

# An R Interface to EQS: The REQS Package

**Patrick Mair**  
Wirtschaftsuniversität Wien

**Eric Wu**  
University of California, Los Angeles

**Peter M. Bentler**  
University of California, Los Angeles

---

## Abstract

In this paper we present the **REQS** package which is an interface between the R environment of statistical computing and the **EQS** software for structural equation modeling. The package consists of three main functions that read **EQS** script files and import the results into R, call **EQS** script files from R, and, finally, run **EQS** script files from R and, again, import the results after **EQS** computation. The components a user needs are R and **EQS**. We give a short introduction to R and **EQS**, elaborate the functionalities of the package, and show how to use the package by means of several examples with special emphasis on simulations.

*Keywords:* **EQS2R**, R, **EQS**.

---

## 1. Introduction

Over the last years R ([R Development Core Team 2008](#)) has become the “lingua franca” among computational statisticians. Currently, R is also becoming more popular outside the statistical community in fields such as psychology, sociology, medicine, education, economics, and many others. In fields related to psychometrics, including factor analytical models and SEM, item response models, techniques of correspondence analysis and dimension reduction, as well as classical test theory, a vast number of packages has been developed during the last few years (see [Mair and Hatzinger 2007](#), for an overview). Recently, the *Journal of Statistical Software* (JSS) published a special issue on psychometrics in R (see [de Leeuw and Mair 2007](#)). We can state the following reasons why R has become so popular during recent years:

- Flexibility: All the outputs and results are stored as objects (vectors, matrices, lists, etc.). The user can process the results in terms of further computations and graphical representations.
- Consistency: All packages including their help files are of the same structure.
- Plot engine: The R plot engine allows for highly customizable graphical visualizations at the publication level (see [Murrell 2005](#)).
- Community: The R community is very active in developing new packages and in discussing and solving various problems and questions in the R help forum.
- Open source: R is published under the GNU General Public License (GPL). It is freely available and the user has full insight into the source code.
- Platform independence: R runs under Windows, MAC OS, and Linux.

- Programming language: The underlying programming language S (Becker, Chambers, and Wilks 1988; Chambers 1998) is strictly vector and matrix oriented and therefore highly suited for multivariate methods.

Of course, if somebody wants to use the S syntax in its entirety, the learning curve is rather steep. But basic things such as extracting objects from package outputs and further processing these results can be learned quickly. In addition, a graphical user interface like the package **Rcmdr** (Fox 2008) makes R is an important didactical tool to make R more accessible to students and researchers outside the area of statistics. Popular introductory R books are Venables and Smith (2002), Everitt and Hothorn (2006) and Braun and Murdoch (2008). Fox (2006) gives a brief introduction to structural equation modeling in R.

We conclude our Introduction by giving some useful R links:

- The R project for statistical computing: <http://www.r-project.org/>
- CRAN main repository (packages, news, downloads, etc.): <http://cran.r-project.org/>
- Online manuals: <http://cran.r-project.org/manuals.html>
- Psychometrics task view: <http://cran.r-project.org/web/views/Psychometrics.html>
- Social Sciences task view: <http://cran.r-project.org/web/views/SocialSciences.html>
- Mailing lists: <http://www.r-project.org/mail.html>
- R help forum: <http://www.nabble.com/R-help-f13820.html>
- Wiki: <http://wiki.r-project.org>
- R-forge development platform: <http://r-forge.r-project.org/>
- Editor (optional, for Windows only): <http://www.sciviews.org/Tinn-R/>

