Red Centre Software
Data Comes Alive

Ruby and R

Version: 3.0, 15 August 2015

Confidential

Ruby and R

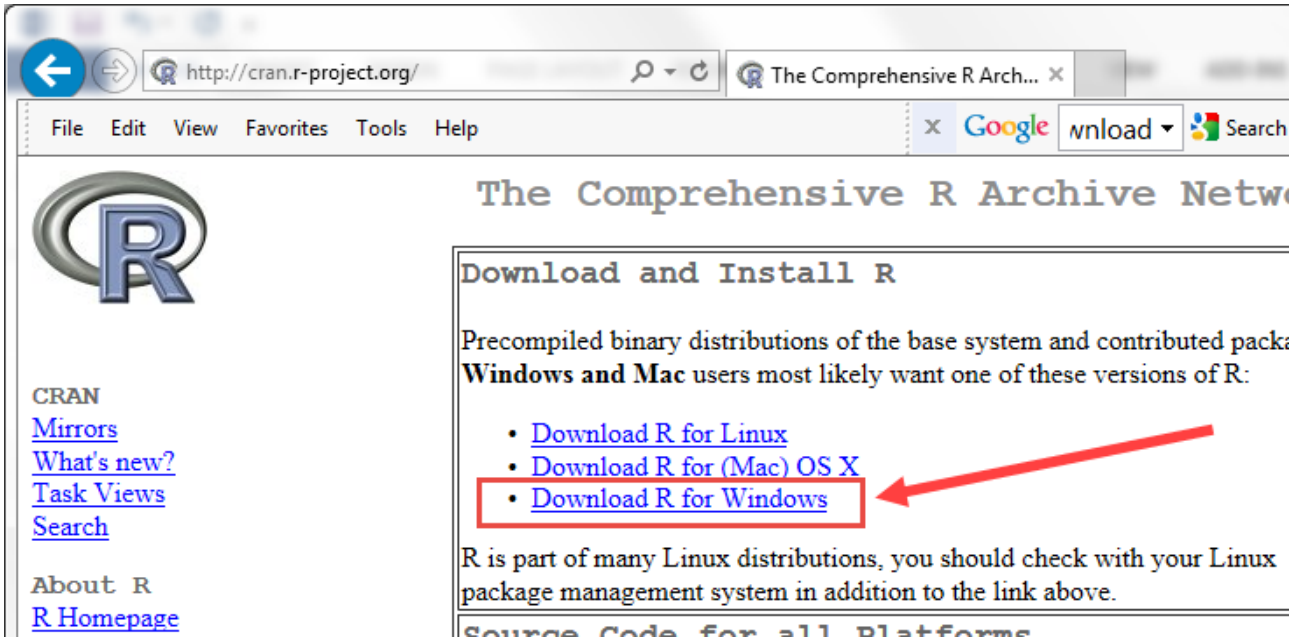
This document shows how to access, modify and execute R routines using a Ruby report as the input.

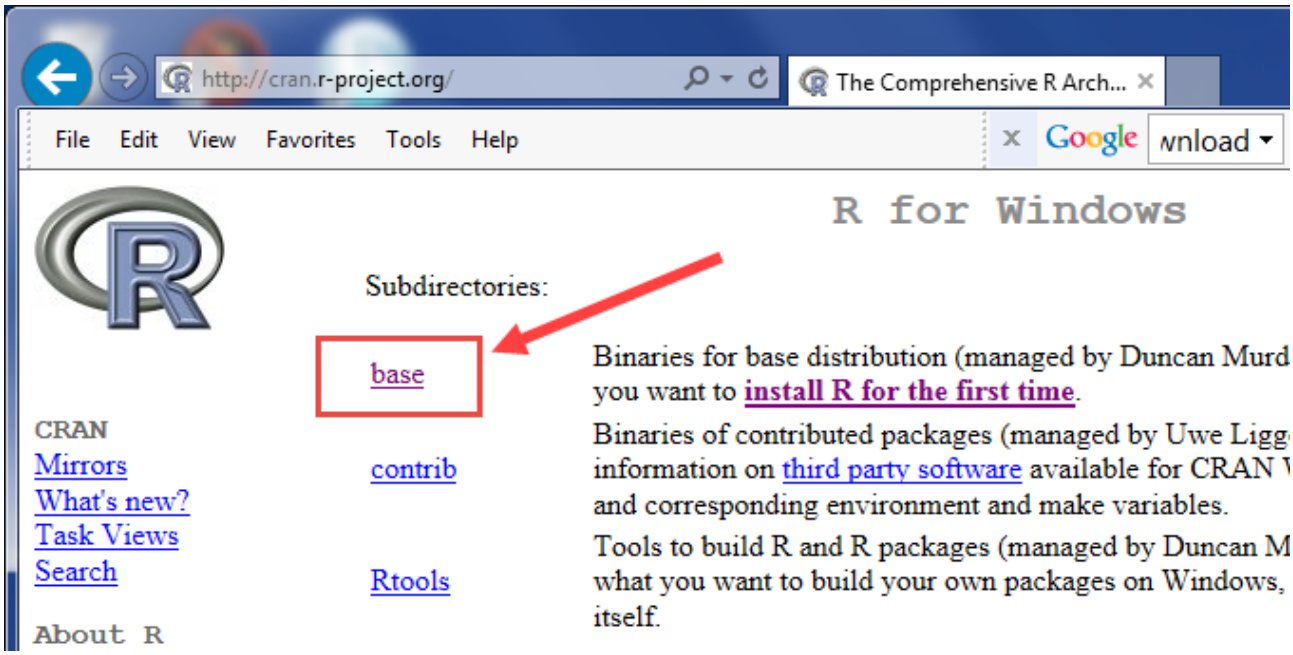
This is not an R tutorial. For details on any of the procedures, see the official R help. It is assumed that you know what you are doing, and can make the appropriate choices regarding inputs. The examples here, though useful in their own right, are intended primarily as the starting point for developing your own set of procedures.

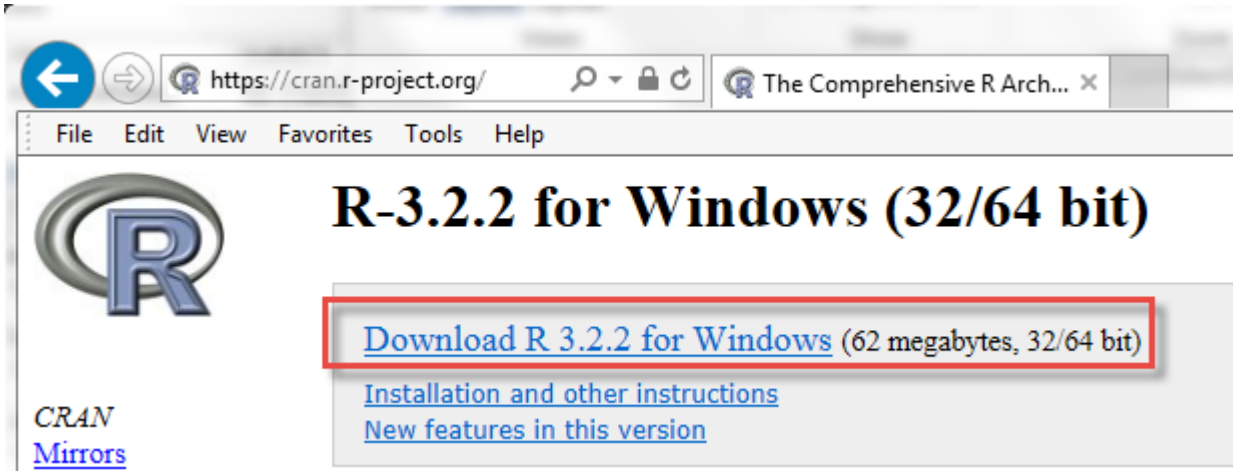
INSTALLING R

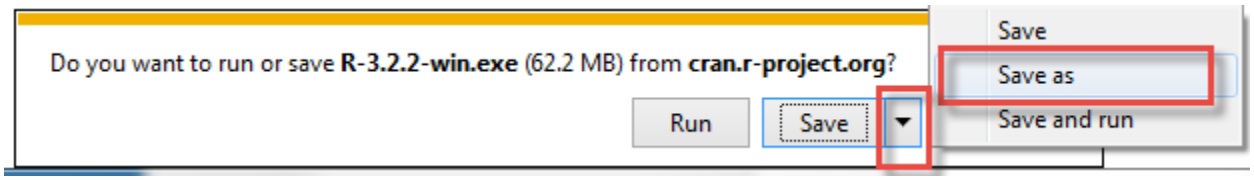
R can be installed from a single download at

http://cran.r-project.org/

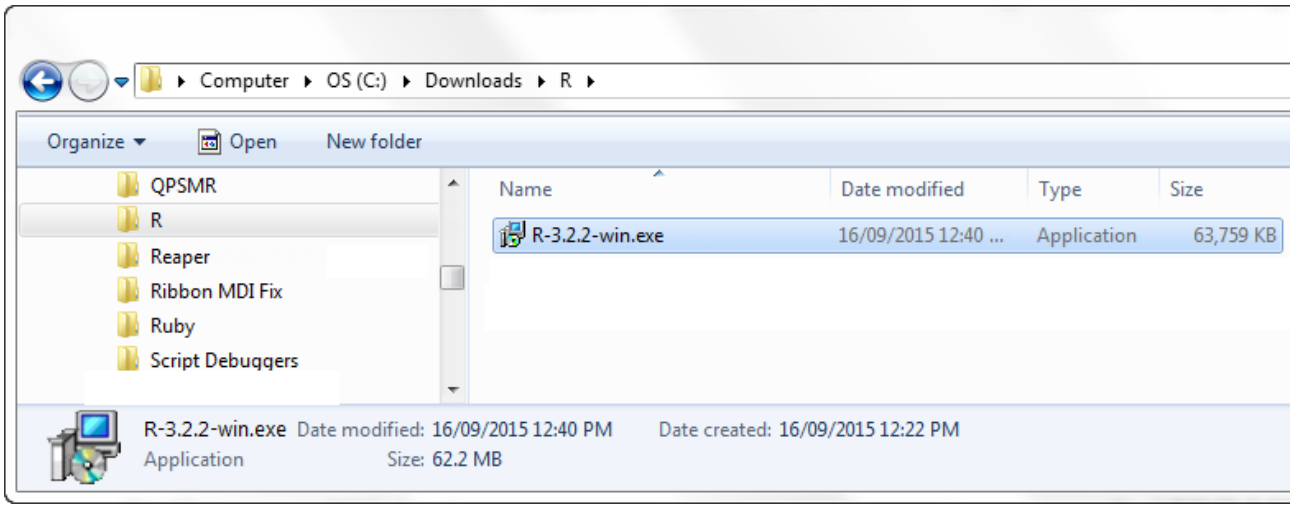




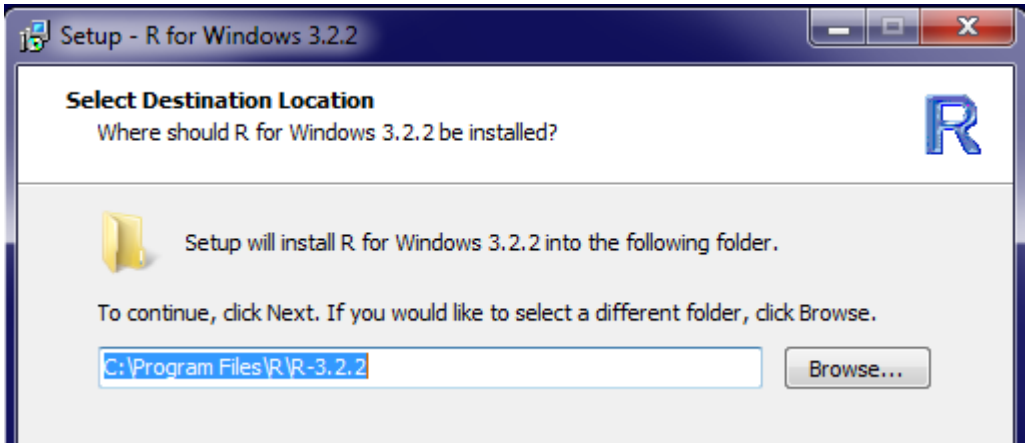




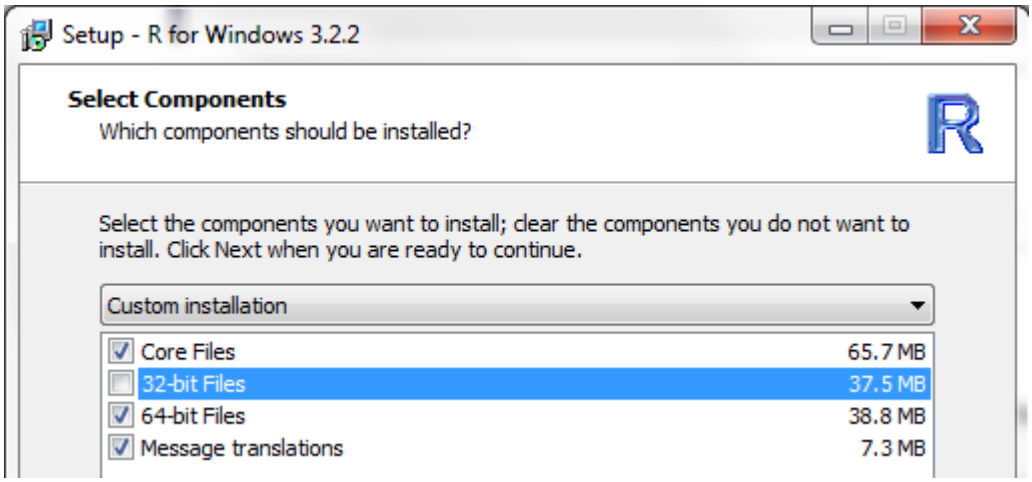
- Save to a subdirectory of your choice
- Run the self-extracting executable by a double-click



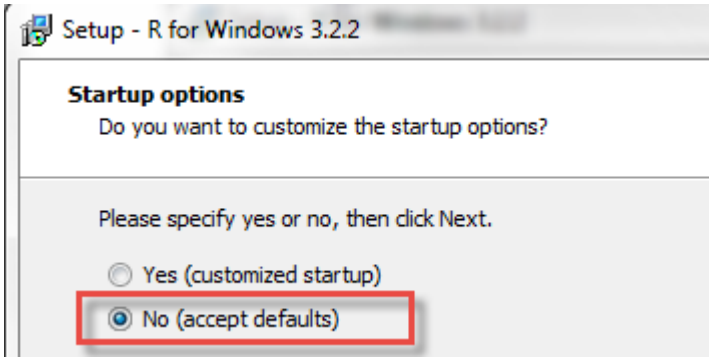
- Install to the default sub-directory



- Deselect the 32 bit option - we want 64 bit (unless you have only Windows 32)



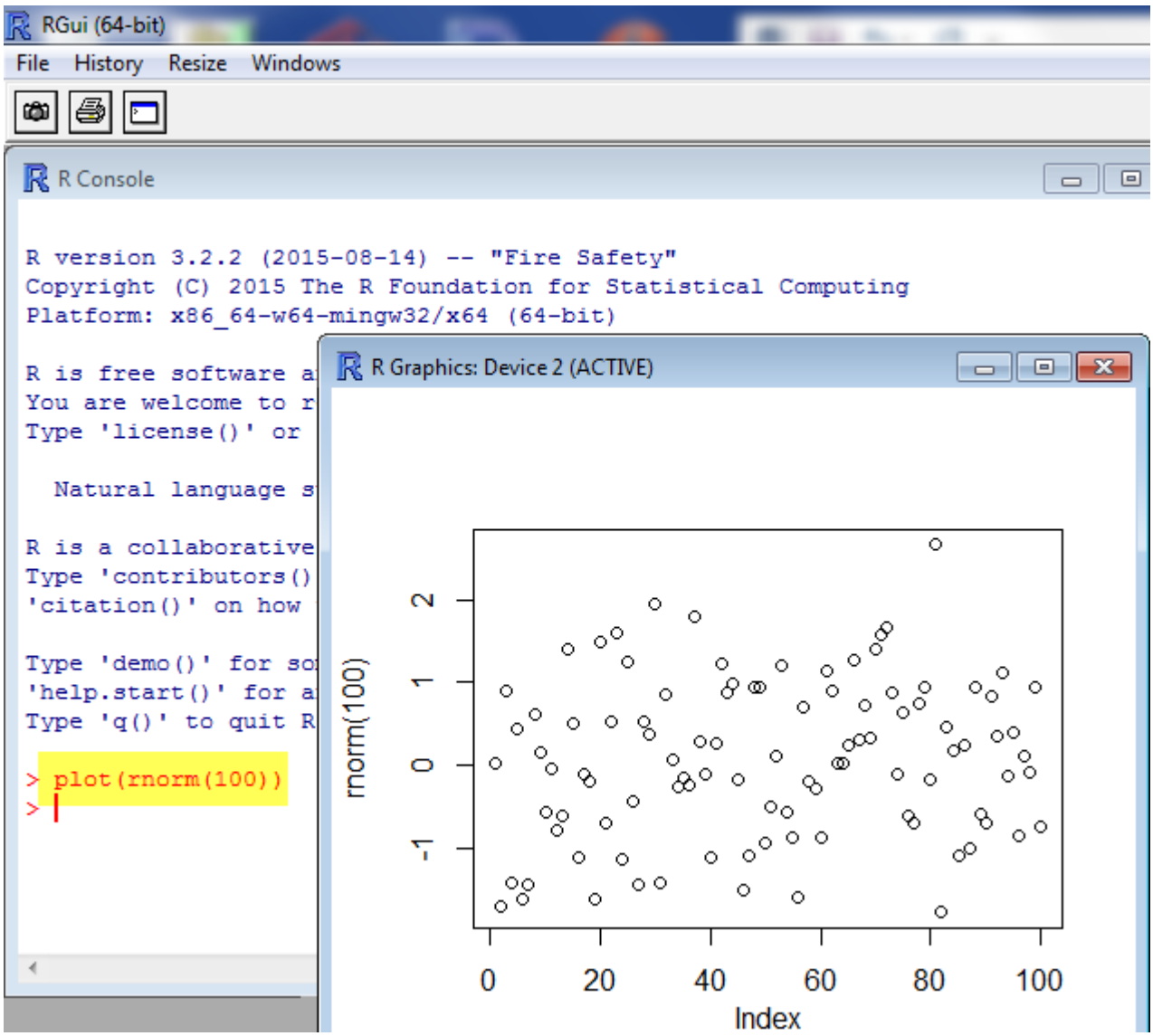
- Accept defaults



- Continue to the end of the installation procedure

R should now be on your desktop.

- Double-click to run
- Type the text *plot(rnorm(100))* at the command prompt and press Enter



A scatter plot appears.

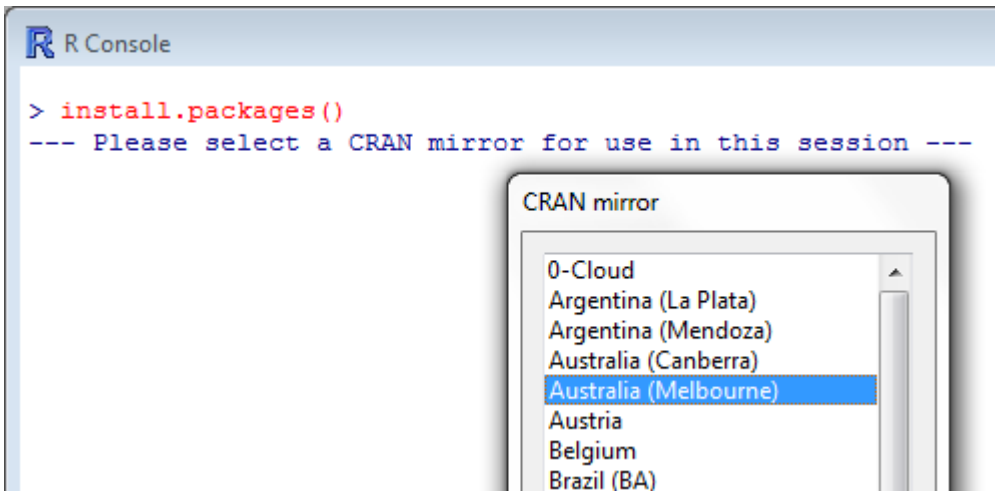
Installing R Packages

In addition to the base functionality, R supports third-party packages for special or extended procedures, and there are thousands of them. The R routines which need packages are

KMeans	psych
Plot	psych
Summary Stats	psych
Correlation	Hmisc, xlsx, corrgram, ellipse

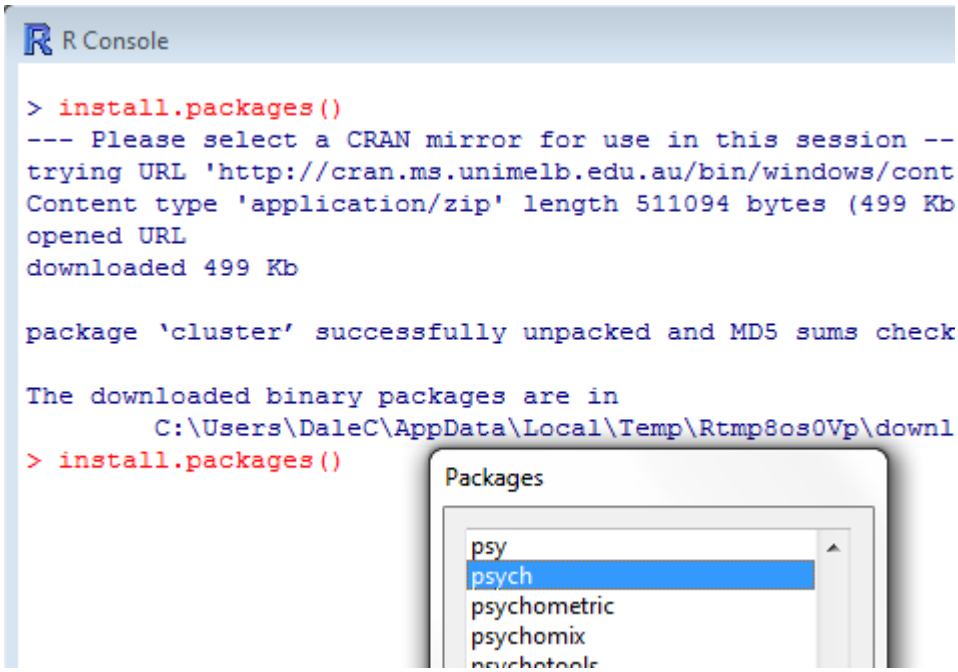
To install the psych package,

- Enter *install.packages()* at the R console, and then select a mirror which is geographically close



The Packages list should then appear.

