa(WASIWRIT.cor, nfactors=2, n.obs=152,
    fm="pa", rotate="varimax")
```

For rotation of the two factors using the promax criterion, use the following syntax.

```
1> fa(WASIWRIT.cor, nfactors=2, n.obs=152,
   fm="pa", rotate="promax")
2 Standardized loadings (pattern matrix)
   based upon correlation matrix
3         PA1    PA2    h2     u2
4 WASI.Voc      .91    .00    .84    .16
5 WASI.BD       .01    .77    .62    .38
6 WASI.Sim      .74    .15    .74    .26
7 WASI.MR       .04    .80    .68    .32
8 WRIT.VerbAn   .64    .20    .63    .37
9 WRIT.Voc      .88    .01    .80    .20
10 WRIT.Mat     .09    .74    .66    .34
11 WRIT.Dia     -.06   .83    .62    .38
12
13 With factor correlations of
14        PA1    PA2
15 PA1    1.00   0.76
16 PA2    0.76   1.00
```

Since promax is an oblique rotation method, the output has the added With factor correlations of section that gives the factor correlations. As oblique rotations cause structure and pattern coefficients to differ, to obtain the structure coefficients, call the Structure object created by the fa() function by either adding $Structure to the end of the syntax or by saving the fa() output as an object and then calling that object with the $Structure suffix.

```
1> fa(WASIWRIT.cor, nfactors=2, n.obs=152,
   fm="pa", rotate="promax")$Structure
2> promax.2fac<-fa(WASIWRIT.cor, nfactors=2,
   n.obs=152, fm="pa", rotate="promax")
3> promax.2fac$Structure
```

Lines two and three are redundant with line one, so only one of the two options needs to be used.

R has the capacity to do more than the "typical" rotations that Canivez et al. (2009) report. The GPArotation package (Bernaards & Jennrich, 2005), for example, which is used in conjunction with the factanal() function allows the user to select from a much larger menu of rotation options. For example, to use a rotation in the Crawford-Ferguson family (Browne, 2001; Crawford &

Ferguson, 1970), use the `factanal()` function with the `rotation` argument being either `cfQ` (for oblique rotations) or `cfT` for orthogonal rotations. By default, the κ parameter is zero. To change the default κ value, use the `kappa` argument within the `control` argument of the `factanal()` function (e.g., for parsimax rotation of a dataset with 10 variables and two factors: `control=list(rotate=list(kappa=(2-1)/(10+2-2))))`.

Below is syntax for a quartimin rotation of Canivez et al.'s (2009) data, using the Crawford-Ferguson (Crawford & Ferguson, 1970) criterion with oblique rotation and setting $\kappa = 0$.

```
1> quartimin<-factanal(covmat=WASIWRIT.cor,
    factors=2, rotation="cfQ",
    control=list(rotate=list(kappa=0)))
```

By default, the `factanal()` function will not print small pattern coefficients. To print all the coefficients, the `factanal()` function needs to be wrapped in the `print()` function with the `cutoff` argument set to some arbitrarily small value (i.e, .001).

Schmid-Leiman Transformation.

Schmid and Leiman (1957) developed a transformation of a higher-order factor model to yield uncorrelated first-order factors that represent both a general (second-order) and more domain specific (first-order) constructs (cf. Yung, Thissen, & McLeod, 1999). This transformation of the factor loadings makes them reflect the incremental influence of both general and specific abilities on the indicator variables. With cognitive ability variables, the Schmid-Leiman (S-L) transformation is usually used to estimate the direct influence of *g* and the first-order constructs on the subtests (e.g., Carroll, 1993). The `psych` package's `schmid()` function will carry out the S-L transformation.

