Confirmatory Factor Analysis with R

A Draft document using **lavaan**, **sem**, and **OpenMx**

Bruce Dudek

2019-07-11

Contents

Preface **3**

1 Introduction and R Setup **4**
1.1 Caveat on this document . . . 5
1.2 Resources . . . 5

2 Prepare and Describe the Data **6**
2.1 The Data Set . . . 6
2.2 Numeric and Graphical Description of the Data . . . 7
2.3 Bivariate Characteristics of the data set . . . 12
2.4 Covariances and Zero Order Correlations . . . 13

3 Using the lavaan package for CFA **15**
3.1 Implement the CFA, First Model . . . 15
3.2 Generate a second model and compare . . . 21
3.3 Compare Model 1 and Model 2 . . . 26
3.4 An additional perspective on estimation and optimization . . . 26

4 Using the sem package for CFA **30**
4.1 Example one . . . 30
4.2 Example two . . . 36
4.3 Compare the two CFA models produced by **sem** . . . 40

5 Using the OpenMx Package for CFA **41**
5.1 First OpenMx Model - Single Factor . . . 41
5.2 Second **OpenMx** Model - the bifactor model . . . 44
5.3 Third **OpenMx** Model . . . 49
5.4 Compare the **OpenMx** models . . . 53

6 Summary and Reproducibility **54**

Preface

The data set and the models evaluated are those used by James Boswell in his APSY613 Multivariate Analysis class in the Psychology Department at the University at Albany. The data set is the WISC-R data set that the multivariate statistics textbook by the Tabachnick textbook (Tabachnick et al., 2019) employs for confirmatory factor analysis illustration. The goal of this document is to outline rudiments of Confirmatory Factor Analysis strategies implmented with three different packages in R. The illustrations here attempt to match the approach taken by Boswell with SAS. The document is targeted to UAlbany graduate students who have already had instruction in R in their introducuctory statistics courses.

This book/monograph uses the **bookdown** package (Xie, 2018a) for R (R Core Team, 2018), which was built on top of **rmarkdown** (Allaire et al., 2018) and **knitr** (Xie, 2015). RStudio (RStudio Team, 2015) was used for all writing and programming.

Chapter 1

Introduction and R Setup

This short monograph outlines three approaches to implementing Confirmatory Factor Analysis with R, by using three separate packages. The illustration is simple, employing a 175 case data set of scores on subsections of the WISC. The idea is to fit a bifactor model where the two latent factors are the verbal and performance constructs. In this primary two-factor model, each observed variable is associated with only one latent factor. Then a second model is fit. It includes a path from both latent factors to one of the variables. Comparisons of models are then performed.

Several R packages are required for the implementations outlined in the succeeding chapters. Since CFA is implemented as a structural equation model, commercial software (e.g., LISREL, EQS, SAS) as well as open-source approaches to CFA all use SEM routines. The three primary R packages to illustrate CFA are **lavaan**, **sem** and **OpenMx**, along with the drawing package, **semPlot**. One major advantage of using R for implementation of these methods is that **semPlot** provides a user-friendly method for producing path diagrams of many styles by simply taking a model object from the CFA fitting functions of the other packages.

Other "housekeeping" packages are loaded here, but the three analytical packages for CFA are loaded at the point in the sequence of their usage since some common function names are shared - thus load order is important.

```
library(car)
library(semPlot)
library(psych)
library(knitr)
library(kableExtra)
library(MVN)
library(dplyr)
library(magrittr)
library(tidyr)
library(corrplot)
library(ggraph)
```

Package citations for packages loaded here (in the above order): **car** (Fox et al., 2018), **semPlot** (Epskamp and with contributions from Simon Stuber, 2017), **psych** (Revelle, 2019), **knitr** (Xie, 2018b), **kableExtra** (Zhu, 2019), **MVN** (Korkmaz et al., 2018), **dplyr** (Wickham et al., 2018), **magrittr** (Bache and Wickham, 2014), **tidyr** (Wickham and Henry, 2018), **corrplot** (Wei and Simko, 2017)

Package citations for packages loaded elsewhere in this document: **bookdown** (Xie, 2018a), **rmarkdown** (Allaire et al., 2018), **sem** (Fox et al., 2017), **lavaan** (Rosseel, 2018), **OpenMx** (Boker et al., 2019)

1.1 Caveat on this document

The present treatment of the CFA procedures is not intended to be an exhaustive analysis of this particular data set. Nor is it intended to be a thorough treatment of the CFA approaches available in R, CFA in general, or SEM in general. Rather, it is intended as a bit more than a simple introduction to CFA using R (and by implication, the nice capabilities of for Structural Equation Modeling). It provides students, who have a basic understanding of how to use R, with a reasonable introduction to CFA modeling code. The R appraoches can then be compared to their class coverage of the same analysis, done with SAS. This document provides some capabilities that may not have been covered in class, and it misses others. The learning curve for software is never at an asymptote……

1.2 Resources

The following list will provide a good start for those needing a broader in CFA modeling and more detailed sources for the primary packages empoyed in this document.

- A comprehensive textbook treatment of SEM and CFA: (Tabachnick et al., 2019)
- Tim Brown's well-regarded book on CFA: (Brown, 2015)
- Rosseel's extensive original article on lavaan: (Rosseel, 2012)
- El-Sheik, et al on a comparison of software for SEM: (El-Sheikh et al., 2017)
- Narayanan's review of eight SEM software approaches (Narayanan, 2012)
- Espkamp's helpful original article on the **semPlot** package: (Narayanan, 2012)

In addition, the following internet resources can be helpful.

- Lavaan package home: [http://lavaan.ugent.be/]
- Google Group for Lavaan: [https://groups.google.com/forum/#!forum/lavaan]
- OpenMx package home: [https://openmx.ssri.psu.edu/wiki/projects]
- OpenMx package online forums: [https://openmx.ssri.psu.edu/forums]
- SEM package page on CRAN: [https://cran.r-project.org/web/packages/sem/index.html]
- lavaan package page on CRAN: [https://cran.r-project.org/web/packages/lavaan/index.html]
- OpenMx package page on Cran: [https://cran.r-project.org/web/packages/OpenMx/index.html]
- MVN package page on Cran: [https://cran.r-project.org/web/packages/MVN/index.html]
- semPlot package page on Cran: [https://cran.r-project.org/web/packages/semPlot/index.html]

Chapter 2

Prepare and Describe the Data

This chapter prepares the data set and does some univariate and multivariate description of its characteristics prior to the CFA implementation in later chapters. Both numeric and graphical description and inference about distribution shape are quickly available with R functions from the **psych** and **MVN** packages.

2.1 The Data Set

The 175 case data set (no missing observations) is loaded from a .csv file. The .csv file was exported from the SPSS system file that is available from the website for the Tabachnick textbook (Tabachnick et al., 2019) It has eleven subscales from the WISC:

- info (Information)
- comp (Comprehension)
- arith (Arithmetic)
- simil (Similarities)
- vocab (Vocabulary)
- digit (Digit Span)
- pictcomp (Picture Completion)
- parang (Picture Arrangement)
- block (Block Design)
- object(Object Assembly)
- coding (Coding - not sure if it is A or B, or a combination)

The user may recognize these scales as commonly discussed subtests of the WISC. The first 6 variables comprise the set of manifest variables for the latent factor known as Verbal. The last five are associated with Performance.

The original data file also contains an ID variable that is dropped for the working object created as wisc2 here.

```
# import the primary data file
wisc1 <- read.csv("wisc1.csv")
knitr::kable(head(wisc1),booktabs=TRUE,format="markdown")
```

ID	info	comp	arith	simil	vocab	digit	pictcomp	parang	block	object	coding
3	8	7	13	9	12	9	6	11	12	7	9
4	9	6	8	7	11	12	6	8	7	12	14
5	13	18	11	16	15	6	18	8	11	12	9
6	8	11	6	12	9	7	13	4	7	12	11
7	10	3	8	9	12	9	7	7	11	4	10

ID	info	comp	arith	simil	vocab	digit	pictcomp	parang	block	object	coding
8	11	7	15	12	10	12	6	12	10	5	10

```
# create the working data frame by removing the ID variable
wisc2 <- wisc1[,2:12]
```

A note about tables in this document: Many of the tables generated by the various R functions in this document are reformatted so that they do not appear as the plain text that is typically output into the R console. The `kable` function in the **knitr** package permits formatting that is well-rendered with **rmarkdown** and **bookdown** document production. `kable` is used frequently.

2.2 Numeric and Graphical Description of the Data

We can explore univariate characteristics of the data with summaries, plots, and evaluation of normality characteristics

2.2.1 Univariate descriptive statistics from the psych package.

```
knitr::kable(describe(wisc2,type=2,fast=T),booktabs=TRUE,format="markdown")
```

	vars	n	mean	sd	min	max	range	se
info	1	175	9.4971	2.9123	3	19	16	0.22015
comp	2	175	10.0000	2.9653	0	18	18	0.22416
arith	3	175	9.0000	2.3069	4	16	12	0.17439
simil	4	175	10.6114	3.1836	2	18	16	0.24066
vocab	5	175	10.7029	2.9327	2	19	17	0.22169
digit	6	175	8.7314	2.7042	0	16	16	0.20442
pictcomp	7	175	10.6800	2.9342	2	19	17	0.22181
parang	8	175	10.3714	2.6597	2	17	15	0.20105
block	9	175	10.3143	2.7098	2	18	16	0.20484
object	10	175	10.9029	2.8440	3	19	16	0.21498
coding	11	175	8.5486	2.8721	0	15	15	0.21711

2.2.2 Univariate Distribution Tests and Plots plus Evaluation of Multivariate Normality

The **MVN** package provides univariate and multivariate normality tests. It is an efficient way to explore characteristics of a set of variables. Several options are available for testing both univariate and Multivariate normality. First, explicit calls for univariate and multivariate tests are made, and then an approach is shown that obtains all at once plus a useful set of plots.

```
x_vars <- wisc2
# use the mvn function for an extensive evaluation
# note that different kinds of tests can be specified with changes in the arguments
result <- mvn(data= x_vars, mvnTest="mardia",  univariateTest="AD")
kable(result$univariateNormality, booktabs=TRUE, format="markdown")
```

Test	Variable	Statistic	p value	Normality
Anderson-Darling	info	1.2049	0.0037	NO
Anderson-Darling	comp	1.3546	0.0016	NO

Test	Variable	Statistic	p value	Normality
Anderson-Darling	arith	1.8656	1e-04	NO
Anderson-Darling	simil	0.9336	0.0175	NO
Anderson-Darling	vocab	1.4992	7e-04	NO
Anderson-Darling	digit	2.5087	<0.001	NO
Anderson-Darling	pictcomp	1.1472	0.0052	NO
Anderson-Darling	parang	1.3391	0.0017	NO
Anderson-Darling	block	1.5225	6e-04	NO
Anderson-Darling	object	1.1593	0.0049	NO
Anderson-Darling	coding	1.2217	0.0034	NO

```
kable(result$multivariateNormality, booktabs=TRUE, format="markdown")
```

Test	Statistic	p value	Result
Mardia Skewness	348.496907430775	0.00673137335852766	NO
Mardia Kurtosis	2.30789388829536	0.0210050390817529	NO
MVN	NA	NA	NO

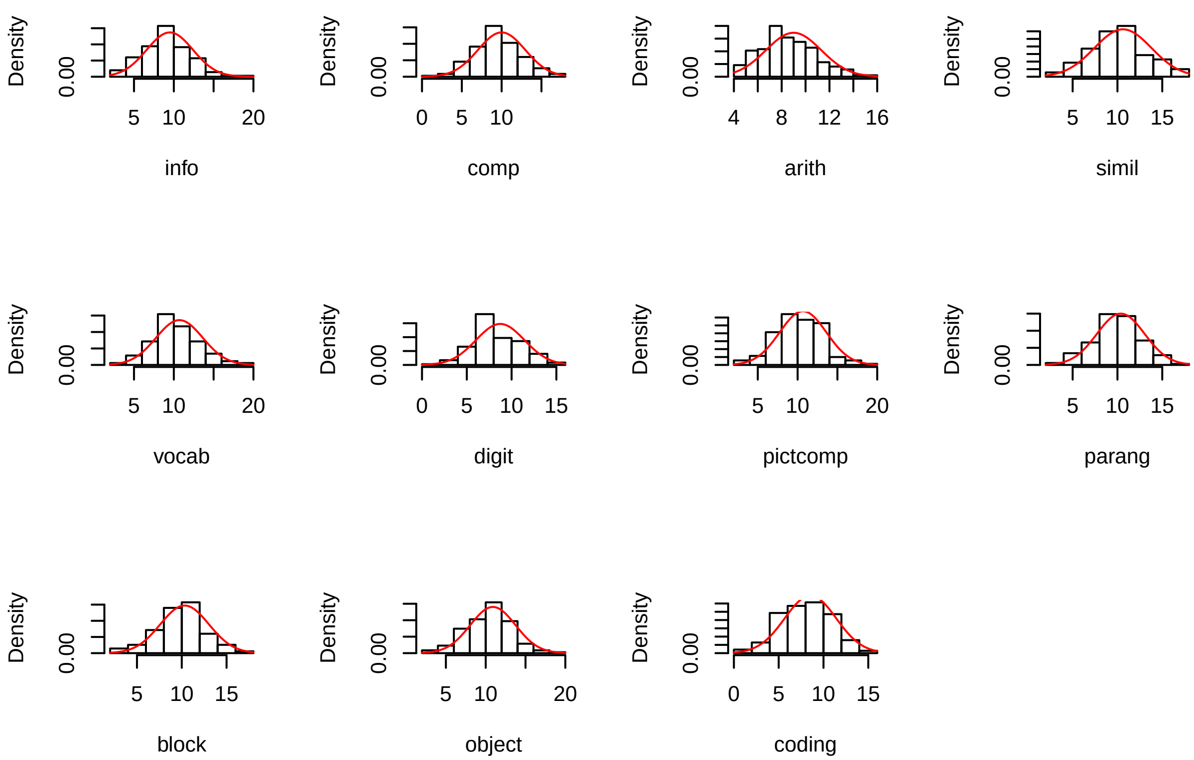
```
kable(mvn(data= x_vars,univariatePlot="histogram"), booktabs=TRUE, format="markdown")
```

Test	Statistic	p value	Result
Mardia Skewness	348.496907430775	0.00673137335852766	NO
Mardia Kurtosis	2.30789388829536	0.0210050390817529	NO
MVN	NA	NA	NO

Test	Variable	Statistic	p value	Normality
Shapiro-Wilk	info	0.9814	0.0194	NO
Shapiro-Wilk	comp	0.9810	0.0170	NO
Shapiro-Wilk	arith	0.9709	0.0010	NO
Shapiro-Wilk	simil	0.9866	0.0937	YES
Shapiro-Wilk	vocab	0.9789	0.0093	NO
Shapiro-Wilk	digit	0.9693	0.0007	NO
Shapiro-Wilk	pictcomp	0.9836	0.0373	NO
Shapiro-Wilk	parang	0.9814	0.0193	NO
Shapiro-Wilk	block	0.9800	0.0127	NO
Shapiro-Wilk	object	0.9853	0.0638	YES
Shapiro-Wilk	coding	0.9820	0.0230	NO

	n	Mean	Std.Dev	Median	Min	Max	25th	75th	Skew	Kurtosis
info	175	9.4971	2.9123	10	3	19	8	11.5	0.08341	-0.08148
comp	175	10.0000	2.9653	10	0	18	8	12.0	0.08547	0.32620
arith	175	9.0000	2.3069	9	4	16	7	10.5	0.39097	-0.17897
simil	175	10.6114	3.1836	11	2	18	9	12.0	0.02233	-0.22985
vocab	175	10.7029	2.9327	10	2	19	9	12.0	0.26702	0.28751
digit	175	8.7314	2.7042	8	0	16	7	11.0	0.26631	0.07205
pictcomp	175	10.6800	2.9342	11	2	19	9	13.0	-0.07150	0.29116
parang	175	10.3714	2.6597	10	2	17	9	12.0	-0.19909	-0.05850

	n	Mean	Std.Dev	Median	Min	Max	25th	75th	Skew	Kurtosis
block	175	10.3143	2.7098	10	2	18	9	12.0	-0.21972	0.50196
object	175	10.9029	2.8440	11	3	19	9	13.0	-0.12289	0.14745
coding	175	8.5486	2.8721	9	0	15	6	11.0	-0.05283	-0.45376