- Scroll down to psych, select it and click OK




```
> install.packages()
trying URL 'http://cran.ms.unimelb.edu.au/bin/windows/contrib/3.1/psych_1.4.4.z$
Content type 'application/zip' length 2909325 bytes (2.8 Mb)
opened URL
downloaded 2.8 Mb

package 'psych' successfully unpacked and MD5 sums checked

The downloaded binary packages are in
        C:\Users\DaleC\AppData\Local\Temp\Rtmp8os0Vp\downloaded_packages
>
```

The package is downloaded and installed. Repeat for Hmisc, xlsx, corrgram and ellipse.

INSTALLING THE REQUIRED RUBY FILES

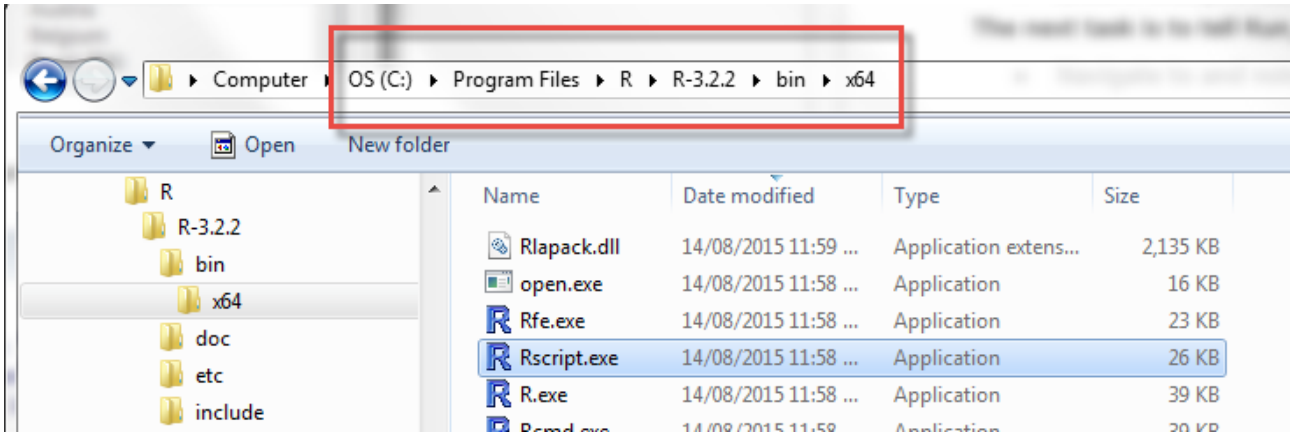
Ruby version 3.0 installations and later will have the files \Ruby\Run_RScripts.vbs, thirteen R procedure files \Ruby*.r, this document \Ruby\Docs\Ruby and R.pdf, and four example reports \Ruby\Jobs\Demo\Reports\Session\r_*.rpt.

Run_RScripts.vbs

- Ensure that you have the file \Ruby\Run_RScripts.vbs

The next task is to tell Run_RScripts.vbs where Rscript.exe is installed on your machine.

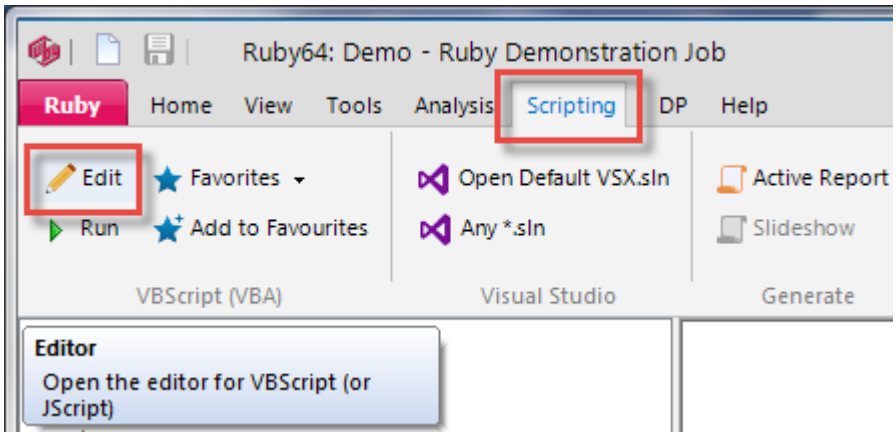
- Navigate to and note the path for Rscript.exe



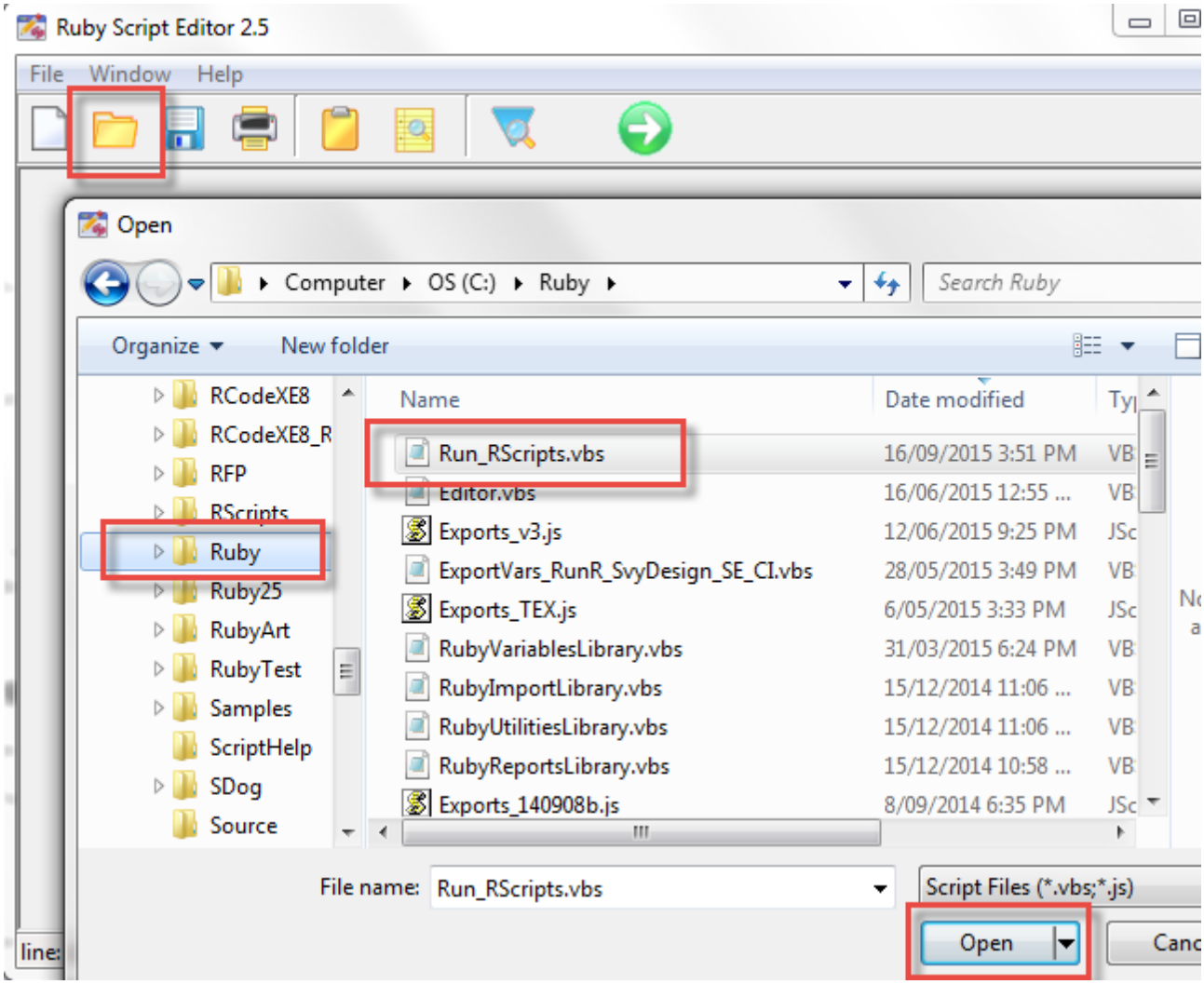
(If 32 bit R and OS, then select the Rscript.exe in just \bin, not \bin\x64.)

The path for this installation is c:\Program Files\R\R-3.2.2\bin\x64\Rscript.exe.

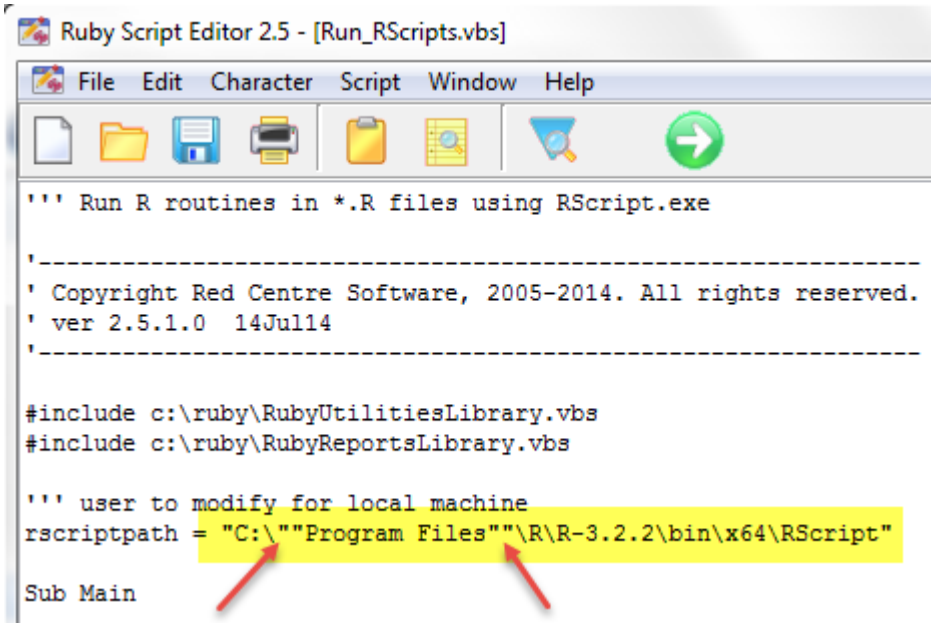
- Scripting | Edit



- File | Open



- Set the path assignment at line 12



Note the double-double quotes, needed because "Program Files" has a space.

The R Procedures

- Ensure that the following R scripts are in your \Ruby subdirectory:

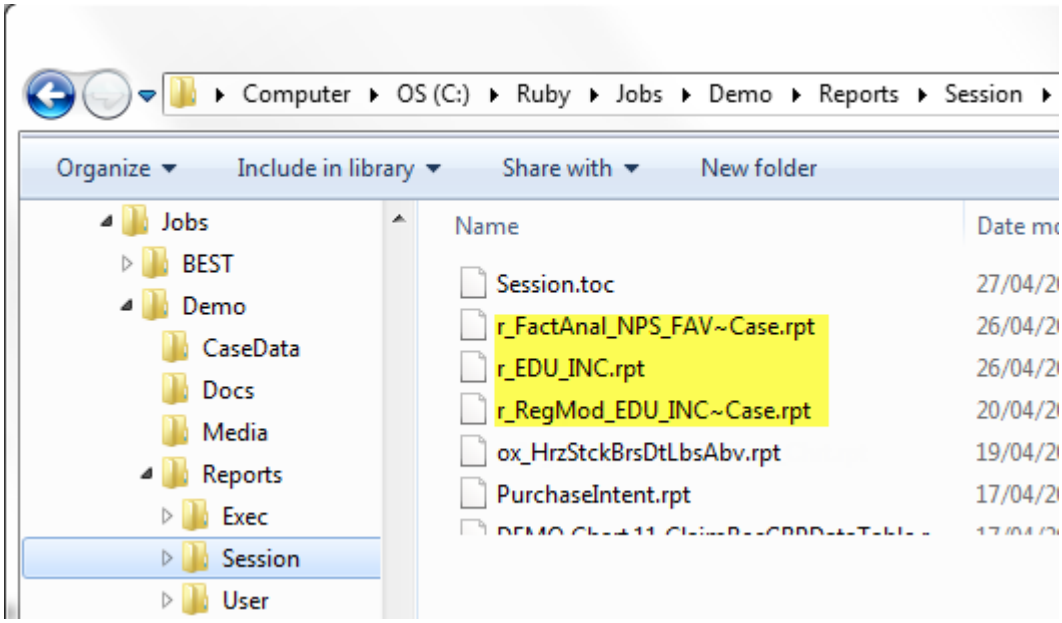
  Anova.r
  ClusterDendro.r
  ClusterPlot.r
  Correlation.r
  CorrespAnaBiPlot.r
  CoVariance.r
  FactorAnalysis.r
  InteractiveKMeans.r
  Plot.r
  PrinCompBiPlot.r
  PrinCompScreePlot.r
  RegressionModel.r
  SummaryStats.r

These scripts implement the procedures. They are Ruby-specific, and will not work unless stored in the \Ruby subdirectory and are called by Run_RScripts.vbs.

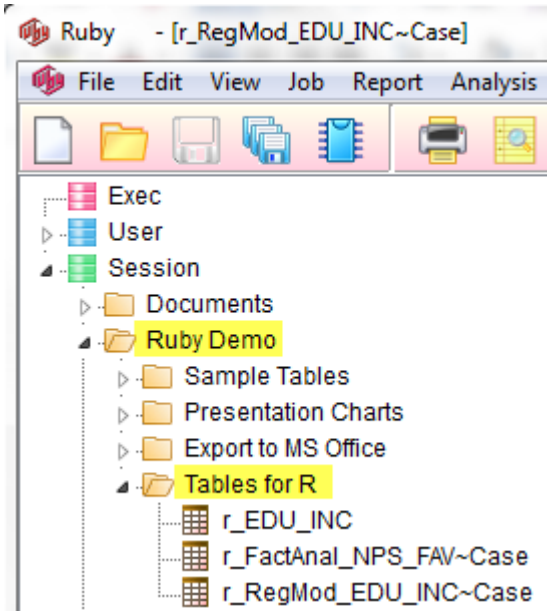
The Example Reports

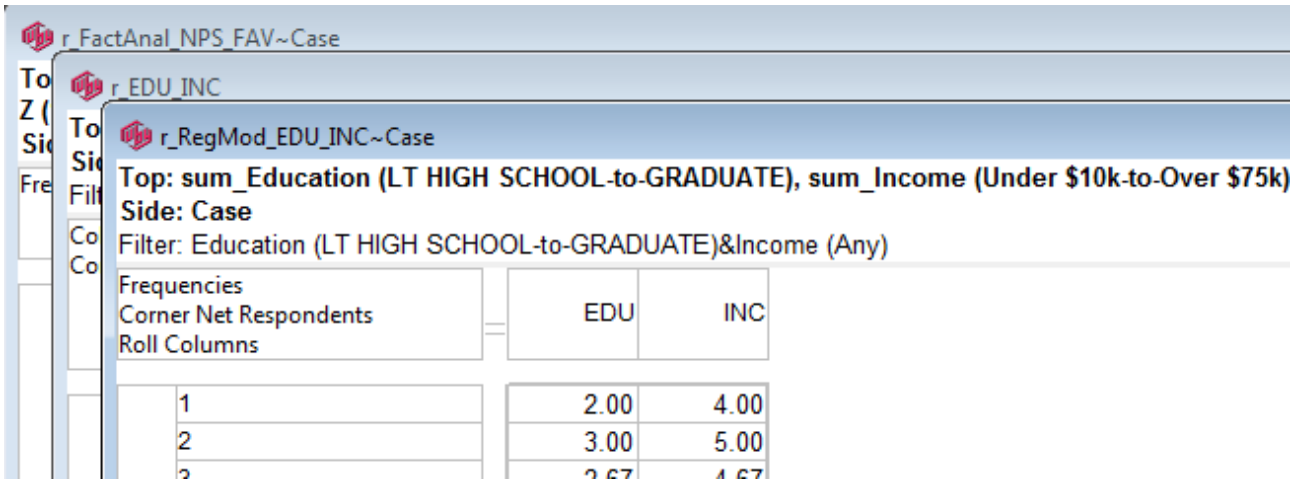
Three reports are used as example inputs.

- Ensure that you have these three *.rpt in ..\Session



- Ensure that you can open these reports from the TOC





If you do not have all of the above files, then download the current Ruby install as a zip, and extract them to the appropriate sub-directories.

The two reports which use Education and Income use actual data from GSS2004, and as such there is a strong positive linear relationship between income and education.

r_EDU_INC

Top: Education
Side: Income
Filter: Education (LT HIGH SCHOOL-to-GRADUATE)&Income (Any)

Column Percents Corner Net Respondents		Education				
		LT HIGH SCHOOL	HIGH SCHOOL	JUNIOR COLLEGE	BACHELOR	GRADUATE
Income	Under $10k	35%	19%	7%	13%	7%
	$10k to 20k	28%	21%	13%	9%	9%
	$20k to 30k	17%	21%	16%	16%	7%
	$30k to 50k	14%	26%	38%	25%	28%
	$50k to 75k	3%	10%	16%	21%	28%
	Over $75k	4%	3%	11%	15%	22%

The NPS and Favourability variables used for factor analysis aggregate as

$Report1

Top: Net Promoter Score Brand
Side: Net Promoter Score Score

Column Percents Corner Net Respondents		Net Promoter Score Brand		
		NPS - BrandX	NPS - BrandY	NPS - BrandZ
Net Promoter Score Score	1	3%	4%	3%
	2	11%	11%	11%
	3	4%	6%	3%
	4	5%	6%	6%
	5	6%	5%	6%
	6	10%	11%	11%
	7	8%	8%	13%
	8	20%	18%	15%
	9	23%	21%	26%
	10	9%	10%	7%
	Net Promoter	-8%	-11%	-7%

$Report2

Top: Favourability Brand
Side: Favourability Rating

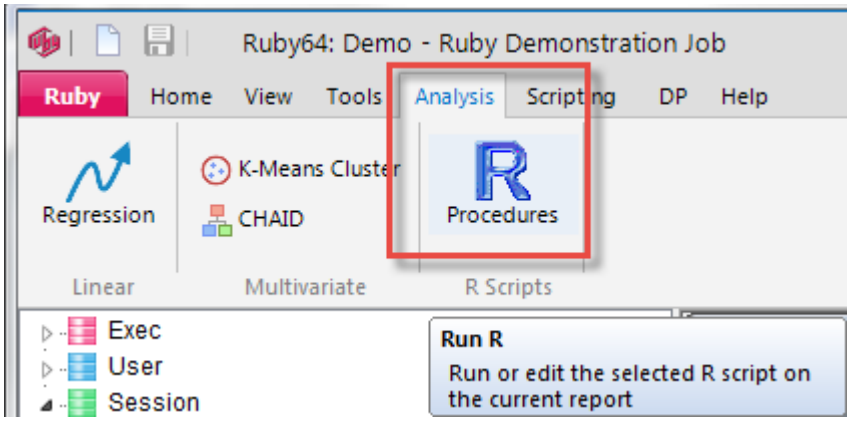
Column Percents Corner Net Respondents		Favourability Brand		
		BrandX	BrandY	BrandZ
Favourability Rating	1	0%	2%	7%
	2	1%	3%	16%
	3	3%	11%	29%
	4	1%	20%	22%
	5	14%	23%	14%
	6	23%	18%	6%
	7	22%	11%	3%
	8	19%	8%	1%
	9	14%	5%	1%
	10	4%	1%	1%
	Code Mean	6.89	5.25	3.71

BrandZ is least favoured, but nonetheless well promoted.

