Aim

The purpose is to get practice doing a set of common data analyses in I/O psychology using R.

Specifically, this means:

- Checking the reliability and factor structure of multi-item scales (e.g., personality tests, performance ratings, work perception scales, etc.) using techniques like exploratory factor analysis, confirmatory factor analysis, and reliability analysis.
- Scoring tests
- Doing preliminary data analysis to prepare and check the raw data
- Reporting univariate statistics
- Reporting correlations between measures
- Running regression (or other) models predicting relevant outcomes for theoretically meaningful sets of predictors

This is then combined with the task of presenting such results in an attractive format that is consistent with standards of reporting (e.g., APA style, etc).

Overview of the dataset

The dataset comes from a published study by Vanenberghe, Strodeur, and D'hoore in European Journal of Personality. Leadership performance was rated by nurses on a multiple item scale looking at transformational and transactional leadership. Further information about the data file is provided separately.

Research Questions

We want to assess what if any aspects of rated leadership behaviour predicts perceived unit performance

The predictors are divided into
- **Transformational**: attributed charisma, intellectual stimulation, individualised consideration
- **Transactional**: passive management by exception, active management by exception, contingent reward

Outcomes:
There are a range of outcome measures. We'll focus on perceived goal attainment of the unit. But we could also look at some of the others such as job satisfaction, commitment or perceptions that the unit retains staff.

Preliminary checks:
- Is the measure of leadership reasonable?

Major research questions:

- How do predictors intercorrelate and how do predictors correlate with performance?
- How does the prediction of transactional compare to transformational leadership?

Preliminary Analyses

- **EFA:** Report the results of an exploratory factor analysis of the leadership measure (i.e., 6 scales, 3 items per scale: MBEA1-3 to IC1-3). Check whether six factors is consistent with data. Present factor loadings in a table. Discuss pattern of loadings
- **CFA:** Provide quick summary of the results of a six factor confirmatory factor analysis with correlated factors. Optionally you could look at some other structures (e.g., one factor model; two factor model dividing items between transformational and transactional leadership).
- **Reliability:** Report Cronbach's alpha reliability for the six scales.

Descriptive Statistics and Correlations

- Present and discuss means, standard deviations and intercorrelations of both predictor and included outcome variables . You will want to comment on (a) the degree to which the predictors are intercorrelated, and (b) the degree to which each predictor correlates with the performance outcome.

Regression Models

You should present a set of regression models which evaluates the relative importance of transactional and transformational leadership in predicting your outcome variables.

- Optionally, you could consider interactions between predictors, the role of control variables, non-linear effects, alternative modelling approaches, and so on.

Answers

Get project set up

1. Download project template from
https://github.com/jeromyanglim/AnglimModifiedProjectTemplate/

or directly from
https://github.com/jeromyanglim/AnglimModifiedProjectTemplate/archive/master.zip

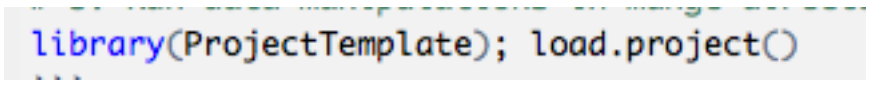
2. Unzip and move directory to an appropriate location on your computer. i.e., not in "downloads, or some arbitrary temporary directory.

3. Rename the directory and rstudio project folder to something meaningful like "vandenberg-analysis" and "vandenberg.rproj"

4. Move the data file and the meta-data files into the "data" folder. If you're interested in the longer process, see the completed version of the project for further details about how the original data files were prepared.

5. Open the project in Rstudio by double clicking on the rstudio project file. This will ensure that you are in the right working directory.

6. Check that the data imports correctly. Open "explore.rmd" and run the following command:



```
library(ProjectTemplate); load.project()
```

The data file is called "mlq". Run some basic commands on it, like:

- head(mlq): See whether all the data looks right
- dim(mlq): to see how many rows are in the data file and whether it looks right
- str(mlq): to check that all the data is numeric
- Hmisc::describe(mlq): to get some basic descriptive statistics on the data.
- or psych::describe(mlq)

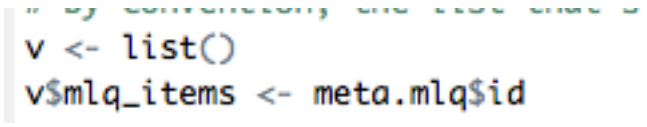
Exploratory factor analysis

There are many resources on exploratory factor analysis.

- http://www.statmethods.net/advstats/factor.html
- http://personality-project.org/r/psych/HowTo/factor.pdf

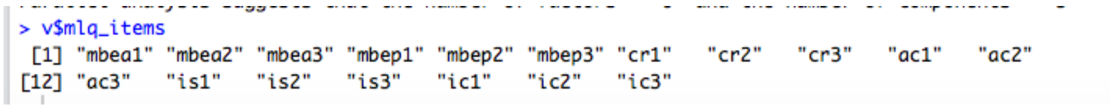
Given that these analyses involve repeated reference to the MLQ items, it is smart to store these variable names in a vector. I use the convention of storing all such variable sets in a list called"v" (v for variable).

This code is stored in the munge directory so that it automatically loaded when you run load.project() and thus always available when running substantive analyses.



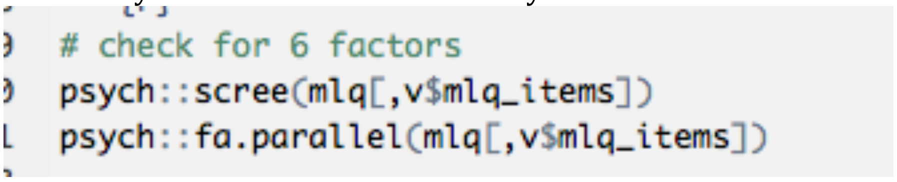
```
v <- list()
v$mlq_items <- meta.mlq$id
```

- v <- list(): creates an empty list
- The next item takes the variable names stored in the meta data and assigns to an element in the list v called mql_items
- The list look like this:



```
> v$mlq_items
 [1] "mbea1" "mbea2" "mbea3" "mbep1" "mbep2" "mbep3" "cr1"   "cr2"   "cr3"   "ac1"   "ac2"
[12] "ac3"   "is1"   "is2"   "is3"   "ic1"   "ic2"   "ic3"
```

How many factors does the factor analysis recommend?

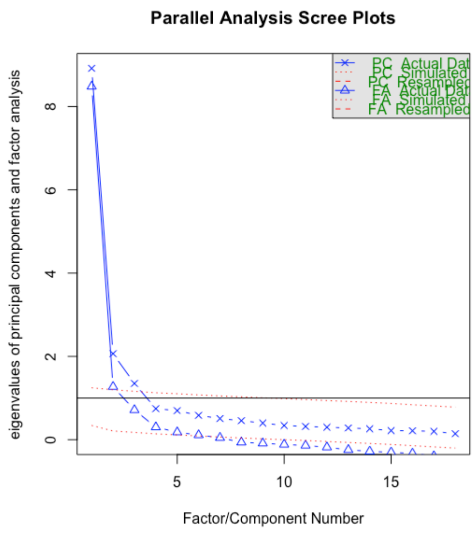


```
# check for 6 factors
psych::scree(mlq[,v$mlq_items])
psych::fa.parallel(mlq[,v$mlq_items])
```

- Note how we are using this list of variable names to extract a subset of variables from the data frame mlq

- scree: Scree plot shows the eigenvalues for unrotated components/factors.
- fa.parallel: This function compares the number of components and factors to random data:

```
> psych::fa.parallel(mlq[,v$mlq_items])
Parallel analysis suggests that the number of factors =  6  and the number of components =  3
```



- The blue points reflect the unrotated eigenvalues for factor analysis and principal components analysis on the data. The red dotted line reflect the eigenvalues of random data. The idea is that obtained data should have more components than random data.
- Interestingly, based on factor analysis, the parallel test suggests six factors, but based on PCA it suggests three.