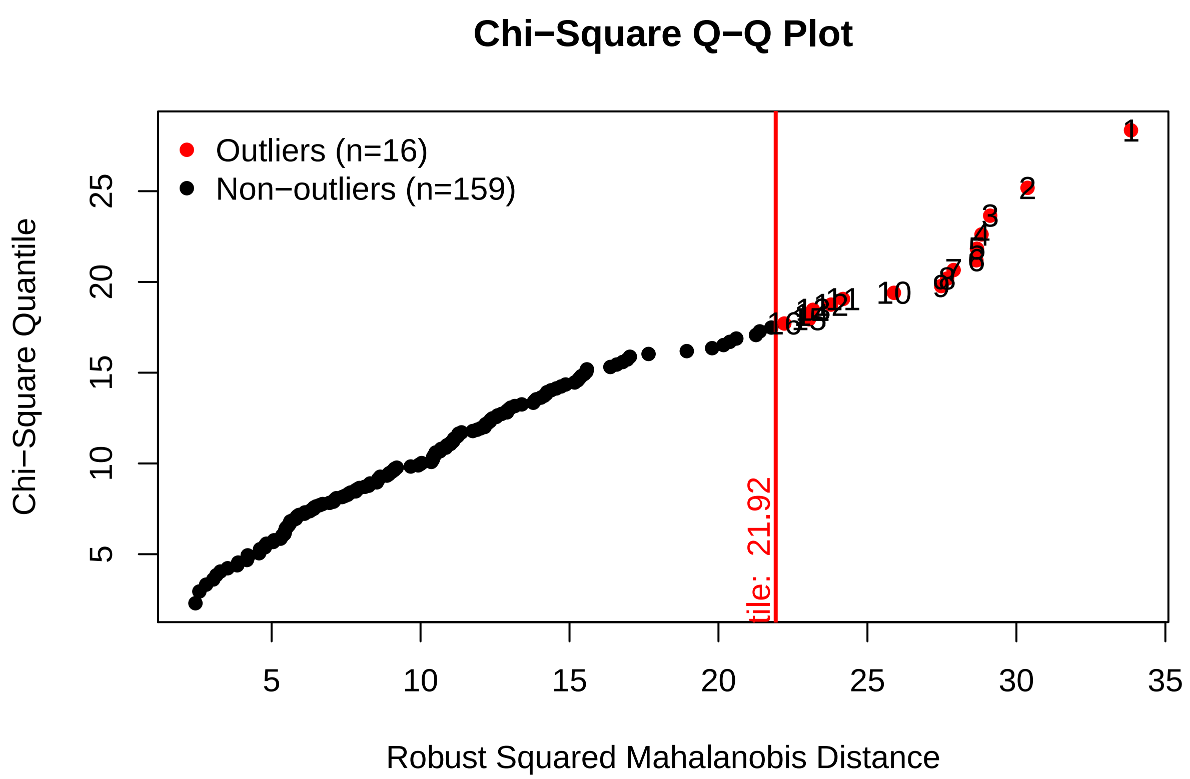


2.2.3 Multivariate Outlier tests

The **MVN** package permits a good array of diagnostic tests/plots for univariate/multivariate shape and outliers.

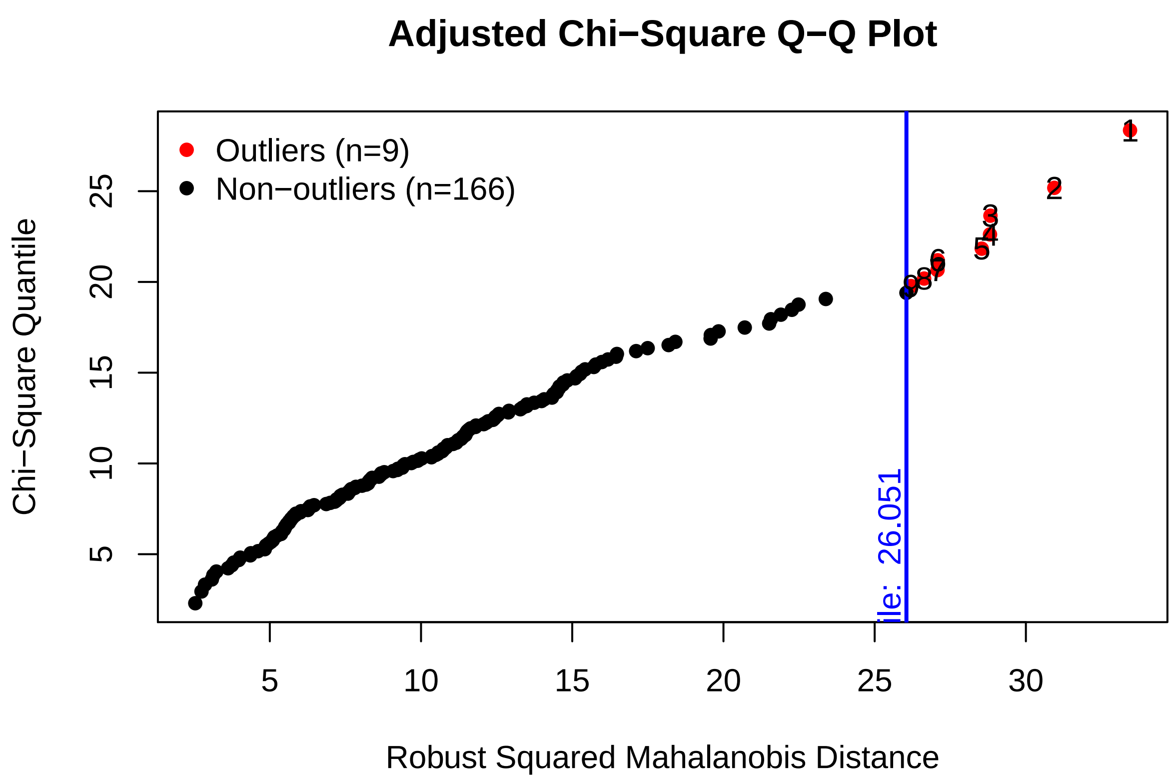
First, multivariate outliers are examined with the Mahalanobis distance:

```
result2 <- mvn(data=x_vars,multivariateOutlierMethod="quan")
```



Next, multivariate outliers are evaluated with the Adjusted-Mahalanobis distance:

```
result2 <- mvn(data=x_vars,multivariateOutlierMethod="adj")
```



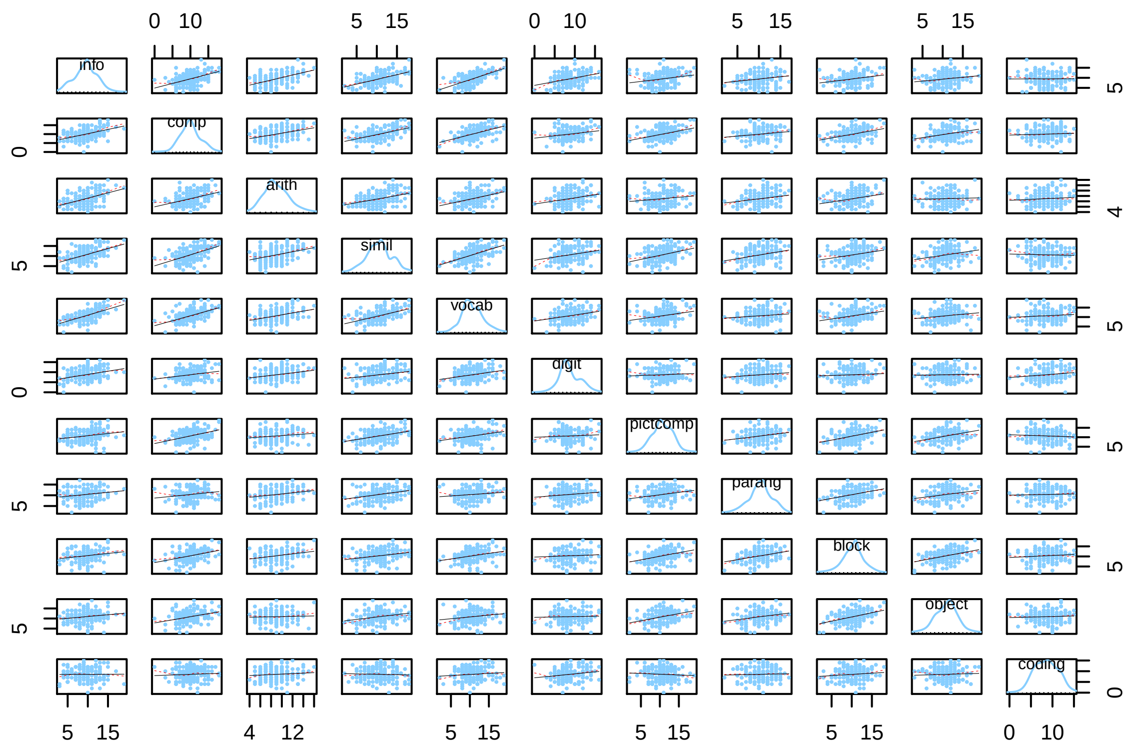
2.3 Bivariate Characteristics of the data set

We can quickly explore numerical and graphical summaries of the eleven variables.

2.3.1 SPLOM

Among the many scatterplot matrix capabilies in R, John Fox' `scatterplot.matrix` function in his **car** package has probably been seen by most students.

```
scatterplotMatrix(wisc2,cex=.2,
                  smooth=list(col.smooth="red", spread=F, lwd.smooth=.3),
                  col="skyblue1",
                  regLine=list(lwd=.3,col="black"))
```



Even with some control over colors and sizes of points/lines, this SPLOM probably has too many variables to be effective - each plot is very small. Nonetheless, the sense of fairly linear relationships among all pairs is somewhat apparent, as is the relative univariate normality of each of the eleven.

Note that the image can be enlarged if the reader is using a pdf version of this document simply by using the increase/decrease size capability of pdf readers. If the user is reading an html version of this document, then try to do a right mouse click on the image and "view image" (in Windows). Then the image can be increased in size in a browser.

2.4 Covariances and Zero Order Correlations

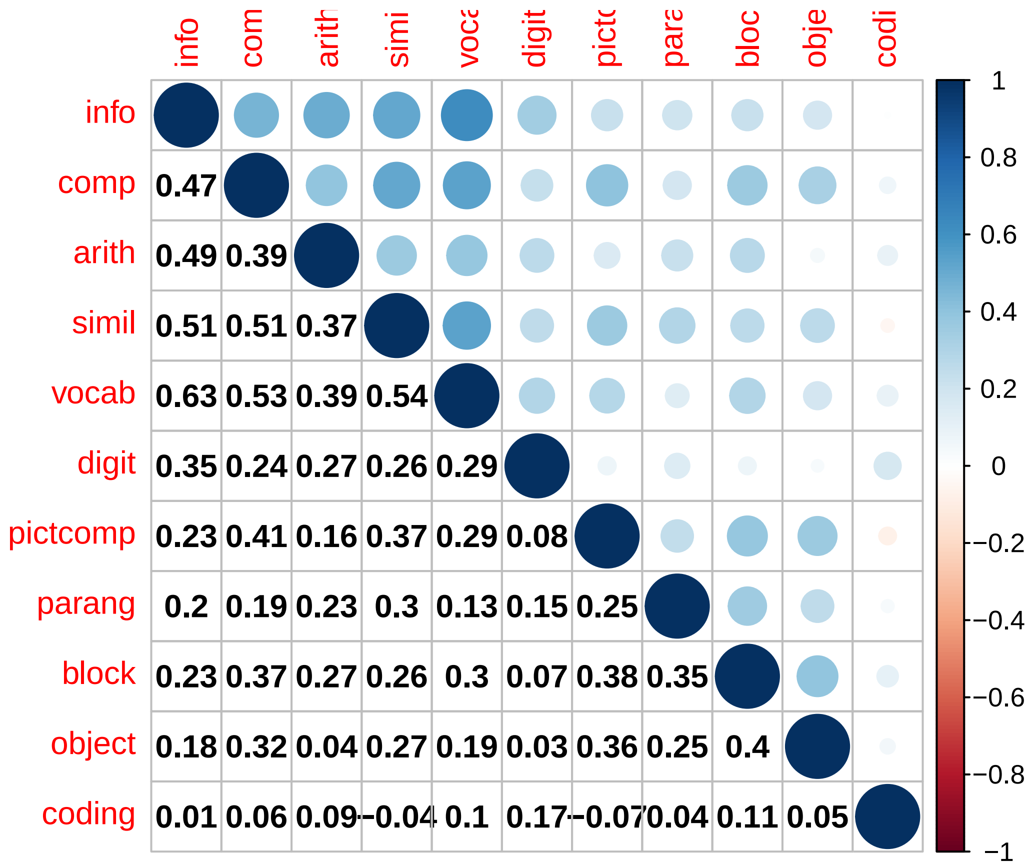
The covariance matrix is the basic input for the CFA algorithms outlined in later chapters.

```
covmatrix1 <- round(cov(wisc2),digits=3)
knitr::kable(covmatrix1,booktabs=TRUE,format="markdown")
```

	info	comp	arith	simil	vocab	digit	pictcomp	parang	block	object	coding
info	8.481	4.034	3.322	4.758	5.338	2.720	1.965	1.561	1.808	1.531	0.059
comp	4.034	8.793	2.684	4.816	4.621	1.891	3.540	1.471	2.966	2.718	0.517
arith	3.322	2.684	5.322	2.713	2.621	1.678	1.052	1.391	1.701	0.282	0.598
simil	4.758	4.816	2.713	10.136	5.022	2.234	3.450	2.524	2.255	2.433	-0.372
vocab	5.338	4.621	2.621	5.022	8.601	2.334	2.456	1.031	2.364	1.546	0.842
digit	2.720	1.891	1.678	2.234	2.334	7.313	0.597	1.066	0.533	0.267	1.344
pictcomp	1.965	3.540	1.052	3.450	2.456	0.597	8.610	1.941	3.038	3.032	-0.605
parang	1.561	1.471	1.391	2.524	1.031	1.066	1.941	7.074	2.532	1.916	0.289
block	1.808	2.966	1.701	2.255	2.364	0.533	3.038	2.532	7.343	3.077	0.832
object	1.531	2.718	0.282	2.433	1.546	0.267	3.032	1.916	3.077	8.088	0.433
coding	0.059	0.517	0.598	-0.372	0.842	1.344	-0.605	0.289	0.832	0.433	8.249

We can use the Corrplot package to produce a useful combination of a schematic and correlation matrix.

```
mat1 <- cor(wisc2)
corrplot(mat1,type="upper",tl.pos="tp")
corrplot(mat1,add=T,type="lower", method="number",
col="black", diag=FALSE,tl.pos="n", cl.pos="n")
```



Chapter 3

Using the lavaan package for CFA

One of the primary tools for SEM in R is the **lavaan** package. It permits path specification with a simple syntax.

3.1 Implement the CFA, First Model

Using the **lavaan** package, we can implemnt directly the CFA with only a few steps. Since this document contains three different packages' approach to CFA, the packages used for each will be loaded at that point, so as to not have confusion over common function names.

```
library(lavaan)
```

3.1.1 Define and fit the first model

The latent variables and their paths to the manifest variables are initially defined as simple textual specifications. The =~ symbol is the key to defining paths. We have two latent variables and no manifest variable has duplicate paths from both latents. This is a so-called "simple structure."

Note that this text string uses variable names that match what is in the wisc2 data set.

```
wisc.model1 <- "verbal  =~ info + comp + arith + simil + vocab + digit
                 performance =~ pictcomp + parang + block + object + coding"
```

Fit the model and obtain the basic summary. Note that in this default approach, the latent factors are permitted to covary and the model estimates this covariance.