```
1> schmid(WASIWRIT.cor,nfactors=2)
2 Schmid Leiman Factor loadings greater than
    0.2
3                g   F1*  F2*  h2   u2   p2
4 WASI.Voc      .81  .46       .86  .14  .76
5 WASI.BD       .69       .35  .60  .40  .80
6 WASI.Sim      .78  .35       .73  .27  .83
7 WASI.MR       .74       .37  .69  .31  .80
8 WRIT.VerbAn   .73  .29       .62  .38  .85
9 WRIT.Voc      .78  .43       .80  .20  .77
10 WRIT.Mat     .73       .38  .67  .33  .79
11 WRIT.Dia     .68       .39  .61  .39  .75
```

A newer rotation whose purpose is similar to the S-L transformation is the bi-factor rotation (Jennrich & Bentler, 2011), which is another rotation option in the `fa()` function as well as is the `GPArotation` package. It is deigned to extract a general factor as well as domain-specific factors. Thus, for Canivez et al.'s (2009) data, three factors should be extracted instead of two; one for *g* and the other two for Comprehension-Knowledge and Fluid Reasoning factors.

```
1> fa(WASIWRIT.cor, nfactors=3, n.obs=152,
    fm="pa", rotate="bifactor", max.iter =
    500)
2 Standardized loadings (pattern matrix)
    based upon correlation matrix
3              PA1   PA2   PA3   h2    u2
4 WASI.Voc     .84   .37   .00   .85   .15
5 WASI.BD      .66   .04   .52   .71   .29
6 WASI.Sim     .81   .28   .06   .74   .26
7 WASI.MR      .73  -.05   .33   .64   .36
8 WRIT.VerbAn  .79   .14  -.03   .64   .36
9 WRIT.Voc     .83   .32  -.03   .79   .21
10 WRIT.Mat    .93  -.60   .00  1.23  -.23
11WRIT.Dia     .65  -.03   .44   .62   .38
```

As expected, the pattern coefficients from the bi-factor and S-L rotations are quite similar.

Determining the Number of Factors.

There are a variety of ways to determine the number of factors to extract, but best practice suggests that minimum average partial (MAP; Velicer, 1976) criteria and parallel analysis (Horn, 1965) are some of the more robust methods (Velicer & Jackson, 1990). As Costello and Osborne (2005) lament, "Unfortunately [these] methods, although accurate and easy to use, are not available in the most frequently used statistical software and must be calculated by hand" (p. 3). The `psych` package has functions for both methods, though, making factor retention decisions easier for the **R** user than the user of "most frequently used statistical software". Below is syntax for both factor retention methods, using Canivez et al.'s (2009) data.

```
1 #Parallel Analysis
```

```
> fa.parallel(WASIWRIT.cor, n.obs=152,
  fm="pa")
Parallel analysis suggests that the number
  of factors =2      and the number of
  components = 1
```

```
#Velicer's minimum average partial (MAP)
> VSS(WASIWRIT.cor, n.obs=152)
The Velicer MAP criterion achieves a
  minimum of NA      with   2
        factors
Velicer MAP
[1] 0.07 0.06 0.09 0.17 0.32 0.52 1.00 NA
```

Confirmatory Factor Analysis

There are multiple packages that can do structural equation modeling in **R** (e.g., `sem`, `OpenMx`), but this article will focus solely on the `lavaan` (Rosseel, 2012) package. Information and documentation about the package can be found at `http://www.lavaan.org`.

Table 3: `lavaan` *Commands*

Syntax	Command	Example
~	Regress onto	Regress B onto A: `B ~ A`
~~	(Co)varaince	Variance of A: `A ~~ A` Covariance of A and B: `A << B`
~1	Constant/ mean	Regress B onto A, and include the intercept in the model: `B ~1 + A`
~=	Define reflective latent variable	Define Factor 1 by A-D: `F1 ~= A + B + C + D`
:=	Define non-model parameters	Define parameter u2 to be 2 times the square of u: `u2 := 2*(u∋2)`
<~	Define formative variables	Define Variable A by $X_1 - X_4$: `A <~ X1 + X2 + X3 + X4`

To compute a model in `lavaan`, the user first needs to specify the model structure as text, using a few pre-defined commands that are shown in Table 3. Within a model, the user can label parameters by premultiplying the label onto one of the variables on the left hand side of an equation, (e.g., `Factor1 = a*A + b*B + c*C + d*D`).

Before fitting the CFA model with summary data, it wise to convert the correlation matrix to a covariance matrix (Cudeck, 1989). This can be done using the `cor2cov()` function, which is included in the `lavaan` package. It takes a correlation matrix and vector of standard deviations as arguments, and returns the covariance matrix.