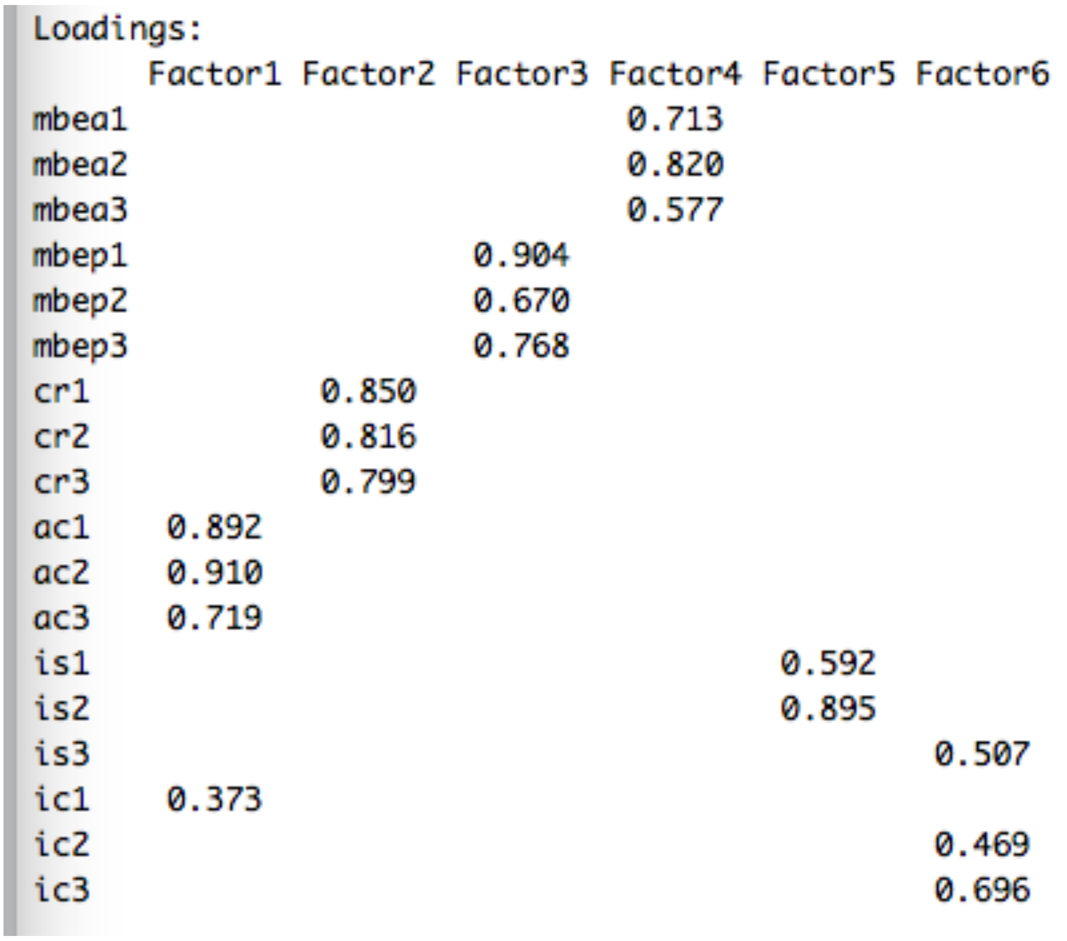
- The main point is that the first factor is huge. Thus, the main factor driving ratings of a supervisors is some global evaluative factor. The second and third factors may be of some relevance, and then the scree appears to begin. Thus, in general, it looks typical of most 360 measures where most ratings are driven by a single evaluative factors, and the tests have many more subscales than are really needed. That said, they have some small incremental value.

We can check out the factor loadings:

```
# examine 6 factor EFA
fac1 <- factanal(mlq[,v$mlq_items], 6, rotation = "promax")
print(fac1, cutoff = .30)

fl <- round(unclass(fac1$loadings), 2)
fl

write.csv(fl, "output/efa-factor-loadings.csv")
```

- factanal: function for performing exploratory factor analysis takes a data.frame, the number of factors to extract (i.e., 6), and optionally a rotation (promax will give a correlated rotation)
- To show just loadings above a certain cutoff, use the print method with a cutoff like .30 on the returned object

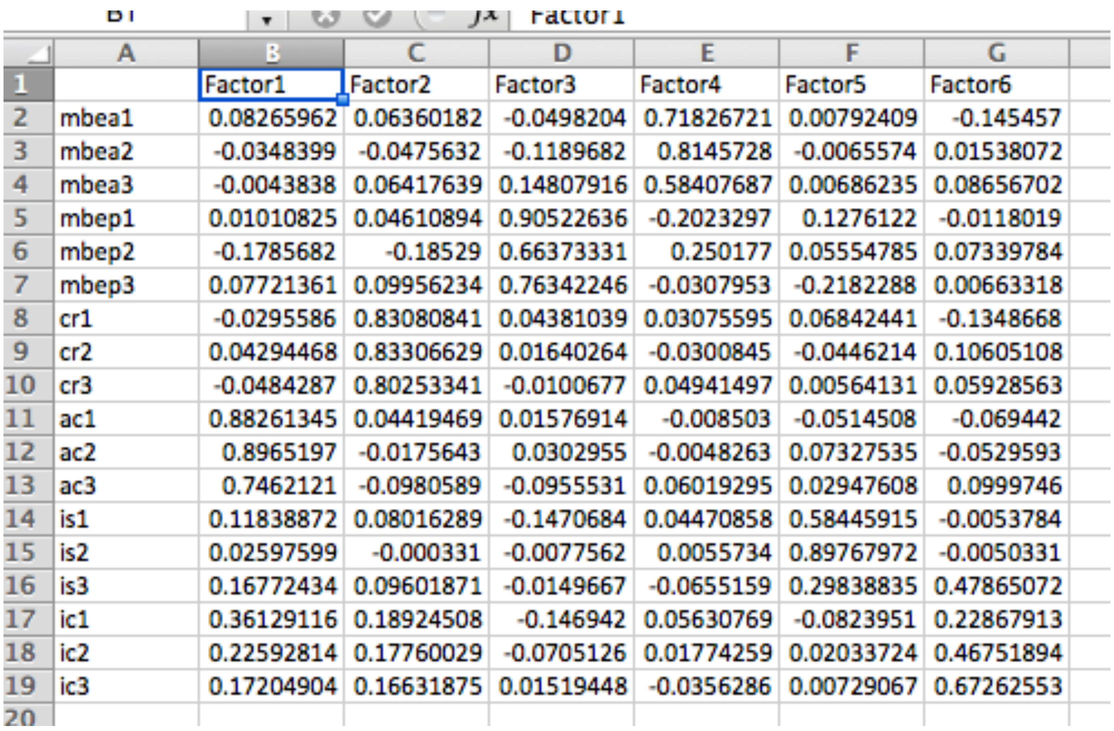
- If you want to save the factor loadings, the subsequent code will work



```
Loadings:
      Factor1 Factor2 Factor3 Factor4 Factor5 Factor6
mbea1                          0.713
mbea2                          0.820
mbea3                          0.577
mbep1                  0.904
mbep2                  0.670
mbep3                  0.768
cr1            0.850
cr2            0.816
cr3            0.799
ac1    0.892
ac2    0.910
ac3    0.719
is1                                    0.592
is2                                    0.895
is3                                            0.507
ic1    0.373
ic2                                            0.469
ic3                                            0.696
```

- The factor loadings are not too bad. IS3 and IC1 are not loading on their theorised factors. But in general, everything else looks okay.

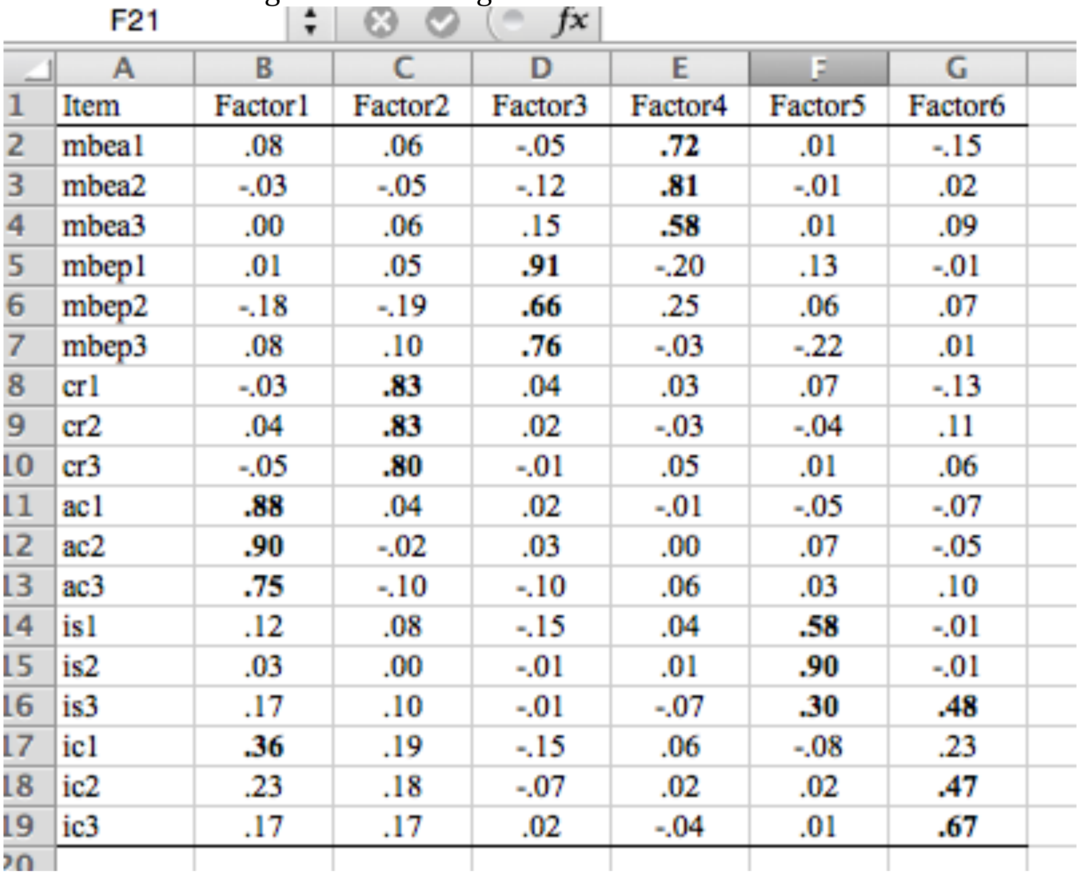
If you wanted to present an attractive loading matrix for a thesis or report. You could open the saved csv file in excel and apply some formatting. I generally copy the text into "output/output-processing.xlsx".