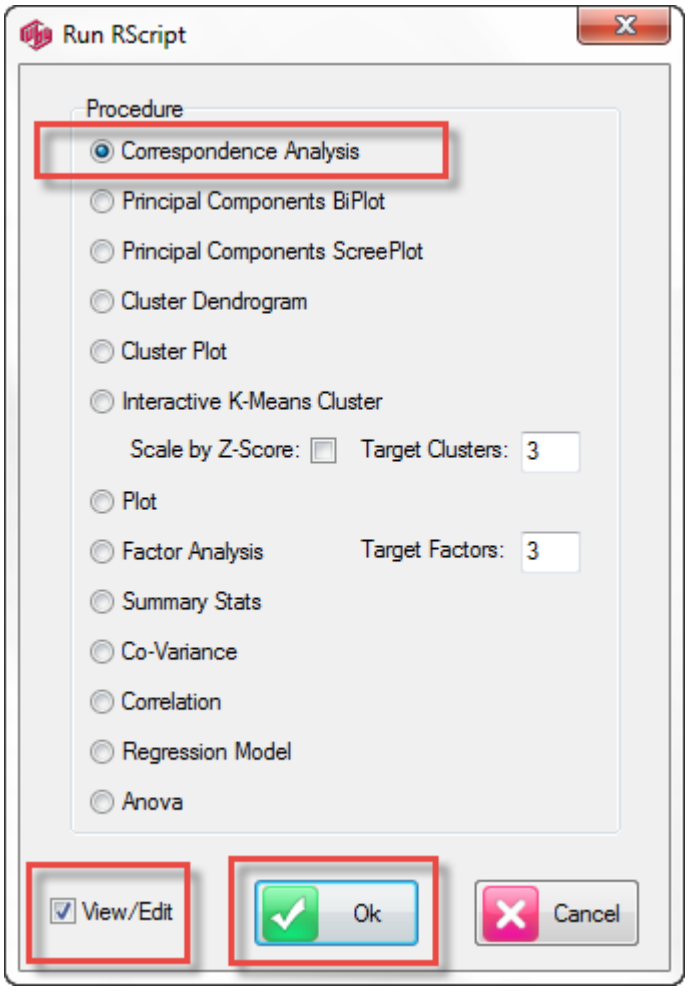
r_EDU_INC is a crosstab, whereas the other two are score tables, one row per case. Examine the specifications to see how they are made.

VIEW OR EDIT AN R SCRIPT

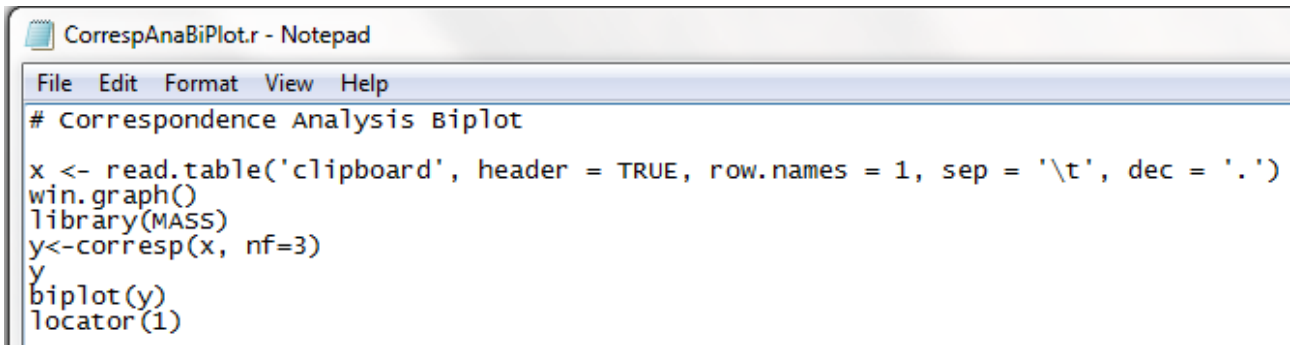
- Analysis | R Procedures



- Select Correspondence Analysis
- Check View/Edit
- Click OK



The R script CorrespAnaBiPlot.r is opened in Notepad.exe.



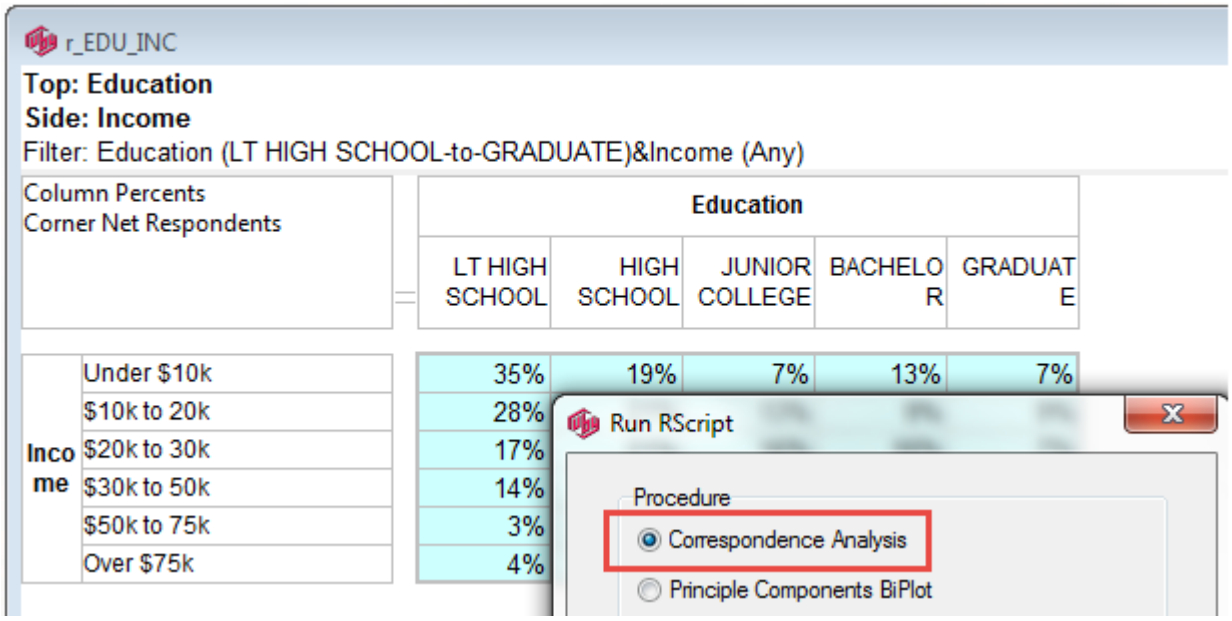
This is how you can both review what R is going to do, and edit the script to meet any custom requirements. Most of the supplied scripts are intentionally as basic as possible, while still doing something useful. The above R script

- Reads the Ruby table from the clipboard (placed there by Run_RScripts.vbs) into x
- Sets up the Windows Graphics device for plotting
- Loads the MASS library (which defines the corresp() routine)
- Runs a correspondence analysis on x and stores the results in y
- Displays the contents of y in a Command window
- Displays the biplot of y on the graphics device
- Waits for a click on the graphics device to close it

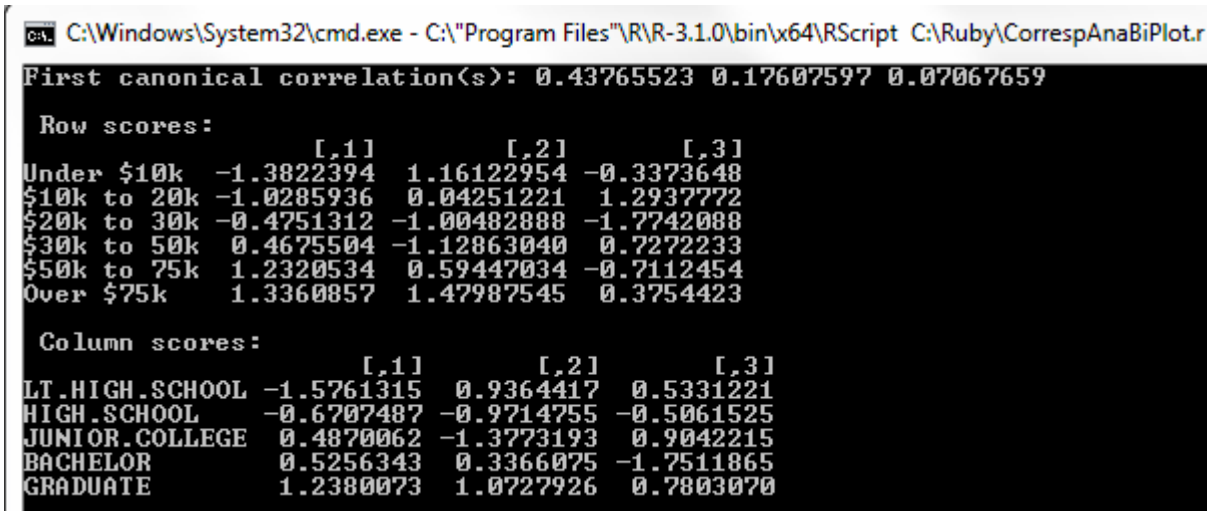
THE PROCEDURES

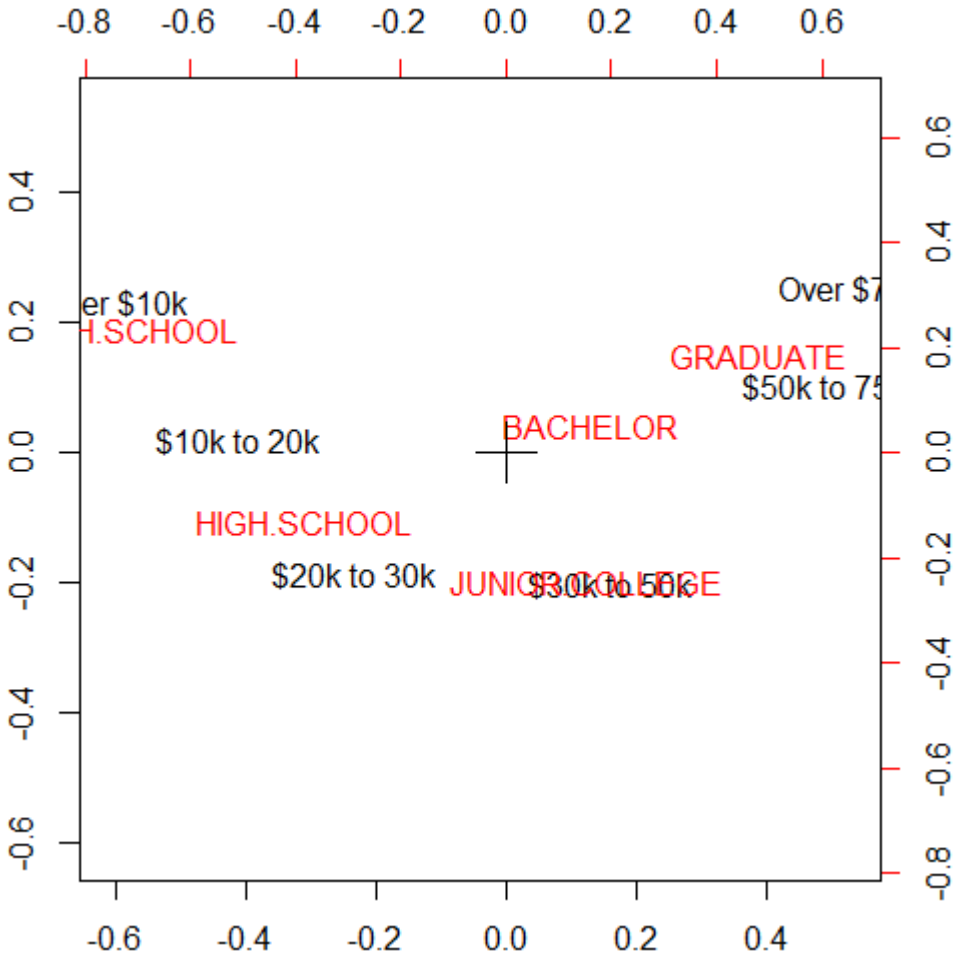
Correspondence Analysis

- Open the supplied table r_EDU_INC
- Job | Scripting | Favourites | Run_RScript
- Select Correspondence Analysis
- Click OK

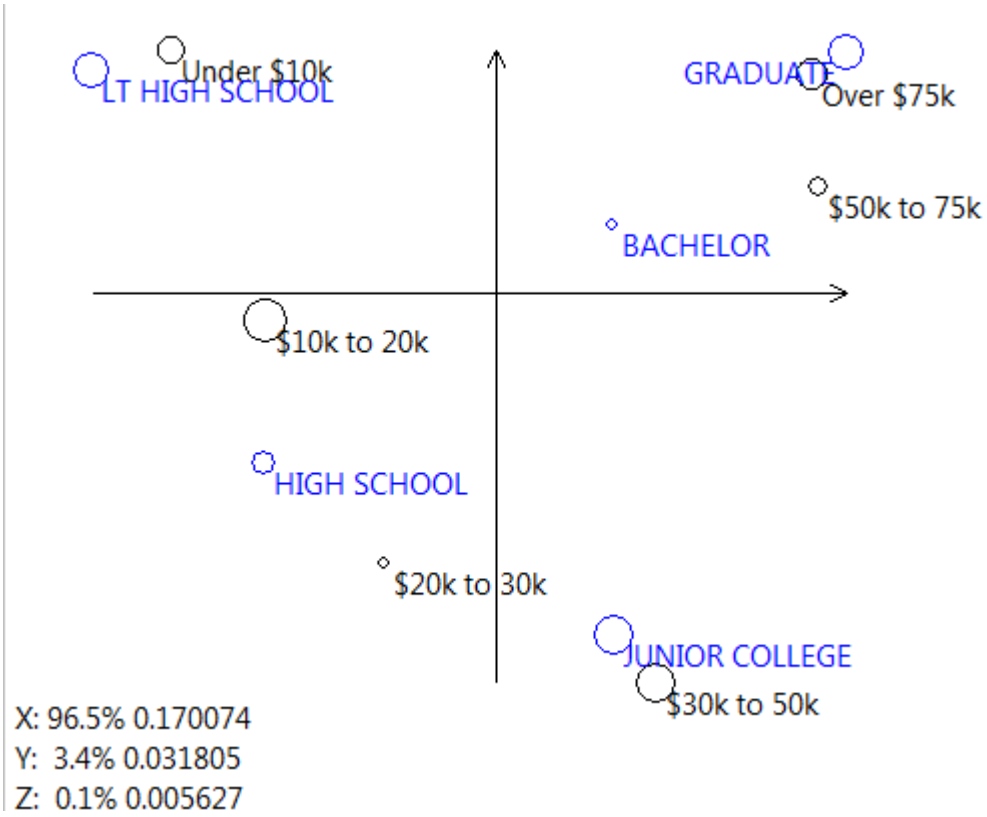


The outputs are





The correspondence map is very close to the Ruby Perceptual Map, allowing for different scaling:



Principal Components Biplot

The R script is

PrinCompBiPlot.r - Notepad

File Edit Format View Help

```
# Principal Components Biplot

x <- read.table('clipboard', header = TRUE, row.names = 1, sep = '\t', dec = '.')
win.graph()
y<-princomp(x)
y
y$loadings
biplot(y)
locator(1)
```

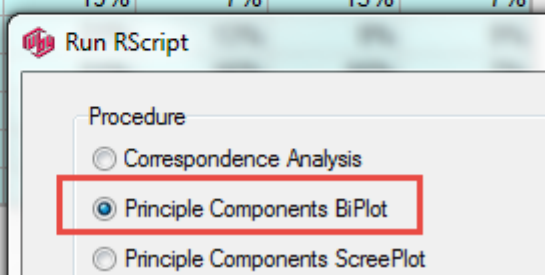
If you are new to R, note the use of . and $. In R, a dot is used like an underscore in most other programming languages. It has no intrinsic meaning - it is just a character. The name could just as well have been wingraph(), but the original programmer decided to call it win.graph(), and in most other languages, it would probably have been win_graph(). In short, the . does not mean 'member of', as it does in C, C#, .Net dialects, VBA, JScript, etc. The 'member of' symbol in R is $, so y$loadings means to display the loadings member of y, which has just been populated by Y<-princomp(x).

Also, the assignment operator in R is usually <-, although nearly always = will do the same thing. Using <- is considered better style.

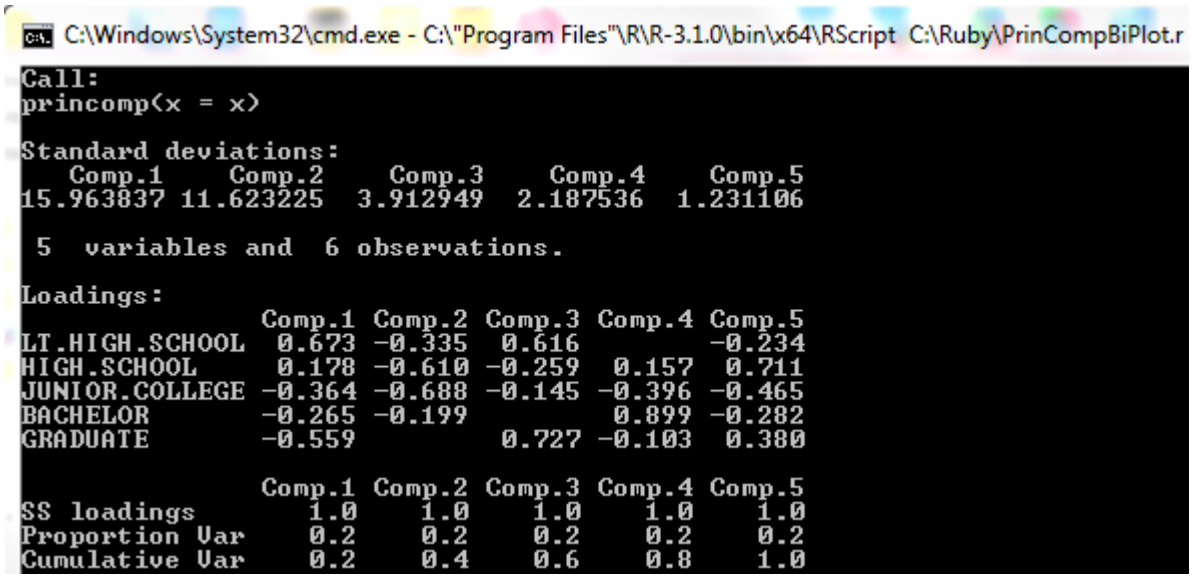
On the same table as above, run as

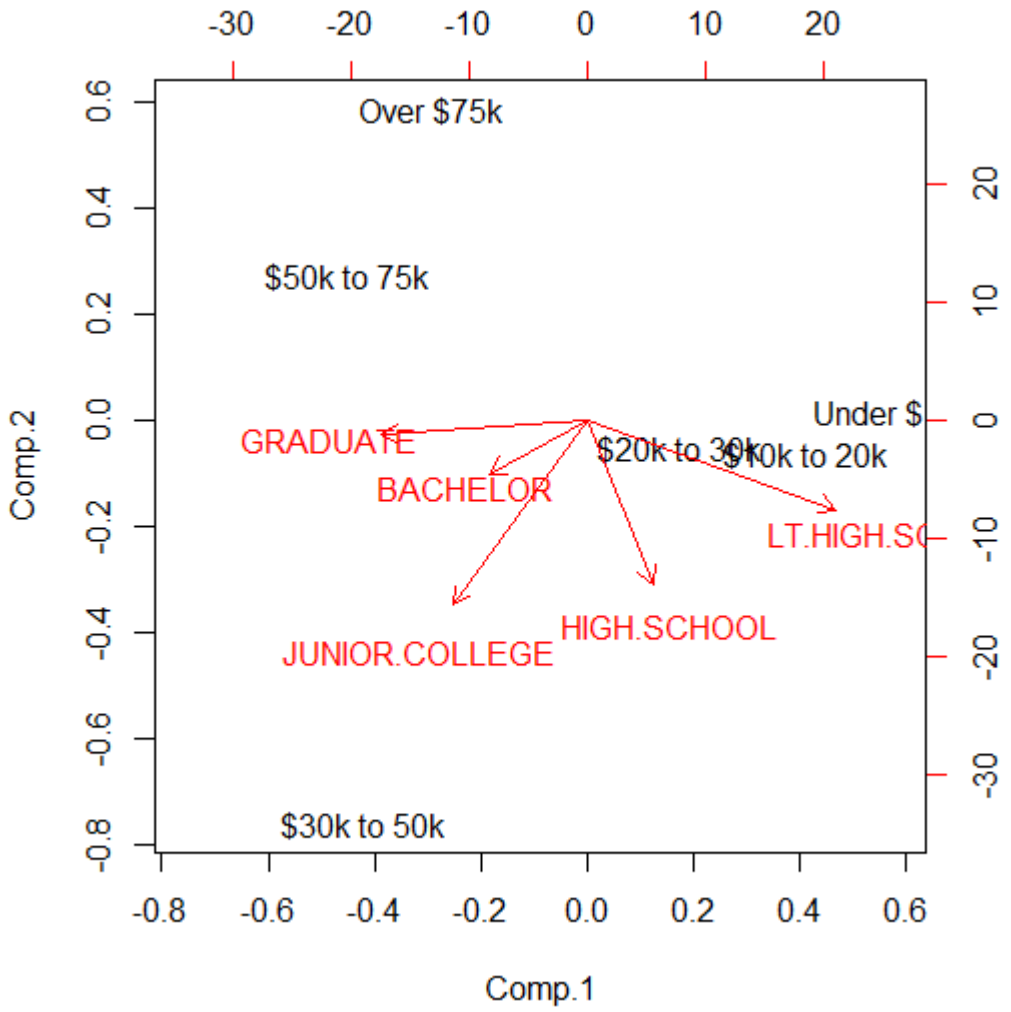
Top: Education
Side: Income
Filter: Education (LT HIGH SCHOOL-to-GRADUATE)&Income (Any)