One R syntax note..... the format here to call the cfa function (`lavaan::cfa(.....)`) is employed to ensure no ambiguity that the correct cfa function is the one from the lavaan package. This precludes confusion when multiple packages contain functions with the same name as is the case with both **lavaan** and **sem** which is also used in this document. Even though **sem** is loaded later in this document, if there is a chance that it may simultaneously exist in the R environment with **lavaan** then the approach here is safer.

```
fit1 <- lavaan::cfa(wisc.model1, data=wisc2,std.lv=TRUE)
summary(fit1, fit.measures=T,standardized=T)
```

```
## lavaan 0.6-3 ended normally after 24 iterations
##
##   Optimization method                           NLMINB
##   Number of free parameters                         23
##
##   Number of observations                           175
```

```
##
##   Estimator                                         ML
##   Model Fit Test Statistic                      70.640
##   Degrees of freedom                                43
##   P-value (Chi-square)                           0.005
##
## Model test baseline model:
##
##   Minimum Function Test Statistic              519.204
##   Degrees of freedom                                55
##   P-value                                        0.000
##
## User model versus baseline model:
##
##   Comparative Fit Index (CFI)                    0.940
##   Tucker-Lewis Index (TLI)                       0.924
##
## Loglikelihood and Information Criteria:
##
##   Loglikelihood user model (H0)              -4491.822
##   Loglikelihood unrestricted model (H1)      -4456.502
##
##   Number of free parameters                         23
##   Akaike (AIC)                                9029.643
##   Bayesian (BIC)                              9102.433
##   Sample-size adjusted Bayesian (BIC)         9029.600
##
## Root Mean Square Error of Approximation:
##
##   RMSEA                                          0.061
##   90 Percent Confidence Interval          0.033  0.085
##   P-value RMSEA <= 0.05                          0.233
##
## Standardized Root Mean Square Residual:
##
##   SRMR                                           0.059
##
## Parameter Estimates:
##
##   Information                                 Expected
##   Information saturated (h1) model          Structured
##   Standard Errors                             Standard
##
## Latent Variables:
##                    Estimate  Std.Err  z-value  P(>|z|)   Std.lv  Std.all
##   verbal =~
##     info              2.206    0.200   11.029    0.000    2.206    0.760
##     comp              2.042    0.210    9.709    0.000    2.042    0.691
##     arith             1.300    0.172    7.555    0.000    1.300    0.565
##     simil             2.232    0.225    9.940    0.000    2.232    0.703
##     vocab             2.250    0.200   11.225    0.000    2.250    0.770
##     digit             1.053    0.212    4.967    0.000    1.053    0.390
##   performance =~
##     pictcomp          1.742    0.242    7.187    0.000    1.742    0.595
```

```
##     parang            1.253    0.224    5.582    0.000    1.253    0.473
##     block             1.846    0.222    8.311    0.000    1.846    0.683
##     object            1.605    0.236    6.800    0.000    1.605    0.566
##     coding            0.207    0.254    0.814    0.416    0.207    0.072
##
## Covariances:
##                    Estimate  Std.Err  z-value  P(>|z|)   Std.lv  Std.all
##   verbal ~~
##     performance       0.589    0.075    7.814    0.000    0.589    0.589
##
## Variances:
##                    Estimate  Std.Err  z-value  P(>|z|)   Std.lv  Std.all
##    .info              3.566    0.507    7.034    0.000    3.566    0.423
##    .comp              4.572    0.585    7.815    0.000    4.572    0.523
##    .arith             3.602    0.420    8.571    0.000    3.602    0.681
##    .simil             5.096    0.662    7.702    0.000    5.096    0.506
##    .vocab             3.487    0.506    6.886    0.000    3.487    0.408
##    .digit             6.162    0.680    9.056    0.000    6.162    0.848
##    .pictcomp          5.526    0.757    7.296    0.000    5.526    0.646
##    .parang            5.463    0.658    8.298    0.000    5.463    0.777
##    .block             3.894    0.640    6.083    0.000    3.894    0.533
##    .object            5.467    0.719    7.600    0.000    5.467    0.680
##    .coding            8.159    0.874    9.335    0.000    8.159    0.995
##     verbal            1.000                               1.000    1.000
##     performance       1.000                               1.000    1.000
```

3.1.2 Obtain coefficients

It is possible to obtain the output from the above `summary` function that does not contain the parameter coefficients part of the table. If that had been the implementation, then the coefficients can be obtained simply, but it is better to obtain the full table of parameter estimates and errors, etc.

```r
# obtain only the coefficients
kable(coef(fit1), booktabs=TRUE, format="markdown")
```

3.1.3 Complete parameter listing

Instead of directly printing the parameter table, I prefer to reformat it with kable when using R Markdown. Without using kable, the code would be the following:

```r
parameterEstimates(fit1,standardized=T)
```

Format the table and clean it up using kable.

```r
parameterEstimates(fit1, standardized=TRUE) %>%
  filter(op == "=~") %>%
  select('Latent Factor'=lhs, Indicator=rhs, B=est, SE=se, Z=z, 'p-value'=pvalue, Beta=std.all) %>%
  knitr::kable(digits = 3, booktabs=TRUE, format="markdown", caption="Factor Loadings")
```

Latent Factor	Indicator	B	SE	Z	p-value	Beta
verbal	info	2.206	0.200	11.029	0.000	0.760
verbal	comp	2.042	0.210	9.709	0.000	0.691
verbal	arith	1.300	0.172	7.555	0.000	0.565
verbal	simil	2.232	0.225	9.940	0.000	0.703
verbal	vocab	2.250	0.200	11.225	0.000	0.770

Latent Factor	Indicator	B	SE	Z	p-value	Beta
verbal	digit	1.053	0.212	4.967	0.000	0.390
performance	pictcomp	1.742	0.242	7.187	0.000	0.595
performance	parang	1.253	0.224	5.582	0.000	0.473
performance	block	1.846	0.222	8.311	0.000	0.683
performance	object	1.605	0.236	6.800	0.000	0.566
performance	coding	0.207	0.254	0.814	0.416	0.072

3.1.4 Residuals correlation matrix

Residuals can be examined.

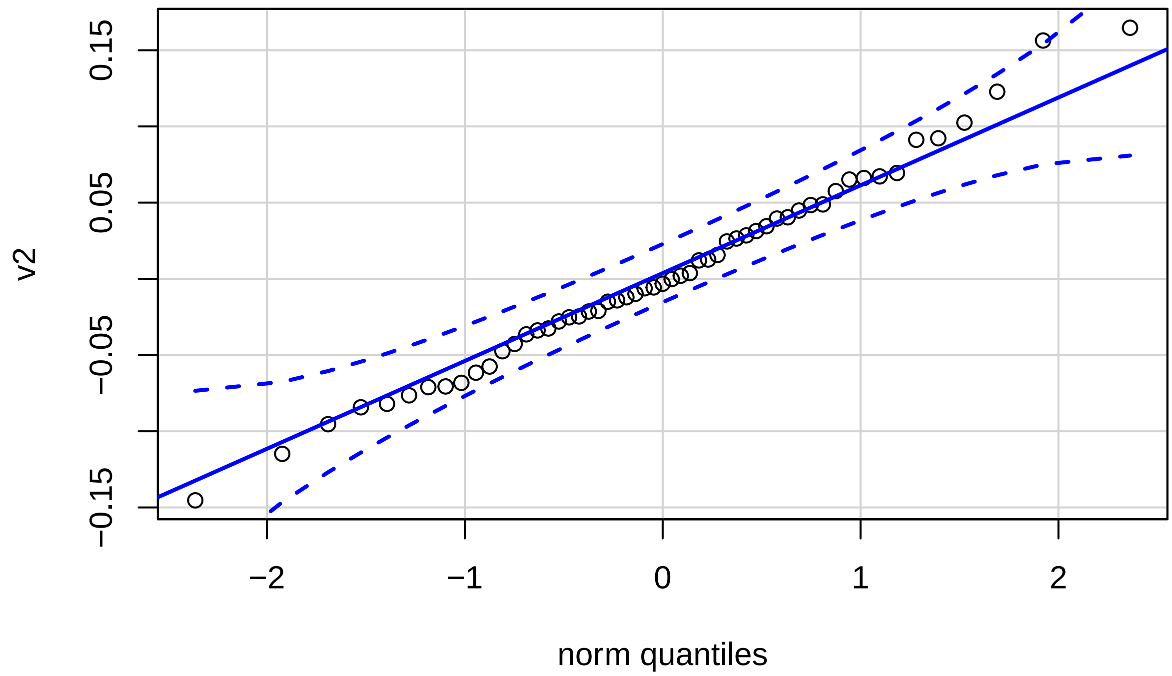
```
cor_table <- residuals(fit1, type = "cor")$cov
#cor_table[upper.tri(cor_table)] <-  # erase the upper triangle
#diag(cor_table) <- NA # erase the diagonal 0's
knitr::kable(cor_table, digits=3,format="markdown", booktabs=TRUE) # makes a nice table and rounds every
```

	info	comp	arith	simil	vocab	digit	pictcomp	parang	block	object	coding
info	0.000	-0.058	0.065	-0.021	0.040	0.049	-0.036	-0.010	-0.076	-0.068	-0.025
comp	-0.058	0.000	0.002	0.025	0.000	-0.034	0.165	-0.006	0.091	0.092	0.031
arith	0.065	0.002	0.000	-0.028	-0.047	0.048	-0.043	0.069	0.045	-0.145	0.066
simil	-0.021	0.025	-0.028	0.000	-0.003	-0.015	0.123	0.102	-0.021	0.034	-0.071
vocab	0.040	0.000	-0.047	-0.003	0.000	-0.006	0.016	-0.082	-0.012	-0.071	0.067
digit	0.049	-0.034	0.048	-0.015	-0.006	0.000	-0.062	0.040	-0.084	-0.095	0.156
pictcomp	-0.036	0.165	-0.043	0.123	0.016	-0.062	0.000	-0.033	-0.025	0.026	-0.115
parang	-0.010	-0.006	0.069	0.102	-0.082	0.040	-0.033	0.000	0.028	-0.014	0.004
block	-0.076	0.091	0.045	-0.021	-0.012	-0.084	-0.025	0.028	0.000	0.013	0.058
object	-0.068	0.092	-0.145	0.034	-0.071	-0.095	0.026	-0.014	0.013	0.000	0.012
coding	-0.025	0.031	0.066	-0.071	0.067	0.156	-0.115	0.004	0.058	0.012	0.000

3.1.5 Plot the residuals.

norm plot

```
# extract the residuals from the fit1 model
# get rid of the duplicates and diagonal values
# create a vector for a
res1 <- residuals(fit1, type = "cor")$cov
res1[upper.tri(res1,diag=T)] <- NA
v1 <- as.vector(res1)
v2  <- v1[!is.na(v1)]
qqPlot(v2,id=F)
```



3.1.6 Modification Indices

Modification indices provide……

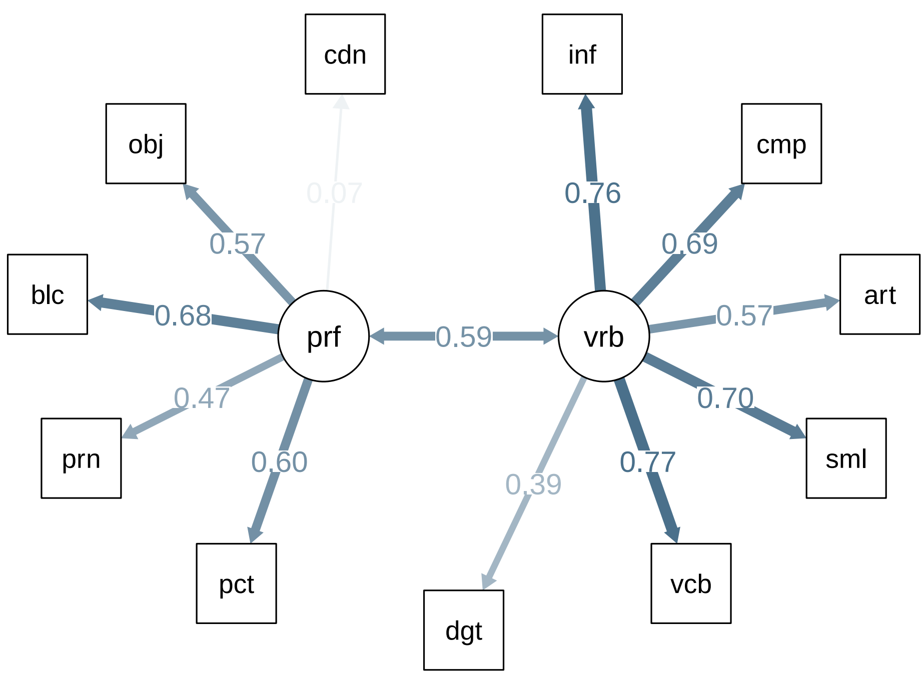
```
kable(modificationIndices(fit1, sort.=TRUE, minimum.value=3), booktabs=TRUE, format="markdown")
```

	lhs	op	rhs	mi	epc	sepc.lv	sepc.all	sepc.nox
32	performance	=~	comp	9.8232	0.93721	0.93721	0.31696	0.31696
62	arith	~~	object	6.3624	-0.94453	-0.94453	-0.21285	-0.21285
37	info	~~	comp	5.2214	-0.98473	-0.98473	-0.24388	-0.24388
81	digit	~~	coding	4.9267	1.20723	1.20723	0.17025	0.17025
51	comp	~~	pictcomp	4.6854	0.96931	0.96931	0.19284	0.19284
85	pictcomp	~~	coding	4.5748	-1.22491	-1.22491	-0.18242	-0.18242
31	performance	=~	info	4.4766	-0.59552	-0.59552	-0.20507	-0.20507
40	info	~~	vocab	4.4029	0.91242	0.91242	0.25875	0.25875
73	vocab	~~	parang	4.1459	-0.79773	-0.79773	-0.18277	-0.18277
38	info	~~	arith	4.1333	0.69702	0.69702	0.19450	0.19450
67	simil	~~	parang	3.3732	0.83050	0.83050	0.15741	0.15741
70	simil	~~	coding	3.2711	-0.95602	-0.95602	-0.14827	-0.14827
66	simil	~~	pictcomp	3.2695	0.86025	0.86025	0.16212	0.16212

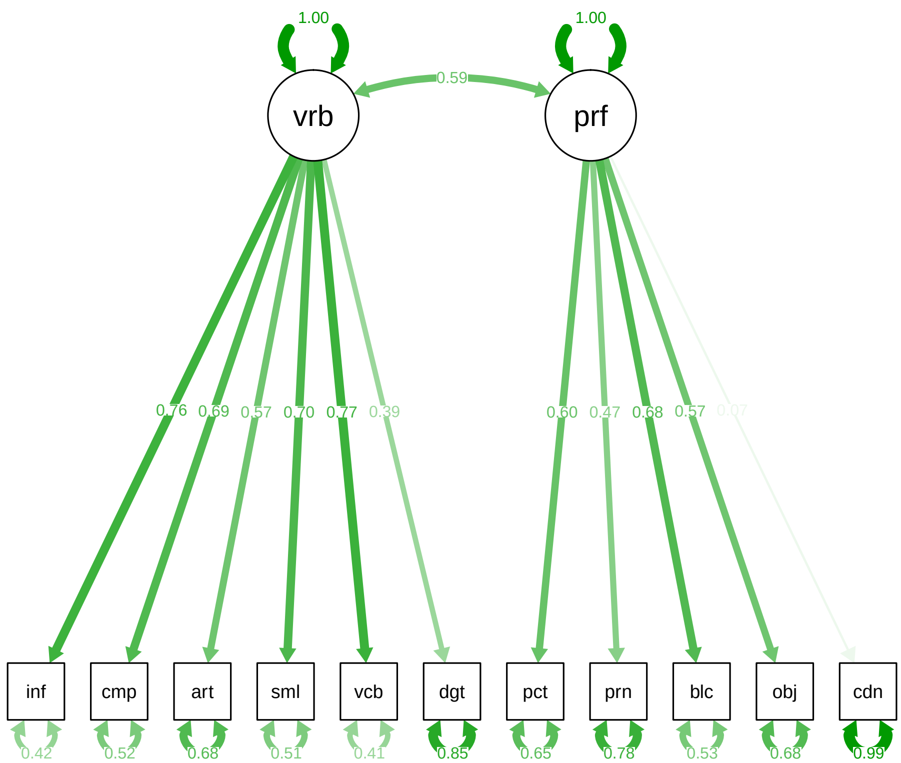
3.1.7 Path Diagram for the bifactor Model 1

The `semPlot` package provides a capability of drawing path diagrams for cfa and other sem models. The `semPaths` function will take a cfa model object and draw the diagram, with several options available. The diagram produced here takes control over font/label sizes, display of residuals, and color of paths/coefficients. These and many more control options are available. There is a challenge in producing these path diagrams to have font sizes large enough for most humans to read. I've taken control of the font sizes for the "edges" with a cex argument. But this causes overlap in the values if the default layout is used. I found that "circle2" worked best here.

```
# Note that the base plot, including standardized path coefficients plots positive coefficients green
# and negative coefficients red.  Red-green colorblindness issues anyone?
# I redrew it here to choose a blue and red.  But all the coefficients in this example are
# positive,so they are shown with the skyblue.
# more challenging to use colors other than red and green.  not in this doc
semPaths(fit1, residuals=F,sizeMan=7,"std",
         posCol=c("skyblue4", "red"),
         #edge.color="skyblue4",
         edge.label.cex=1.2,layout="circle2")
```




```
# or we could draw the paths in such  a way to include the residuals:
#semPaths(fit1, sizeMan=7,"std",edge.color="skyblue4",edge.label.cex=1,layout="circle2")
# the base path diagram can be drawn much more simply:
#semPaths(fit1)
# or
semPaths(fit1,"std")
```



3.1.8 orthognal fit comparison

Compare to a one factor solution.

```
fit1orth <- lavaan::cfa(wisc.model1, data=wisc1,orthogonal=T)
kable(anova(fit1,fit1orth), booktabs=TRUE, format="markdown")
```

	Df	AIC	BIC	Chisq	Chisq diff	Df diff	Pr(>Chisq)
fit1	43	9029.6	9102.4	70.64	NA	NA	NA
fit1orth	44	9065.6	9135.2	108.61	37.97	1	0

3.2 Generate a second model and compare

Next, generate a second CFA model. Thre are theoretical reasons why paths from both latent factors to "comp" might be warranted. The "comp" variable also has the largest Modification index in the first model. This second modelimplements this one-path/parmaeter change.