	A	B	C	D	E	F	G
1		Factor1	Factor2	Factor3	Factor4	Factor5	Factor6
2	mbea1	0.08265962	0.06360182	-0.0498204	0.71826721	0.00792409	-0.145457
3	mbea2	-0.0348399	-0.0475632	-0.1189682	0.8145728	-0.0065574	0.01538072
4	mbea3	-0.0043838	0.06417639	0.14807916	0.58407687	0.00686235	0.08656702
5	mbep1	0.01010825	0.04610894	0.90522636	-0.2023297	0.1276122	-0.0118019
6	mbep2	-0.1785682	-0.18529	0.66373331	0.250177	0.05554785	0.07339784
7	mbep3	0.07721361	0.09956234	0.76342246	-0.0307953	-0.2182288	0.00663318
8	cr1	-0.0295586	0.83080841	0.04381039	0.03075595	0.06842441	-0.1348668
9	cr2	0.04294468	0.83306629	0.01640264	-0.0300845	-0.0446214	0.10605108
10	cr3	-0.0484287	0.80253341	-0.0100677	0.04941497	0.00564131	0.05928563
11	ac1	0.88261345	0.04419469	0.01576914	-0.008503	-0.0514508	-0.069442
12	ac2	0.8965197	-0.0175643	0.0302955	-0.0048263	0.07327535	-0.0529593
13	ac3	0.7462121	-0.0980589	-0.0955531	0.06019295	0.02947608	0.0999746
14	is1	0.11838872	0.08016289	-0.1470684	0.04470858	0.58445915	-0.0053784
15	is2	0.02597599	-0.000331	-0.0077562	0.0055734	0.89767972	-0.0050331
16	is3	0.16772434	0.09601871	-0.0149667	-0.0655159	0.29838835	0.47865072
17	ic1	0.36129116	0.18924508	-0.146942	0.05630769	-0.0823951	0.22867913
18	ic2	0.22592814	0.17760029	-0.0705126	0.01774259	0.02033724	0.46751894
19	ic3	0.17204904	0.16631875	0.01519448	-0.0356286	0.00729067	0.67262553
20							

Tips for making it pretty:

- Highlight numbers - cell formats - custom ".00"; this will show just two decimal places without leading zeros.
- Go through and bold any large numbers (e.g., abs(loading) > .30)
- Add table lines for header and the bottom of the table
- Tweak centering and formatting

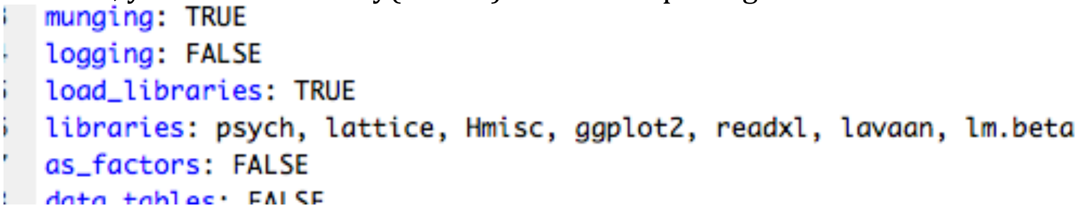


	A	B	C	D	E	F	G
1	Item	Factor1	Factor2	Factor3	Factor4	Factor5	Factor6
2	mbea1	.08	.06	-.05	**.72**	.01	-.15
3	mbea2	-.03	-.05	-.12	**.81**	-.01	.02
4	mbea3	.00	.06	.15	**.58**	.01	.09
5	mbep1	.01	.05	**.91**	-.20	.13	-.01
6	mbep2	-.18	-.19	**.66**	.25	.06	.07
7	mbep3	.08	.10	**.76**	-.03	-.22	.01
8	cr1	-.03	**.83**	.04	.03	.07	-.13
9	cr2	.04	**.83**	.02	-.03	-.04	.11
10	cr3	-.05	**.80**	-.01	.05	.01	.06
11	ac1	**.88**	.04	.02	-.01	-.05	-.07
12	ac2	**.90**	-.02	.03	.00	.07	-.05
13	ac3	**.75**	-.10	-.10	.06	.03	.10
14	is1	.12	.08	-.15	.04	**.58**	-.01
15	is2	.03	.00	-.01	.01	**.90**	-.01
16	is3	.17	.10	-.01	-.07	**.30**	**.48**
17	ic1	**.36**	.19	-.15	.06	-.08	.23
18	ic2	.23	.18	-.07	.02	.02	**.47**
19	ic3	.17	.17	.02	-.04	.01	**.67**
20							

- Paste into word and add any table notes and so on.

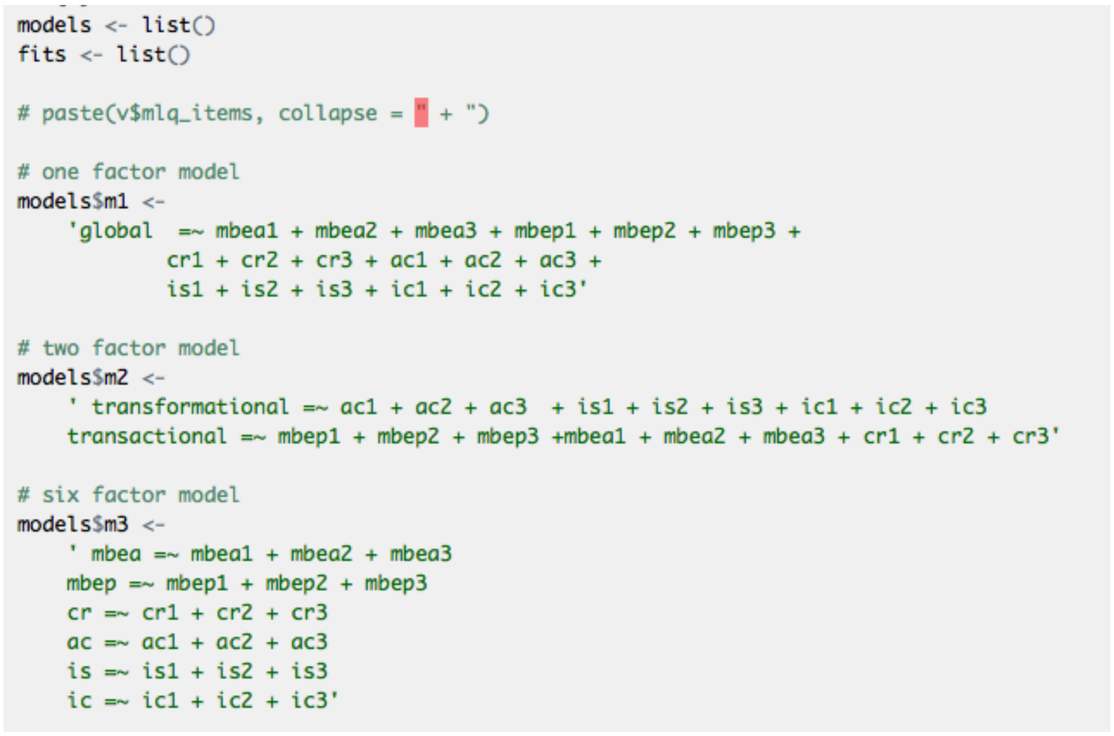
Confirmatory factor analysis
We will use the lavaan package to perform confirmatory factor analysis.
For tutorials, see:
http://lavaan.ugent.be/tutorial/index.html

- Add "lavaan" to the list of libraries in "config/global.dcf"; this ensures that the package gets loaded next time you run load.project. If you're doing it real time, you can run library(lavaan) to load the package.



```
munging: TRUE
logging: FALSE
load_libraries: TRUE
libraries: psych, lattice, Hmisc, ggplot2, readxl, lavaan, lm.beta
as_factors: FALSE
data_tables: FALSE
```

I thought I show three models, also to highlight model comparison: a global factor model, a two factor model (transactional versus transformative) and the six factor model.