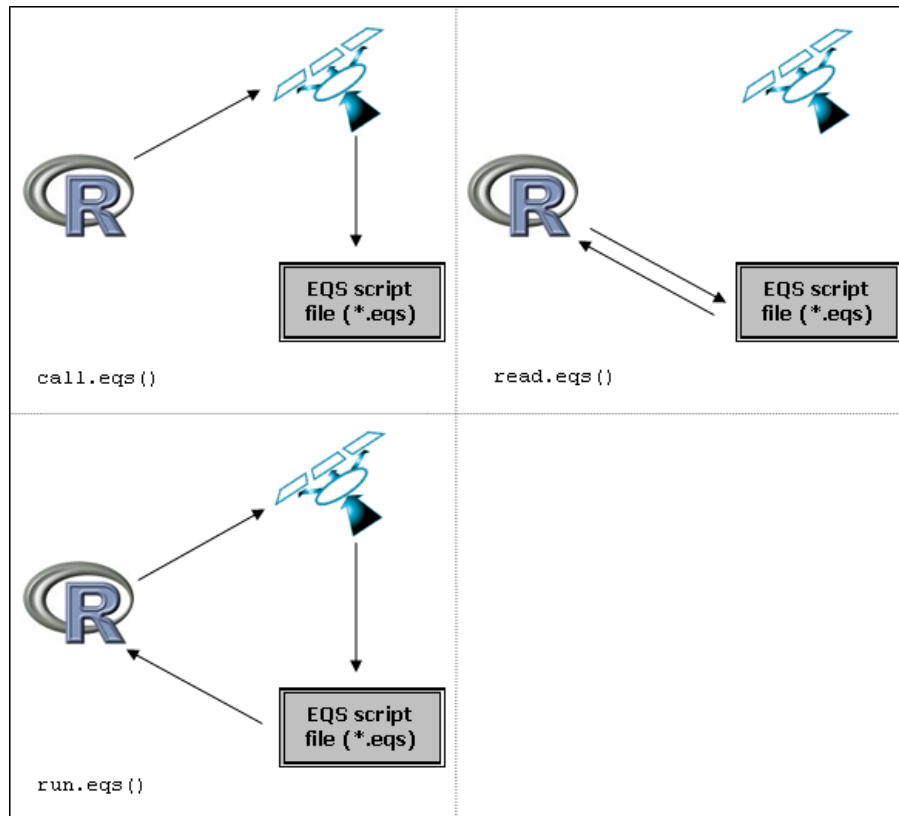
The packages on CRAN are grouped according to specific fields of research (so called “Task Views”). SEM researchers might be interested in exploring the Social Sciences and the Psychometrics task view. The links are given above.

## 2. The package REQS

### 2.1. SEM in R

Before we focus on the presentation of the package we discuss the *status quo* of SEM in R. The **sem** package (Fox 2006) is the main R package for fitting SEM. It uses the reticular action model (RAM) formulation for model specification and estimates the parameters by maximum likelihood, based on the multinormality assumption. The parameter estimates including standard errors, model  $\chi^2$ , (adjusted) GFI, RMSEA, NFI, NNFI, CFI, SRMR, and BIC are returned. The main function called **sem()** takes the correlation matrix as data input argument. Having ordinal or mixed scale data, the **polycor** package (Fox 2007) can be used to compute the polychoric-polyserial correlation matrix by means of the function **hetcor()**.

Compared to EQS (Bentler 2006; Bentler and Wu 2002) and other available SEM software, several important SEM approaches are limited or not implemented in the **sem** package. It allows only ML estimation based on underlying multinormality. In addition, EQS provides LS, GLS, elliptical, heterogeneous kurtosis, ADF, and a robust ML approach suited for non-normal data, as well as statistically correct correlation structure methods including categorical variable handling based on polychoric and polyserial correlations. Robust standard errors based on a sandwich estimator are also available in EQS for both complete and incomplete data. In terms of model testing

Figure 1: Workflow of **REQS** functions

and evaluation, EQS offers univariate and multivariate Wald and LM-tests as well a variety of additional fit indices such as robust MFI and IFI. Furthermore, additional model types such as multigroup, multilevel, latent growth, and mixture approaches are implemented. Finally, a model to be run can be specified by means of a graphical user interface.

Thus, EQS offers several additional important SEM features and R implements a huge number of available statistical methods. Therefore, an interface between EQS and R that connects both environments is highly valuable. As an example, we mention simulation studies. R is very powerful for data simulation and for each simulated data set the user can call EQS, estimate the parameters there, import any desired associated results as back into R, and store them as R objects. These objects can be processed further in terms of additional manipulations, statistical analyses, or graphical visualizations.