```
> library(lavaan)
> WASIWRIT.cov<-cor2cov(WASIWRIT.cor,
   WASIWRIT.sd)
```

Canivez et al. (2009) fit two confirmatory factor analytic models with their data: (*a*) a single factor (i.e., *g*); and (*b*) a two-factor model (i.e., Fluid Reasoning and Comprehension Knowledge). For other possible CFA models, see Beaujean and Parkin (in press).

The `lavaan` syntax for the single factor model is given below.

```
1> singleFactor.model<-'
2 + g=~WASI.Voc + WASI.Sim + WRIT.VerbAn +
    WRIT.Voc + WASI.BD + WASI.MR + WRIT.Mat
    + WRIT.Dia
3 + '
```

Line 1 consists of the name for the model (`singleFactor.model`), the assignment operator (`<-`) and single apostrophe, which indicates that the subsequent syntax will be passed as text. Line 2 defines the single factor structural model, while line 3 is another single apostrophe that tells **R** to stop interpreting the input as text.

Once the model is specified, then the next step is to estimate the parameters. This can be done using either the `cfa()` (for factor analysis models), or `sem()` (for structural equation models) function. The `cfa()` and `sem()` functions are both calls to a more general `lavaan()` function with some of the default arguments specified differently. To obtain the default values for all the `cfa()` or `sem()` functions' arguments, type `?cfa` or `?sem`, respectively, in **R**.

`lavaan` accepts either a covariance matrix (with sample size) or raw data as input, using the `sample.cov=` (with `sample.nobs=`) or `data=` arguments, respectively. If using the covariance matrix as input, the user can (optionally) also input a mean vector using the `sample.mean=` argument. By default, `lavaan` uses normal-theory maximum likelihood as the parameter estimation technique for continuous-variable indicators, but the user can change this to generalized least squares, weighted least squares (ADF), unweighted least squares, or diagonally weighted least squares (for categorical indicators) by specifying them in the `estimator=` argument.

The resulting parameters from the `cfa()` or `sem()` functions are not shown automatically, but can be requested. One such way to request the parameter estimates is by using the `summary()` function. By default, the `summary()` function for `lavaan` objects will produce: (*a*) a note indicating if the minimization algorithm converged; (*b*) the sample size; (*c*) estimator; (*d*) χ^2; (*e*) χ^2 degrees of freedom; (*f*) *p*-value of the χ^2; (*g*) the unstandardized parameter estimates; (*h*) the parameter estimates' standard errors; (*i*) the ratio of the parameter estimates to their standard errors; and (*j*) the that ratio's *p*-value. The `summary()` function has these default specifications for its arguments:

`standardized = FALSE, fit.measures = FALSE, rsquare = FALSE, modindices = FALSE`. Setting `standardized = TRUE` will include standardized estimates in the results, setting `fit.measures = TRUE` will include various fit indices in the results, setting `rsquare = TRUE` will include the R^2 for each exogenous variable, and setting `modindices = TRUE` will include modification indices.

The default option in `lavaan` is to set the pattern coefficient for the first indicator equal to one and estimate the latent variable's variance. One way to override that default is to use the `std.lv=TRUE` argument, which sets the variance of all the latent variable(s) to one, and estimates each factor pattern coefficient. The syntax for fitting Canivez et al.'s (2009) single factor model, as well as obtaining the results using the `summary()` function, are given below. The actual output is given in Appendix A.