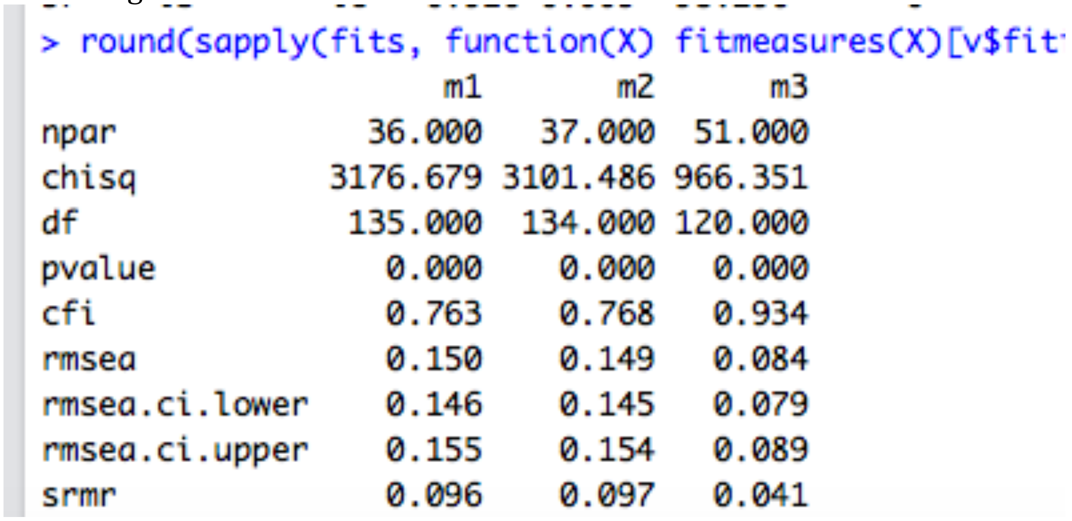


```
models <- list()
fits <- list()

# paste(v$mlq_items, collapse = " + ")

# one factor model
models$m1 <-
    'global   =~ mbea1 + mbea2 + mbea3 + mbep1 + mbep2 + mbep3 +
            cr1 + cr2 + cr3 + ac1 + ac2 + ac3 +
            is1 + is2 + is3 + ic1 + ic2 + ic3'

# two factor model
models$m2 <-
    ' transformational =~ ac1 + ac2 + ac3  + is1 + is2 + is3 + ic1 + ic2 + ic3
    transactional =~ mbep1 + mbep2 + mbep3 +mbea1 + mbea2 + mbea3 + cr1 + cr2 + cr3'

# six factor model
models$m3 <-
    ' mbea =~ mbea1 + mbea2 + mbea3
    mbep =~ mbep1 + mbep2 + mbep3
    cr =~ cr1 + cr2 + cr3
    ac =~ ac1 + ac2 + ac3
    is =~ is1 + is2 + is3
    ic =~ ic1 + ic2 + ic3'
```

- Two lists are declared at first. This can be useful as its easier to iterate certain functions over models or fits when objects are stored in lists.
- The first model declares a latent variable "global" and says that all items load on it. The "loads on" operator is "=~". All other items are separated by the plus symbol
- The other models declare several latent variable models. In particular, the main goal is to estimate fit statistics for the six factor model proposed by theory.

```
fits$m1 <- lavaan::cfa(models$m1, data = mlq)
fits$m2 <- lavaan::cfa(models$m2, data = mlq)
fits$m3 <- lavaan::cfa(models$m3, data = mlq)


summary(fits$m3, fit.measures = TRUE)
standardizedSolution(fits$m3)

v$fitindicies <- c("npar",  "chisq", "df", "pvalue", "cfi", "rmsea",
                   "rmsea.ci.lower", "rmsea.ci.upper", "srmr")

round(sapply(fits, function(X) fitmeasures(X)[v$fitindicies]), 3)
```

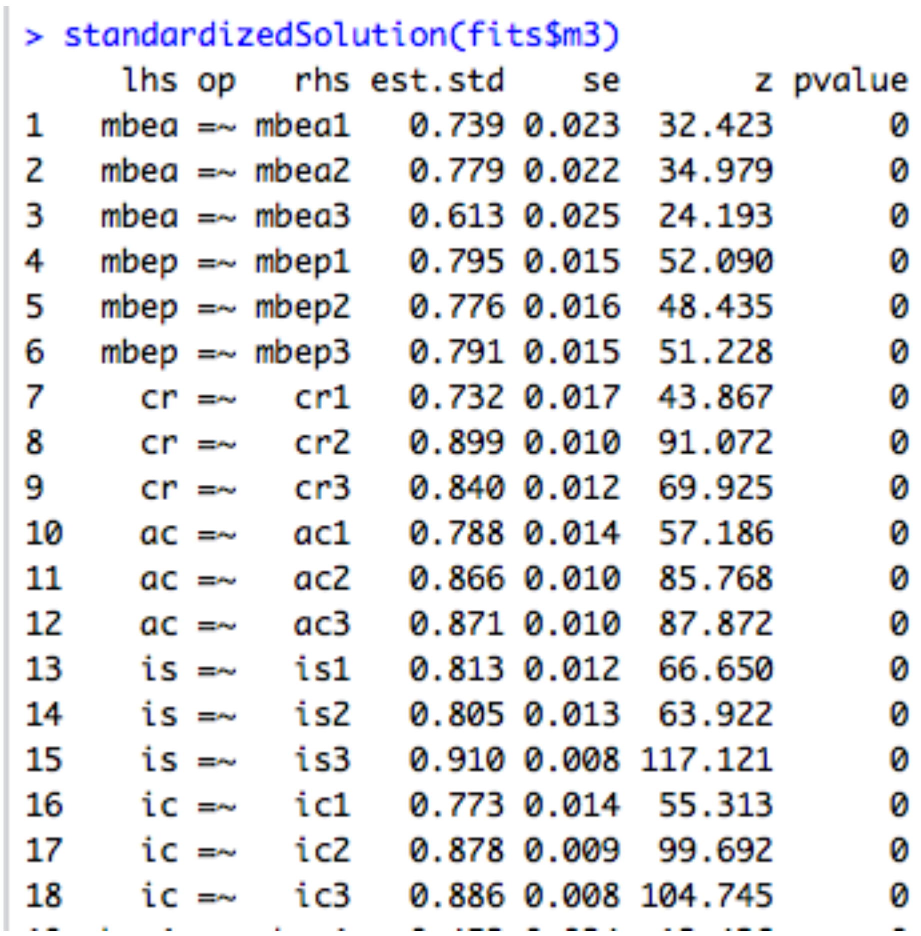
- cfa: This function fits the models to the data using a correlated factor model.
- summary: When passed a fit object, this returns fit information (e.g., coefficients, overall fit measures and so on).
- standardizedSolution: this returns standardised coefficients which is often useful in psychology
- The final line extracts a subset of the fit measures from each of the fits and arranges them in a table



```
> round(sapply(fits, function(X) fitmeasures(X)[v$fit
                      m1       m2      m3
npar              36.000   37.000  51.000
chisq           3176.679 3101.486 966.351
df               135.000  134.000 120.000
pvalue             0.000    0.000   0.000
cfi                0.763    0.768   0.934
rmsea              0.150    0.149   0.084
rmsea.ci.lower     0.146    0.145   0.079
rmsea.ci.upper     0.155    0.154   0.089
srmr               0.096    0.097   0.041
```

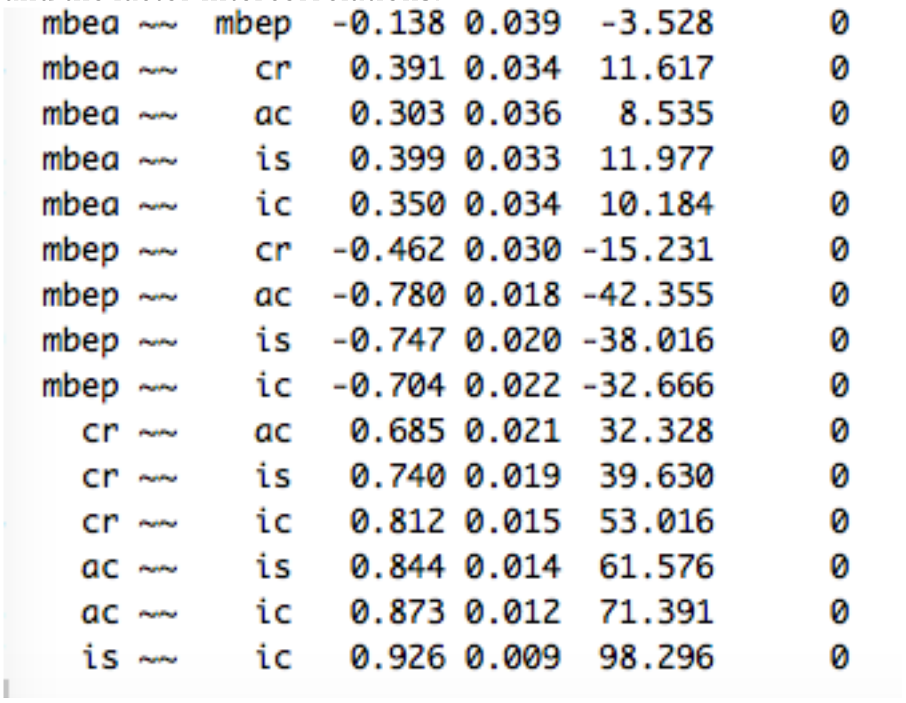
- It shows clearly that the six factor model provides a much better fit.
- Chi-square is reduced by much more than 2 times the change in degrees of freedom
- CFI is above .90 and getting close to the often cited .95 target
- RMSEA is not great at .084 but it is at least respectable