## 2.2. Package description

The **REQS** package is available from the CRAN repository. Installation and how-to-use guidelines will be given in the next section. In this section we describe briefly the structure and the main functions of the package. Corresponding examples are given in Section 3.2 and more technical details about the functions can be found in the corresponding R help files.

Basically, **REQS** provides three functions. The work flow for each function is given in Figure 1. The first function, given in the top left corner, calls EQS and executes the specified script file. The results are stored in a EQS output file without importing them into R. The corresponding syntax is

```
> call.eqs(EQSpgm, EQSmodel, serial, data = NA, datname = NA, LEN = 2e+06)
```

with the following arguments: `EQSpgm` (path where EQS program is installed), `EQSmodel` (path where `.eqs` script file is located), `serial` (EQS serial number), `data` (optional matrix argument if data or covariances are stored in R), `datname` (optional filename for saving data), `LEN` (number of working array units; by default it is set 2000000 8 bytes units). Optionally, having the data (correlation or covariance matrix) in R, the user can assign these data to the script file. It is important that the argument `datname` and the file name quoted in the `.ets` file (see Appendix) match.

The second function reads EQS output files (i.e., `.ets`, `.CBK` and `.ETP` which are provided by EQS) into R and stores the results as objects (work flow see Figure 1, top right). EQS is not called by this function.

```
> read.eqs(file)
```

It contains only one argument which is the file name of the `.ets` file.

Basically, what `read.eqs()` does is to parse the `.ets` file using the meta information in `.CBK` and `.ETP` and, consequently, to import these results into R.

The following R objects (object names in parenthesis) are created: General model information (`model.info`),  $p$ -values for test statistics (`pval`), fit indices (`fit.indices`), descriptive measures (`model.desc`), Phi matrix (`Phi`), Gamma matrix (`Gamma`), Beta matrix (`Beta`), parameter table (with standard errors) (`par.table`), sample covariance matrix (`sample.cov`), model covariance matrix (`sigma.hat`), inverse information matrix (`inv.infmat`), robust inverse information matrix (`rinv.infmat`), corrected inverse information matrix (`cinv.infmat`), first derivatives (`derivatives`), 4th moments weight matrix (`moment4`), standardized elements (`ssolution`), R-squared measures (`Rsquared`), factor means (`fac.means`), descriptive variable measures (univariate statistics) (`var.desc`), independent variable standardization vector (`indstd`), dependent variable standardization vector (`depstd`). Note that if any of these outputs is not provided in the `.ets` file, the corresponding R object becomes NA (missing).

The third function basically combines `call.eqs` and `read.eqs` (work flow see Figure 1 bottom):

```
> run.eqs(EQSpgm, EQSmodel, serial, data = NA, datname = NA, LEN = 2e+06)
```

It calls the `.ets` script file, EQS does the computation, and imports the output as R objects. The arguments are the same as in `call.eqs()` and the returning values are the same R objects as in `read.eqs()`.

## 3. How to use REQS

### 3.1. Installing and loading the package

First of all, R and EQS have to be installed. The R installation can be carried out by going to <http://cran.r-project.org/> where at the top the user can find “Download and Install R”. By choosing the corresponding operating system, installation instructions will be given. At this point we give a special advice for Windows users: When executing the `R-x.x.x-win32.exe` select “Customized Installation” and then the check box referring to *SDI*. If the user decides to work with the TINN-R editor (see link list above), an interface between TINN-R and R is established. This allows, for instance, to write the R script in TINN-R and send it (stepwise) to R. TINN-R offers many additional nice features such that we suggest this editor for Windows users. In addition we strongly recommend that the users always have the the most recent R version installed. To get notified about R updates, users can subscribe to the corresponding mailing list (see link list in the Introduction).

Once R is installed and started, the user has to install and load the **REQS** package:

```
> install.packages("REQS")
> library("REQS")
```

The first command downloads the package from CRAN and installs it on the hard disk. The second command loads the package into the R workspace. Note that each time R is started, the package has to be loaded.