```
singleFactor.fit<-cfa(singleFactor.model,
   sample.cov=WASIWRIT.cov,
   sample.nobs=152, std.lv=TRUE)
summary(singleFactor.fit,
   fit.measures=TRUE, standardized=TRUE,
   rsquare=TRUE)
```

Canivez et al.'s (2009) two factor model is specified in `lavaan` using the following syntax.

```
twoFactor.model<-'
Gc=~WASI.Voc + WASI.Sim + WRIT.VerbAn +
   WRIT.Voc
Gv=~WASI.BD + WASI.MR + WRIT.Mat +
   WRIT.Dia
'
twoFactor.fit<-cfa(twoFactor.model,
   sample.cov=WASIWRIT.cov,
   sample.nobs=152, std.lv= TRUE)
summary(twoFactor.fit, fit.measures=TRUE,
   standardized=TRUE, rsquare=TRUE)
```

Within rounding error, the results (i.e., standardized pattern coefficients, fit statistics) from `lavaan` are identical to those reported by Canivez et al. (2009).

Conclusion

With the many steps involved in factor analysis (FA), it can be difficult finding a program that does everything that one may desire. While **R** (R Development Core Team, 2011) does not have a function for everything involved in FA, it does have many, including those for extraction, rotation, determination of the number of factors, as well as confirmatory methods. This, alone, sets it apart **R** from many other programing options currently available. What makes **R** even more powerful, though, is that because its syntax is open source, the user can write his/her own functions to conduct any analyses not currently available in any **R** package. When added to fact that the only cost involved in using **R** is the time it takes to learn the language, it is easy to see why Kelley et al. (2008) write that "there is no time like the present to begin incorporating **R** into one's set of statistical tools" (p. 569).

References

Beaujean, A. A. (2012). BaylorEdPsych: R package for Baylor University Educational Psychology quantitative courses (Version 0.5) [Computer software]. Waco, TX: Baylor University.

Beaujean, A. A., & Parkin, J. (in press). Using R for the analysis of cognitive abilities and behavior genetic data.

In J. Kush (Ed.), *Intelligence quotient: Testing, role of genetics and the environment and social outcomes*. New York: Nova Science.

Bernaards, C. A., & Jennrich, R. I. (2005). Gradient projection algorithms and software for arbitrary rotation criteria in factor analysis. *Educational and Psychological Measurement* , *65* (5), 676-696. doi: 10.1177/0013164404272507

Browne, M. W. (2001). An overview of analytic rotation in exploratory factor analysis. *Multivariate Behavioral Research*, *36* (1), 111-150. doi: 10.1207/s15327906mbr3601_05

Canivez, G. L., Konold, T. R., Collins, J. M., & Wilson, G. (2009). Construct validity of the Wechsler Abbreviated Scale of Intelligence and Wide Range Intelligence Test: Convergent and structural validity. *School Psychology Quarterly*, *24* (4), 252-265. doi: 10.1037/a0018030

Carroll, J. B. (1993). *Human cognitive abilities: A survey of factor-analytic studies*. New York, NY: Cambridge University Press.

Costello, A. B., & Osborne, J. W. (2005). Best practices in exploratory factor analysis: Four recommendations for getting the most from your analysis. *Practical Assessment Research Evaluation*, *10* (7), 1-9.

Crawford, C., & Ferguson, G. (1970). A general rotation criterion and its use in orthogonal rotation. *Psychometrika*, *35* (3), 321-332. doi: 10.1007/bf02310792

Crawley, M. J. (2007). *The R book*. Hoboken, NJ: Wiley.

Cudeck, R. (1989). Analysis of correlation matrices using covariance structure models. *Psychological Bulletin*, *105* (2), 317-327. doi: 10.1037/0033-2909.105.2.317

Field, A., Miles, J., & Field, Z. (2012). *Discovering statistics using R*. Thousand Oaks, CA: Sage.

Fox, J. (2005). The R Commander: A basic-statistics graphical user interface to R. *Journal of Statistical Software*, *14* (9).

Glutting, J., Adams, W., & Sheslow, D. (2000). *Wide range intelligence test*. Wilmington, DE: Wide Range.

Horn, J. (1965). A rationale and test for the number of factors in factor analysis. *Psychometrika*, *30* (2), 179-185. doi: 10.1007/bf02289447