Standarized solution output of most relevance includes the loadings:



```
> standardizedSolution(fits$m3)
     lhs op   rhs est.std    se       z pvalue
1   mbea =~ mbea1   0.739 0.023  32.423      0
2   mbea =~ mbea2   0.779 0.022  34.979      0
3   mbea =~ mbea3   0.613 0.025  24.193      0
4   mbep =~ mbep1   0.795 0.015  52.090      0
5   mbep =~ mbep2   0.776 0.016  48.435      0
6   mbep =~ mbep3   0.791 0.015  51.228      0
7     cr =~   cr1   0.732 0.017  43.867      0
8     cr =~   cr2   0.899 0.010  91.072      0
9     cr =~   cr3   0.840 0.012  69.925      0
10    ac =~   ac1   0.788 0.014  57.186      0
11    ac =~   ac2   0.866 0.010  85.768      0
12    ac =~   ac3   0.871 0.010  87.872      0
13    is =~   is1   0.813 0.012  66.650      0
14    is =~   is2   0.805 0.013  63.922      0
15    is =~   is3   0.910 0.008 117.121      0
16    ic =~   ic1   0.773 0.014  55.313      0
17    ic =~   ic2   0.878 0.009  99.692      0
18    ic =~   ic3   0.886 0.008 104.745      0
```

and the factor intercorrelations:



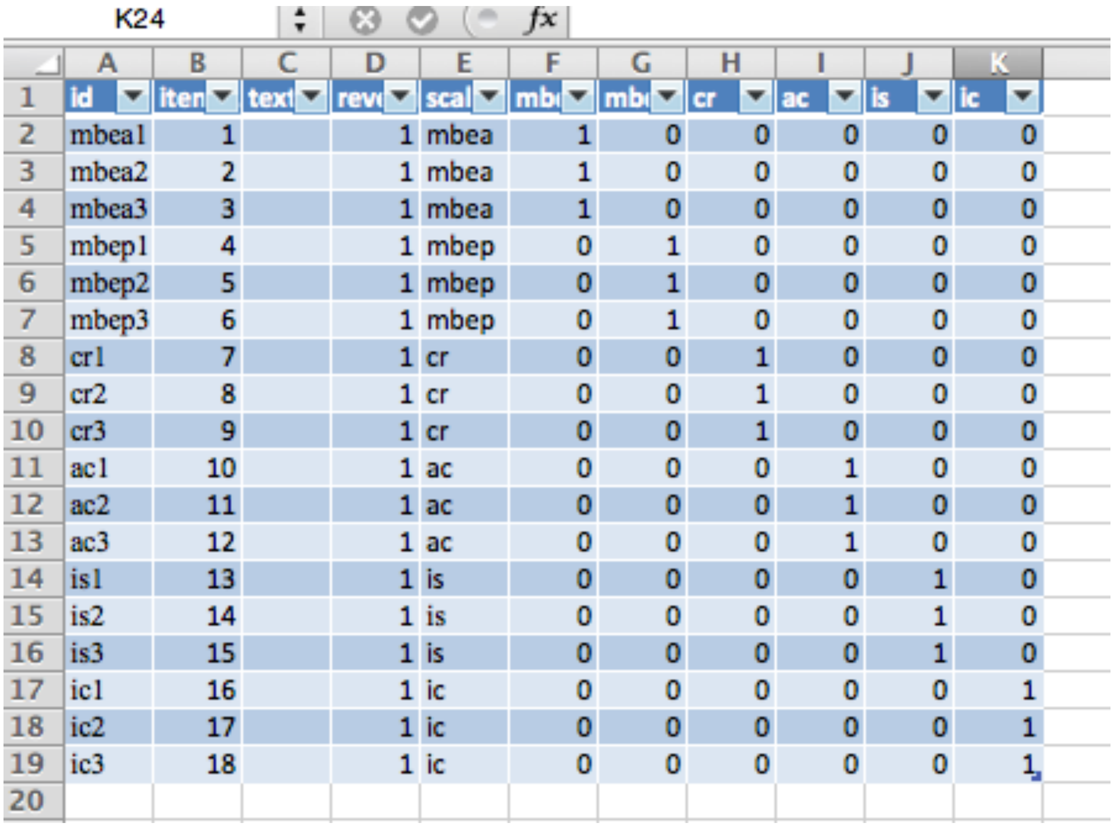
```
  mbea ~~  mbep  -0.138 0.039  -3.528      0
  mbea ~~    cr   0.391 0.034  11.617      0
  mbea ~~    ac   0.303 0.036   8.535      0
  mbea ~~    is   0.399 0.033  11.977      0
  mbea ~~    ic   0.350 0.034  10.184      0
  mbep ~~    cr  -0.462 0.030 -15.231      0
  mbep ~~    ac  -0.780 0.018 -42.355      0
  mbep ~~    is  -0.747 0.020 -38.016      0
  mbep ~~    ic  -0.704 0.022 -32.666      0
    cr ~~    ac   0.685 0.021  32.328      0
    cr ~~    is   0.740 0.019  39.630      0
    cr ~~    ic   0.812 0.015  53.016      0
    ac ~~    is   0.844 0.014  61.576      0
    ac ~~    ic   0.873 0.012  71.391      0
    is ~~    ic   0.926 0.009  98.296      0
```

- Loadings are uniformly high (i.e., above .60 and generally above .7 or .8). This is a good sign for the model.

- Correlations: Some of them are particularly high such those between AC, IS, and IC. MBEP also correlates quite highly with these three (albeit negatively). CR and MBEA seem to be a bit different.

Overall, these analyses support the six factor mode. But more broadly from the perspective of the EFA, it seems like there is one main evaluative factor, and then a few smaller meaningful components. It may be that six factors is required to get all the nuances of the ratings, but that with further exploration, perhaps three might do a reasonable job, and one factor will actually get you a long way.

Reliability Analysis
A good way to run reliability analysis in R is to use the psych function scoreItems. The main challenge is that it requires a scoring key. This is where it comes in handy having a meta data file with one row per item for the various scales.
Here is the meta data that I created:

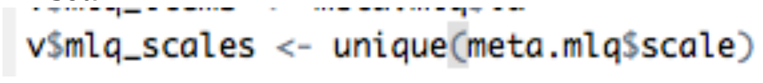


K24

	A	B	C	D	E	F	G	H	I	J	K
1	id	iten	text	reve	scal	mb	mb	cr	ac	is	ic
2	mbea1	1		1	mbea	1	0	0	0	0	0
3	mbea2	2		1	mbea	1	0	0	0	0	0
4	mbea3	3		1	mbea	1	0	0	0	0	0
5	mbep1	4		1	mbep	0	1	0	0	0	0
6	mbep2	5		1	mbep	0	1	0	0	0	0
7	mbep3	6		1	mbep	0	1	0	0	0	0
8	cr1	7		1	cr	0	0	1	0	0	0
9	cr2	8		1	cr	0	0	1	0	0	0
10	cr3	9		1	cr	0	0	1	0	0	0
11	ac1	10		1	ac	0	0	0	1	0	0
12	ac2	11		1	ac	0	0	0	1	0	0
13	ac3	12		1	ac	0	0	0	1	0	0
14	is1	13		1	is	0	0	0	0	1	0
15	is2	14		1	is	0	0	0	0	1	0
16	is3	15		1	is	0	0	0	0	1	0
17	ic1	16		1	ic	0	0	0	0	0	1
18	ic2	17		1	ic	0	0	0	0	0	1
19	ic3	18		1	ic	0	0	0	0	0	1
20											

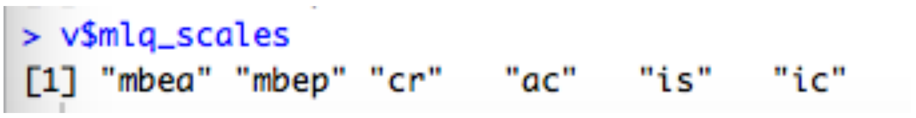
- id: This corresponds to the variable name for the item in R
- six scale columns: These correspond to the six scales where values are 1 (positively worded item),-1 (reversed item), 0 (not included in scale).