3.2.1 Add a path (Perf to comp) and Fit the second CFA model

Define the addditional path in the model text string.

```
wisc.model2 <- 'verbal  =~ info + comp + arith + simil + vocab + digit
                performance =~ pictcomp + parang + block + object + coding + comp'
```

Fit the model and obtain the basic summary

```
fit2 <- lavaan::cfa(wisc.model2, data=wisc1,std.lv=TRUE)
summary(fit2, fit.measures=T,standardized=T)
```

```
## lavaan 0.6-3 ended normally after 26 iterations
##
##   Optimization method                           NLMINB
##   Number of free parameters                         24
##
##   Number of observations                           175
##
##   Estimator                                         ML
##   Model Fit Test Statistic                      60.642
##   Degrees of freedom                                42
##   P-value (Chi-square)                           0.031
##
## Model test baseline model:
##
##   Minimum Function Test Statistic              519.204
##   Degrees of freedom                                55
##   P-value                                        0.000
##
## User model versus baseline model:
##
##   Comparative Fit Index (CFI)                    0.960
##   Tucker-Lewis Index (TLI)                       0.947
##
## Loglikelihood and Information Criteria:
##
##   Loglikelihood user model (H0)              -4486.823
##   Loglikelihood unrestricted model (H1)      -4456.502
##
##   Number of free parameters                         24
##   Akaike (AIC)                                9021.645
##   Bayesian (BIC)                              9097.600
##   Sample-size adjusted Bayesian (BIC)         9021.600
##
## Root Mean Square Error of Approximation:
##
##   RMSEA                                          0.050
##   90 Percent Confidence Interval          0.016  0.077
##   P-value RMSEA <= 0.05                          0.465
##
## Standardized Root Mean Square Residual:
##
##   SRMR                                           0.054
##
## Parameter Estimates:
##
##   Information                                 Expected
##   Information saturated (h1) model          Structured
##   Standard Errors                             Standard
##
## Latent Variables:
##                    Estimate  Std.Err  z-value  P(>|z|)   Std.lv  Std.all
```

```
##   verbal =~
##     info              2.256    0.199   11.318    0.000    2.256    0.777
##     comp              1.491    0.254    5.877    0.000    1.491    0.504
##     arith             1.307    0.172    7.584    0.000    1.307    0.568
##     simil             2.205    0.226    9.748    0.000    2.205    0.695
##     vocab             2.273    0.201   11.329    0.000    2.273    0.777
##     digit             1.075    0.212    5.068    0.000    1.075    0.399
##   performance =~
##     pictcomp          1.790    0.239    7.495    0.000    1.790    0.612
##     parang            1.189    0.224    5.317    0.000    1.189    0.448
##     block             1.823    0.219    8.334    0.000    1.823    0.675
##     object            1.633    0.233    7.010    0.000    1.633    0.576
##     coding            0.200    0.253    0.793    0.428    0.200    0.070
##     comp              0.884    0.266    3.324    0.001    0.884    0.299
##
## Covariances:
##                    Estimate  Std.Err  z-value  P(>|z|)   Std.lv  Std.all
##   verbal ~~
##     performance       0.533    0.081    6.594    0.000    0.533    0.533
##
## Variances:
##                    Estimate  Std.Err  z-value  P(>|z|)   Std.lv  Std.all
##    .info              3.343    0.502    6.665    0.000    3.343    0.396
##    .comp              4.331    0.554    7.819    0.000    4.331    0.495
##    .arith             3.584    0.420    8.533    0.000    3.584    0.677
##    .simil             5.217    0.675    7.726    0.000    5.217    0.518
##    .vocab             3.383    0.508    6.655    0.000    3.383    0.396
##    .digit             6.116    0.677    9.032    0.000    6.116    0.841
##    .pictcomp          5.358    0.741    7.231    0.000    5.358    0.626
##    .parang            5.620    0.663    8.480    0.000    5.620    0.799
##    .block             3.979    0.623    6.385    0.000    3.979    0.545
##    .object            5.376    0.707    7.603    0.000    5.376    0.668
##    .coding            8.162    0.874    9.337    0.000    8.162    0.995
##     verbal            1.000                               1.000    1.000
##     performance       1.000                               1.000    1.000
```

3.2.2 Obtain coefficients

Coefficients can be obtained simply with the `coef` function, but it is better to do the full parameter listing in the next section

```
knitr::kable(coef(fit2),booktabs=TRUE, format="markdown")
```

It is simple to obtain the full parameter list, but I prefer to use kable for tables when I can.

```
parameterEstimates(fit2,standardized=TRUE)
```

Reformat the table and clean it up by using kable.

```
parameterEstimates(fit2, standardized=TRUE) %>%
  filter(op == "=~") %>%
  select('Latent Factor'=lhs, Indicator=rhs, B=est, SE=se, Z=z, 'p-value'=pvalue, Beta=std.all) %>%
  knitr::kable(digits = 3, format="markdown", booktabs=TRUE, caption="Factor Loadings")
```

Latent Factor	Indicator	B	SE	Z	p-value	Beta
verbal	info	2.256	0.199	11.318	0.000	0.777

Latent Factor	Indicator	B	SE	Z	p-value	Beta
verbal	comp	1.491	0.254	5.877	0.000	0.504
verbal	arith	1.307	0.172	7.584	0.000	0.568
verbal	simil	2.205	0.226	9.748	0.000	0.695
verbal	vocab	2.273	0.201	11.329	0.000	0.777
verbal	digit	1.075	0.212	5.068	0.000	0.399
performance	pictcomp	1.790	0.239	7.495	0.000	0.612
performance	parang	1.189	0.224	5.317	0.000	0.448
performance	block	1.823	0.219	8.334	0.000	0.675
performance	object	1.633	0.233	7.010	0.000	0.576
performance	coding	0.200	0.253	0.793	0.428	0.070
performance	comp	0.884	0.266	3.324	0.001	0.299

3.2.3 Residuals correlation matrix

Residuals can be examined.

```
cor_table2 <- residuals(fit2, type = "cor")$cov

#cor_table[upper.tri(cor_table)] <-  # erase the upper triangle
#diag(cor_table) <- NA # erase the diagonal 0's
knitr::kable(cor_table2, digits=3,format="markdown",booktabs=TRUE) # makes a nice table and rounds ever
```

	info	comp	arith	simil	vocab	digit	pictcomp	parang	block	object	coding
info	0.000	-0.049	0.053	-0.026	0.021	0.036	-0.023	0.016	-0.050	-0.054	-0.022
comp	-0.049	0.000	0.015	0.049	0.015	-0.029	0.059	-0.068	-0.014	-0.005	0.021
arith	0.053	0.015	0.000	-0.025	-0.054	0.043	-0.030	0.091	0.068	-0.131	0.069
simil	-0.026	0.049	-0.025	0.000	-0.002	-0.017	0.143	0.132	0.012	0.056	-0.067
vocab	0.021	0.015	-0.054	-0.002	0.000	-0.016	0.032	-0.054	0.018	-0.053	0.071
digit	0.036	-0.029	0.043	-0.017	-0.016	0.000	-0.055	0.053	-0.071	-0.088	0.158
pictcomp	-0.023	0.059	-0.030	0.143	0.032	-0.055	0.000	-0.025	-0.031	0.011	-0.115
parang	0.016	-0.068	0.091	0.132	-0.054	0.053	-0.025	0.000	0.049	-0.005	0.006
block	-0.050	-0.014	0.068	0.012	0.018	-0.071	-0.031	0.049	0.000	0.011	0.060
object	-0.054	-0.005	-0.131	0.056	-0.053	-0.088	0.011	-0.005	0.011	0.000	0.013
coding	-0.022	0.021	0.069	-0.067	0.071	0.158	-0.115	0.006	0.060	0.013	0.000

3.2.4 Modification Indices for Model 2

Modification indices provide……………….

```
kable(modificationIndices(fit2, sort.=TRUE, minimum.value=3), booktabs=TRUE, format="markdown")
```

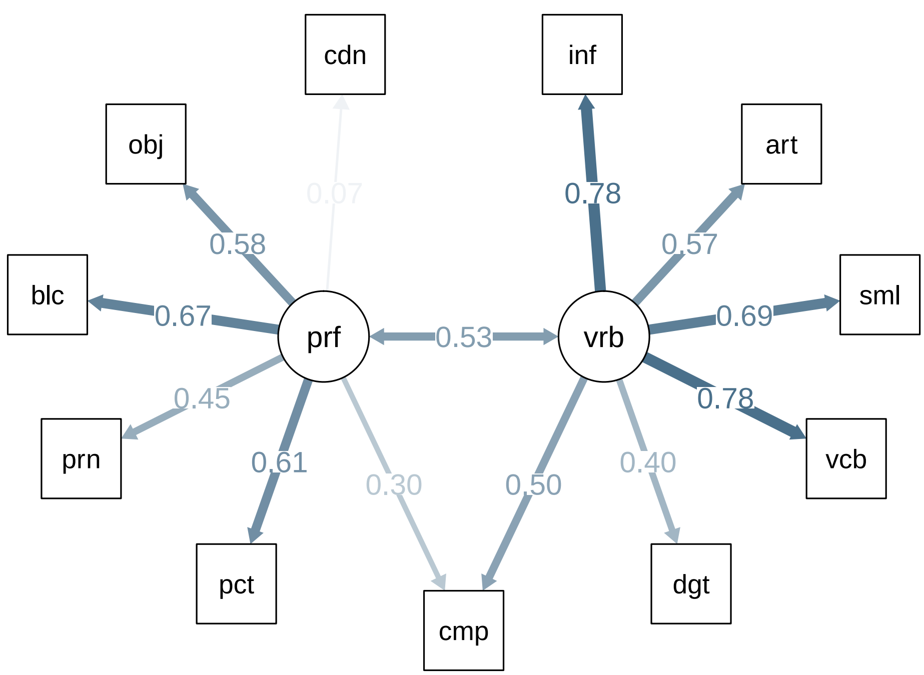
	lhs	op	rhs	mi	epc	sepc.lv	sepc.all	sepc.nox
34	performance	=~	simil	6.7637	0.79457	0.79457	0.25030	0.25030
62	arith	~~	object	5.5514	-0.87412	-0.87412	-0.19915	-0.19915
81	digit	~~	coding	4.9756	1.20983	1.20983	0.17124	0.17124
85	pictcomp	~~	coding	4.7421	-1.23384	-1.23384	-0.18659	-0.18659
66	simil	~~	pictcomp	3.8109	0.92525	0.92525	0.17501	0.17501
52	comp	~~	parang	3.7688	-0.85250	-0.85250	-0.17279	-0.17279
73	vocab	~~	parang	3.6900	-0.75282	-0.75282	-0.17265	-0.17265
67	simil	~~	parang	3.5789	0.86835	0.86835	0.16037	0.16037
57	arith	~~	vocab	3.3775	-0.64063	-0.64063	-0.18399	-0.18399

	lhs	op	rhs	mi	epc	sepc.lv	sepc.all	sepc.nox
61	arith	~~	block	3.2569	0.60993	0.60993	0.16152	0.16152
38	info	~~	arith	3.2166	0.62088	0.62088	0.17938	0.17938
70	simil	~~	coding	3.1652	-0.94894	-0.94894	-0.14543	-0.14543

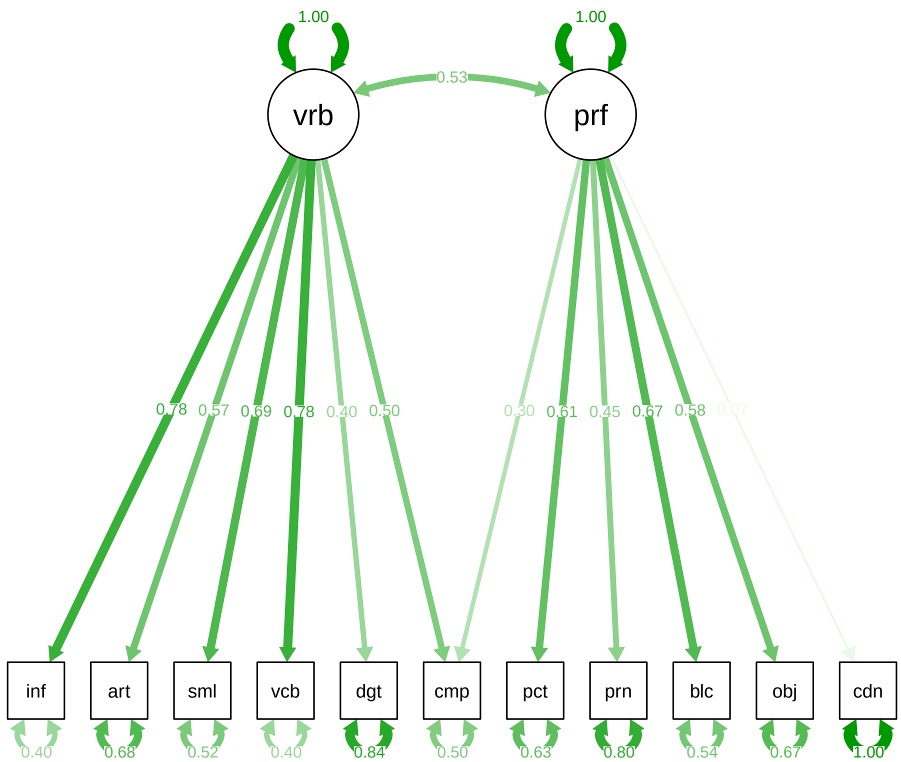
3.2.5 Path Diagram for Model 2

The `semPlot` package provides a capability of drawing path diagrams for cfa and other sem models. The `semPaths` function will take a cfa model object and draw the diagram, with several options available. The diagram produced here takes control over font/label sizes, display of residuals, and color of paths/coefficients. These and many more control options are available. There is a challenge in producing these path diagrams to have font sizes large enough for most humans to read. I've taken control of the font sizes for the "edges" with a cex argument. But this causes overlap in the values if the default layout is used. I found that "circle2" worked best here.

```
# Note that the base plot, including standardized path coefficients plots positive coefficients green
# and negative coefficients red.  Red-green colorblindness issues anyone?
# I redrew it here to choose a blue and red.  But all the coefficients in this example are
# positive,so they are shown with the skyblue.
# more challenging to use colors other than red and green.  not in this doc
semPaths(fit2, residuals=F,sizeMan=7,"std",
         posCol=c("skyblue4", "red"),
         #edge.color="skyblue4",
         edge.label.cex=1.2,layout="circle2")
```



```
# or we could draw the paths in such  a way to include the residuals:
#semPaths(fit1, sizeMan=7,"std",edge.color="skyblue4",edge.label.cex=1,layout="circle2")
# the base path diagram can be drawn much more simply:
#semPaths(fit2)
# or
semPaths(fit2,"std")
```



3.3 Compare Model 1 and Model 2

Model 1 is first/base. Model 2 adds a path from Perf to comp... Compare these models.

```
kable(anova(fit1,fit2), booktabs=TRUE, format="markdown")
```

	Df	AIC	BIC	Chisq	Chisq diff	Df diff	Pr(>Chisq)
fit2	42	9021.6	9097.6	60.642	NA	NA	NA
fit1	43	9029.6	9102.4	70.640	9.998	1	0.00157

3.4 An additional perspective on estimation and optimization

Subtle differences in algorithmic strategies for Structural Equation Modeling exist among software packages. Users are often familiar with only the default approaches and will only be moved to learn other approaches when the default approach fails to converge or produces problematic models. The student first confronted with these methods will often assume that the same SEM model evauated in different software (e.g., LISREL, MPlus, EQS, lavaan, sem, OpenMx) will produce the same model outcome. This may not be a safe assumption. Different "dialects" exist for the various software products. The commercial products may use algorithms that are proprietary and not available to understand their approach. The open source products (e.g., lavaan) have made source code available for inspection.