Column Percents Corner Net Respondents		Education				
		LT HIGH SCHOOL	HIGH SCHOOL	JUNIOR COLLEGE	BACHELOR	GRADUATE
Income	Under $10k	35%	19%	7%	13%	7%
	$10k to 20k	28%				
	$20k to 30k	17%				
	$30k to 50k	14%				
	$50k to 75k	3%				
	Over $75k	4%				



The outputs are





Principal Components Scree Plot

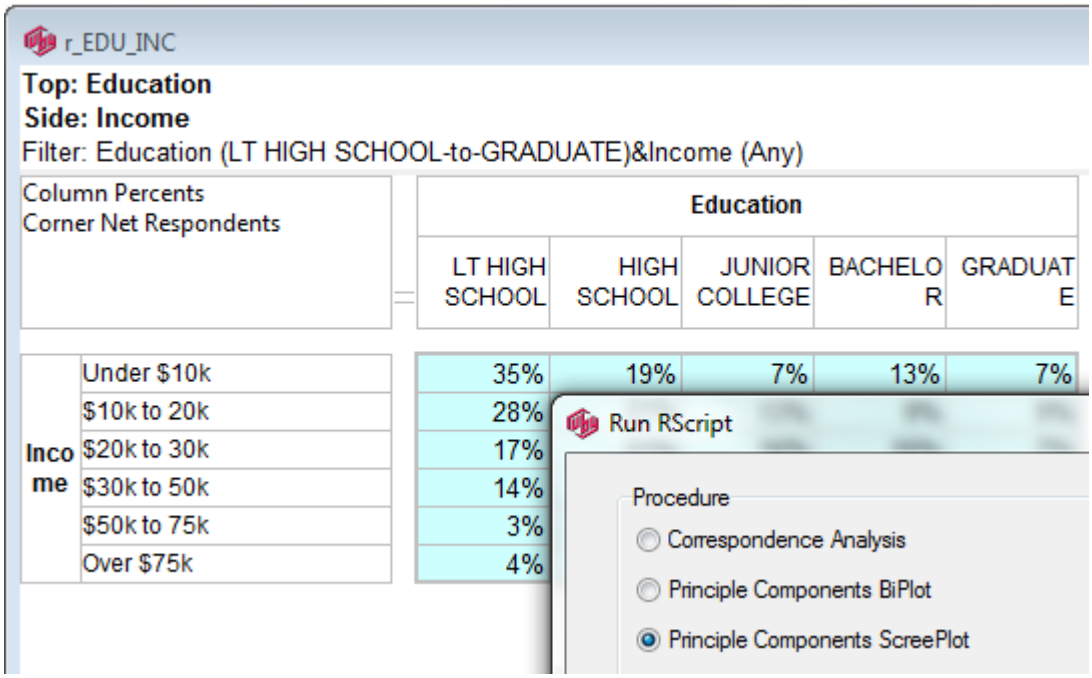
The script is the same as for the biplot, except screeplot() is called instead.

PrinCompScreePlot.r - Notepad

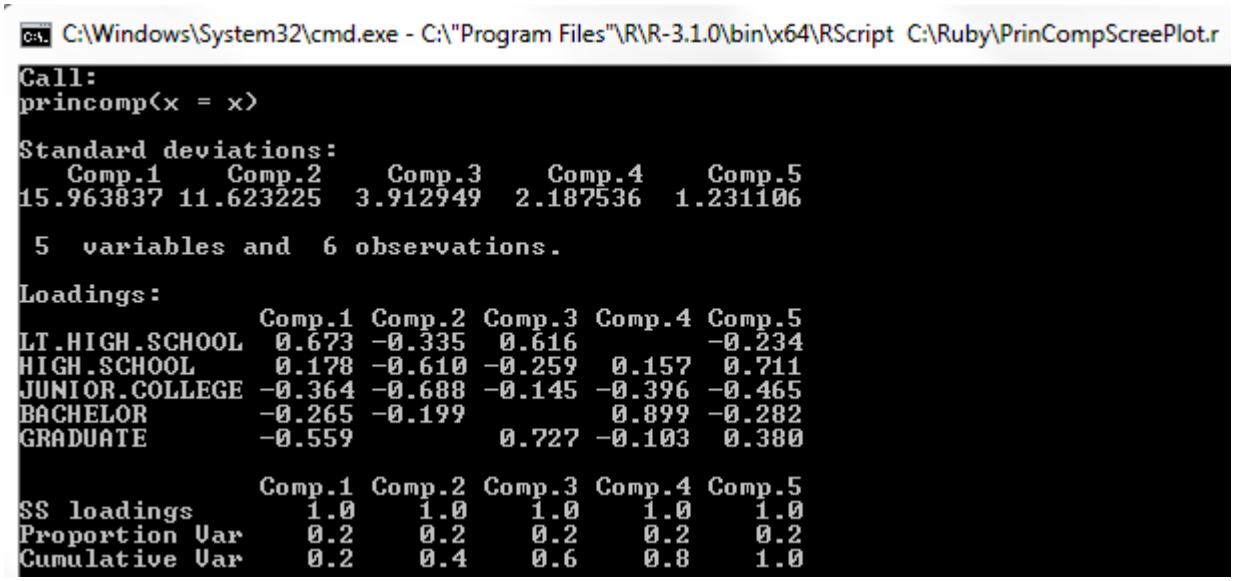
File Edit Format View Help

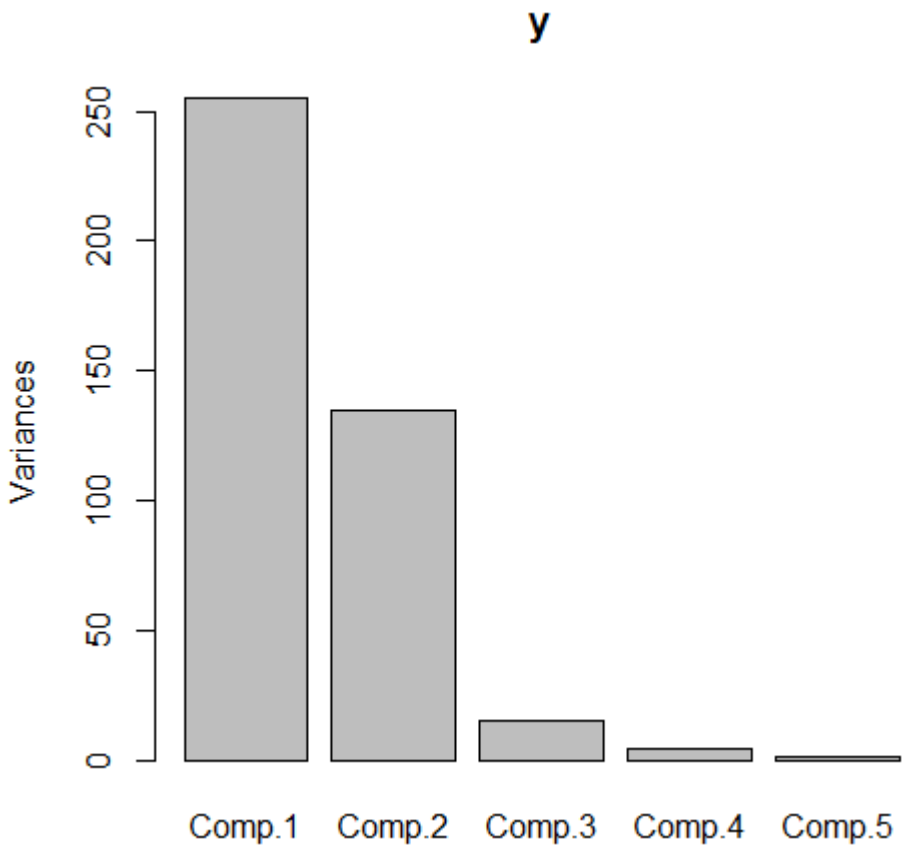
```
# Principal Componments Scree Plot

x <- read.table('clipboard', header = TRUE, row.names = 1, sep = '\t', dec = '.')
win.graph()
y <- princomp(x)
y
y$loadings
screeplot(y)
locator(1)
```

On the same table as above, run as



The outputs are





This shows that there are two dominant components.

Cluster Dendrogram

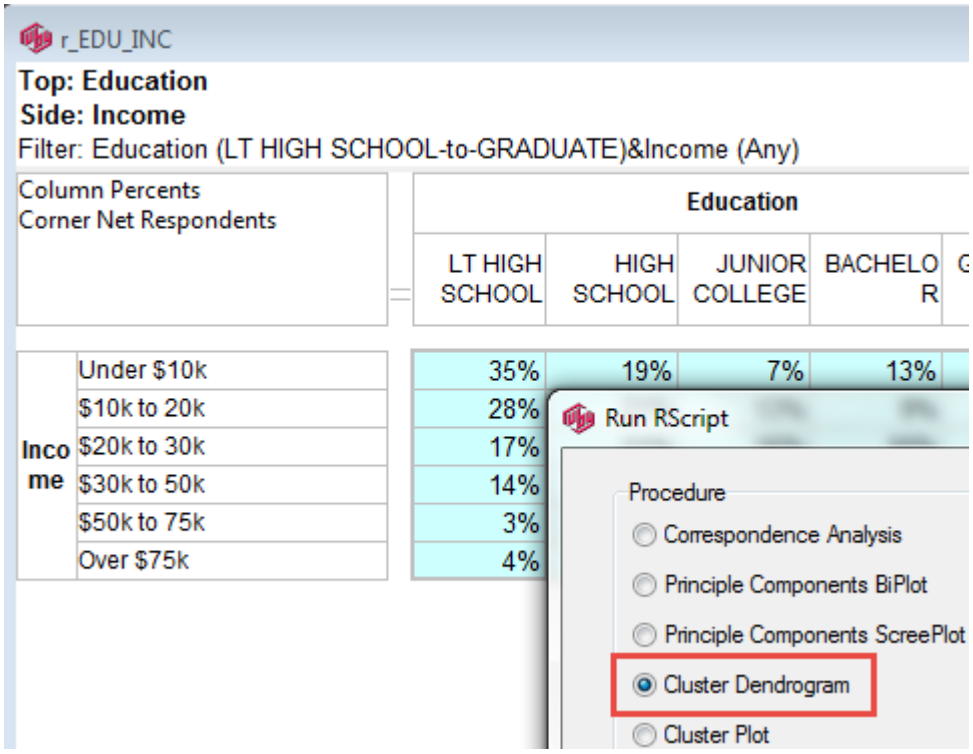
The script is

ClusterDendro.r - Notepad

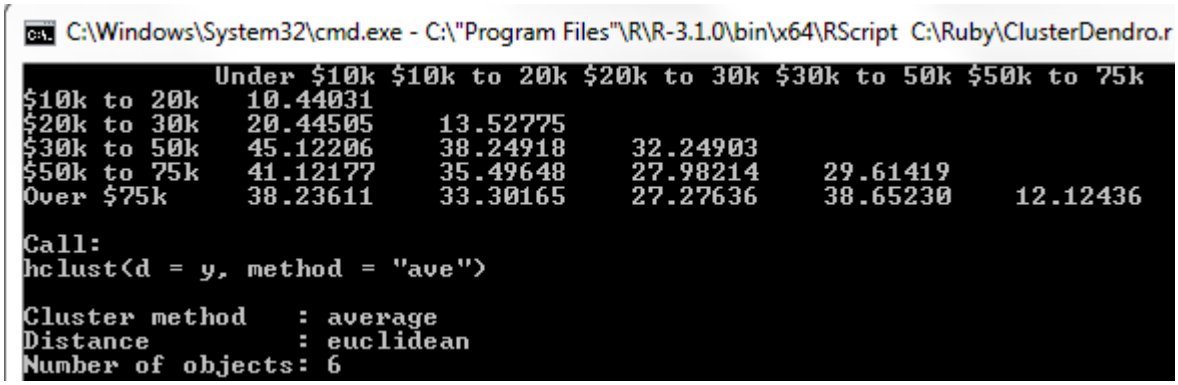
File Edit Format View Help

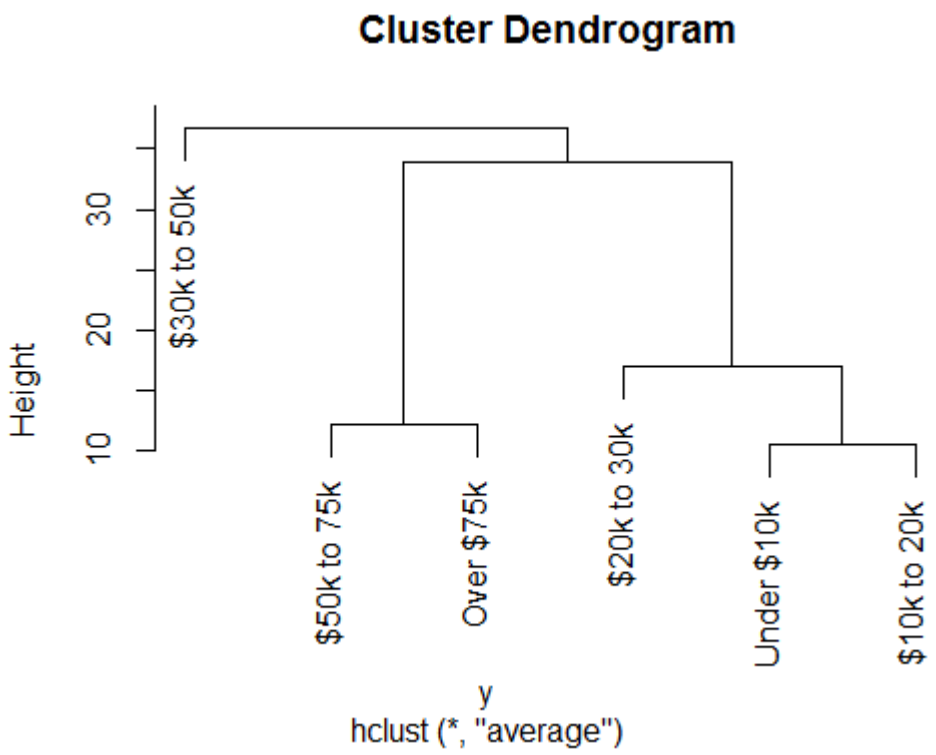
```
# Cluster Dendrogram

x <- read.table('clipboard', header = TRUE, row.names = 1, sep = '\t', dec = '.')
win.graph()
y <- dist(x)
y
z <- hclust(y, 'ave')
z
plot(z)
locator(1)
```

Run on the same table as above, as



The outputs are





Cluster Plot

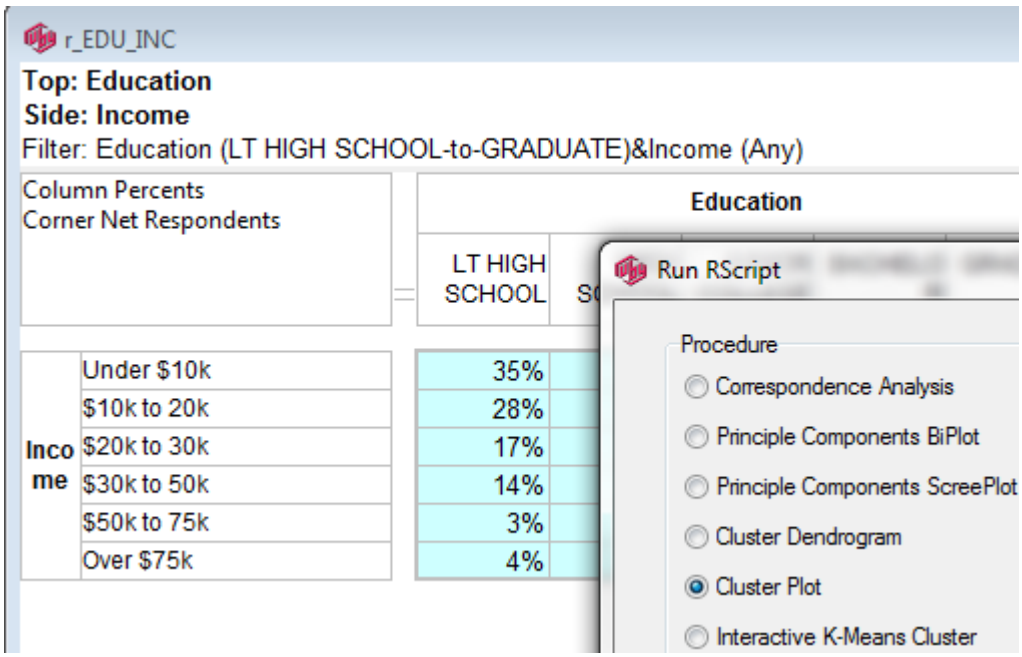
This script is a bit more complicated:

ClusterPlot.r - Notepad

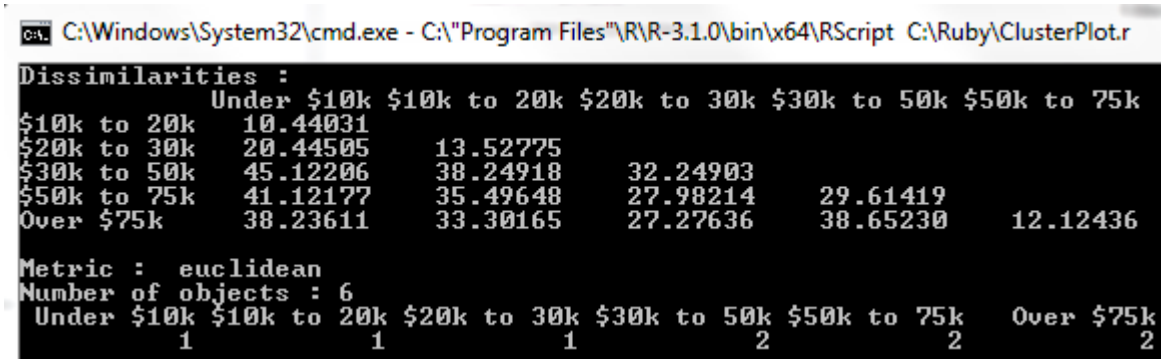
File Edit Format View Help

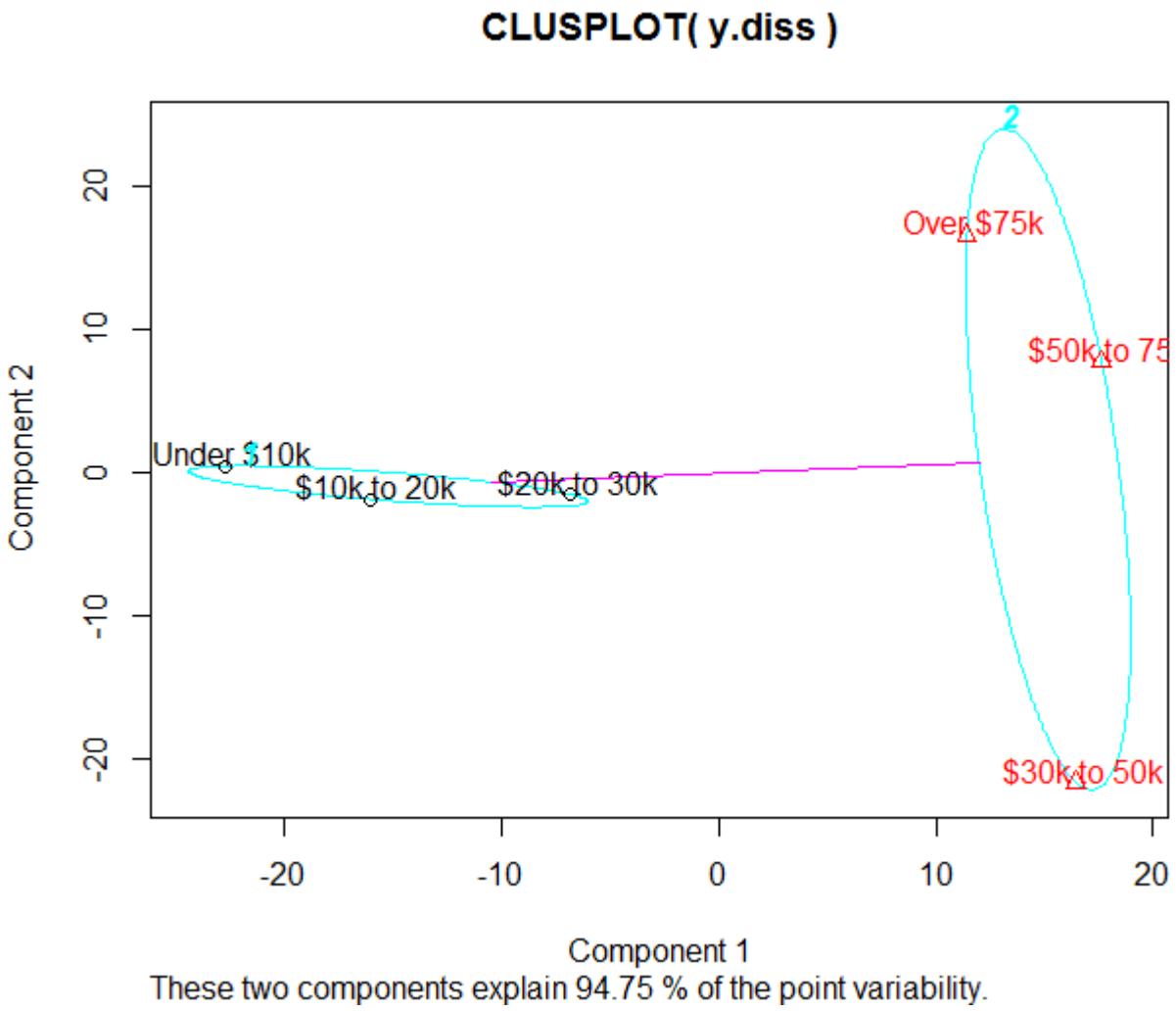
```
# Cluster Plot

x <- read.table('clipboard', header = TRUE, row.names = 1, sep = '\t', dec = '.')
win.graph()
library(cluster)
y.diss <- daisy(x)
y.diss
y.clus <- pam(y.diss, 2, diss = TRUE)$clustering
y.clus
###uncomment the preferred clusplot
clusplot(y.diss, y.clus, diss = TRUE, col.p = y.clus, labels = 2)     # color points and labels
#clusplot(y.diss, y.clus, diss = TRUE, labels=2, shade = TRUE)
#clusplot(y.diss, y.clus, diss = TRUE, labels=2, span = FALSE)        # simple ellipses

locator(1)
```