As these six scales will be used regularly, I've added them as another variable set:



```
v$mlq_scales <- unique(meta.mlq$scale)
```

I.e., it looks like this:

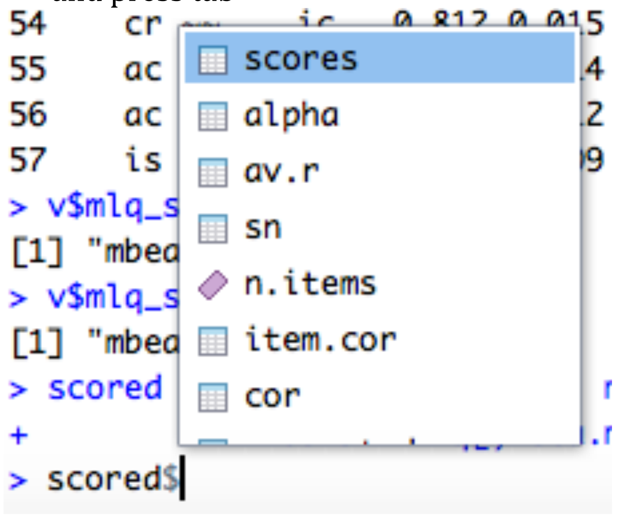


```
> v$mlq_scales
[1] "mbea" "mbep" "cr"   "ac"   "is"   "ic"
```

Thus, to calculate reliability:

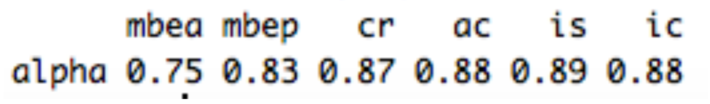
```
scored <- scoreItems(keys =  meta.mlq[,v$mlq_scales],
           items = mlq[,meta.mlq$id])
round(scored$alpha, 2)
```

- keys: The meta data is used as the key. Just the six scale columns are extracted
- items: meta.mlq$id corresponds to the variable names in the key. Its important to do this to ensure that you extract the participant data where the variables are in the same order as the key.
- This object is then assigned to an object that I have called "scored"
- If you didn't know what objects were inside scored, you could type "scored$" and press tab



```
54     cr                 ic   0.812 0.015
55     ac                               4
56     ac                               2
57     is                               9
> v$mlq_s
[1] "mbea
> v$mlq_s
[1] "mbea
> scored
+
> scored$
```

- alpha stores Cronbach's alphas
- Rounding to two decimal places makes it easier to read.



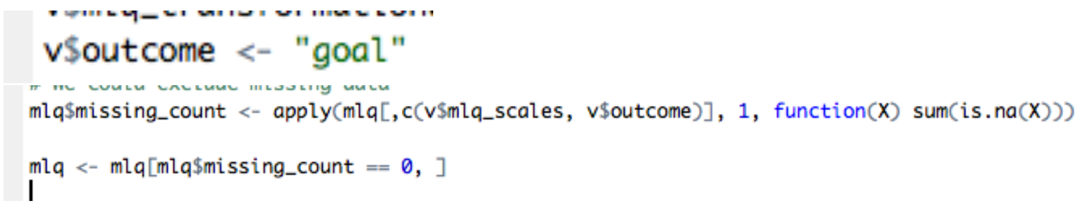
```
      mbea mbep   cr   ac   is   ic
alpha 0.75 0.83 0.87 0.88 0.89 0.88
```

- In general, the alphas are pretty good: i.e., most above .80; all above .70.

Descriptive Statistics and correlations

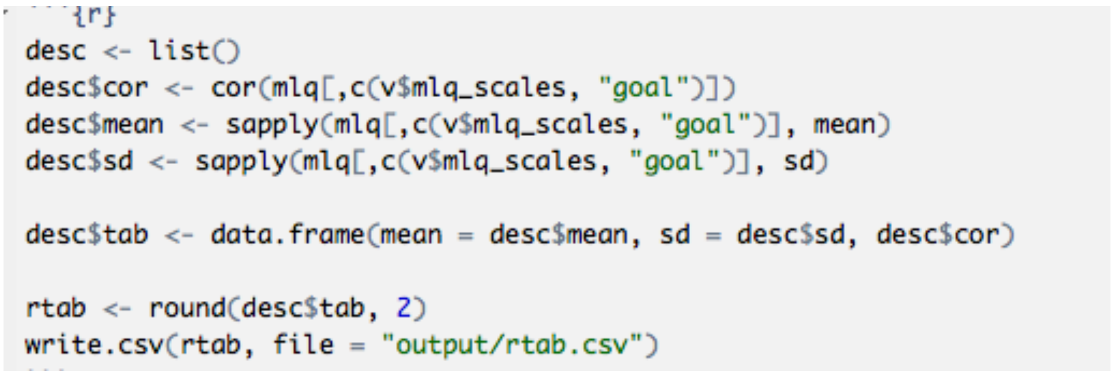
There are various standard approaches to get descriptive statistics and correlations.

- mean, sd, cor: are the standard functions
- It as at this point in the analysis, that I realised that there was some missing data on the outcome variable "goal" (i.e., perception of the unit achieving its goal). The above functions will throw an error, and it's often simpler just to have a consistent sample.



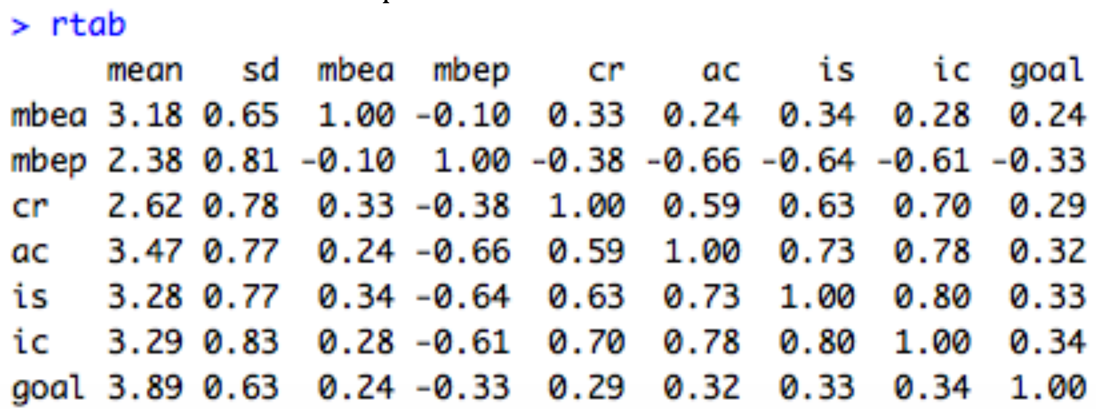
```
[illegible]
v$outcome <- "goal"
# we could exclude missing data
mlq$missing_count <- apply(mlq[,c(v$mlq_scales, v$outcome)], 1, function(X) sum(is.na(X)))

mlq <- mlq[mlq$missing_count == 0, ]
```

- This code removed cases that had any missing data on the variables of interest. More could be said about this, but this is a simple approach



~~~
```{r}
desc <- list()
desc$cor <- cor(mlq[,c(v$mlq_scales, "goal")])
desc$mean <- sapply(mlq[,c(v$mlq_scales, "goal")], mean)
desc$sd <- sapply(mlq[,c(v$mlq_scales, "goal")], sd)

desc$tab <- data.frame(mean = desc$mean, sd = desc$sd, desc$cor)