The perceptive reader will notice that the default solutions given by the three R packages employed in this document (**lavaan**, **sem**, **OpenMx**) all give the same values for parameter estimates and goodnes of fit statistics for the same two models run with each. However, there are slight differences in these quantities compared to the published LISREL analysis of the same data in the Tabachnik, et al textbook ((2019)). Other readers will have noticed that the LISREL output in this textbook matches SAS output that they may have seen with class coverage of the CFA topic. The R packages, while agreeing with each other, vary slightly from the LISREL and SAS values. The degree of difference is slight, but its existence may puzzle some students. The answer to understanding these differences goes well beyond the scope of this document and involves an advanced understanding of the computational algorithms employed. Even though all of the approaches are Maximum Liklihood methods some optimization and estimation strategies can differ.

The **lavaan** package permits some insight into this with one of the argments available for the `cfa` function used in this chapter. The reader might want to examine the help docs for this function: `?cfa`. That help page directs readers to another help document on Lavaan Options (`lavoptions`). There, one can find an argument that can be passed to `cfa`, called "mimic". Here is the text from that section, describing the various "mimic" possibilities:

"If"Mplus", an attempt is made to mimic the Mplus program. If"EQS", an attempt is made to mimic the EQS program. If"default", the value is (currently) set to to"lavaan", which is very close to"Mplus"."

The reader may have been exposed to a SAS Proc Calis approach to this problem that employed the default Method called LINEQS. The following rerun of the first model from this chapter above, employs the mimic argument to be specified as "EQS". The product of this model is a set of parameter values and fit statistics (e.g., Chi Sq) that match the SAS output (and the Tabachnick et al LISREL output) exactly. Demonstrating the equivalence with the addition of the mimic argument does not fully explain why such differences originally existed, but that, again, is well beyond the scope of this document. The reference section to this document includes some articles that address these differences (El-Sheikh et al., 2017; Narayanan, 2012). Rosseel (2012) has discussed use of the 'mimic' function that guides **lavaan** to emulate the approach of some of the commercial products. His website is also a valuable resource on this [http://lavaan.ugent.be/].

```
wisc.model1 <- "verbal  =~ info + comp + arith + simil + vocab + digit
                 performance =~ pictcomp + parang + block + object + coding"
```

```
fit1eqs <- lavaan::cfa(wisc.model1, data=wisc2,std.lv=TRUE, mimic="EQS")
summary(fit1eqs, fit.measures=T,standardized=T)
```

```
## lavaan 0.6-3 ended normally after 25 iterations
##
##   Optimization method                           NLMINB
##   Number of free parameters                         23
##
##   Number of observations                           175
##
##   Estimator                                         ML
##   Model Fit Test Statistic                      70.236
##   Degrees of freedom                                43
##   P-value (Chi-square)                           0.005
##
## Model test baseline model:
##
##   Minimum Function Test Statistic              516.237
##   Degrees of freedom                                55
##   P-value                                        0.000
```

```
##
## User model versus baseline model:
##
##   Comparative Fit Index (CFI)                    0.941
##   Tucker-Lewis Index (TLI)                       0.924
##
## Loglikelihood and Information Criteria:
##
##   Loglikelihood user model (H0)              -4497.337
##   Loglikelihood unrestricted model (H1)      -4462.018
##
##   Number of free parameters                         23
##   Akaike (AIC)                                9040.675
##   Bayesian (BIC)                              9113.465
##   Sample-size adjusted Bayesian (BIC)         9040.631
##
## Root Mean Square Error of Approximation:
##
##   RMSEA                                          0.060
##   90 Percent Confidence Interval          0.033  0.085
##   P-value RMSEA <= 0.05                          0.239
##
## Standardized Root Mean Square Residual:
##
##   SRMR                                           0.059
##
## Parameter Estimates:
##
##   Information                                 Expected
##   Information saturated (h1) model          Structured
##   Standard Errors                             Standard
##
## Latent Variables:
##                    Estimate  Std.Err  z-value  P(>|z|)   Std.lv  Std.all
##   verbal =~
##     info              2.212    0.201   10.997    0.000    2.212    0.760
##     comp              2.048    0.212    9.682    0.000    2.048    0.691
##     arith             1.304    0.173    7.534    0.000    1.304    0.565
##     simil             2.238    0.226    9.911    0.000    2.238    0.703
##     vocab             2.257    0.202   11.193    0.000    2.257    0.770
##     digit             1.056    0.213    4.952    0.000    1.056    0.390
##   performance =~
##     pictcomp          1.747    0.244    7.166    0.000    1.747    0.595
##     parang            1.257    0.226    5.566    0.000    1.257    0.473
##     block             1.851    0.223    8.287    0.000    1.851    0.683
##     object            1.609    0.237    6.780    0.000    1.609    0.566
##     coding            0.208    0.256    0.811    0.417    0.208    0.072
##
## Covariances:
##                    Estimate  Std.Err  z-value  P(>|z|)   Std.lv  Std.all
##   verbal ~~
##     performance       0.589    0.076    7.792    0.000    0.589    0.589
##
## Variances:
```

```
##                    Estimate  Std.Err  z-value  P(>|z|)   Std.lv  Std.all
##    .info             3.586    0.511    7.014    0.000    3.586    0.423
##    .comp             4.599    0.590    7.793    0.000    4.599    0.523
##    .arith            3.623    0.424    8.547    0.000    3.623    0.681
##    .simil            5.125    0.667    7.680    0.000    5.125    0.506
##    .vocab            3.507    0.511    6.866    0.000    3.507    0.408
##    .digit            6.198    0.686    9.030    0.000    6.198    0.848
##    .pictcomp         5.558    0.764    7.276    0.000    5.558    0.646
##    .parang           5.494    0.664    8.275    0.000    5.494    0.777
##    .block            3.916    0.646    6.066    0.000    3.916    0.533
##    .object           5.499    0.726    7.578    0.000    5.499    0.680
##    .coding           8.206    0.882    9.309    0.000    8.206    0.995
##     verbal           1.000                               1.000    1.000
##     performance      1.000                               1.000    1.000
```

Chapter 4

Using the sem package for CFA

In this chapter, we use the **sem** package to implement the same two CFA analyses that we produced with **lavaan** in chapter 3. **sem** provides an equally simple way to obtain the models and only the basics are shown here. The code in this chapter is modeled after a document by James Steiger

4.1 Example one

Once again, the bifactor model with Verbal and Performance factors is specified. Each manifest factor has a path from only one of the two factors.

4.1.1 Data Setup

In **sem**, it is helpful to have covariance and correlation matrices available as objects, as well ans a sample size object.

```
# same data file and extraction of the wisc2 data frame with only the 11 manifests
#wisc1 <- read.csv("wisc1.csv")
#wisc2 <- wisc1[,2:12]
# covariance and correlation matrices are saved as objects
covwisc <- cov(wisc2)
corwisc <-cor(wisc2)
# list of manifest variables for potential use
manifests <- names(wisc2)
# this gives an object that is the sample size
wobs <- length(wisc2)
```

We also need to load the package.

```
library(sem)
```

4.1.2 Define the first model

In **sem**, the structure of the model is created with a text string to define the paths. The `specifyModel` function permits this in several ways. The simplest is to enter text as an argument. Some explanation of the strucure is needed.

- Each line defines a path, a label for the parameter, and the starting value for the parameter value.
- This is symbolized as: , , .

- Note that paths can be double or single-headed arrows.
- The unique name specified by the parameter symbol is for free parameters.
- If the parameter value is NA, then its starting point value is system determined.
- A numerical value following the NA can fix a variance at the value. E.g.,: F1 <-> F1, NA, 1 would fix the factor variance a 1.
- Unique variances can be specified for manifest variables. e.g.,: manifest1 <-> manifest1, e1, NA. If they are not specified, they default to "free to vary"
- Factors can be set to a fixed relationship to each other, e.g, 0, or 1. Or they can be left free (estimable) as in the example here.

```
# this text could have been saved in a file and read in with the file argument to be efficient
# commented here to show the argument options
# m1.model <- specifyModel(file="sem1.txt")
m1.model <- specifyModel(text="
  ## Factor 1 is Verbal
  Verbal -> info, t01, NA
  Verbal -> comp, t02, NA
  Verbal -> arith, t03, NA
  Verbal -> simil, t04, NA
  Verbal -> vocab, t05, NA
  Verbal -> digit, t06, NA
  ## Factor 2 is performance
  Performance -> pictcomp, t07, NA
  Performance -> parang, t08, NA
  Performance -> block, t09, NA
  Performance -> object, t10, NA
  Performance -> coding, t11, NA
  ## Set factor variances
  Verbal <-> Verbal, NA, 1
  Performance <-> Performance, NA, 1
  ## Set factor covariance to be estimable
  Verbal <-> Performance, p1, NA"
)
```

```
## NOTE: it is generally simpler to use specifyEquations() or cfa()
##       see ?specifyEquations
```

```
## NOTE: adding 11 variances to the model
```

This m1.model is now available to be used in the model function. The first "note" refers to the fact that the model can be specified interactivly (I think). I found it easier to do this way, and more reproducible.

4.1.3 Fit model 1 and examine the results

In this chapter, we will just look at the basics of the model fit and draw the path diagram. Other aspects of the model can be extracted, but are passed over here to save space. The interested reader might try `str(m1)` to see what is available from the model object.

```
# all that is required is the model specification, the covariance matrix, and the sample size
m1 <- sem(m1.model,covwisc,175)
options(digits=5)
summary(m1)
```

```
##
##  Model Chisquare =  70.236   Df =  43 Pr(>Chisq) = 0.0054454
##  AIC =  116.24
##  BIC =  -151.85
##
##  Normalized Residuals
##     Min.  1st Qu.   Median     Mean  3rd Qu.     Max.
## -1.88413 -0.41375 -0.00001  0.03511  0.45976  2.11208
##
##  R-square for Endogenous Variables
##     info     comp    arith    simil    vocab    digit pictcomp   parang
##   0.5772   0.4770   0.3193   0.4944   0.5922   0.1524   0.3545   0.2233
##    block   object   coding
##   0.4667   0.3202   0.0052
##
##  Parameter Estimates
##             Estimate Std Error z value  Pr(>|z|)
## t01          2.21249  0.201190  10.99700 3.9507e-28
## t02          2.04806  0.211541   9.68163 3.6090e-22
## t03          1.30357  0.173035   7.53358 4.9367e-14
## t04          2.23845  0.225846   9.91142 3.7135e-23
## t05          2.25691  0.201642  11.19269 4.4281e-29
## t06          1.05579  0.213185   4.95244 7.3287e-07
## t07          1.74699  0.243776   7.16638 7.7008e-13
## t08          1.25683  0.225785   5.56647 2.5995e-08
## t09          1.85123  0.223392   8.28693 1.1621e-16
## t10          1.60918  0.237324   6.78052 1.1974e-11
## t11          0.20761  0.255890   0.81132 4.1718e-01
## p1           0.58883  0.075569   7.79198 6.5965e-15
## V[info]      3.58620  0.511281   7.01414 2.3137e-12
## V[comp]      4.59856  0.590106   7.79277 6.5556e-15
## V[arith]     3.62254  0.423850   8.54675 1.2660e-17
## V[simil]     5.12484  0.667296   7.68001 1.5908e-14
## V[vocab]     3.50720  0.510787   6.86626 6.5906e-12
## V[digit]     6.19783  0.686345   9.03020 1.7136e-19
## V[pictcomp]  5.55770  0.763882   7.27560 3.4488e-13
## V[parang]    5.49428  0.663998   8.27454 1.2895e-16
## V[block]     3.91612  0.645638   6.06550 1.3154e-09
## V[object]    5.49872  0.725619   7.57798 3.5099e-14
## V[coding]    8.20596  0.881552   9.30853 1.2961e-20
##
## t01         info <--- Verbal
## t02         comp <--- Verbal
## t03         arith <--- Verbal
```

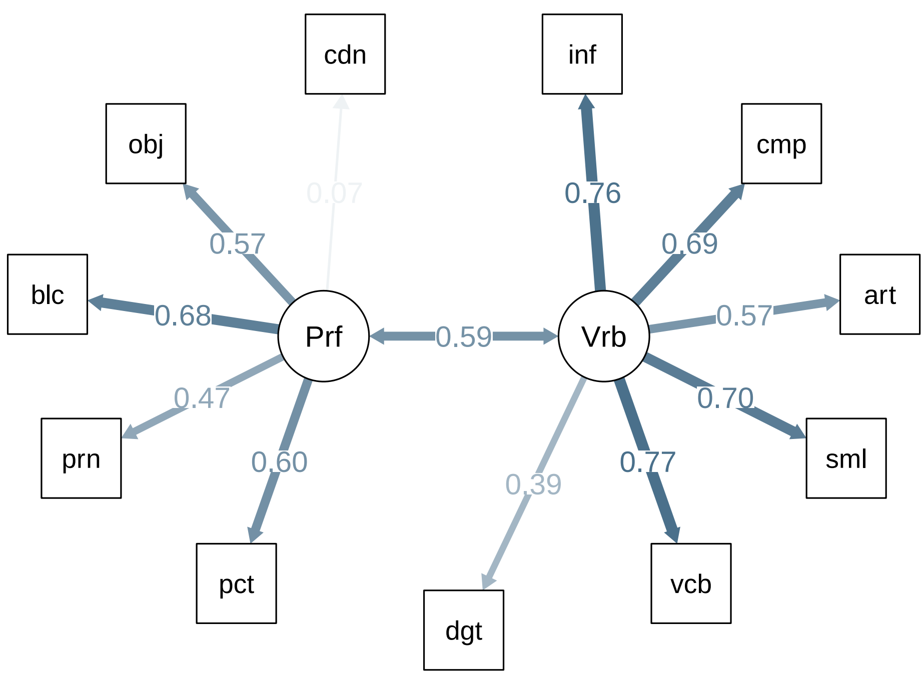
```
## t04         simil <--- Verbal
## t05         vocab <--- Verbal
## t06         digit <--- Verbal
## t07         pictcomp <--- Performance
## t08         parang <--- Performance
## t09         block <--- Performance
## t10         object <--- Performance
## t11         coding <--- Performance
## p1          Performance <--> Verbal
## V[info]     info <--> info
## V[comp]     comp <--> comp
## V[arith]    arith <--> arith
## V[simil]    simil <--> simil
## V[vocab]    vocab <--> vocab
## V[digit]    digit <--> digit
## V[pictcomp] pictcomp <--> pictcomp
## V[parang]   parang <--> parang
## V[block]    block <--> block
## V[object]   object <--> object
## V[coding]   coding <--> coding
##
##  Iterations =  74
```

The table listing of parameter estimates uses the labeling strategy that was defined in the `modelSpecify` function. The names, such as theta01 are arbitrary.

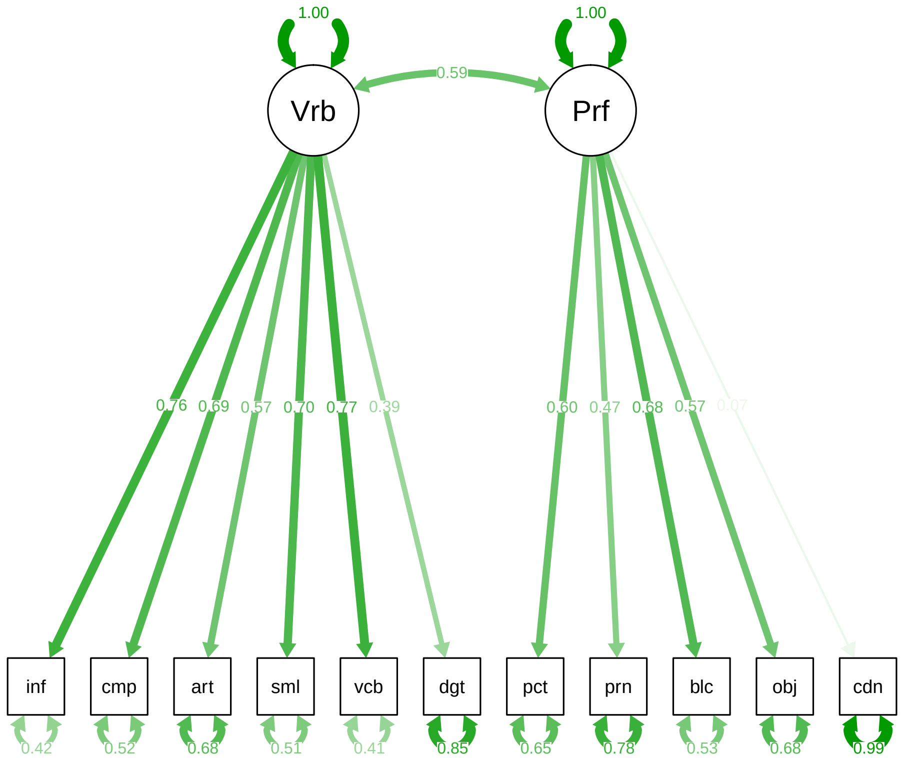
4.1.4 Can we draw the path diagram from a sem model?