Run on the same table as above, as


r_EDU_INC
Top: Education
Side: Income
Filter: Education (LT HIGH SCHOOL-to-GRADUATE)&Income (Any)
Column Percents
Corner Net Respondents
Education
LT HIGH SCHOOL
Income
Under $10k
$10k to 20k
$20k to 30k
$30k to 50k
$50k to 75k
Over $75k
35%
28%
17%
14%
3%
4%
Run RScript
Procedure
Correspondence Analysis
Principle Components BiPlot
Principle Components ScreePlot
Cluster Dendrogram
Cluster Plot
Interactive K-Means Cluster

Outputs are


C:\Windows\System32\cmd.exe - C:\"Program Files"\R\R-3.1.0\bin\x64\RScript C:\Ruby\ClusterPlot.r
Dissimilarities :
Under $10k $10k to 20k $20k to 30k $30k to 50k $50k to 75k
$10k to 20k 10.44031
$20k to 30k 20.44505 13.52775
$30k to 50k 45.12206 38.24918 32.24903
$50k to 75k 41.12177 35.49648 27.98214 29.61419
Over $75k 38.23611 33.30165 27.27636 38.65230 12.12436
Metric : euclidean
Number of objects : 6
Under $10k $10k to 20k $20k to 30k $30k to 50k $50k to 75k Over $75k
1 1 1 2 2 2



K-Means Cluster Analysis

This is a much more sophisticated example. The script is too long to review here.

The analysis both determines the optimal number of clusters and does the clustering itself. On the first pass, the number of cluster targets is automatically set to number of rows -1, and you make an educated guess as to the most likely number of target clusters. You then examine the scree plots for an 'elbow' and use the elbow value as the target clusters for a second run if the educated guess was not appropriate. You may of course prefer a coarser clustering than the scree plots suggest, but there is never any reason to ask for more, since the extra clusters will be just noise.

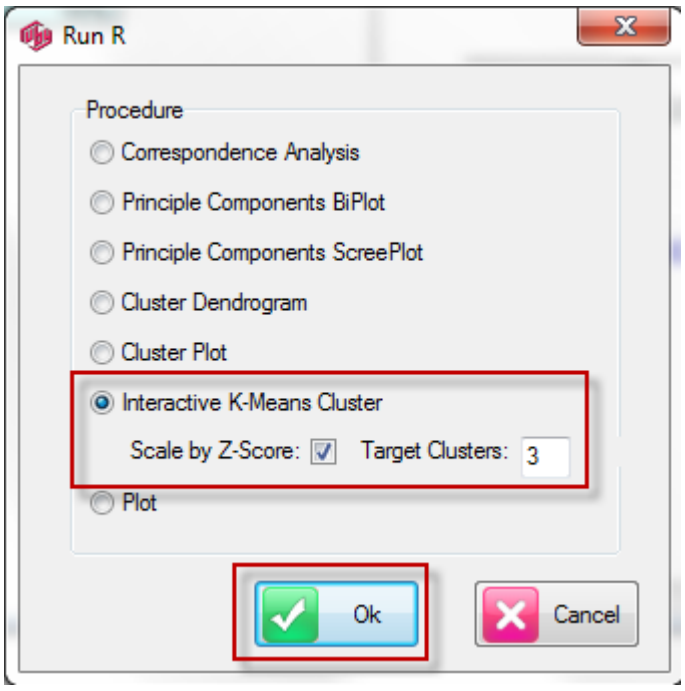
The k-means routine is adapted from

http://www.mattpeeples.net/kmeans.html

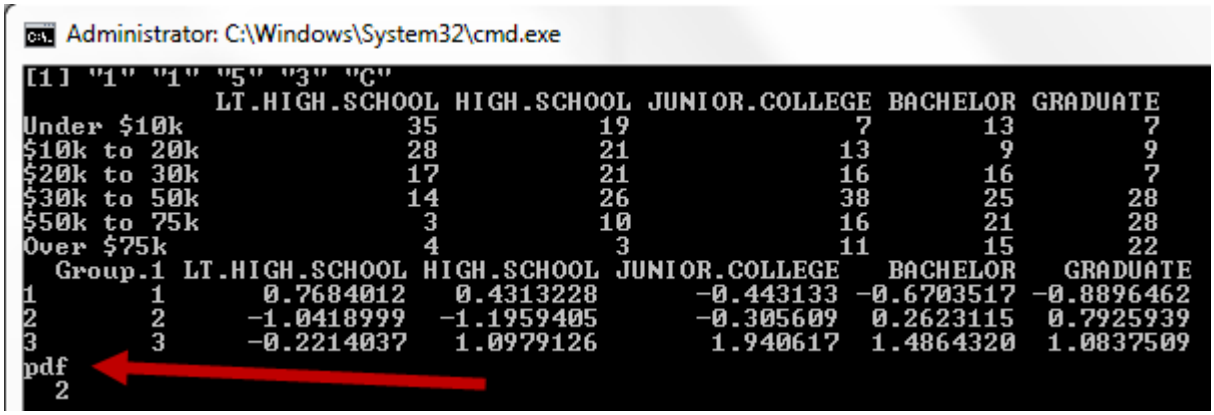
(Peeples, Matthew A. (2011) R Script for K-Means Cluster Analysis)

where the meaning of the various charts is fully discussed. The original version of this script has four user prompts: percents yes/no, zscore scaling yes/no, number of test clusters and number of target clusters. These inputs here come from the Ruby table (percent status) and from the Run_RScripts form, with test clusters set to number of rows -1. Apart from this, the only other difference is that a different example table is used.

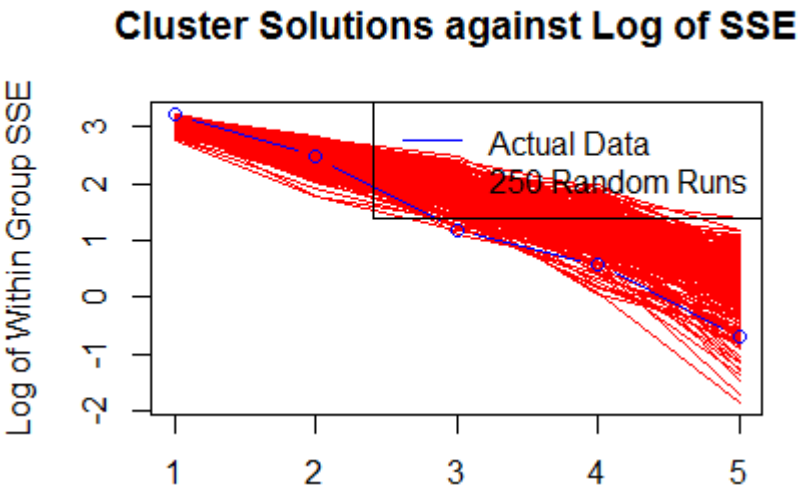
Continuing with the above table, run as

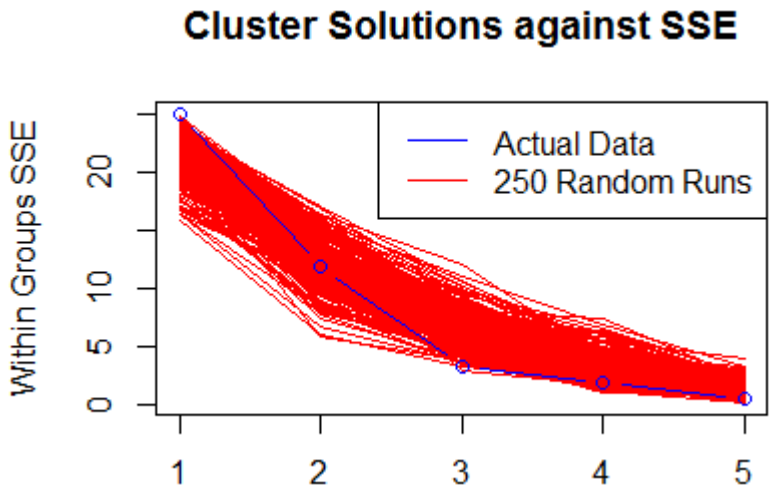


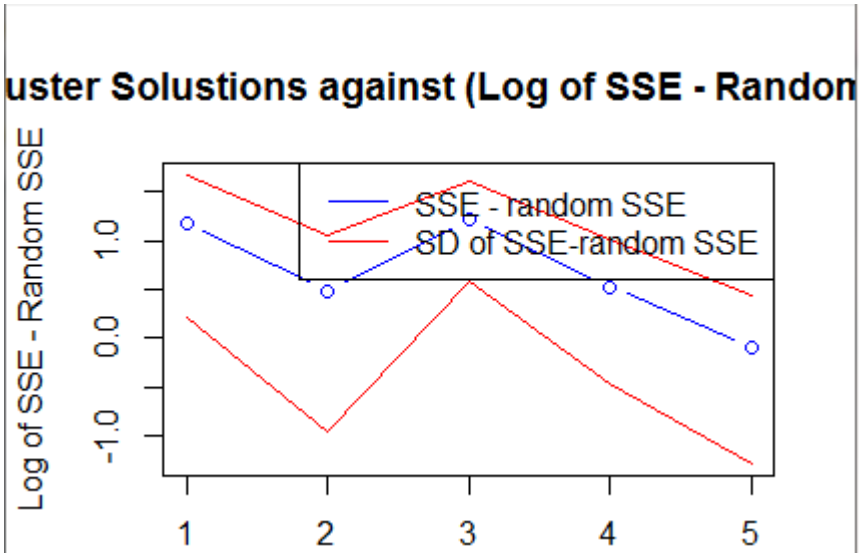
The Command window output is

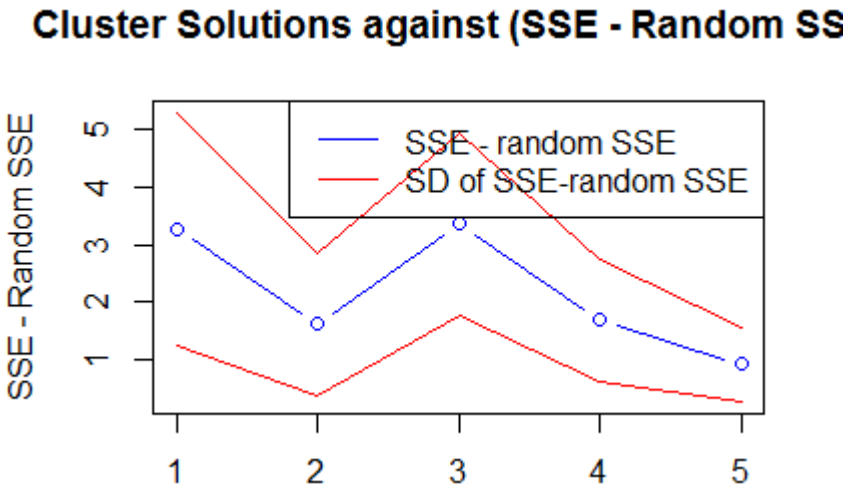


Note the pdf - this means that a PDF file has been written. On closing the Command window, the PDF is displayed. Scroll down to see the charts.

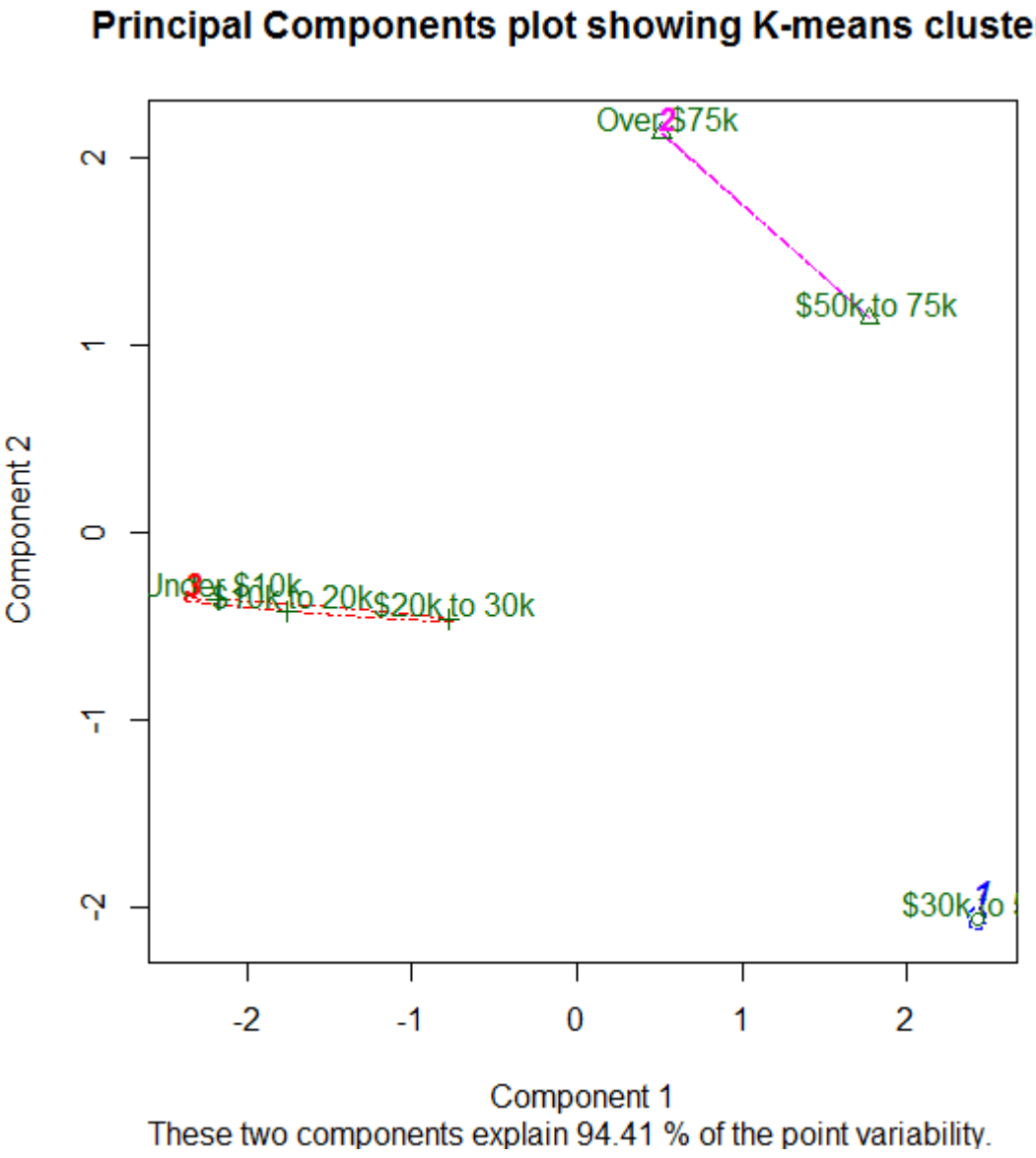








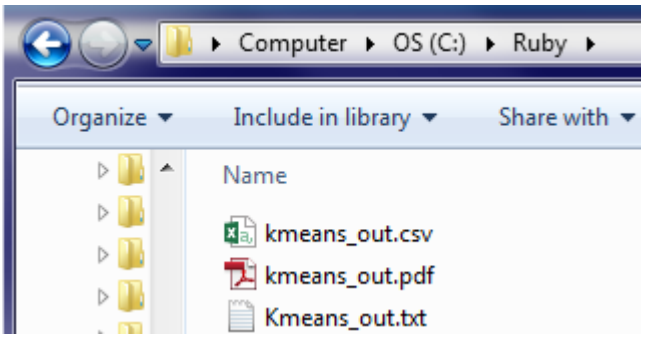
Finally, the cluster plot is shown:



This shows three clear clusters as $0-30k, $30-50k and over $50k.

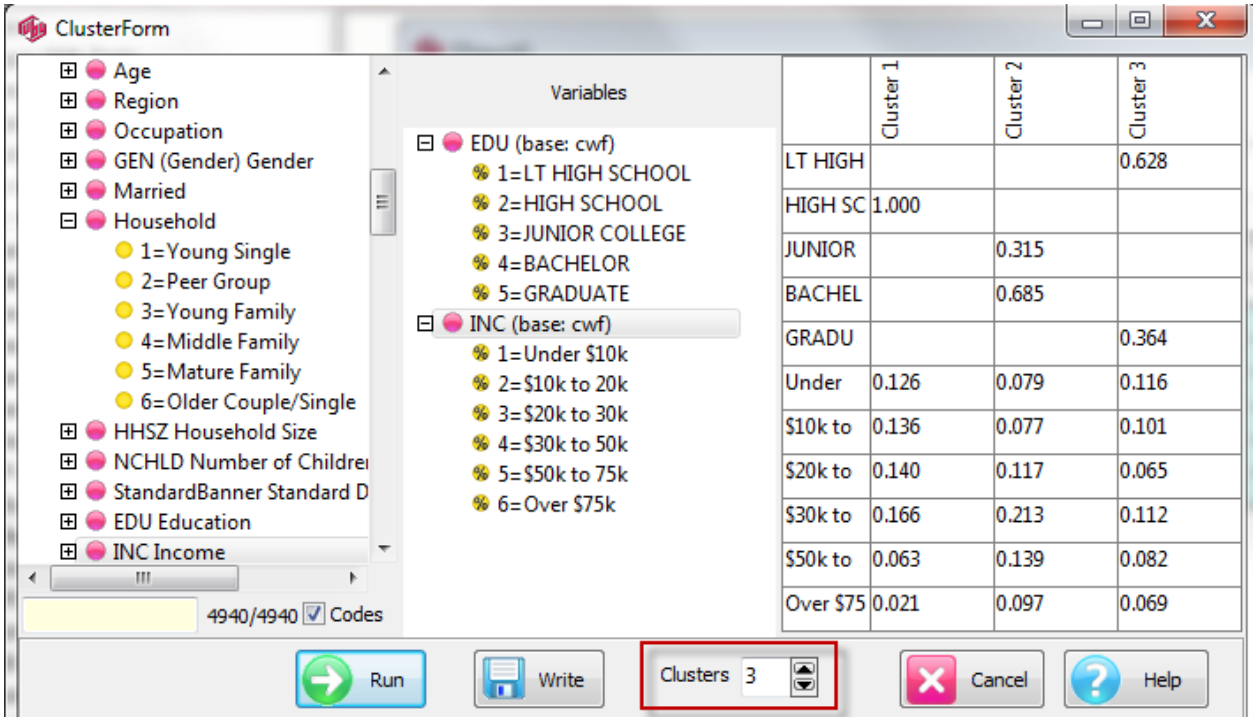
Diagnostic and Chart Outputs

Three files are written to the \Ruby subdirectory:



kmeans_out.csv is the source table, Kmeans_out.txt has diagnostic details on the clustering, and kmeans_out.pdf for the charts as above.

Comparing to Ruby Cluster



EDU(6) No Answer has been removed.