### 3.2. Examples - Reading, calling, running EQS from R

The easiest way to apply all the **REQS** functions is to change the R working directory into the folder where your EQS output/script files are located. This can be done with the command `setwd()`. For instance, under Windows this could look like

```
> setwd("C:/EQS61/examples/")
```

Note that R requires forward slashes instead of back slashes.

Throughout this section we will use “manul7” example that is included in the EQS basic installation. Let us start with the import of EQS results contained in the `manul7.eqs` output file.

```
> out <- read.eqs("manul7.ets")
```

Note that if the directory is not changed as indicated above, the user has to specify the full path where `manul7.eqs` is located. At this point `out` is an R list object. By typing

```
> names(out)
```

```
[1] "model.info"  "pval"          "fit.indices"  "model.desc"  "Phi"
[6] "Gamma"      "Beta"          "par.table"    "sample.cov"  "sigma.hat"
[11] "inv.infmtat" "rinv.infmtat" "cinv.infmtat" "derivatives" "moment4"
[16] "ssolution"   "Rsquared"     "fac.means"   "var.desc"    "indstd"
[21] "depstd"
```

we get all the names elements of the list which can be a single number, matrices, data frames, vectors, characters, or lists again. List elements can be accessed by means of the `$` symbol. For instance,

```
> parameter <- out$par.table
> parameter
```

	Parameter	SE	RSE	CSE	Gradient
(F1,F2)	0.4126106	NA	NA	NA	NA
(E1,E1)	0.6702828	NA	NA	NA	NA
(E2,E2)	0.7206135	NA	NA	NA	NA
(E3,E3)	0.2888152	NA	NA	NA	NA
(E4,E4)	0.6373565	NA	NA	NA	NA
(E5,E5)	0.2010225	NA	NA	NA	NA
(E6,E6)	0.5536968	NA	NA	NA	NA
(V1,F1)	0.6916699	NA	NA	NA	NA
(V2,F1)	0.5089503	NA	NA	NA	NA
(V3,F1)	1.0244305	NA	NA	NA	NA
(V4,F2)	0.4271626	NA	NA	NA	NA
(V5,F2)	0.8078310	NA	NA	NA	NA
(V6,F2)	0.4552483	NA	NA	NA	NA

gives us a data frame containing parameter estimates and standard errors. Note that because in this example no standard errors were computed, the corresponding values are NA (i.e. missing).

If a researcher is interested in the path coefficients only (i.e., the elements from the 8th to the 13th row, first column) he can extract them by doing

```
> VFpar <- parameter[8:13, 1]
> VFpar

      (V1,F1)  (V2,F1)  (V3,F1)  (V4,F2)  (V5,F2)  (V6,F2)
0.6916699 0.5089503 1.0244305 0.4271626 0.8078310 0.4552483
```

This returns a vector. Other output information can be extracted and further processed in an analogous manner.

The second function is `call.eqs()` which calls EQS from R. The arguments were described in Section 2.2. Our starting point is the EQS script file `manul7.eqs`. Additional information regarding the script file can be found in the Appendix.

```
> call.eqs(EQSpgm = '"C:/Program Files/EQS61/WINEQS"', EQSmodel = "manul7.eqs",
+         serial = 1234)
```

This command calls EQS, does the computation based on the specified script file in EQS, and writes the corresponding `.out`, `.ets`, `.ets`, and `.etp` file to the disk. There is one important issue regarding the EQS path specification we have to point out: If this path contains a blank (as above), the user must provide single quote - double quote - pathname - double quote - single quote. The additional single quote should not be provided if the path contains no blank (e.g. `"C:/Program Files/EQS61/"`). The same applies to the `EQSmodel` argument if the user does not change the R working directory.

Now assume that the user has the data in R rather than in a `.ess` file. As an example we simulate a data matrix `X`.

```
> X <- matrix(rnorm(300), ncol = 6)
> res.run <- run.eqs(EQSpgm = '"C:/Program Files/EQS61/WINEQS"',
+   EQSmodel = "manul7X.eqs", serial = 1234, Rmatrix = X, datname = "mydata.dat")
```

The `Rmatrix` argument refers to this new data matrix. By means of the `datname` argument we specify the filename of the dataset for saving it to the hard disk. It is important that this file name matches the specification in the script file. In our example we can simply change `manul7.eqs` to `manul7X.eqs`: By modifying the data line from `DATA='MANUL7.ESS'` to `DATA='mydata.dat'`. The whole script files is given in Appendix A. This calls the new data file `mydata.dat` generated within `run.eqs()`.