Horton, N. J., Brown, E. R., & Qian, L. (2004). Use of R as a toolbox for mathematical statistics exploration. *The American Statistician*, *58* (4), 343-357. doi: 10.1198/000313004X5572

Jennrich, R., & Bentler, P. (2011). Exploratory bi-factor analysis. *Psychometrika*, *76* (4), 537-549. doi: 10.1007/s11336-011-9218-4

Jensen, A. R. (1998). *The g factor: The science of mental ability*. Westport, CT: Praeger Publishers/Greenwood.

Kelley, K., Lai, K., & Wu, P.-J. (2008). Using R for data analysis: A best practice for research. In J. W. Osborne (Ed.), *Best practices in quantitative methods* (p. 535-572). Thousand Oaks, CA: Sage.

McCammon, R. B. (1966). Principal component analysis and its application in large-scale correlation studies. *Journal of Geology*, *74* (5-2), 721-733.

Newton, J. H., & McGrew, K. S. (2010). Introduction to the special issue: Current research in Cattell-Horn-Carroll-based assessment. *Psychology in the Schools*, *47* (7), 621-634. doi: 10.1002/pits.20495

Paradis, E. (2005). *R for beginners*. Montpellier, France: Universite Montpellier II. Retrieved from http://cran.r-project.org/doc/contrib/Paradis-rdebuts_en.pdf

Pashler, H., & Wagenmakers, E. (2012). EditorsâĂŹ introduction to the special section on replicability in psychological science: A crisis of confidence? *Perspectives on Psychological Science*, *7* (6), 528-530. doi: 10.1177/1745691612465253

R Development Core Team. (2011). *R: A language and environment for statistical computing*. Vienna, Austria: R Foundation for Statistical Computing. Available from http://www.R-project.org .

Revelle, W. (2012). *psych: Procedures for psychological, psychometric, and personality research (Version 1.2.4) [Computer software]*. Evanston, IL: NorthwesternUniversity.

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

Schmid, J., & Leiman, J. (1957). The development of hierarchical factor solutions. *Psychometrika*, *22* (1), 53-61. doi: 10.1007/bf02289209

The Psychological Corporation. (1999). *Wechsler abbreviated scale of intelligence*. San Antonio, TX: Author.

Vance, A. (2009, Jan 6). Data analysts captivated by R's power. *New York Times*. Retrieved from http://www.nytimes.com/2009/01/07/technology/business-computing/07program.html/?pagewanted=all

Velicer, W. F. (1976). Determining the number of components from the matrix of partial correlations. *Psychometrika*, *41* (3), 321-327. doi: 10.1007/bf02293557

Velicer, W. F., & Jackson, D. N. (1990). Component analysis versus common factor analysis: Some issues in selecting an appropriate procedure. *Multivariate Behavioral Research*, *25* (1), 1-28. doi: 10.1207/s15327906mbr2501_1

Venables, W. N., Smith, D. M., & R Development Core Team. (2012). *An introduction to R*. R Development Core Team. Retrieved from http://cran.r-project.org/doc/manuals/R-intro.pdf

Yung, Y.-F., Thissen, D., & McLeod, L. D. (1999). On the relationship between the higher-order factor model and the hierarchical factor model. *Psychometrika*, *64* (2), 113-128. doi: 10.1007/bf02294531.

Appendix 1

Wechsler Abbreviated Scale of Intelligence and Wide Range Intelligence Test Summary Statistics

	WASI. Voc	WASI. BD	WASI. Sim	WASI. MR	WRIT. VerbAn	WRIT. Voc	WRIT. Mat	WRIT. Dia
WASI.Voc	1.00	0.57	0.79	0.62	0.69	0.83	0.56	0.51
WASI.BD		1.00	0.57	0.65	0.51	0.54	0.59	0.66
WASI.Sim			1.00	0.60	0.70	0.74	0.58	0.55
WASI.MR				1.00	0.53	0.57	0.71	0.62
WRIT.VerbAn					1.00	0.71	0.65	0.51
WRIT.Voc						1.00	0.58	0.53
WRIT.Mat							1.00	0.62
WRIT.Dia								1.00
Mean	97.75	97.87	103.81	99.81	101.51	100.63	101.45	100.64
SD	17.37	14.49	17.26	16.61	14.77	16.42	16.17	13.92

Note. Data taken from Canivez et al. (2009, p. 257). WASI: Wechselr Abbreviated Scale of Intelligence; WRIT: Wide Range Intelligence Test; Voc: Vocabulary; BD: Block Design; Sim: Similarities; MR: Matrix Reasoning; VerbAn: Verbal Analogies; Mat: Matrices; Dia: Diamonds.

Appendix 2

lavaan Results from the Single-Factor Model