- Run and save as EDU_INC_Clust3

To check the clusters

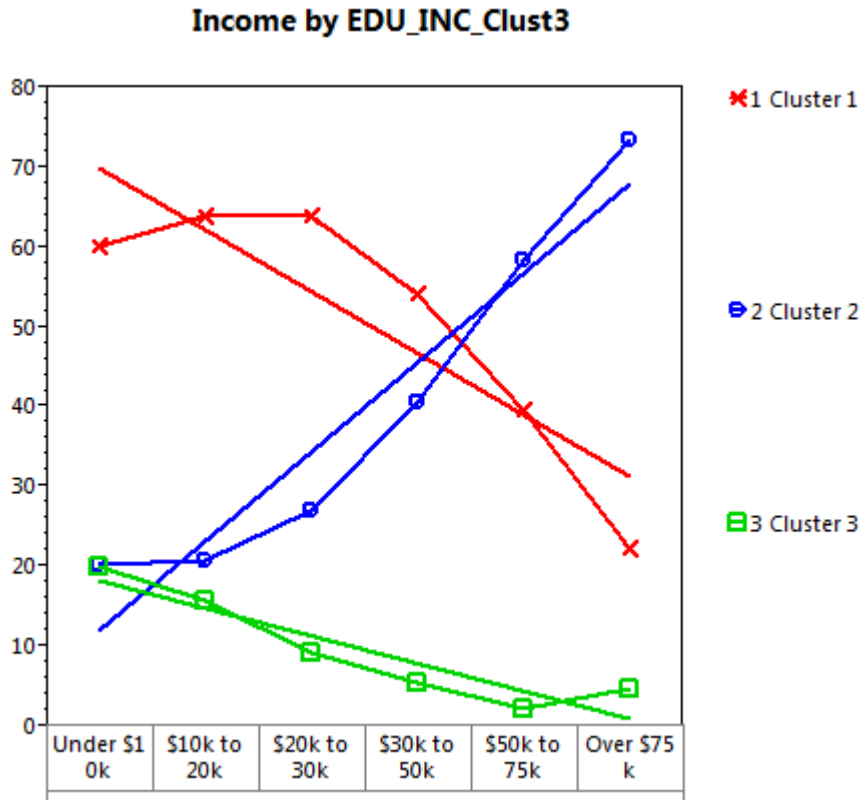
- Run the table INC, EDU by EDU_INC_Clust3

This shows that the three clusters are High School, College/Graduate and Less than High School. The highlighting on the left is an intuitive guess as to the income clusters.

Top: Income, Education
Side: EDU_INC_Clust3

Column Percents		Income						Education				
		Under $10k	$10k to 20k	$20k to 30k	$30k to 50k	$50k to 75k	Over $75k	LT HIGH SCH	HIGH SCHO	JUNIOR COL	BACHELOR	GRADUATE
EDU_IN	Cluster 1	60%	64%	64%	54%	40%	22%	0%	100%	0%	0%	0%
	Cluster 2	20%	21%	27%	41%	58%	73%	0%	0%	100%	100%	100%
	Cluster 3	20%	16%	9%	5%	2%	5%	100%	0%	0%	0%	0%

Notice how the Income rows are close to linear. A chart of Income by EDU_INC_Clust3 with trend lines is



So, there is some difference about where to place $30-50k between R and Ruby, but otherwise agreement on the cluster solution is reasonable.

Plot

This procedure simply demonstrates various plotting techniques. The R script is

Plot.r - Notepad

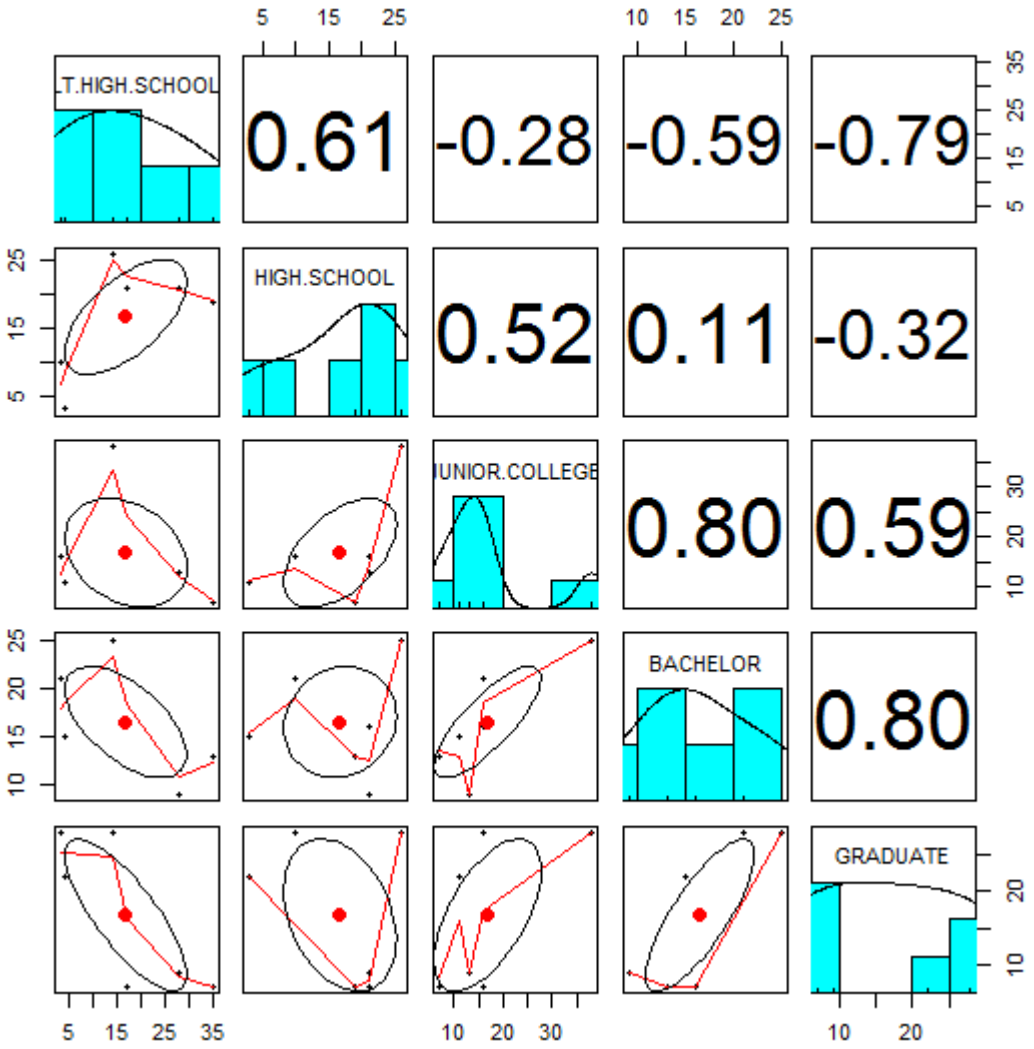
File Edit Format View Help

```
# Plot (chart) the table

args <- commandArgs(trailingOnly=T)
print(args)
title = args[1]
xlabel = args[2]
ylabel = args[3]

x <- read.table('clipboard', header = TRUE, row.names = 1, sep = '\t', dec = '.')
win.graph()
# uncomment the required plot method, or add new ones
library(psych); pairs.panels(x)
#boxplot(x)
#plot(x)
#library(lattice); plot.new(); xyplot(ts(x))
#matplot(as.matrix(x), type = "b")
#barplot(as.matrix(x), main=title, xlab=xlabel, ylab=ylabel, col=c("blue","red","gre
locator(1)
```

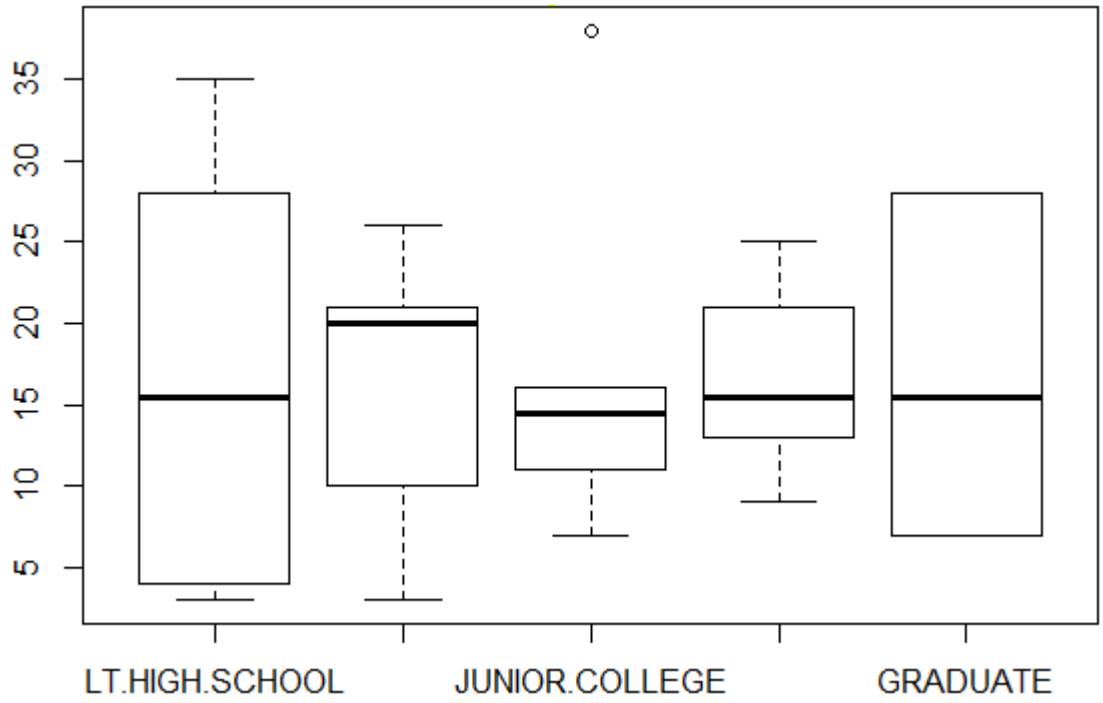
Only one of the plot methods can be uncommented.

The plot types are

pairs.panels:



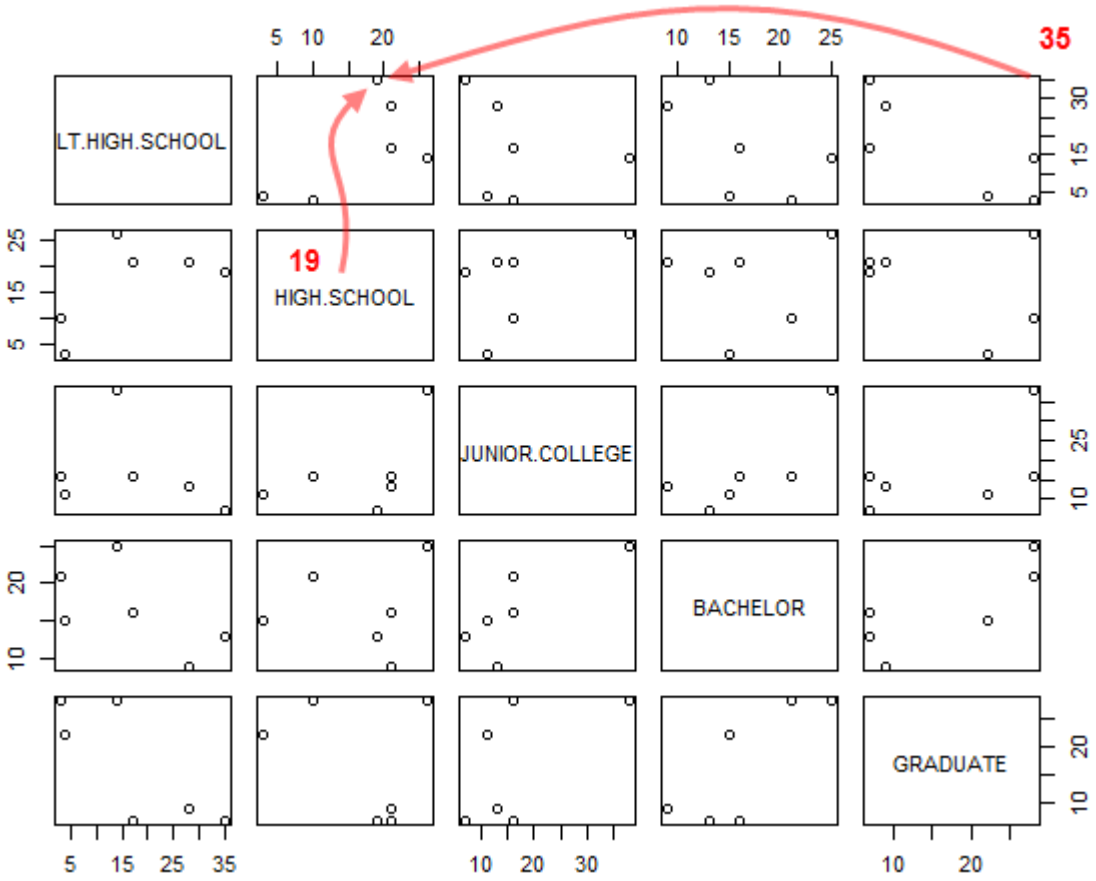
boxplot:



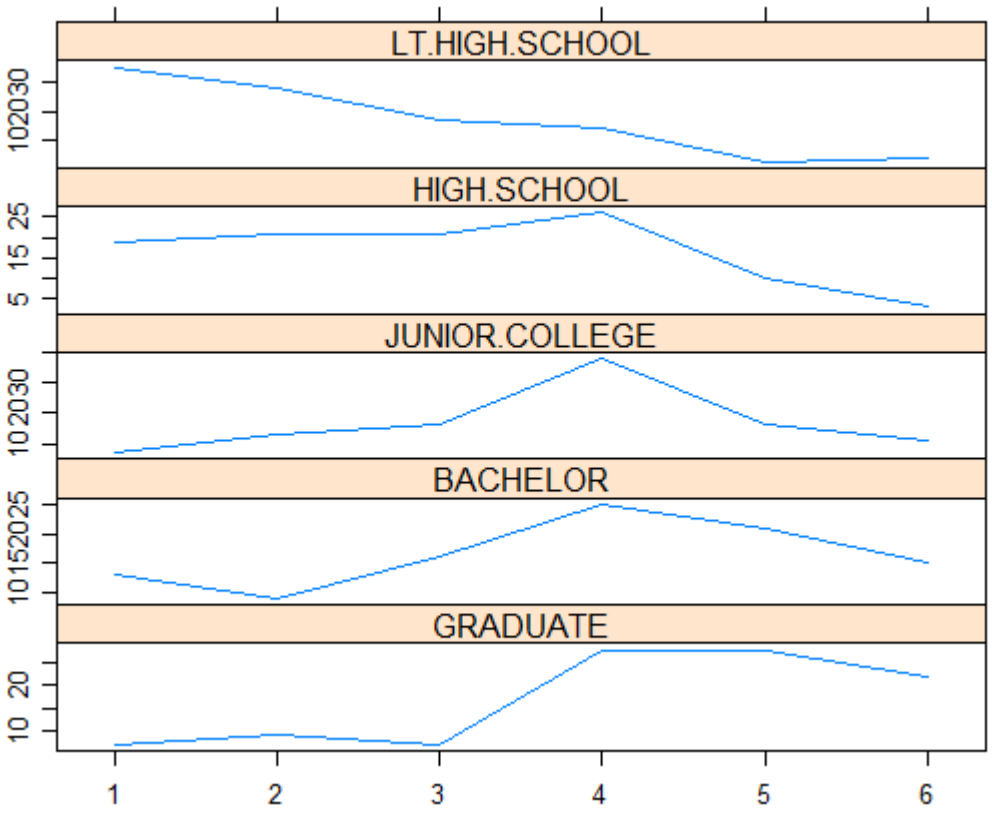
plot:

The annotated point is LT High School=35%, High School=19%

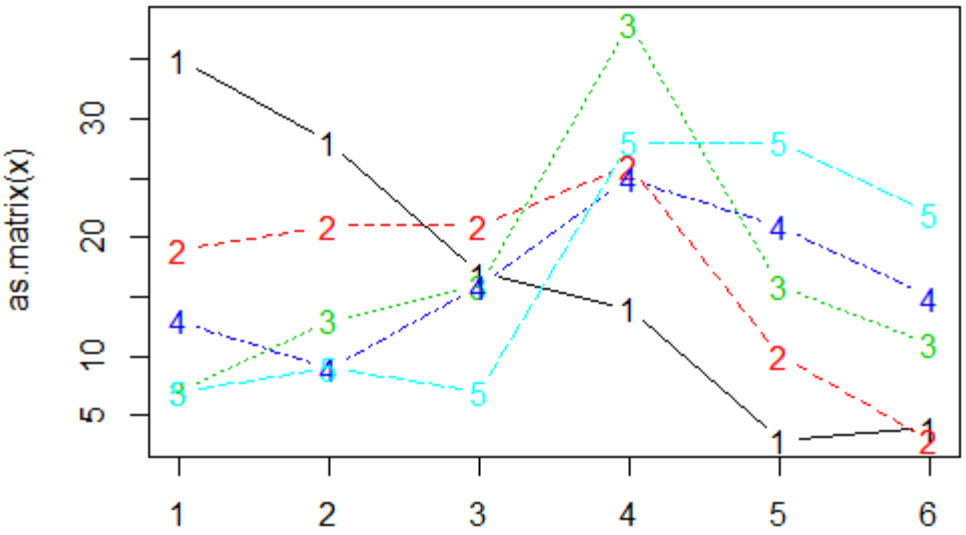
	LT HIGH SCHOOL	HIGH SCHOOL
Under $10k	35%	19%
$10k to 20k	28%	21%


LT.HIGH.SCHOOL
HIGH.SCHOOL
JUNIOR.COLLEGE
BACHELOR
GRADUATE
35
19

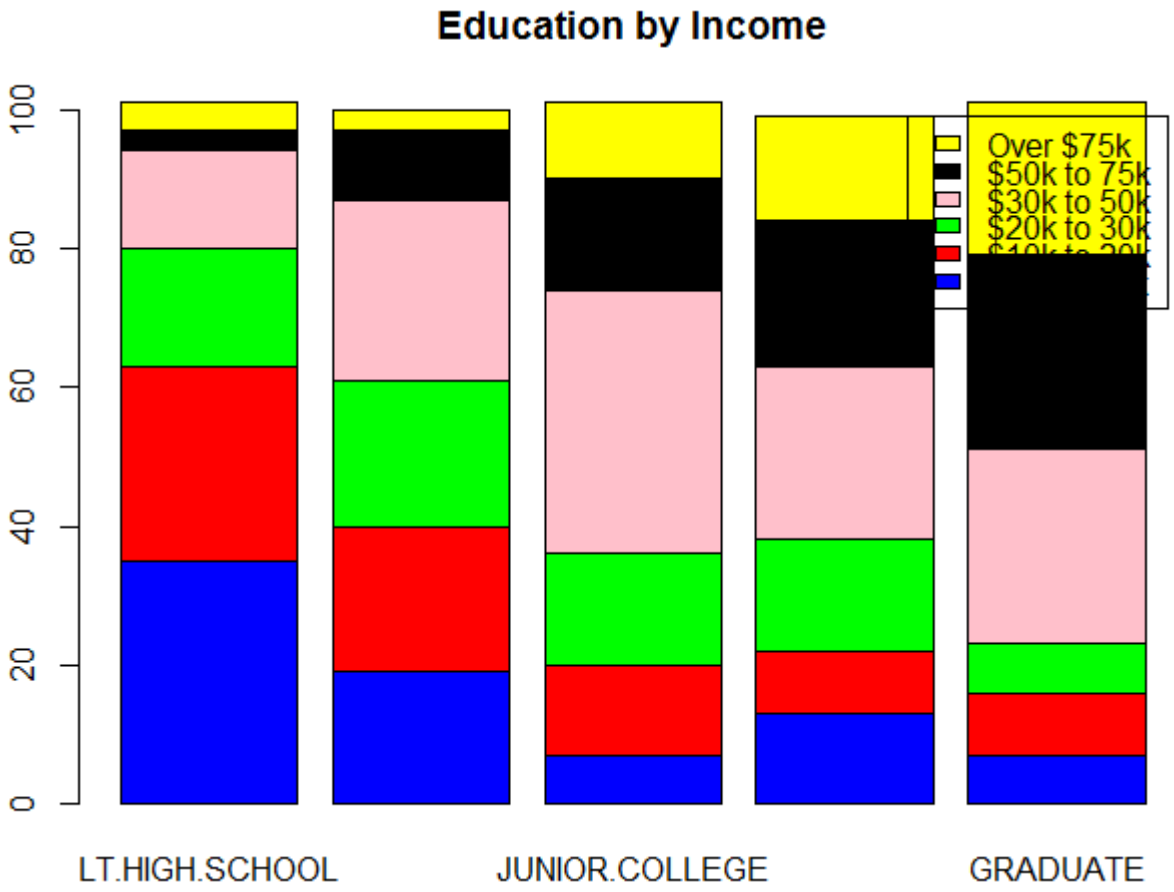
xyplot:


LT.HIGH.SCHOOL
HIGH.SCHOOL
JUNIOR.COLLEGE
BACHELOR
GRADUATE

matplot:



barplot:

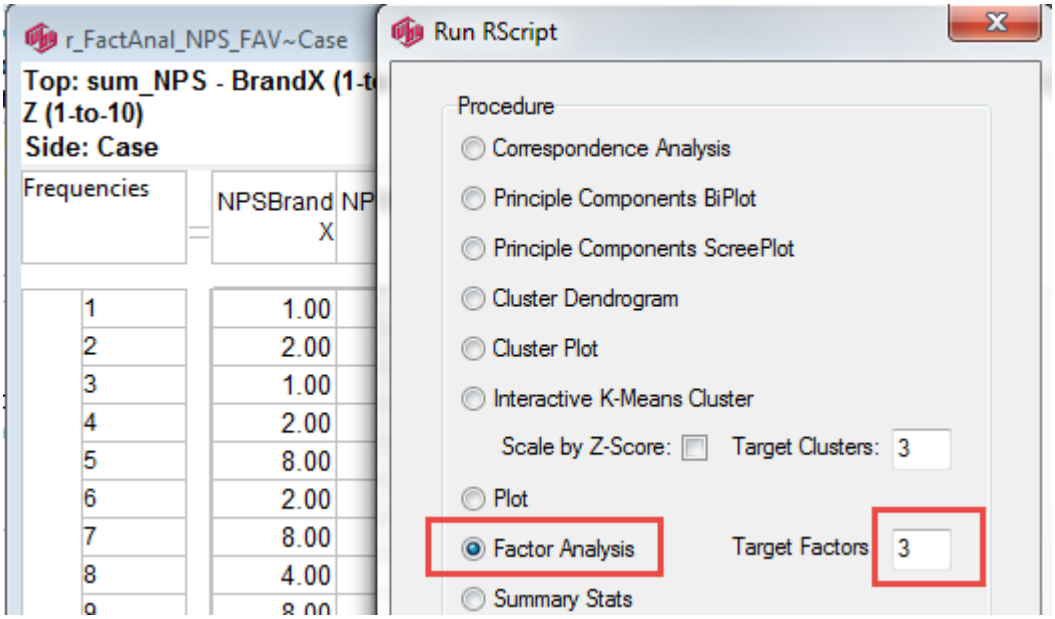


Factor Analysis

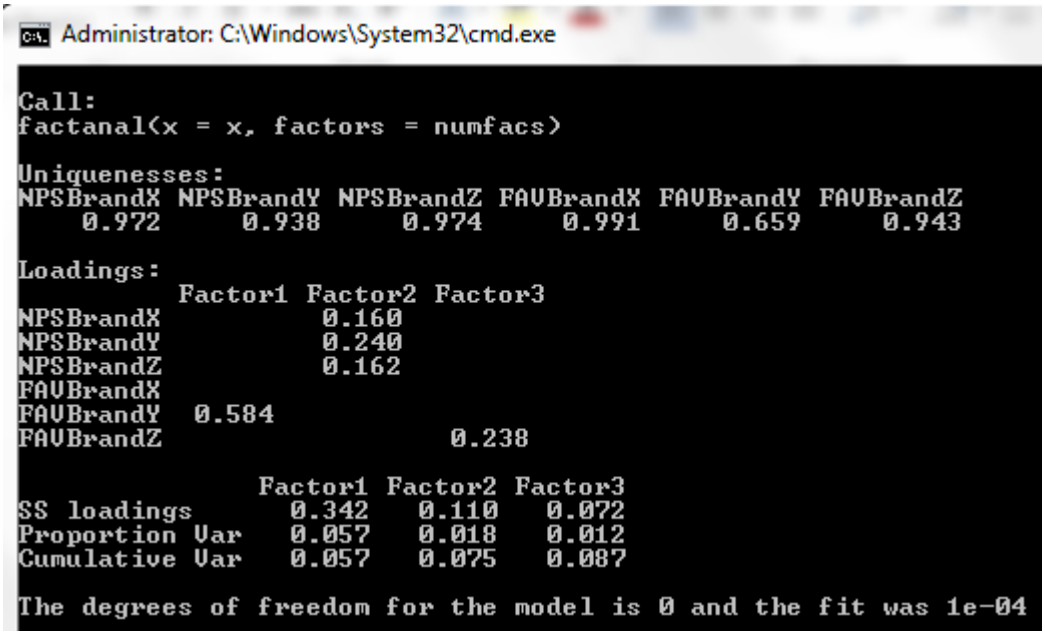
For factor analysis the rows would usually be cases, so a different report is used.

- Open the table r_FactAnal_NPS_FAV~Case

- Run as

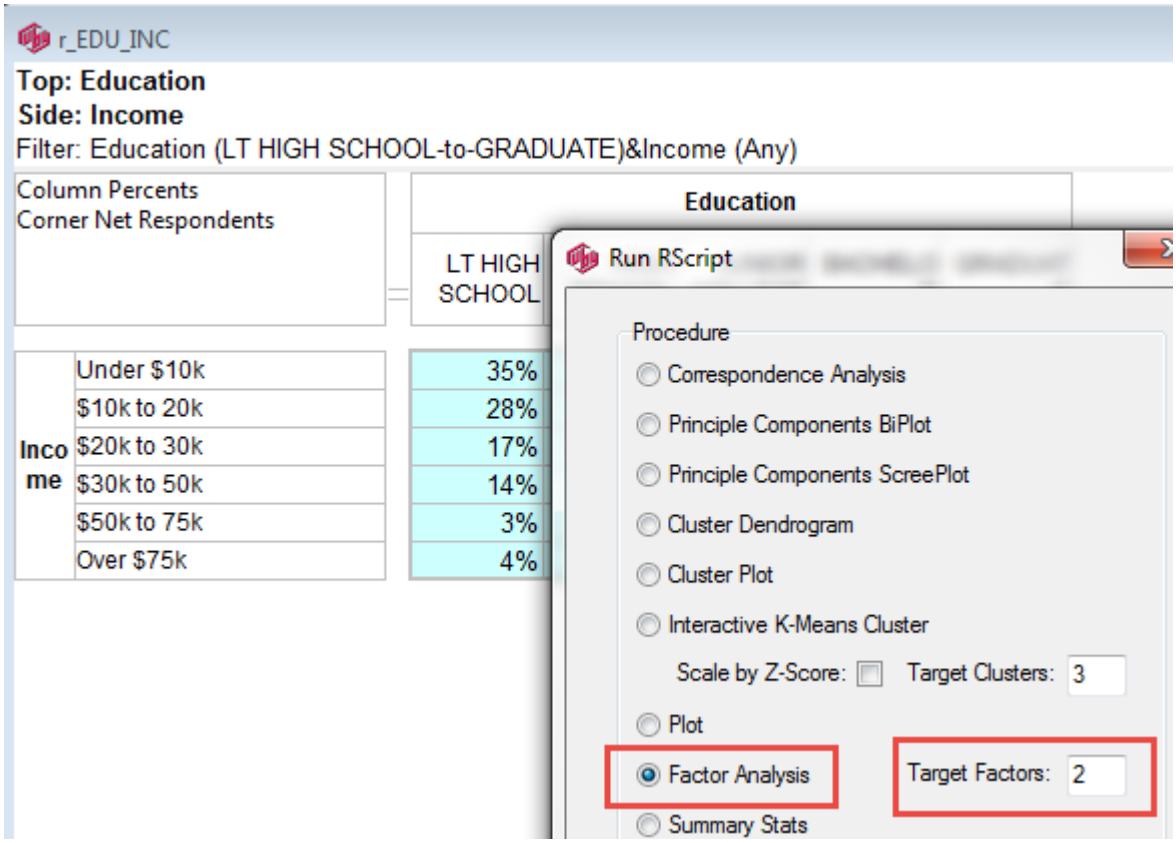


The output is

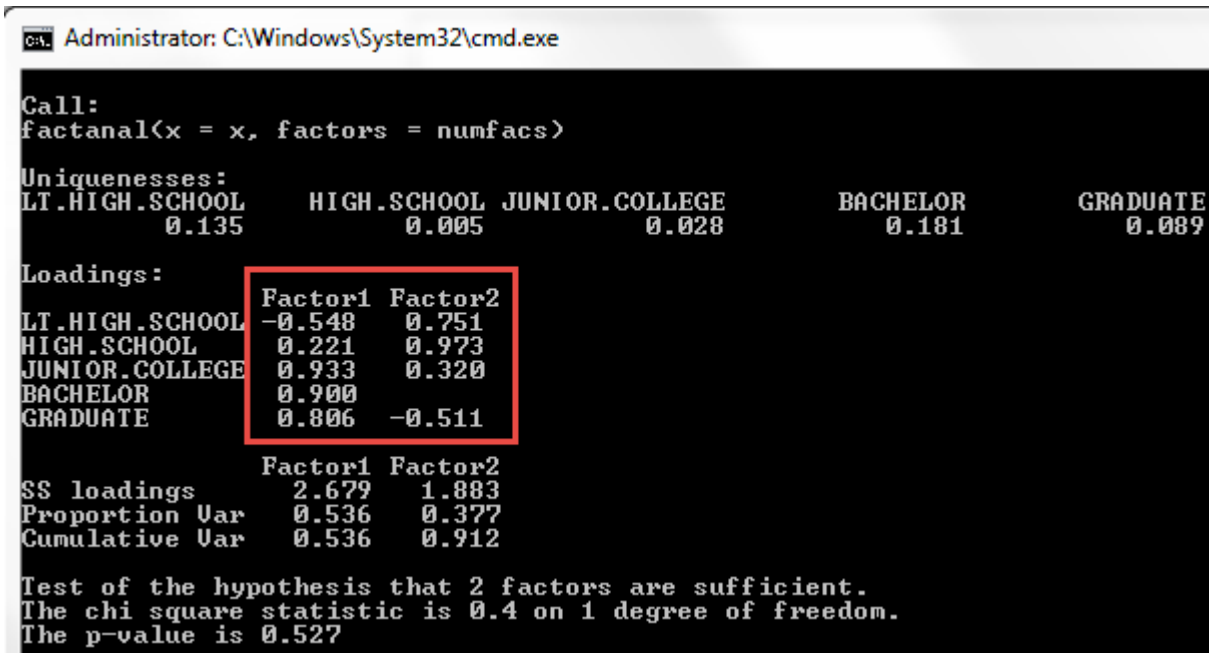


Since factor analysis takes each row as a case, you could use it on r_EDU_INC by considering a 'case' as a circumstance, ie the income range.

- On the report r_EDU_INC, run as



Note target factors = 2. The output is



The loadings show a strong diagonal structure.

Summary Stats

The script is

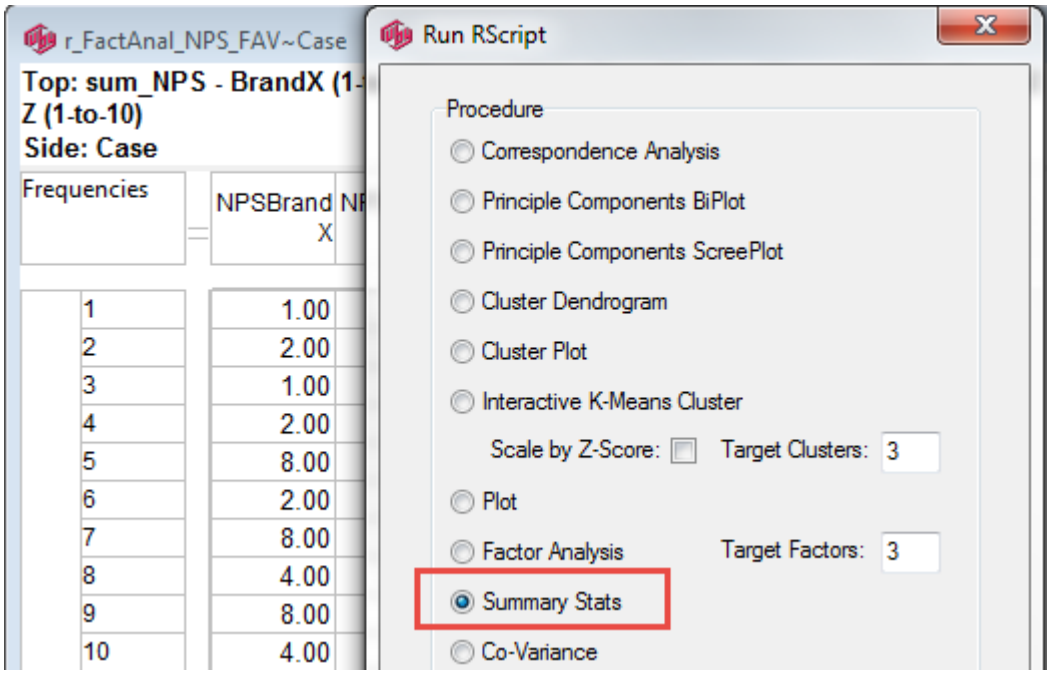
SummaryStats.r - Notepad

File Edit Format View Help

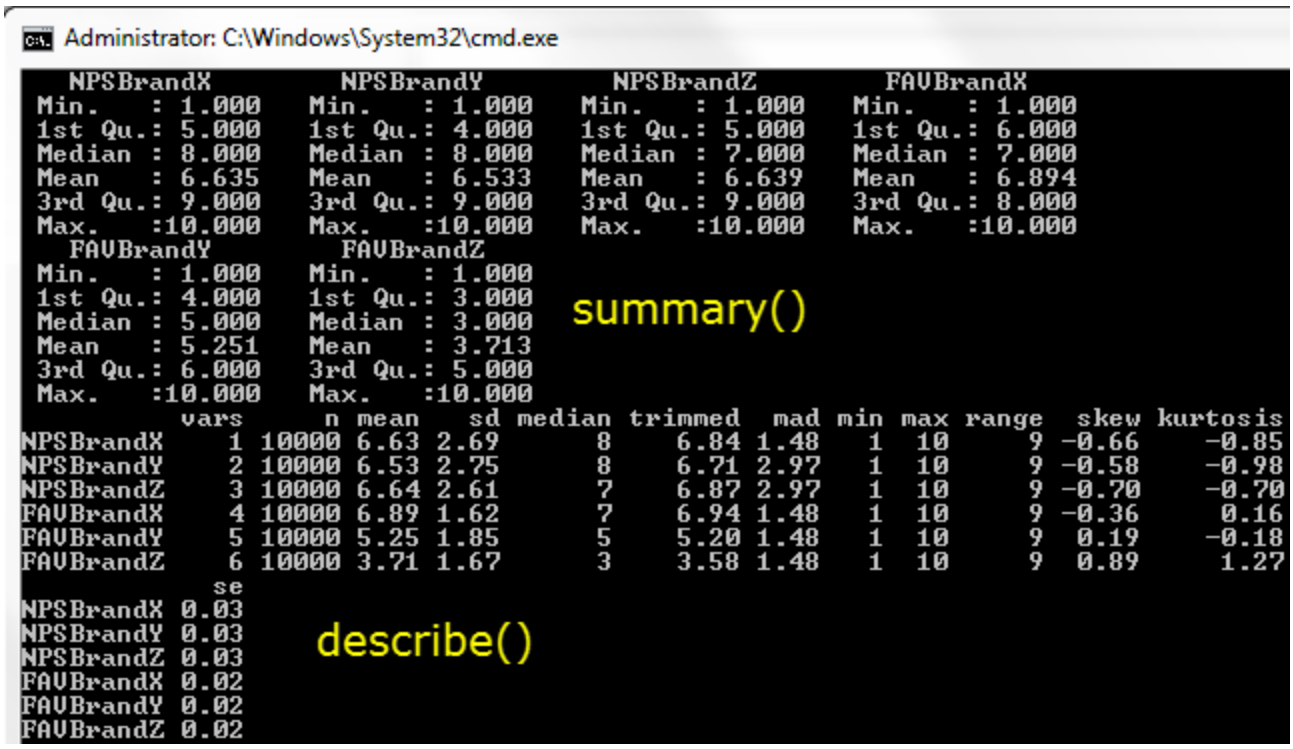
```
# Summary Stats per column

x <- read.table('clipboard', header = TRUE, row.names = 1, sep = '\t', dec = '.')
summary(x)
library(psych); describe(x)
```

The summaries are by column, so this procedure will work on any table.

Using the factor analysis table above, run as



The output is in two parts:



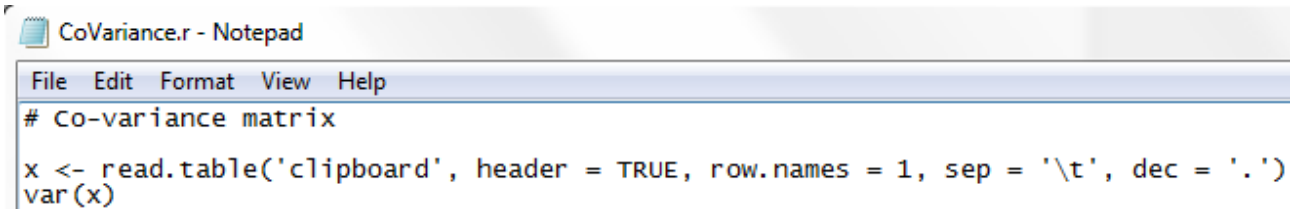
summary() gives 1st and 3rd quartiles.

describe() does not deliver quartiles, but gives the values for number of cases (rows), standard deviation, trimmed, mad, skew, kurtosis and standard error.

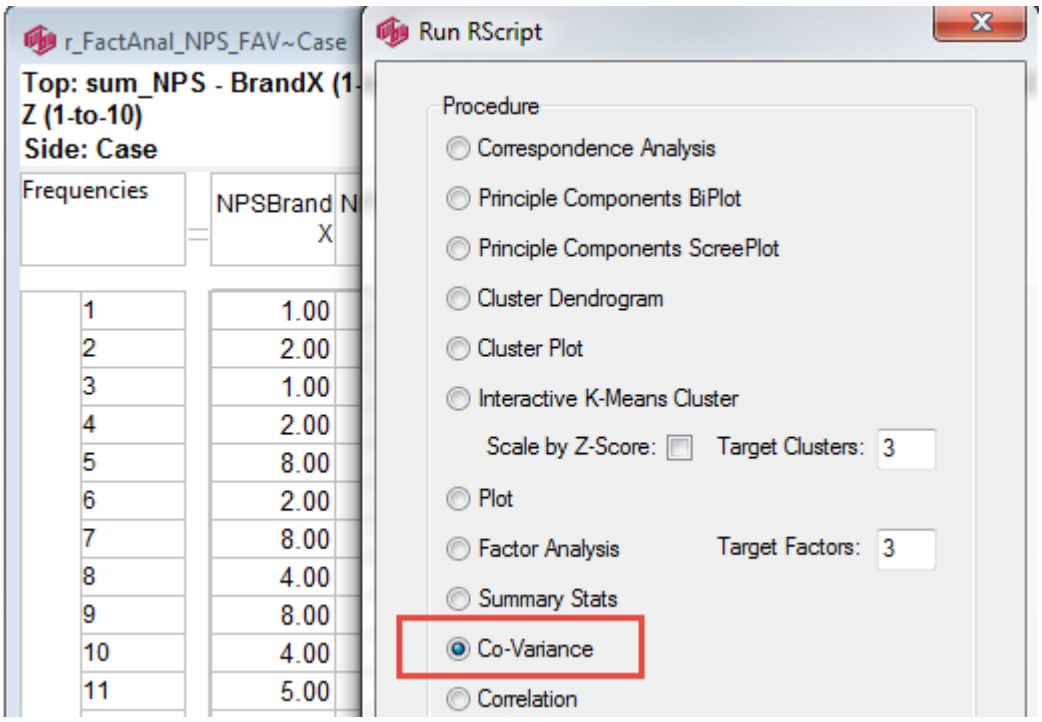
Both methods give min, max, median and mean.