Consequently, `manul7X.out`, `manul7X.ets`, `manul7X.cbk`, and `manul7X.etp` are created. The R object `res.call` becomes 0 if the estimation was successful, 1 otherwise. Note that the data can also be a correlation or covariance matrix, but, in this case, the specification has to be given in the `.eqs` file.

Finally we present the function `run.eqs()` which combines both working tasks. This is especially helpful if the user wants to carry out simulation studies. We will give a small example (100 replications, extracting CFI each time):

```
> cfivec <- rep(NA, 100)
> for (i in 1:100) {
+   X <- matrix(rnorm(300), ncol = 6)
+   out <- run.eqs(EQSpgm = '"C:/Program Files/EQS61/WINEQS"',
```

```
+      EQSmodel = "manul7X.eqs", serial = 1234, Rmatrix = X,
+      datname = "mydata.dat")
+      cfivec[i] <- out$fit.indices["CFI", ]
+ }
```

The first line initializes the CFI vector. Within each replication  $i$  we randomly create a data matrix  $X$  and perform the 2-factor model specified in `manul7X.eqs`. After EQS is done with the estimation, this function automatically imports the results (cf. `read.eqs()`) and stores them as list. In the last line we access the data frame of fit indices and in particular the line which contains the CFI. Now, for instance, the user can create a histogram based on the CFIs.

Of course, the extraction of the CFIs is only one possibility the user has within each run. Basically, all EQS output is organized internally as R objects such that the user can access them easily and study the behavior across the simulation runs.

## 4. Discussion

We presented the **REQS** package that allows for a bilateral communication between R and EQS. The benefit of this package concerns both “worlds”: From the R perspective we can state that it extends the SEM modeling options for R users, given that EQS is available. From the EQS perspective it is obvious that the whole spectrum of statistical methods provided by R is now available to SEM researchers. They can perform additional analyses and research on the SEM results, run simulations, plot results in a highly-customized manner, etc. Having such an integrated framework for SEM modeling vastly facilitates methodological developments in the area of SEM and, for practitioners, it provides a comprehensive environment for post-processing, representing, and visualizing their results.

## References

- Becker RA, Chambers JM, Wilks AR (1988). *The New S Language: A Programming Environment for Data Analysis and Graphics*. Wadsworth, Pacific Grove, CA.
- Bentler PM (2006). *EQS 6 Structural Equations Program Manual*. Multivariate Software, Inc., Encino, CA.
- Bentler PM, Raykov T (2000). “On measures of explained variance in nonrecursive structural equation models.” *Journal of Applied Psychology*, **85**, 125–131.
- Bentler PM, Wu E (2002). *EQS 6 for Windows User’s Guide*. Multivariate Software, Inc., Encino, CA.
- Bentler PM, Yuan K (2000). “On adding a mean structure to a covariance structure model.” *Educational and Psychological Measurement*, **60**, 326–339.
- Braun WJ, Murdoch DJ (2008). *A First Course in Statistical Programming with R*. Cambridge University Press, Cambridge, MA.
- Chambers JM (1998). *Programming with Data: A Guide to the S Language*. Springer, New York.
- de Leeuw J, Mair P (2007). “An Introduction to the Special Volume on “Psychometrics in R”.” *Journal of Statistical Software*, **20**(1), 1–5.
- Everitt BS, Hothorn T (2006). *A Handbook of Statistical Analyses Using R*. Chapman & Hall/CRC, Boca Raton, FL.

Fox J (2006). “Structural Equation Modeling With the sem Package in R.” *Structural Equation Modeling*, **13**, 465–486.

Fox J (2007). *polycor: Polychoric and Polyserial Correlations*. R package version 0.7-5.

Fox J (2008). *Rcmdr: R Commander*. R package version 1.4-0, URL <http://www.r-project.org>.