```
lavaan (0.5-9) converged normally after    61 iterations
Number of observations      152
Estimator      ML
Minimum Function Chi-square 121.483
Degrees of freedom    20
P-value 0.000
 Chi-square test baseline model:
Minimum Function Chi-square 918.115
Degrees of freedom    28
 P-value 0.000
 Full model versus baseline model:
 Comparative Fit Index (CFI)  0.886
 Tucker-Lewis Index (TLI)     0.840
Loglikelihood and Information Criteria:
 Loglikelihood user model (H0)       -4681.416
 Loglikelihood unrestricted model (H1)      -4620.674
 Number of free parameters   16
 Akaike (AIC)   9394.832
 Bayesian (BIC) 9443.214
 Sample-size adjusted Bayesian (BIC) 9392.574
 Root Mean Square Error of Approximation:
 RMSEA   0.183
 90 Percent Confidence Interval     0.152  0.215
 P-value RMSEA <= 0.05 0.000
```

```
Standardized Root Mean Square Residual:
 SRMR    0.068

                    Estimate  Std.err Z-value P(>|z|)  Std.lv Std.all
Latent variables:
 g =~
 WASI.Voc         15.192     1.122     13.538     0.000     15.192     0.877
 WASI.Sim         14.701     1.134     12.967     0.000     14.701     0.855
 WRIT.VerbAn      11.735     1.008     11.645     0.000     11.735     0.797
 WRIT.Voc         14.088     1.074     13.120     0.000     14.088     0.861
 WASI.BD          10.107     1.043      9.686     0.000     10.107     0.700
 WASI.MR          12.284     1.170     10.498     0.000     12.284     0.742
 WRIT.Mat         11.866     1.143     10.384     0.000     11.866     0.736
 WRIT.Dia          9.387     1.014      9.261     0.000      9.387     0.677
Variances:
 WASI.Voc         68.947     0.264                          68.947     0.230
 WASI.Sim         79.832    11.241                          79.832     0.270
 WRIT.VerbAn      79.000    10.282                          79.000     0.365
 WRIT.Voc         69.380     9.898                          69.380     0.259
 WASI.BD         106.433    13.057                         106.433     0.510
 WASI.MR         123.169    15.409                         123.169     0.449
 WRIT.Mat        118.942    14.834                         118.942     0.458
 WRIT.Dia        104.381    12.698                         104.381     0.542
 g                 1.000                                     1.000     1.000
```

Citation:

Beaujean, A. Alexander (2013). Factor Analysis using R. *Practical Assessment, Research & Evaluation*, 18(4). Available online: http://pareonline.net/getvn.asp?v=18&n=4

Author:

A. Alexander Beaujean
Department of Educational Psychology
Baylor University
One Bear Place #97301
Waco, TX 76798-7301
Alex_Beaujean [at] Baylor.edu