The `semPaths` function from **semPlot** is capable of recognizing a model object from **sem**. This code is identical to what was used in chapter 2 for the **lavaan** object. The fit object name is just changed to m1 here.

```
# Note that the base plot, including standardized path coefficients plots positive coefficients green
# and negative coefficients red.  Red-green colorblindness issues anyone?
# I redrew it here to choose a blue and red.  But all the coefficients in this example are
# positive,so they are shown with the skyblue.
# more challenging to use colors other than red and green.  not in this doc
semPaths(m1, residuals=F,sizeMan=7,"std",
         posCol=c("skyblue4", "red"),
         #edge.color="skyblue4",
         edge.label.cex=1.2,layout="circle2")
```




```
# or we could draw the paths in such  a way to include the residuals:
#semPaths(m1, sizeMan=7,"std",edge.color="skyblue4",edge.label.cex=1,layout="circle2")
# the base path diagram can be drawn much more simply:
#semPaths(m1)
# or
semPaths(m1,"std")
```


1.00
1.00
0.59
Vrb
Prf
0.76
0.69
0.57
0.70
0.77
0.39
0.60
0.47
0.68
0.57
inf
cmp
art
sml
vcb
dgt
pct
prn
blc
obj
cdn
0.42
0.52
0.68
0.51
0.41
0.85
0.65
0.78
0.53
0.68
0.99

4.2 Example two

Model with added path still under construction in this chapter.

4.2.1 Define the second model

In the `specifyModel` text argument we just need to add one path, from Performance to "comp", rerun the model, and compare to the first.

```
# this text could have been saved in a file and read in with the file argument to be efficient
# commented here to show the argument options
# m1.model <- specifyModel(file="sem2.txt")
m2.model <- specifyModel(text="
  ## Factor 1 is Verbal
  Verbal -> info, t01, NA
  Verbal -> comp, t02, NA
  Verbal -> arith, t03, NA
  Verbal -> simil, t04, NA
  Verbal -> vocab, t05, NA
  Verbal -> digit, t06, NA
  ## Factor 2 is performance
  Performance -> pictcomp, t07, NA
  Performance -> parang, t08, NA
  Performance -> block, t09, NA
  Performance -> object, t10, NA
  Performance -> coding, t11, NA
  Performance -> comp, t12, NA
  ## Set factor variances
  Verbal <-> Verbal, NA, 1
  Performance <-> Performance, NA, 1
  ## Set factor covariance to be estimable
  Verbal <-> Performance, p1, NA"
)
```

```
## NOTE: it is generally simpler to use specifyEquations() or cfa()
##       see ?specifyEquations

## NOTE: adding 11 variances to the model
```

4.2.2 Fit model 2 and examine the results

In this chapter, we will just look at the basics of the model fit and draw the path diagram. Other aspects of the model can be extracted, but are passed over here to save space. The interested reader might try `str(m1)` to see what is available from the model object.

```
# all that is required is the model specification, the covariance matrix, and the sample size
m2 <- sem(m2.model,covwisc,175)
options(digits=5)
summary(m2)
```

```
##
##  Model Chisquare =  60.295   Df =  42 Pr(>Chisq) = 0.033354
##  AIC =  108.3
##  BIC =  -156.63
##
##  Normalized Residuals
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
```

```
## -1.7087 -0.3670  0.0000  0.0439  0.4509  2.0855
##
##  R-square for Endogenous Variables
##     info     comp    arith    simil    vocab    digit pictcomp   parang
##   0.6036   0.5046   0.3227   0.4823   0.6044   0.1589   0.3741   0.2009
##    block   object   coding
##   0.4551   0.3315   0.0049
##
##  Parameter Estimates
##             Estimate Std Error z value  Pr(>|z|)
## t01         2.26254  0.200471  11.28609 1.5373e-29
## t02         1.49558  0.255230   5.85973 4.6361e-09
## t03         1.31053  0.173300   7.56223 3.9620e-14
## t04         2.21107  0.227463   9.72056 2.4641e-22
## t05         2.28001  0.201829  11.29675 1.3616e-29
## t06         1.07784  0.213286   5.05351 4.3377e-07
## t07         1.79476  0.240149   7.47352 7.8078e-14
## t08         1.19223  0.224879   5.30164 1.1477e-07
## t09         1.82799  0.219975   8.30998 9.5719e-17
## t10         1.63752  0.234254   6.99037 2.7416e-12
## t11         0.20104  0.254172   0.79098 4.2896e-01
## t12         0.88667  0.267519   3.31443 9.1829e-04
## p1          0.53322  0.081099   6.57487 4.8695e-11
## V[info]     3.36224  0.505946   6.64546 3.0227e-11
## V[comp]     4.35598  0.558735   7.79614 6.3830e-15
## V[arith]    3.60434  0.423598   8.50888 1.7563e-17
## V[simil]    5.24665  0.681040   7.70388 1.3199e-14
## V[vocab]    3.40241  0.512709   6.63614 3.2200e-11
## V[digit]    6.15077  0.682983   9.00574 2.1421e-19
## V[pictcomp] 5.38850  0.747358   7.21007 5.5922e-13
## V[parang]   5.65249  0.668468   8.45589 2.7697e-17
## V[block]    4.00164  0.628568   6.36628 1.9367e-10
## V[object]   5.40673  0.713201   7.58094 3.4306e-14
## V[coding]   8.20865  0.881648   9.31057 1.2715e-20
##
## t01         info <--- Verbal
## t02         comp <--- Verbal
## t03         arith <--- Verbal
## t04         simil <--- Verbal
## t05         vocab <--- Verbal
## t06         digit <--- Verbal
## t07         pictcomp <--- Performance
## t08         parang <--- Performance
## t09         block <--- Performance
## t10         object <--- Performance
## t11         coding <--- Performance
## t12         comp <--- Performance
## p1          Performance <--> Verbal
## V[info]     info <--> info
## V[comp]     comp <--> comp
## V[arith]    arith <--> arith
## V[simil]    simil <--> simil
## V[vocab]    vocab <--> vocab
## V[digit]    digit <--> digit
```

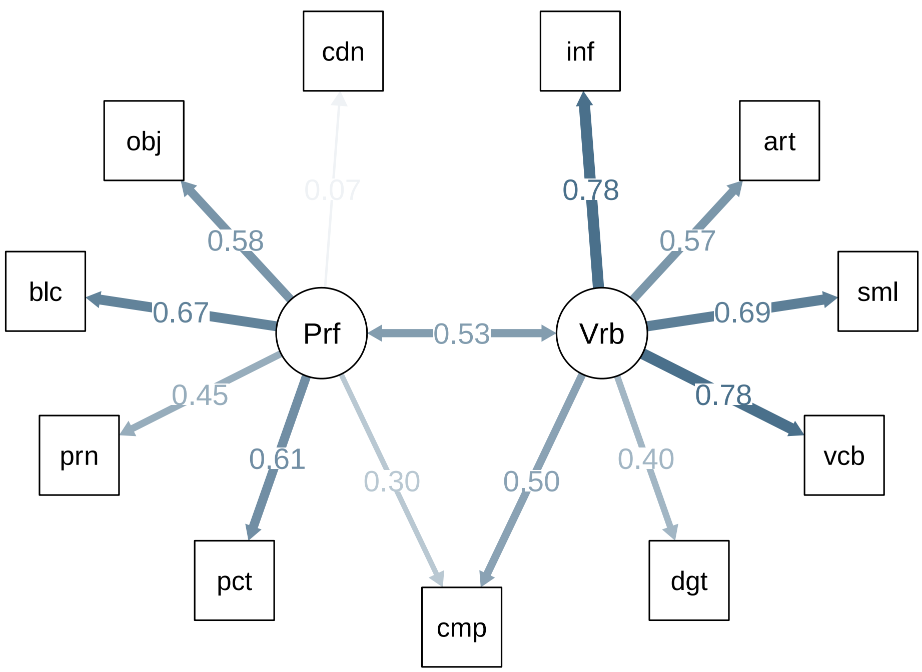
```
## V[pictcomp] pictcomp <--> pictcomp
## V[parang]   parang <--> parang
## V[block]    block <--> block
## V[object]   object <--> object
## V[coding]   coding <--> coding
##
##  Iterations =  37
```

The table listing of parameter estimates uses the labeling strategy that was defined in the `modelSpecify` function. The names, such as theta01 are arbitrary.

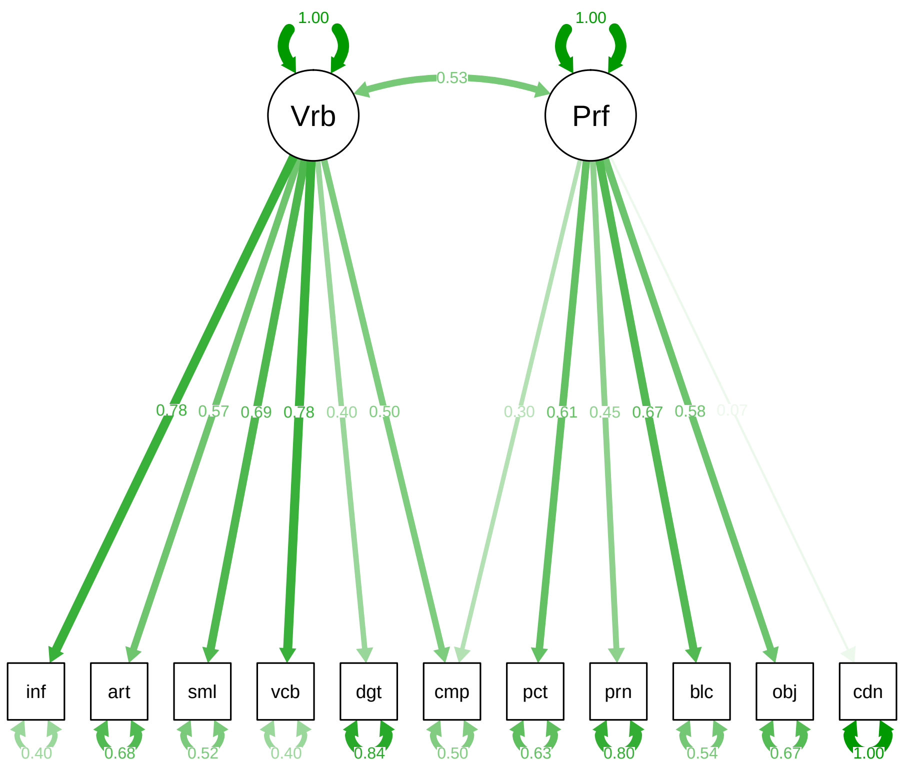
4.2.3 Can we draw the path diagram from a sem model?

The `semPaths` function from **semPlot** is capable of recognizing a model object from **sem**. This code is identical to what was used in chapter 2 for the **lavaan** object. The fit object name is just changed to m2 here.

```
# Note that the base plot, including standardized path coefficients plots positive coefficients green
# and negative coefficients red.  Red-green colorblindness issues anyone?
# I redrew it here to choose a blue and red.  But all the coefficients in this example are
# positive,so they are shown with the skyblue.
# more challenging to use colors other than red and green.  not in this doc
semPaths(m2, residuals=F,sizeMan=7,"std",
         posCol=c("skyblue4", "red"),
         #edge.color="skyblue4",
         edge.label.cex=1.2,layout="circle2")
```




```
# or we could draw the paths in such  a way to include the residuals:
#semPaths(m1, sizeMan=7,"std",edge.color="skyblue4",edge.label.cex=1,layout="circle2")
# the base path diagram can be drawn much more simply:
#semPaths(m1)
# or
semPaths(m2,"std")
```



4.3 Compare the two CFA models produced by sem

We can use the anova function to compare the two models, as we did for **lavaan**.

```
kable(anova(m2,m1), booktabs=TRUE, format="markdown")
```

	Model Df	Model Chisq	Df	LR Chisq	Pr(>Chisq)
m2	42	60.295	NA	NA	NA
m1	43	70.236	1	9.9409	0.00162

Chapter 5

Using the OpenMx Package for CFA

The **OpenMX** package in R is a port of the well-respected MX analytical software. It handles SEM and can easily be used for CFA. Here, the same two models that were run in **lavaan** will be run again, but an additional model will be run first.

```
library(OpenMx)
```

5.1 First OpenMx Model - Single Factor

First, fit a bad model that posits only one underlying factor

5.1.1 set new dataframe and set up basics

Some initial setups

```
# grab data
#wisc2 <- wisc1[2:12]   # remove ID from df
# set up basics
manifests=names(wisc2)
observedCov <- cov(wisc2)
numSubjects <- nrow(wisc2)
```

5.1.2 Create the model using the mxModel function

This code initializes the model and sets vars/paths

```
latents="F1"
cfa2a <- mxModel("Common Factor Model",type="RAM",
                 manifestVars = manifests, latentVars = latents,
                 # Now set the residual variance for manifest variables
                 mxPath(from=manifests, arrows=2,free=T,values=1,labels=paste("error",1:11,sep="")),
                 # set latent factor variance to 1
                 mxPath(from="F1",arrows=2,free=F,values=1,labels="varF1"),
                 # specify factor loadings
                 mxPath(from="F1",to=manifests, arrows=1,
                        free=T,values=1,labels=paste("i", 1:11,sep="")),
                 # specify the covariance matrix
                 mxData(observed=observedCov,type="cov",numObs=numSubjects)
) # close the model
```

5.1.3 Now use `mxRun` to fit the model

`mxRun` uses the model defined above.

```
cfa2a <- mxRun(cfa2a)
```

```
## Running Common Factor Model with 22 parameters
```

Summarize the fit:

```
summary(cfa2a)
```

```
## Summary of Common Factor Model
##
## free parameters:
##       name matrix      row      col Estimate Std.Error A
## 1       i1      A     info       F1  2.10843   0.20489
## 2       i2      A     comp       F1  2.10258   0.20876
## 3       i3      A    arith       F1  1.27253   0.17310
## 4       i4      A    simil       F1  2.25978   0.22323
## 5       i5      A    vocab       F1  2.17498   0.20342
## 6       i6      A    digit       F1  1.00892   0.21308
## 7       i7      A pictcomp       F1  1.35055   0.22867
## 8       i8      A   parang       F1  0.89834   0.21227
## 9       i9      A    block       F1  1.23160   0.21130
## 10     i10      A   object       F1  1.01784   0.22717
## 11     i11      A   coding       F1  0.20128   0.23480
## 12  error1      S     info     info  3.98738   0.54543
## 13  error2      S     comp     comp  4.32199   0.56950
## 14  error3      S    arith    arith  3.67209   0.42704
## 15  error4      S    simil    simil  4.97100   0.64992
## 16  error5      S    vocab    vocab  3.82117   0.53031
## 17  error6      S    digit    digit  6.25280   0.68878
## 18  error7      S pictcomp pictcomp  6.73647   0.76246
## 19  error8      S   parang   parang  6.22646   0.68288
## 20  error9      S    block    block  5.78440   0.65307
## 21 error10      S   object   object  7.00600   0.77245
## 22 error11      S   coding   coding  8.16142   0.87333
##
## Model Statistics:
##              |  Parameters  |  Degrees of Freedom  |  Fit (-2lnL units)
##       Model:             22                     44              5492.0
##   Saturated:             66                      0              5375.1
## Independence:            11                     55              5894.3
## Number of observations/statistics: 175/66
##
## chi-square:  <U+03C7>² ( df=44 ) = 116.85,  p = 1.5993e-08
## Information Criteria:
##       |  df Penalty  |  Parameters Penalty  |  Sample-Size Adjusted
## AIC:         28.851                 160.85                   167.51
## BIC:       -110.400                 230.48                   160.81
## CFI: 0.84306
## TLI: 0.80383   (also known as NNFI)
## RMSEA:  0.097268  [95% CI (0.071763, 0.12282)]
## Prob(RMSEA <= 0.05): 0.00026603
## timestamp: 2019-07-11 11:27:19
```

```
## Wall clock time: 0.055 secs
## optimizer:  CSOLNP
## OpenMx version number: 2.12.2
## Need help?  See help(mxSummary)
```

What is in the model?

```r
slotNames(cfa2a@output)
```

```
## NULL
```

```r
names(cfa2a@output)
```

```
##  [1] "matrices"                "algebras"
##  [3] "data"                    "SaturatedLikelihood"
##  [5] "IndependenceLikelihood"  "calculatedHessian"
##  [7] "standardErrors"          "gradient"
##  [9] "hessian"                 "expectations"
## [11] "fit"                     "fitUnits"
## [13] "Minus2LogLikelihood"     "maxRelativeOrdinalError"
## [15] "minimum"                 "estimate"
## [17] "infoDefinite"            "conditionNumber"
## [19] "status"                  "iterations"
## [21] "evaluations"             "mxVersion"
## [23] "frontendTime"            "backendTime"
## [25] "independentTime"         "wallTime"
## [27] "timestamp"               "cpuTime"
```

5.2 Second OpenMx Model - the bifactor model

Now fit a model that is the same as the initial model fit with **lavaan** in chapter 3. Two factors, verbal and performance are established with each manifest variable uniquely specified by only one of the two latent factors.