rtab <- round(desc$tab, 2)
write.csv(rtab, file = "output/rtab.csv")
~~~

- In order to create a combined table with means, sd, and intercorrelations, elements of this were added to a list called "desc"
- cor: This function generates a correlation matrix between variables
- sapply(..., mean): calculates the mean for each variable included
- This was repeated for the standard deviation
- These elements were then combined into a data frame and the values were rounded to two decimal places:

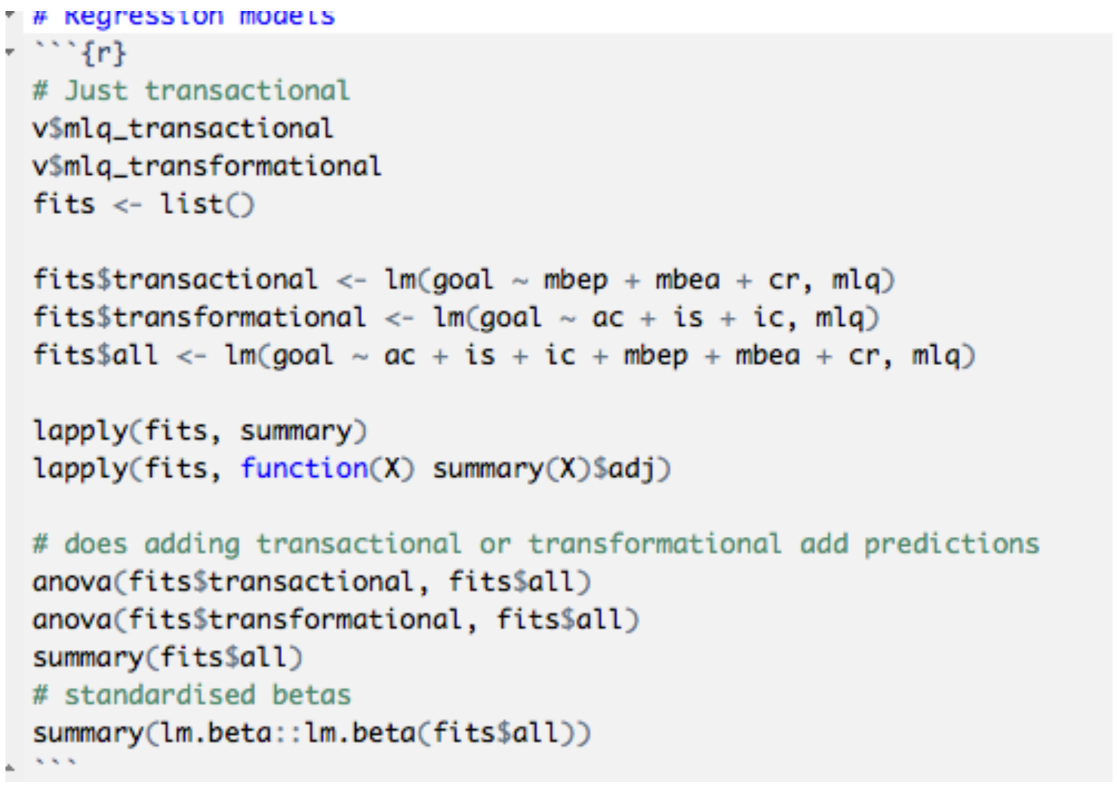


```
> rtab
     mean   sd  mbea  mbep    cr    ac    is    ic  goal
mbea 3.18 0.65  1.00 -0.10  0.33  0.24  0.34  0.28  0.24
mbep 2.38 0.81 -0.10  1.00 -0.38 -0.66 -0.64 -0.61 -0.33
cr   2.62 0.78  0.33 -0.38  1.00  0.59  0.63  0.70  0.29
ac   3.47 0.77  0.24 -0.66  0.59  1.00  0.73  0.78  0.32
is   3.28 0.77  0.34 -0.64  0.63  0.73  1.00  0.80  0.33
ic   3.29 0.83  0.28 -0.61  0.70  0.78  0.80  1.00  0.34
goal 3.89 0.63  0.24 -0.33  0.29  0.32  0.33  0.34  1.00
```

- It highlights the large correlations between several of the leadership scales as discussed earlier with the CFA.
- In general, there are fairly consistent correlations between leadership and perceptions that the unit is achieving it's goals with correlations in the .2 to .35 range. Note that passive management by exception (MBEP) appears to be negatively related to unit performance. This scale is a transactional leadership scale included in the MLQ more as a comparison for transformational leadership.

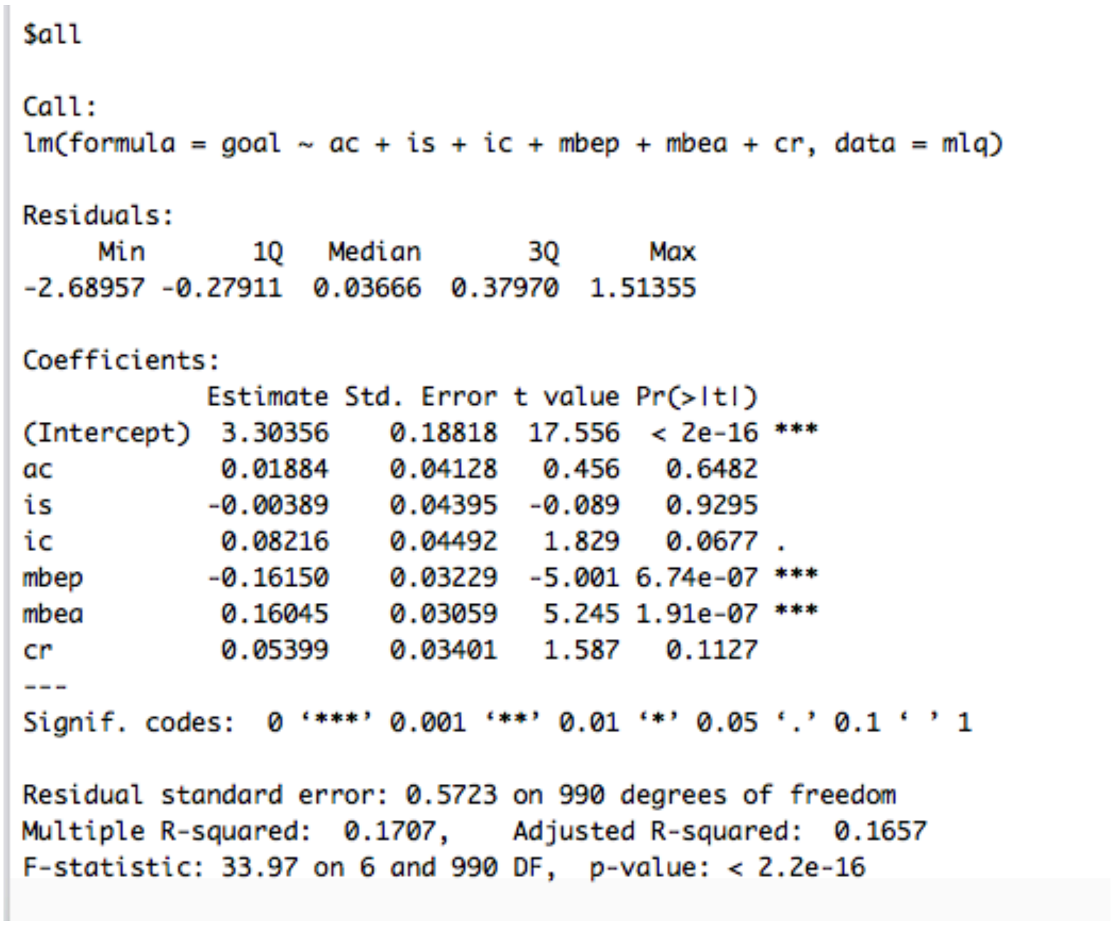
By exporting this to csv, we could further polish this into an attractive table. See here for a few notes:
http://jeromyanglim.blogspot.com.au/2009/02/formatting-correlation-matrices-in.html

Regression
We could fit three regression models predicting perceptions of goal attainment: (1) transactional, (2) transformational, (3) all predictors:



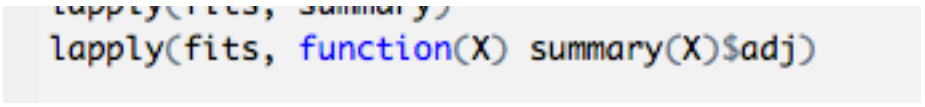
~~~
# Regression models
```{r}
# Just transactional
v$mlq_transactional
v$mlq_transformational
fits <- list()

fits$transactional <- lm(goal ~ mbep + mbea + cr, mlq)
fits$transformational <- lm(goal ~ ac + is + ic, mlq)
fits$all <- lm(goal ~ ac + is + ic + mbep + mbea + cr, mlq)

lapply(fits, summary)
lapply(fits, function(X) summary(X)$adj)

# does adding transactional or transformational add predictions
anova(fits$transactional, fits$all)
anova(fits$transformational, fits$all)
summary(fits$all)
# standardised betas
summary(lm.beta::lm.beta(fits$all))
```
~~~

- list of fits: I've stored the results of the model fits, to make it easier to loop over them.
- lm: lm stands for linear model and performs standard multiple regression with the dv predicted by (~) the predictors separated by the plus symbol.
- lapply(fits, summary). This gets a summary for each fit. For example, for the all predictor model:



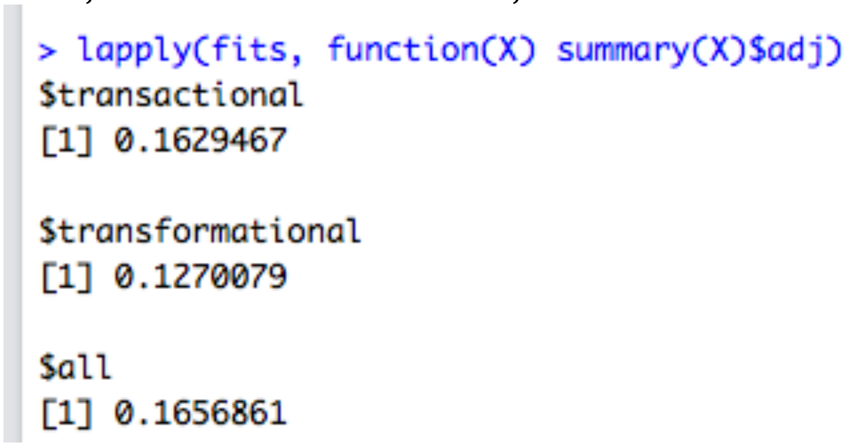
```
$all