Mair P, Hatzinger R (2007). “Psychometrics Task View.” *R News*, **7/3**, 38–40.

Murrell P (2005). *R Graphics*. Chapman & Hall/CRC, Boca Raton, FL.

R Development Core Team (2008). *R: A Language and Environment for Statistical Computing*. R Foundation for Statistical Computing, Vienna, Austria. ISBN 3-900051-07-0, URL <http://www.R-project.org>.

Venables WN, Smith DN (2002). *An Introduction to R*. Network Theory, Ltd., Bristol, UK.

## Appendix A - EQS script file for external data

Here we give the `manul7X.eqs` example from Section 3.2. The crucial lines are both DATA statements. In line 6 we have to refer to the file (of the whole path) where R stores the matrix  $X$ . In the second to last line we quote the file where the results from EQS computation are stored. In our case, `manul7X.ets`, `manul7X.CBK`, and `manul7X.ETP` are created.

```

/TITLE
SIMULATED CONFIRMATORY FACTOR ANALYSIS EXAMPLE (EXAMPLE IN EQS MANUAL P.117)
RAW SCORES IN BENTLER (1985, P.105)
DEFAULT START VALUES
/SPECIFICATIONS
CASES = 50; VARIABLES = 6; ME = ML, ROBUST;
MA = RA; DATA='mydata.dat';
/EQUATIONS
V1 = *F1 + E1;
V2 = *F1 + E2;
V3 = *F1 + E3;
V4 = *F2 + E4;
V5 = *F2 + E5;
V6 = *F2 + E6;
/VARIANCES
F1 TO F2 = 1;
E1 TO E6 = *;
/COVARIANCE
F1,F2 = *;
/OUTPUT
LISTING;
PARAMETER;
DATA='manul7X.ets';
/END

```

## Appendix B - Creating the EQS script file

A prerequisite to using **EQS2R** is to have a license to use EQS and to know how to construct an EQS model script. EQS provides two methods, namely Diagrammer and Equation Tables, to assist you to build an EQS model. The step by step procedures on how to use Diagrammer and Equation Tables can be found in the EQS 6.1 User's Guide.

In this Appendix, we will concentrate how to prepare EQS internal information so that it can be exported to R data objects. Let's use following model as an illustrated example.

```

/TITLE
PATH ANALYSIS MODEL
/SPECIFICATIONS
CAS=932; VAR=6; ME=ML;           ! ML = maximum likelihood
DATA='MANUL4.ESS';             ! Input covariance matrix in external file
/LABEL                          ! Optional
V1 = ANOMIE67; V2 = POWRLS67;   ! At most 8 characters per label
V3 = ANOMIE71; V4 = POWRLS71;
/EQUATIONS
V3 = 1*V1 + 1*V2 + E3;          ! Equations with labeled vars ok
V4 = 1*V1 + 1*V2 + E4;          ! No labels possible for Es
/VARIANCES
V3= 10*; V4 = 10*;
E3 - E4 = 2*;                   ! Dash (-) or T0 sets several in sequence
/COVARIANCES
V2,V1 = 7*;
/OUTPUT
DATA='PATH.ETS';
LISTING;
(OTHER OUTPUT OPTIONS)
/END

```

As illustrated in the example, an `/OUTPUT` section is inserted into a regular EQS model script. This section is designed to create EQS technical output in a compact form and with greater precision. If `/OUTPUT` is requested, minimal output is put into the standard output file (an echo of the model, a guide to technical output that will be produced, and format of the technical output), while the specifically requested output is stored in an external file that can be easily accessed by other programs for further analysis. The keywords that can be specified in `/OUTPUT` are `DATA`, `LISTING`, and a variety of specific information as given below.

The file name where technical output will be stored is given as

```
DATA = 'file name';
```

where file name is any appropriate name in the computer system. If no file name is given, `EQSOUT.ETS` will be used as a default. No format is needed; the output file will be in free format.