5.2.1 set new dataframe and set up basics

Same initial setups as above.

```
# grab data
wisc2 <- wisc1[2:12]   # remove ID from df
# set up basics
manifests <- names(wisc2)
verbalVars <- names(wisc2[1:6])
perfVars <- names(wisc2[7:11])
latents2 <- c("verbal","perf")
#manifests <- c("verbalVars","perfVars")
observedCov <- cov(wisc2[,1:11])
numSubjects <- nrow(wisc2)
```

5.2.2 Create the model using the mxModel function

Essentially initializes the model and sets vars/paths

```
cfa2b <- mxModel("Two Factor Model",type="RAM",
                 manifestVars = manifests, latentVars = latents2,
                 # Now set the residual variance for manifest variables
                 mxPath(from = manifests, arrows=2,free=T,values=1, labels=paste("e",1:11,sep="")),
                 # set latent factor variances
                 mxPath(from=latents2,arrows=2,free=F,values=1,labels=c("p1","p2")),
                 # speciffy factor loadings
                 # Allow the factors to covary as per the lavaan model
                 mxPath(from="verbal", to="perf", arrows=2,
                        free=T, values=1, labels="latcov1"),
                 # Specify the latent factor paths to manifests
                 mxPath(from="verbal", to = verbalVars,
                        arrows=1,
                        free=T,values=1),
                 mxPath(from="perf", to = perfVars,
                        arrows=1,
                        free=T,values=1),
                 mxData(observed=observedCov,type="cov",numObs=numSubjects)
) # close the model
```

5.2.3 Now use 'mxRun'to fit the model

mxRun uses the model defined above

```
cfa2b <- mxRun(cfa2b)
```

```
## Running Two Factor Model with 23 parameters
```

Summarize the fit

```
summary(cfa2b)
```

```
## Summary of Two Factor Model
##
## free parameters:
##                          name matrix      row      col Estimate Std.Error A
## 1   Two Factor Model.A[1,12]      A     info   verbal  2.20616  0.201309
## 2   Two Factor Model.A[2,12]      A     comp   verbal  2.04220  0.211937
## 3   Two Factor Model.A[3,12]      A    arith   verbal  1.29984  0.172679
## 4   Two Factor Model.A[4,12]      A    simil   verbal  2.23205  0.225198
## 5   Two Factor Model.A[5,12]      A    vocab   verbal  2.25045  0.200598
## 6   Two Factor Model.A[6,12]      A    digit   verbal  1.05277  0.212308
## 7   Two Factor Model.A[7,13]      A pictcomp     perf  1.74200  0.246173
## 8   Two Factor Model.A[8,13]      A   parang     perf  1.25323  0.224773
## 9   Two Factor Model.A[9,13]      A    block     perf  1.84593  0.224562
## 10 Two Factor Model.A[10,13]      A   object     perf  1.60457  0.236048
## 11 Two Factor Model.A[11,13]      A   coding     perf  0.20701  0.257118
## 12                         e1      S     info     info  3.56570  0.516752
## 13                         e2      S     comp     comp  4.57229  0.594874
## 14                         e3      S    arith    arith  3.60184  0.422011
## 15                         e4      S    simil    simil  5.09555  0.666203
## 16                         e5      S    vocab    vocab  3.48717  0.507536
## 17                         e6      S    digit    digit  6.16241  0.680961
## 18                         e7      S pictcomp pictcomp  5.52590  0.772061
## 19                         e8      S   parang   parang  5.46287  0.658943
## 20                         e9      S    block    block  3.89376  0.651701
## 21                        e10      S   object   object  5.46733  0.719732
## 22                        e11      S   coding   coding  8.15907  0.874195
## 23                    latcov1      S   verbal     perf  0.58883  0.077178
##
## Model Statistics:
##                |  Parameters  |  Degrees of Freedom  |  Fit (-2lnL units)
##        Model:             23                     43               5445.7
##    Saturated:             66                      0               5375.1
## Independence:             11                     55               5894.3
## Number of observations/statistics: 175/66
##
## chi-square:  <U+03C7>² ( df=43 ) = 70.608,  p = 0.0050089
## Information Criteria:
##       |  df Penalty  |  Parameters Penalty  |  Sample-Size Adjusted
## AIC:       -15.392               116.61                   123.92
## BIC:      -151.478               189.40                   116.56
## CFI: 0.94053
## TLI: 0.92393   (also known as NNFI)
## RMSEA:  0.060571  [95% CI (0.026575, 0.089595)]
## Prob(RMSEA <= 0.05): 0.23348
```

```
## timestamp: 2019-07-11 11:27:19
## Wall clock time: 0.0322 secs
## optimizer:  CSOLNP
## OpenMx version number: 2.12.2
## Need help?  See help(mxSummary)
```

What is in the model?

```
slotNames(cfa2b@output)
```

```
## NULL
```

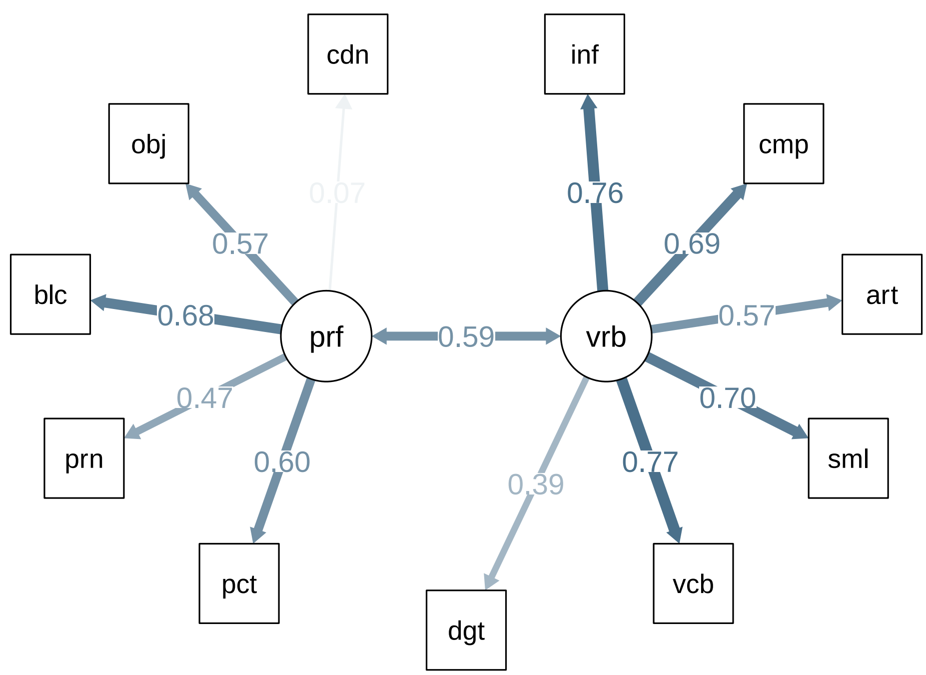
```
names(cfa2b@output)
```

```
##  [1] "matrices"                "algebras"
##  [3] "data"                    "SaturatedLikelihood"
##  [5] "IndependenceLikelihood"  "calculatedHessian"
##  [7] "standardErrors"          "gradient"
##  [9] "hessian"                 "expectations"
## [11] "fit"                     "fitUnits"
## [13] "Minus2LogLikelihood"     "maxRelativeOrdinalError"
## [15] "minimum"                 "estimate"
## [17] "infoDefinite"            "conditionNumber"
## [19] "status"                  "iterations"
## [21] "evaluations"             "mxVersion"
## [23] "frontendTime"            "backendTime"
## [25] "independentTime"         "wallTime"
## [27] "timestamp"               "cpuTime"
```

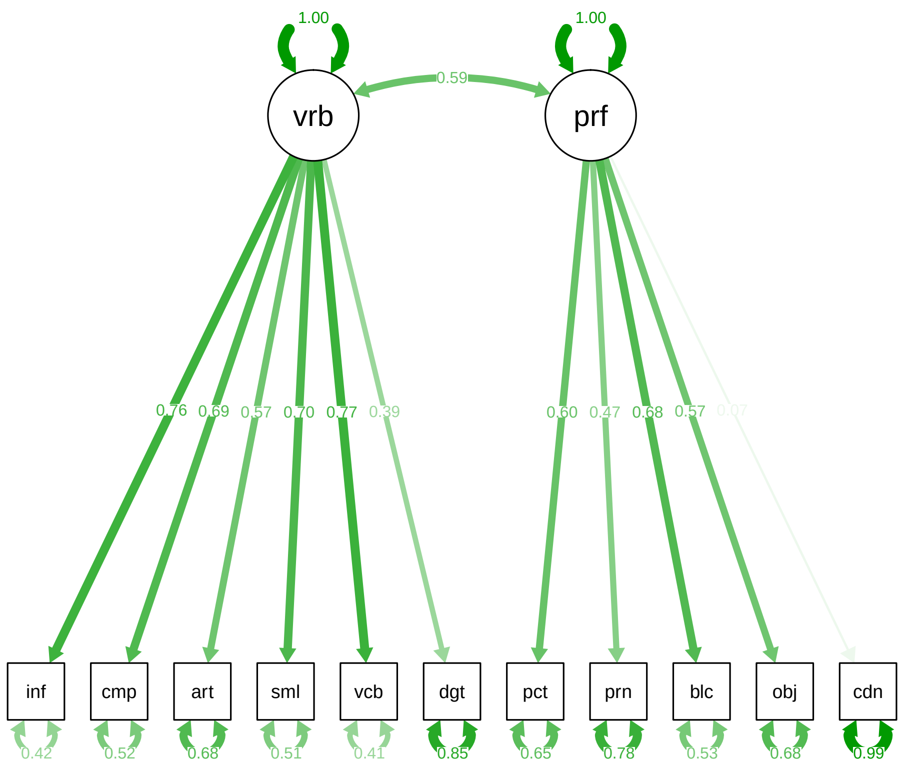
5.2.4 Can we use semPlot to draw the OpenMx model fit?

The **semPlot** package is very powerful and can recognize many lm and sem model objects. We can use the identical code that we used in chapter 2 for the **lavaan** model.

```
# Note that the base plot, including standardized path coefficients plots positive coefficients green
# and negative coefficients red.  Red-green colorblindness issues anyone?
# I redrew it here to choose a blue and red.  But all the coefficients in this example are
# positive,so they are shown with the skyblue.
# more challenging to use colors other than red and green.  not in this doc
semPaths(cfa2b, residuals=F,sizeMan=7,"std",
         posCol=c("skyblue4", "red"),
         #edge.color="skyblue4",
         edge.label.cex=1.2,layout="circle2")
```




```
# or we could draw the paths in such  a way to include the residuals:
#semPaths(fit1, sizeMan=7,"std",edge.color="skyblue4",edge.label.cex=1,layout="circle2")
# the base path diagram can be drawn much more simply:
#semPaths(fit1)
# or
semPaths(cfa2b,"std")
```


1.00
1.00
0.59
vrb
prf
0.76
0.69
0.57
0.70
0.77
0.39
0.60
0.47
0.68
0.57
0.07
inf
cmp
art
sml
vcb
dgt
pct
prn
blc
obj
cdn
0.42
0.52
0.68
0.51
0.41
0.85
0.65
0.78
0.53
0.68
0.99

5.3 Third OpenMx Model

Now fit a model that is the same as the second model fit with lavaan above. Two factors, verbal and performance, plus paths from both latents to "comp".

5.3.1 set new dataframe and set up basics

Same initial setups as above.

```
# grab data
wisc2 <- wisc1[2:12]   # remove ID from df
# set up basics
manifests <- names(wisc2)
verbalVars <- names(wisc2[1:6])
perfVars <- c(names(wisc2[7:11]),"comp")
latents2 <- c("verbal","perf")
#manifests <- c("verbalVars","perfVars")
observedCov <- cov(wisc2[,1:11])
numSubjects <- nrow(wisc2)
```

5.3.2 Create the model using the mxModel function

Essentially initializes the model and sets vars/paths

```
cfa2c <- mxModel("TwoFac, Comp Common",type="RAM",
                manifestVars = manifests, latentVars = latents2,
                # Now set the residual variance for manifest variables
                mxPath(from = manifests, arrows=2,free=T,values=1, labels=paste("e",1:11,sep="")),
                # set latent factor variances
                mxPath(from=latents2,arrows=2,free=F,values=1,labels=c("p1","p2")),
                # speciffy factor loadings
                # Allow the factors to covary as per the lavaan model
                mxPath(from="verbal", to="perf", arrows=2,
                       free=T, values=1, labels="latcov1"),
                # Specify the latent factor paths to manifests
                mxPath(from="verbal", to = verbalVars,
                       arrows=1,
                       free=T,values=1),
                mxPath(from="perf", to = perfVars,
                       arrows=1,
                       free=T,values=1),
                mxData(observed=observedCov,type="cov",numObs=numSubjects)
) # close the model
```

5.3.3 Now use 'mxRun' to fit the model

mxRun uses the model defined above

```
cfa2c <- mxRun(cfa2c)
```

```
## Running TwoFac, Comp Common with 24 parameters
```

Summarize the fit

```
summary(cfa2c)
```

```
## Summary of TwoFac, Comp Common
```

```
##
## free parameters:
##                            name matrix      row      col Estimate
## 1   TwoFac, Comp Common.A[1,12]      A     info   verbal  2.25606
## 2   TwoFac, Comp Common.A[2,12]      A     comp   verbal  1.49130
## 3   TwoFac, Comp Common.A[3,12]      A    arith   verbal  1.30678
## 4   TwoFac, Comp Common.A[4,12]      A    simil   verbal  2.20475
## 5   TwoFac, Comp Common.A[5,12]      A    vocab   verbal  2.27348
## 6   TwoFac, Comp Common.A[6,12]      A    digit   verbal  1.07476
## 7   TwoFac, Comp Common.A[2,13]      A     comp     perf  0.88414
## 8   TwoFac, Comp Common.A[7,13]      A pictcomp     perf  1.78962
## 9   TwoFac, Comp Common.A[8,13]      A   parang     perf  1.18881
## 10  TwoFac, Comp Common.A[9,13]      A    block     perf  1.82276
## 11 TwoFac, Comp Common.A[10,13]      A   object     perf  1.63283
## 12 TwoFac, Comp Common.A[11,13]      A   coding     perf  0.20047
## 13                           e1      S     info     info  3.34302
## 14                           e2      S     comp     comp  4.33109
## 15                           e3      S    arith    arith  3.58375
## 16                           e4      S    simil    simil  5.21667
## 17                           e5      S    vocab    vocab  3.38297
## 18                           e6      S    digit    digit  6.11561
## 19                           e7      S pictcomp pictcomp  5.35770
## 20                           e8      S   parang   parang  5.62019
## 21                           e9      S    block    block  3.97877
## 22                          e10      S   object   object  5.37584
## 23                          e11      S   coding   coding  8.16174
## 24                      latcov1      S   verbal     perf  0.53322
##    Std.Error A
## 1   0.200315
## 2   0.256354
## 3   0.173048
## 4   0.227320
## 5   0.200572
## 6   0.212373
## 7   0.270140
## 8   0.242338
## 9   0.225100
## 10  0.221685
## 11  0.233235
## 12  0.255009
## 13  0.509394
## 14  0.557177
## 15  0.422007
## 16  0.682585
## 17  0.507400
## 18  0.677590
## 19  0.755634
## 20  0.665637
## 21  0.636905
## 22  0.708170
## 23  0.874112
## 24  0.082033
##
## Model Statistics:
```

```
##                |  Parameters  |  Degrees of Freedom  |  Fit (-2lnL units)
##        Model:             24                     42              5435.7
##    Saturated:             66                      0              5375.1
## Independence:             11                     55              5894.3
## Number of observations/statistics: 175/66
##
## chi-square:  <U+03C7>² ( df=42 ) = 60.61,  p = 0.031396
## Information Criteria:
##       |  df Penalty  |  Parameters Penalty  |  Sample-Size Adjusted
## AIC:         -23.39                108.61                  116.61
## BIC:        -156.31                184.56                  108.56
## CFI: 0.95991
## TLI: 0.9475   (also known as NNFI)
## RMSEA:  0.050319  [95% CI (0, 0.081389)]
## Prob(RMSEA <= 0.05): 0.4664
## timestamp: 2019-07-11 11:27:24
## Wall clock time: 0.0488 secs
## optimizer:  CSOLNP
## OpenMx version number: 2.12.2
## Need help?  See help(mxSummary)
```