Call:
lm(formula = goal ~ ac + is + ic + mbep + mbea + cr, data = mlq)

Residuals:
     Min       1Q   Median       3Q      Max 
-2.68957 -0.27911  0.03666  0.37970  1.51355 

Coefficients:
             Estimate Std. Error t value Pr(>|t|)    
(Intercept)  3.30356    0.18818  17.556  < 2e-16 ***
ac           0.01884    0.04128   0.456   0.6482    
is          -0.00389    0.04395  -0.089   0.9295    
ic           0.08216    0.04492   1.829   0.0677 .  
mbep        -0.16150    0.03229  -5.001 6.74e-07 ***
mbea         0.16045    0.03059   5.245 1.91e-07 ***
cr           0.05399    0.03401   1.587   0.1127    
---
Signif. codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1

Residual standard error: 0.5723 on 990 degrees of freedom
Multiple R-squared:  0.1707,	Adjusted R-squared:  0.1657 
F-statistic: 33.97 on 6 and 990 DF,  p-value: < 2.2e-16
```

- It shows the unstandardized coefficients and their statistical significance, along with important summary information such as significance, R-squared, and adjusted r-squared.
- So this model explains 16.6% of variance MBEA is a significant positive predictor of goal attainment, and MBEP is a significant negative predictor.



```
lapply(fits, summary)
lapply(fits, function(X) summary(X)$adj)
```

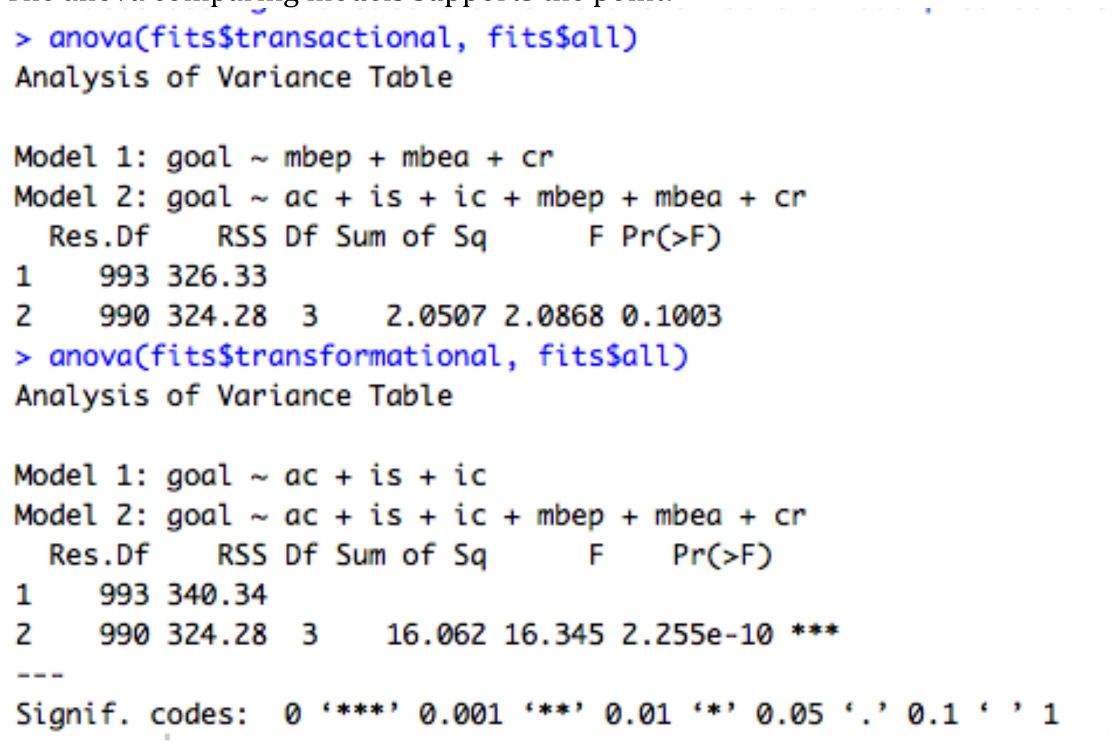
- This extracts the adjusted r-squared from each fit.
- If you didn't know this, you could assign the result of summary(X) to an object and then use tab on that object to see what it contained. Or just google.



```
> lapply(fits, function(X) summary(X)$adj)
$transactional
[1] 0.1629467

$transformational
[1] 0.1270079

$all
[1] 0.1656861
```

- It shows that transactional seems to be a better predictor of goal attainment than transformational.

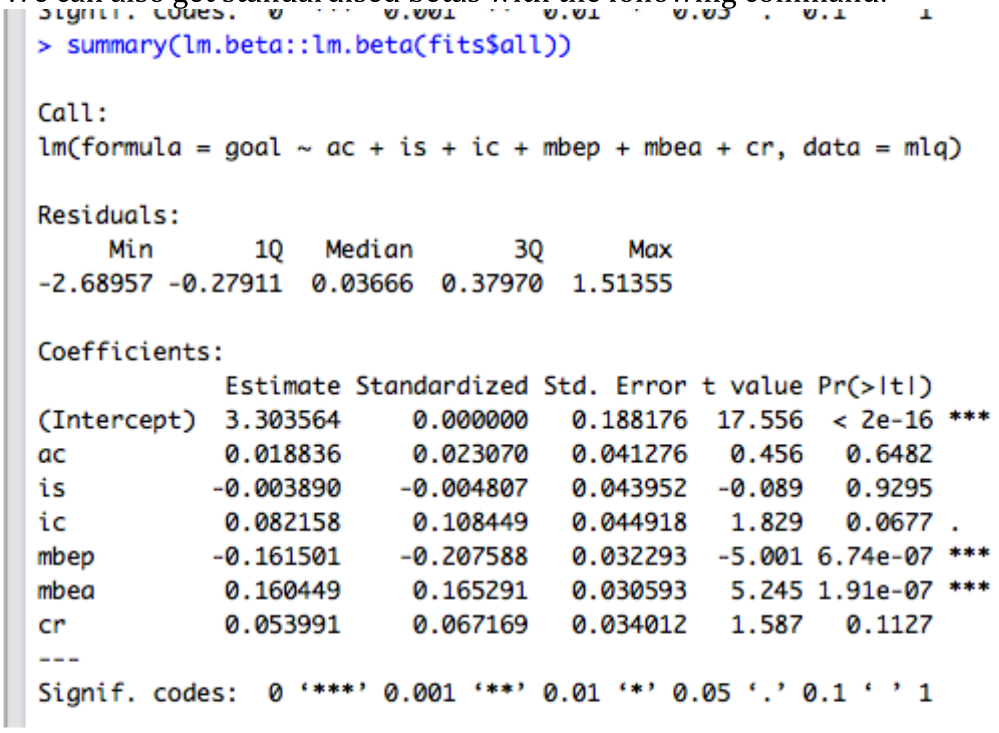
The anova comparing models supports the point:



```
> anova(fits$transactional, fits$all)
Analysis of Variance Table

Model 1: goal ~ mbep + mbea + cr
Model 2: goal ~ ac + is + ic + mbep + mbea + cr
  Res.Df    RSS Df Sum of Sq      F Pr(>F)
1    993 326.33
2    990 324.28  3    2.0507 2.0868 0.1003
> anova(fits$transformational, fits$all)
Analysis of Variance Table

Model 1: goal ~ ac + is + ic
Model 2: goal ~ ac + is + ic + mbep + mbea + cr
  Res.Df    RSS Df Sum of Sq      F    Pr(>F)
1    993 340.34
2    990 324.28  3    16.062 16.345 2.255e-10 ***
---
Signif. codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1
```

- transactional adds prediction over and above transformational, but vice versa is not true.

We can also get standardised betas with the following command:



```
> summary(lm.beta::lm.beta(fits$all))

Call:
lm(formula = goal ~ ac + is + ic + mbep + mbea + cr, data = mlq)

Residuals:
     Min       1Q   Median       3Q      Max
-2.68957 -0.27911  0.03666  0.37970  1.51355

Coefficients:
             Estimate Standardized Std. Error t value Pr(>|t|)
(Intercept)  3.303564     0.000000   0.188176  17.556  < 2e-16 ***
ac           0.018836     0.023070   0.041276   0.456   0.6482
is          -0.003890    -0.004807   0.043952  -0.089   0.9295
ic           0.082158     0.108449   0.044918   1.829   0.0677 .
mbep        -0.161501    -0.207588   0.032293  -5.001 6.74e-07 ***
mbea         0.160449     0.165291   0.030593   5.245 1.91e-07 ***
cr           0.053991     0.067169   0.034012   1.587   0.1127
---
Signif. codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1
```

- E.g., a one standard deviation increase on MBEP predicts a .16 staadard deviation reduction on GOAL