The listing option controls only the regular EQS output, and not the technical output requested in `/OUTPUT`. When this keyword is not given, the regular output will be turned off, and only an abbreviated output will be produced. When this keyword is given, the regular EQS output will also be produced. The format is simply to state the name `LISTING`. An example that includes some specific output items (see below) is:

```

/OUTPUT
Listing; parameter estimates; standard errors;

```

In order to be more precise to describe each part of information to be exported, we use the following

Keyword	Abbreviation	Default	Meaning
DATA	DA = 'XY.ETS';	EQSOUT.ETS	File where technical output will be stored
LISTING	LI;	Short LOG file	Regular LOG file produced
ALL	AL;		Turns on all options below
PARAMETERS	PA;		Parameter estimates
STANDARD or SE	ST;		Standard errors of parameters
GRADIENTS	GR;		Derivatives of minimized function w.r.t. parameters
COVARIANCE	CO;		Sample covariance matrix
SIGMA	SI;		Model covariance matrix
INVERSE	IN;		Inverted information matrix
DERIVATIVES	DE;		Derivatives of model covariance matrix w.r.t. parameters
WEIGHT	WE;		AGLS weight matrix
SS	SS;	Standardized solution	
RSQUARE	RS;	Bentler-Raykov R-squares	
FMEANS	FM;	Factor means	
UNIVARIATE	UN;	Univariate statistics for all variables	

notation:  $p$  is the number of measured variables,  $f$  is the number of factors, and  $q$  is the number of parameters. In addition we define  $p^* = p(p + 1)/2$ . We get the following output:

- **P**arameter estimates provides the vector  $\hat{\theta}$  at the solution. It is a vector of  $q$  elements.
- **S**tandard errors give the estimated sampling variability of  $\hat{\theta}$ , based on the covariance matrix of parameter estimates as defined in “**I**nverted information matrix”, below. When **ME = xx**, **R**obust is used, two sets of standard errors are provided. Robust standard errors follow those of **ME = xx**. It is a vector of  $q$  elements.
- **G**radient elements are  $\partial Q/\partial \theta$  for the particular function chosen. For **ML**, the elements are  $\partial Q/\partial \theta$ . Gradient is also a vector with  $q$  elements.
- **C**ovariance matrix is the sample matrix used in the analysis, rearranged with dependent variables first, followed by independent variables. In structured means models, this is the augmented covariance/mean matrix. It describes the sample data after all transformations (if any) have been applied. It is a matrix of dimension  $p \times p$ .
- **S**igma is the model covariance matrix rearranged with dependent variables first, followed by independent variables. In structured means models, this is the augmented covariance/mean model matrix. It is a matrix of dimension  $p \times p$ .
- **I**nverted information matrix is  $n$  times the covariance matrix of parameter estimates. In the case of **LS** and **ELS** estimation, it is the sandwich matrix with a normal optimal matrix. When **ME= xx**, **R**obust; is used, two full matrices are printed: first, the usual inverse information, then the sandwich with a distribution-free optimal matrix. It is a matrix of dimension  $q \times q$ .
- **D**erivatives are the elements of  $\partial \sigma/\partial \theta$  for the model and free parameters chosen. Each row of the matrix represents a parameter, and, for each parameter, the derivatives are given in the sequence of the rearranged lower triangular model matrix. It has a dimension of  $q \times p^*$ .
- **W**eight matrix refers to the **AGLS** (**ADF**) weight matrix  $W$ , which is the inverse of the optimal matrix for the theory being invoked. The rows and columns of this matrix correspond to the ordering of the rearranged covariance matrix. It is a matrix of dimension  $p^* \times p^*$ .

- **SS** is the standardized solution.
- **RSquare** is the Bentler-Raykov corrected  $R^2$  (Bentler and Raykov 2000). It is a vector of length  $p + f$ .
- **FMeans** are the estimated factor means (Bentler and Yuan 2000). It is a vector of length  $f$ .
- **UNivariate** are the elements of all univariate statistics (i.e. sample size, mean, skewness, kurtosis, and standard deviation) for all variables. It is a  $5 \times p$  matrix.

More details can be found in the EQS user's guide (Bentler and Wu 2002) and in the EQS technical manual (Bentler 2006).