What is in the model?

```r
slotNames(cfa2c@output)
```

```
## NULL
```

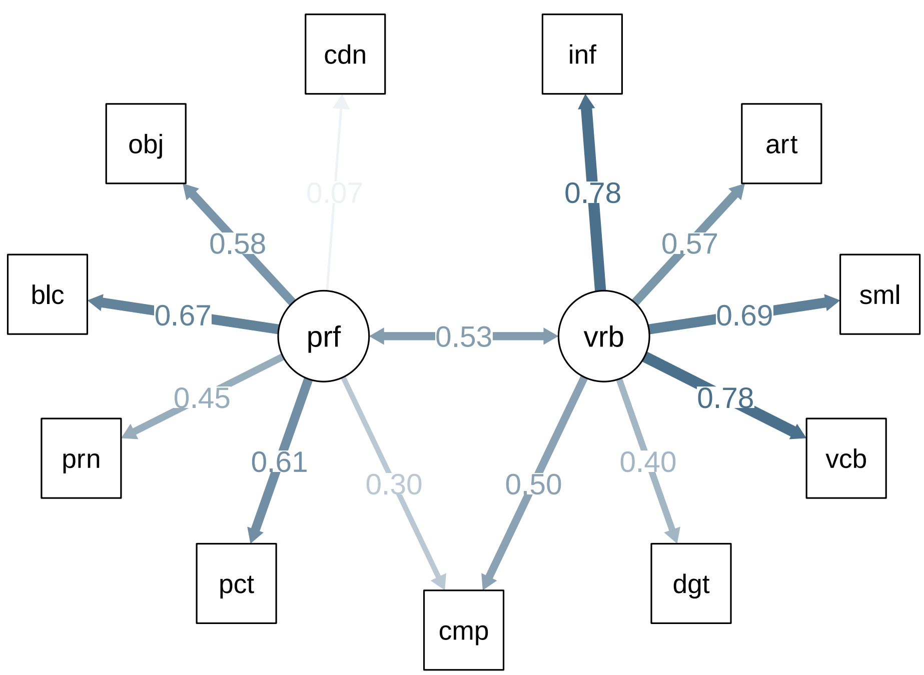
```r
names(cfa2c@output)
```

```
##  [1] "matrices"                "algebras"
##  [3] "data"                    "SaturatedLikelihood"
##  [5] "IndependenceLikelihood"  "calculatedHessian"
##  [7] "standardErrors"          "gradient"
##  [9] "hessian"                 "expectations"
## [11] "fit"                     "fitUnits"
## [13] "Minus2LogLikelihood"     "maxRelativeOrdinalError"
## [15] "minimum"                 "estimate"
## [17] "infoDefinite"            "conditionNumber"
## [19] "status"                  "iterations"
## [21] "evaluations"             "mxVersion"
## [23] "frontendTime"            "backendTime"
## [25] "independentTime"         "wallTime"
## [27] "timestamp"               "cpuTime"
```

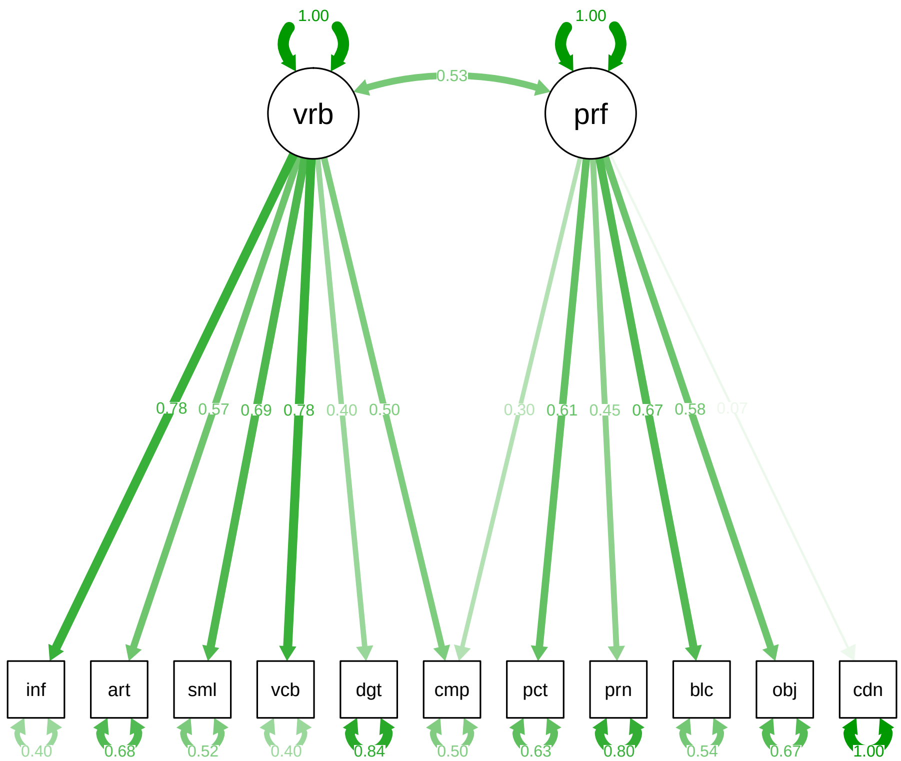
5.3.4 Draw the OpenMx model number 3

The **semPlot** package is very powerful and can recognize many 'lm' and SEM model objects. We can use the identical code that we used in chapter 2 for the **lavaan** model.

```
# Note that the base plot, including standardized path coefficients plots positive coefficients green
# and negative coefficients red.  Red-green colorblindness issues anyone?
# I redrew it here to choose a blue and red.  But all the coefficients in this example are
# positive,so they are shown with the skyblue.
# more challenging to use colors other than red and green.  not in this doc
semPaths(cfa2c, residuals=F,sizeMan=7,"std",
         posCol=c("skyblue4", "red"),
         #edge.color="skyblue4",
         edge.label.cex=1.2,layout="circle2")
```




```
# or we could draw the paths in such  a way to include the residuals:
#semPaths(fit1, sizeMan=7,"std",edge.color="skyblue4",edge.label.cex=1,layout="circle2")
# the base path diagram can be drawn much more simply:
#semPaths(fit1)
# or
semPaths(cfa2c,"std")
```



5.4 Compare the OpenMx models

In **OpenMx** a convenient function exists for model comparisons. This code compares the same two models that we initially compared in the **lavaan** approach that used the `anova` function.

```
kable(mxCompare(cfa2c,cfa2b), booktabs=TRUE, format="markdown")
```

base	comparison	ep	minus2LL	df	AIC	diffLL	diffdf	p
TwoFac, Comp Common	NA	24	5435.7	42	-23.390	NA	NA	NA
TwoFac, Comp Common	Two Factor Model	23	5445.7	43	-15.392	9.998	1	0.00157

Chapter 6

Summary and Reproducibility

We have finished a nice book……..

Here is some information for for reproducibility:

```r
sessionInfo()
```

```
## R version 3.5.2 (2018-12-20)
## Platform: x86_64-w64-mingw32/x64 (64-bit)
## Running under: Windows 7 x64 (build 7601) Service Pack 1
##
## Matrix products: default
##
## locale:
## [1] LC_COLLATE=English_United States.1252
## [2] LC_CTYPE=English_United States.1252
## [3] LC_MONETARY=English_United States.1252
## [4] LC_NUMERIC=C
## [5] LC_TIME=English_United States.1252
##
## attached base packages:
## [1] stats     graphics  grDevices utils     datasets  methods   base
##
## other attached packages:
##  [1] OpenMx_2.12.2    sem_3.1-9        lavaan_0.6-3     ggraph_1.0.2
##  [5] ggplot2_3.1.0    corrplot_0.84    tidyr_0.8.2      magrittr_1.5
##  [9] dplyr_0.8.0.1    MVN_5.6          kableExtra_1.0.1 knitr_1.21
## [13] psych_1.8.12     semPlot_1.1      car_3.0-2        carData_3.0-2
##
## loaded via a namespace (and not attached):
##   [1] readxl_1.3.0          backports_1.1.3       Hmisc_4.2-0
##   [4] BDgraph_2.55          plyr_1.8.4            igraph_1.2.4
##   [7] lazyeval_0.2.2        sp_1.3-1              splines_3.5.2
##  [10] digest_0.6.18         htmltools_0.3.6       viridis_0.5.1
##  [13] matrixcalc_1.0-3      checkmate_1.9.1       lisrelToR_0.1.4
##  [16] cluster_2.0.7-1       openxlsx_4.1.0        readr_1.3.1
##  [19] jpeg_0.1-8            colorspace_1.4-1      ggrepel_0.8.0
##  [22] rvest_0.3.2           rrcov_1.4-7           haven_2.1.0
##  [25] xfun_0.4              crayon_1.3.4          lme4_1.1-20
##  [28] zoo_1.8-4             survival_2.43-3       glue_1.3.1
```

```
##  [31] gtable_0.3.0          webshot_0.5.1         mi_1.0
##  [34] kernlab_0.9-27        prabclus_2.2-7        DEoptimR_1.0-8
##  [37] ggm_2.3               abind_1.4-5           VIM_4.8.0
##  [40] scales_1.0.0          sgeostat_1.0-27       mvtnorm_1.0-8
##  [43] GGally_1.4.0          sROC_0.1-2            Rcpp_1.0.1
##  [46] viridisLite_0.3.0     xtable_1.8-3          laeken_0.5.0
##  [49] htmlTable_1.13.1      units_0.6-2           foreign_0.8-71
##  [52] mclust_5.4.2          Formula_1.2-3         stats4_3.5.2
##  [55] truncnorm_1.0-8       vcd_1.4-4             htmlwidgets_1.3
##  [58] httr_1.4.0            fpc_2.1-11.1          RColorBrewer_1.1-2
##  [61] modeltools_0.2-22     acepack_1.4.1         farver_1.1.0
##  [64] NADA_1.6-1            flexmix_2.3-15        pkgconfig_2.0.2
##  [67] reshape_0.8.8         XML_3.98-1.17         nnet_7.3-12
##  [70] kutils_1.60           tidyselect_0.2.5      rlang_0.3.1
##  [73] reshape2_1.4.3        munsell_0.5.0         cellranger_1.1.0
##  [76] tools_3.5.2           moments_0.14          ranger_0.11.1
##  [79] pls_2.7-0             cvTools_0.3.2         fdrtool_1.2.15
##  [82] evaluate_0.13         stringr_1.4.0         arm_1.10-1
##  [85] yaml_2.2.0            zip_1.0.0             robustbase_0.93-3
##  [88] purrr_0.3.2           glasso_1.10           pbapply_1.4-0
##  [91] nlme_3.1-137          whisker_0.3-2         xml2_1.2.0
##  [94] compiler_3.5.2        rstudioapi_0.9.0      curl_3.3
##  [97] png_0.1-7             e1071_1.7-0.1         zCompositions_1.2.0
## [100] huge_1.2.7            tweenr_1.0.1          tibble_2.0.1
## [103] robCompositions_2.0.10 pbivnorm_0.6.0       pcaPP_1.9-73
## [106] stringi_1.3.1         highr_0.7             qgraph_1.6.1
## [109] rockchalk_1.8.140     forcats_0.4.0         trimcluster_0.1-2.1
## [112] lattice_0.20-38       Matrix_1.2-15         nloptr_1.2.1
## [115] pillar_1.3.1          lmtest_0.9-36         data.table_1.12.0
## [118] corpcor_1.6.9         R6_2.4.0              latticeExtra_0.6-28
## [121] bookdown_0.9          gridExtra_2.3         rio_0.5.16
## [124] boot_1.3-20           energy_1.7-5          MASS_7.3-51.1
## [127] gtools_3.8.1          assertthat_0.2.1      rjson_0.2.20
## [130] withr_2.1.2           nortest_1.0-4         mnormt_1.5-5
## [133] diptest_0.75-7        parallel_3.5.2        hms_0.4.2
## [136] grid_3.5.2            rpart_4.1-13          coda_0.19-2
## [139] class_7.3-15          minqa_1.2.4           rmarkdown_1.11
## [142] mvoutlier_2.0.9       d3Network_0.5.2.1     ggforce_0.1.3
## [145] numDeriv_2016.8-1     semTools_0.5-1        base64enc_0.1-3
```

Bibliography

Allaire, J., Xie, Y., McPherson, J., Luraschi, J., Ushey, K., Atkins, A., Wickham, H., Cheng, J., Chang, W., and Iannone, R. (2018). *rmarkdown: Dynamic Documents for R*. R package version 1.11.

Bache, S. M. and Wickham, H. (2014). *magrittr: A Forward-Pipe Operator for R*. R package version 1.5.

Boker, S. M., Neale, M. C., Maes, H. H., Spiegel, M., Brick, T. R., Estabrook, R., Bates, T. C., Gore, R. J., Hunter, M. D., Pritikin, J. N., Zahery, M., and Kirkpatrick, R. M. (2019). *OpenMx: Extended Structural Equation Modelling*. R package version 2.12.1.

Brown, T. A. (2015). *Confirmatory factor analysis for applied research*. Methodology in the social sciences. The Guilford Press, New York ; London, second edition. edition.

El-Sheikh, A. A., Abonazel, M. R., and Gamil, N. (2017). A review of software packages for structural equation modeling: A comparative study. *Applied Mathematics and Physics*, 5(3):85–94.

Epskamp, S. and with contributions from Simon Stuber (2017). *semPlot: Path Diagrams and Visual Analysis of Various SEM Packages' Output*. R package version 1.1.

Fox, J., Nie, Z., and Byrnes, J. (2017). *sem: Structural Equation Models*. R package version 3.1-9.

Fox, J., Weisberg, S., and Price, B. (2018). *car: Companion to Applied Regression*. R package version 3.0-2.

Korkmaz, S., Goksuluk, D., and Zararsiz, G. (2018). *MVN: Multivariate Normality Tests*. R package version 5.5.

Narayanan, A. (2012). A review of eight software packages for structural equation modeling. *The American Statistician*, 66(2):129–138.

R Core Team (2018). *R: A Language and Environment for Statistical Computing*. R Foundation for Statistical Computing, Vienna, Austria.

Revelle, W. (2019). *psych: Procedures for Psychological, Psychometric, and Personality Research*. R package version 1.8.12.

Rosseel, Y. (2012). Lavaan: An r package for structural equation modeling and more. version 0.5–12 (beta). *Journal of statistical software*, 48(2):1–36.

Rosseel, Y. (2018). *lavaan: Latent Variable Analysis*. R package version 0.6-3.

RStudio Team (2015). *RStudio: Integrated Development Environment for R*. RStudio, Inc., Boston, MA.

Tabachnick, B. G., Fidell, L. S., and Ullman, J. B. (2019). *Using multivariate statistics*. Pearson, Boston, seventh edition. edition.

Wei, T. and Simko, V. (2017). *corrplot: Visualization of a Correlation Matrix*. R package version 0.84.

Wickham, H., François, R., Henry, L., and Müller, K. (2018). *dplyr: A Grammar of Data Manipulation*. R package version 0.7.8.

Wickham, H. and Henry, L. (2018). *tidyr: Easily Tidy Data with 'spread()' and 'gather()' Functions.* R package version 0.8.2.

Xie, Y. (2015). *Dynamic Documents with R and knitr.* Chapman and Hall/CRC, Boca Raton, Florida, 2nd edition. ISBN 978-1498716963.

Xie, Y. (2018a). *bookdown: Authoring Books and Technical Documents with R Markdown.* R package version 0.9.

Xie, Y. (2018b). *knitr: A General-Purpose Package for Dynamic Report Generation in R.* R package version 1.21.

Zhu, H. (2019). *kableExtra: Construct Complex Table with 'kable' and Pipe Syntax.* R package version 1.0.1.