Covariance

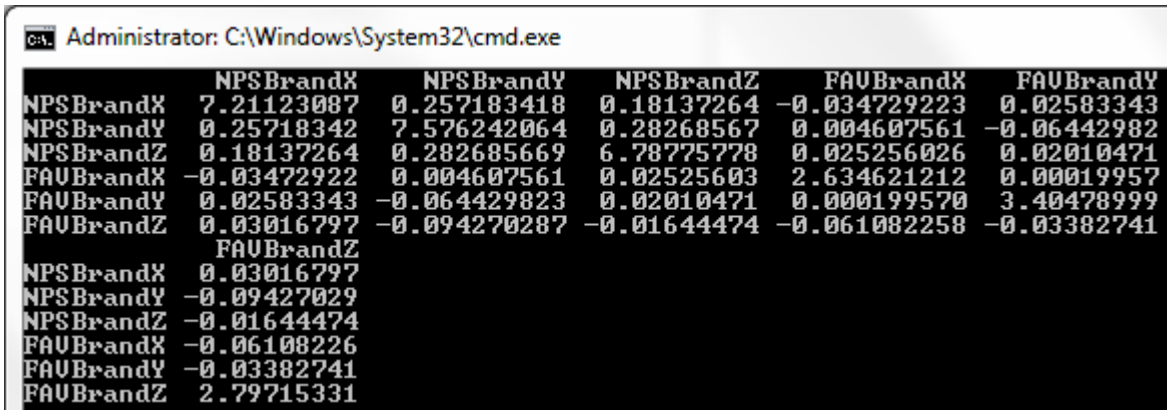
The script is



Call on the FactAnal table, as

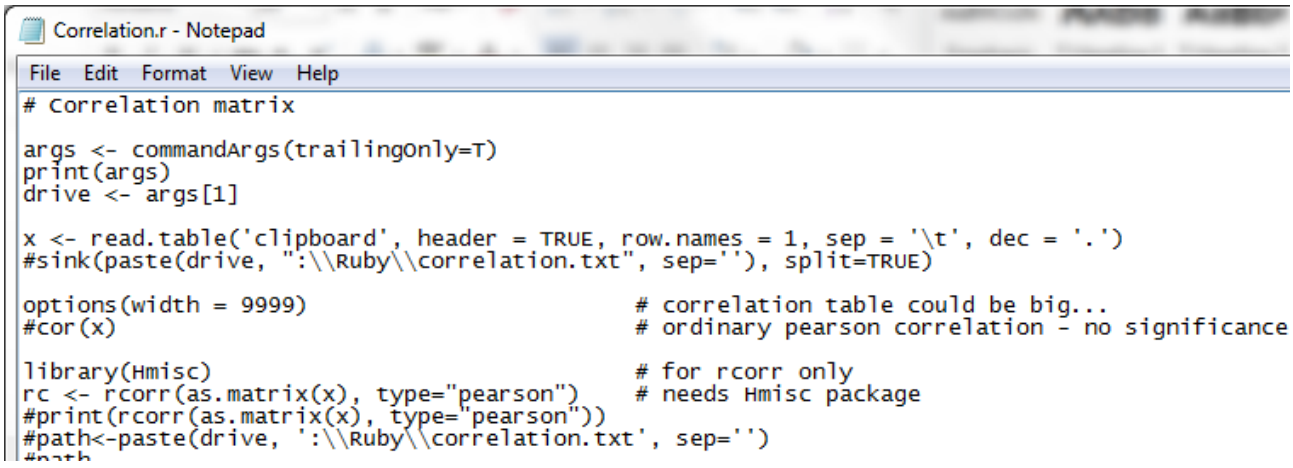


The output is



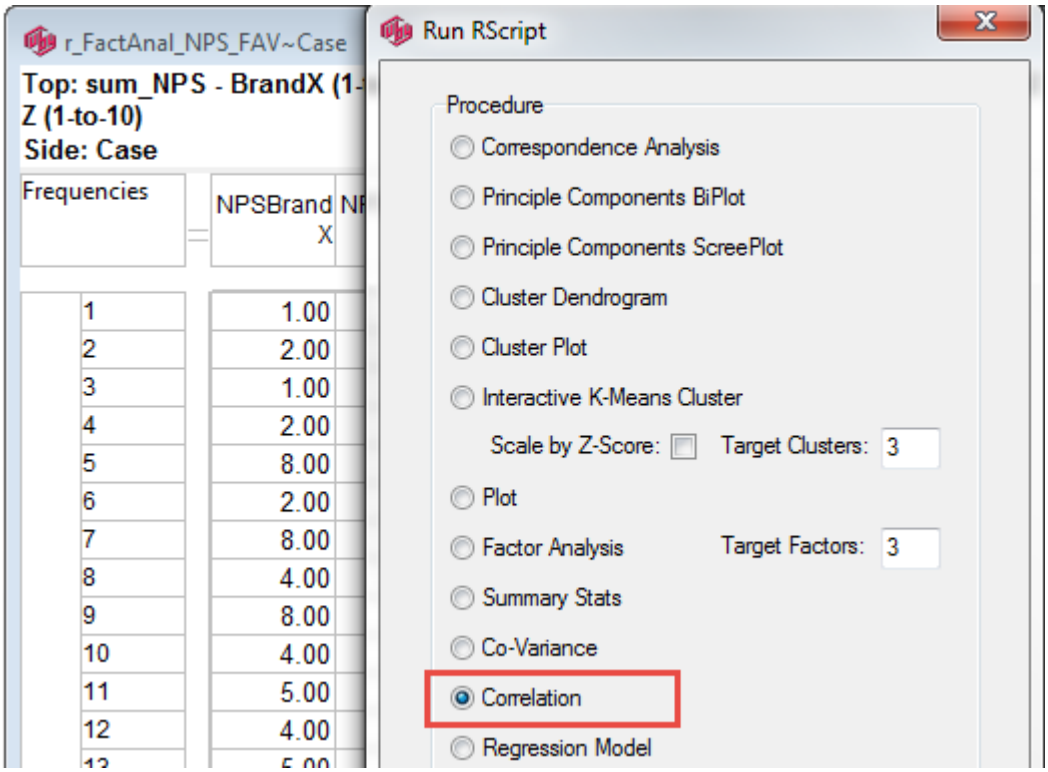
Correlation

The top part of the R script is

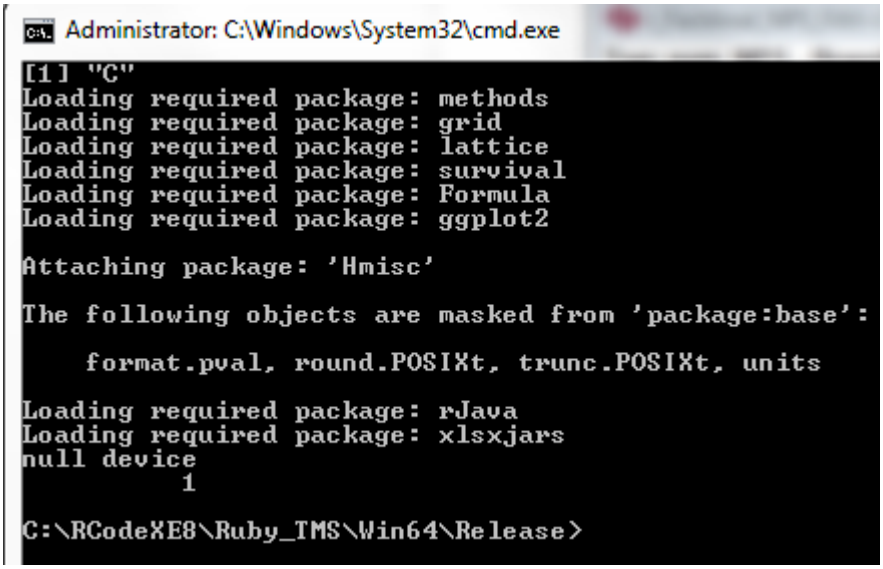


It is a bit too long to show all of it here. This script shows several techniques: sinkiing the output to a text file (this is commented out), how to call a simple Pearson correlation (commented out), how to call correlation with significance (active), and how to send output to Excel and a PDF.

Call on the FactAnal table, as



The runtime outputs are

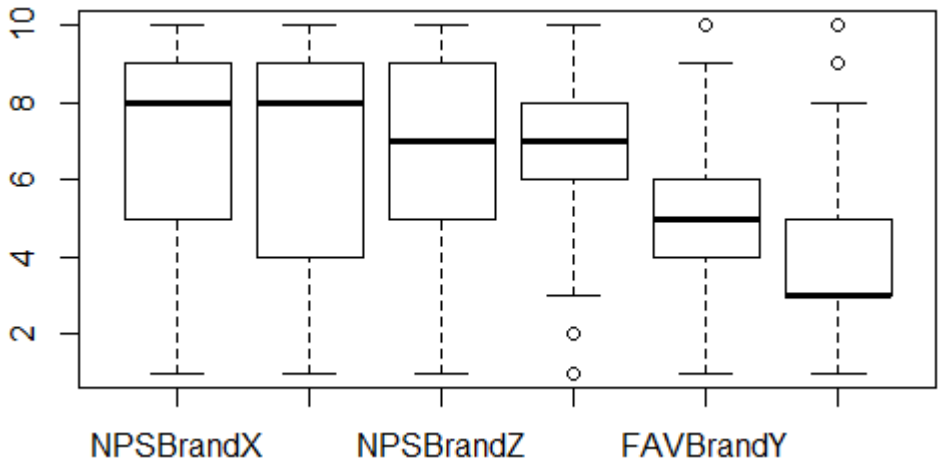


On closing the Command winbdow, the three correlation tables are displayed in Excel:

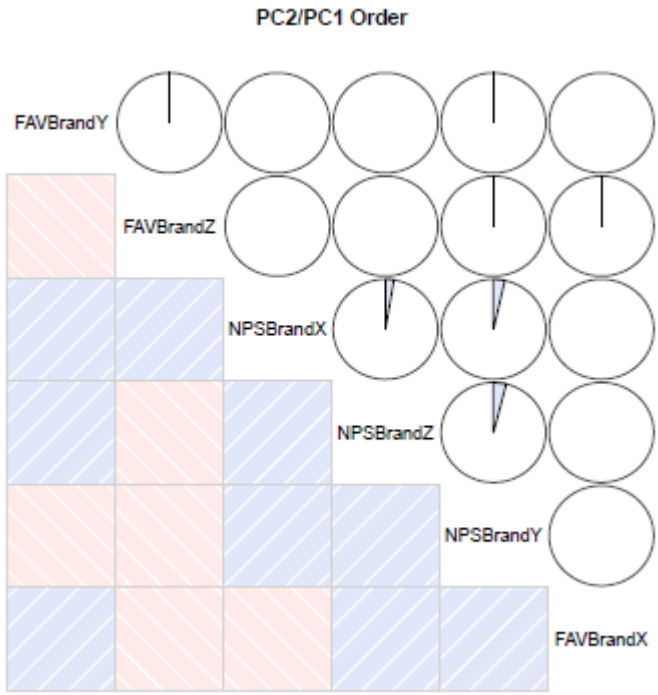
	A	B	C	D	E	F	G
1		NPSBrandX	NPSBrandY	NPSBrandZ	FAVBrandX	FAVBrandY	FAVBrandZ
2	NPSBrandX	1	0.035	0.026	-0.008	0.005	0.007
3	NPSBrandY	0.035	1	0.039	0.001	-0.013	-0.02
4	NPSBrandZ	0.026	0.039	1	0.006	0.004	-0.004
5	FAVBrandX	-0.008	0.001	0.006	1	0	-0.023
6	FAVBrandY	0.005	-0.013	0.004	0	1	-0.011
7	FAVBrandZ	0.007	-0.02	-0.004	-0.023	-0.011	1
8							

Correlation | Frequencies | Probabilities

And a series of visualisations are displayed in a PDF:



The box plot shows min, max, median and the 25% and 75% percentiles. Various corrgrams follow. The first is



You would not want all these - coment out unrequired.

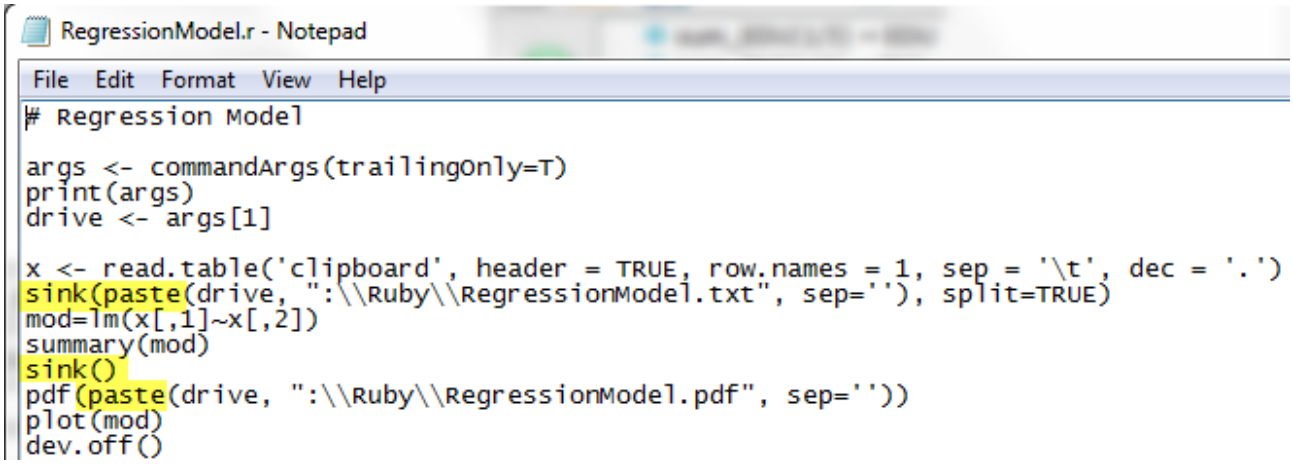
Regression Model

For a regression model, we need a table with an independent variable on the left, and the dependent variable(s) on the right.

- Open the supplied table r_RegMod_EDU_INC~Case

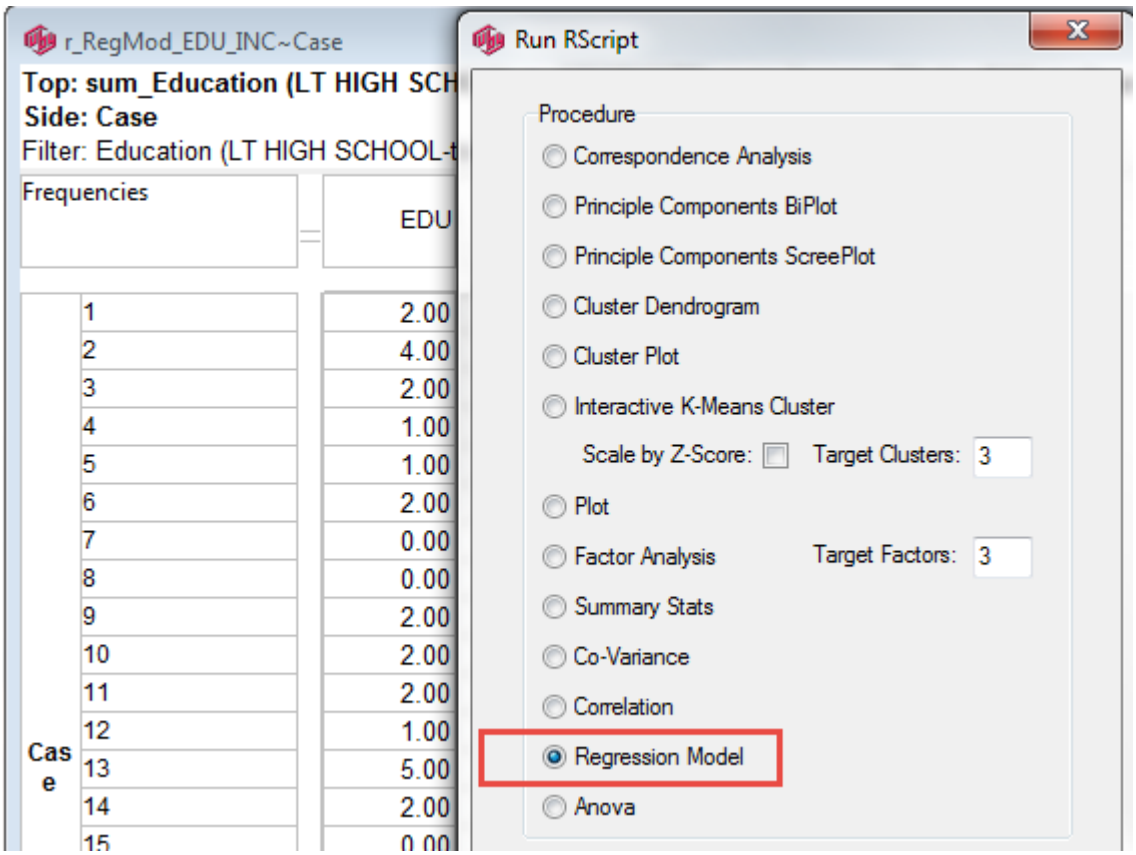
The idea is that education drives income, so the first column is EDU.

The R script is

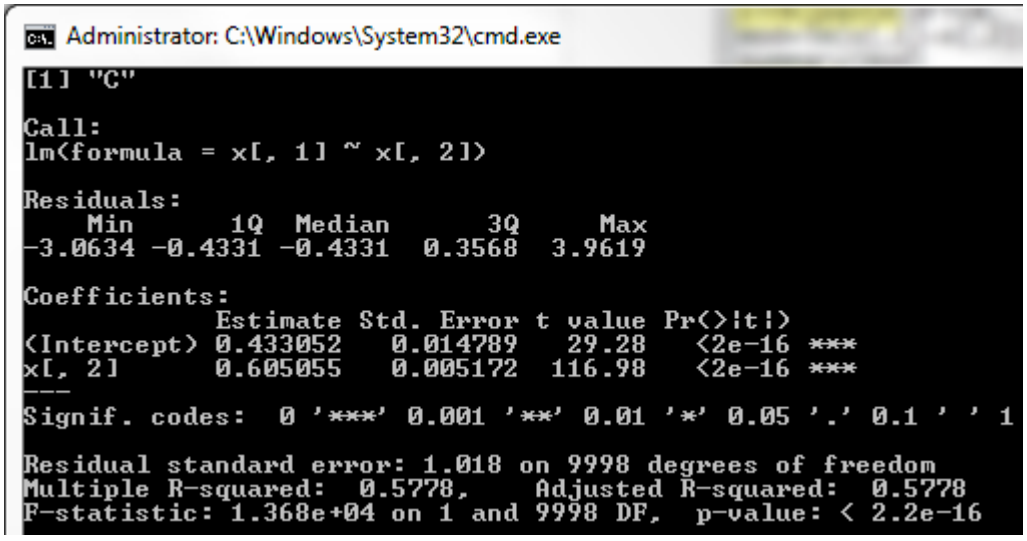


Note the use of sink() to 'sink' the outputs to file, and the use of paste() to concatenate strings. The drive letter is passed as an argument.

Run as



The Command Window output is



The sunk file comprises the above.

On closing the window, a PDF of various diagnostic charts appears.

Anova

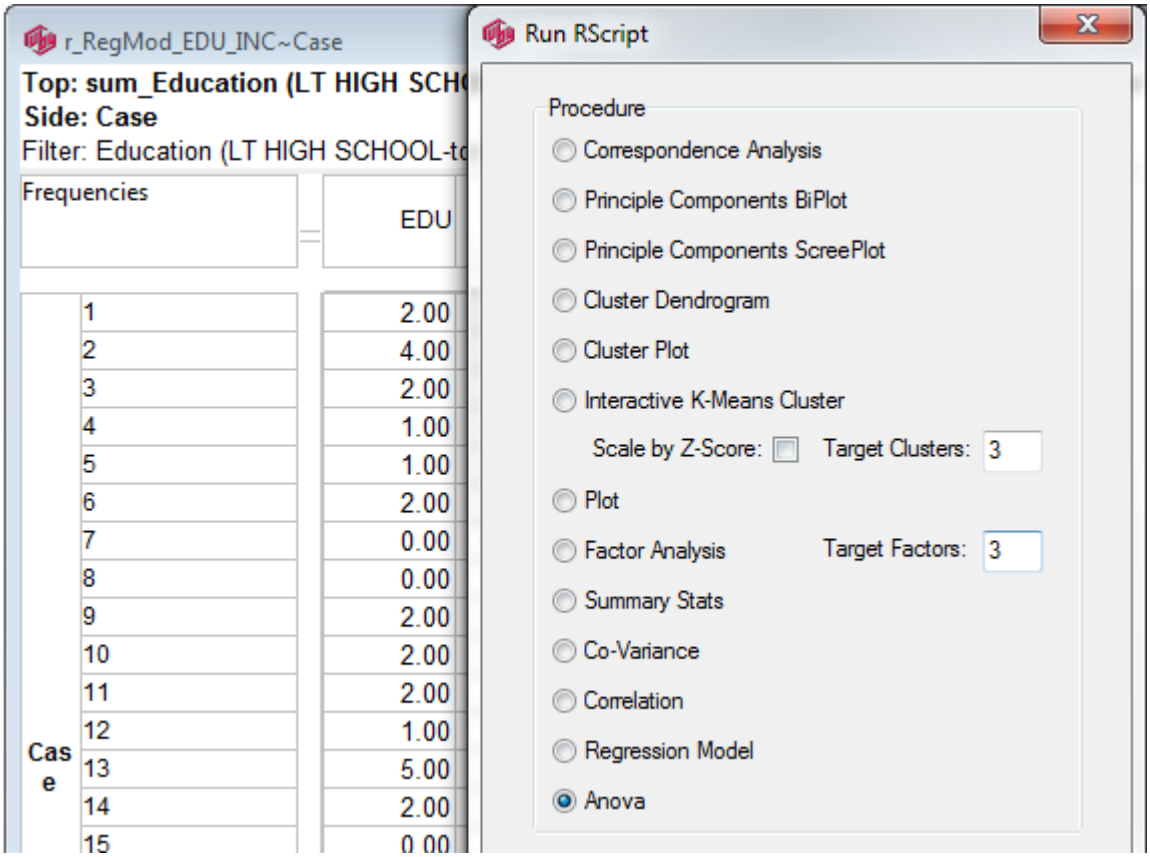
The R script is

Anova.r - Notepad

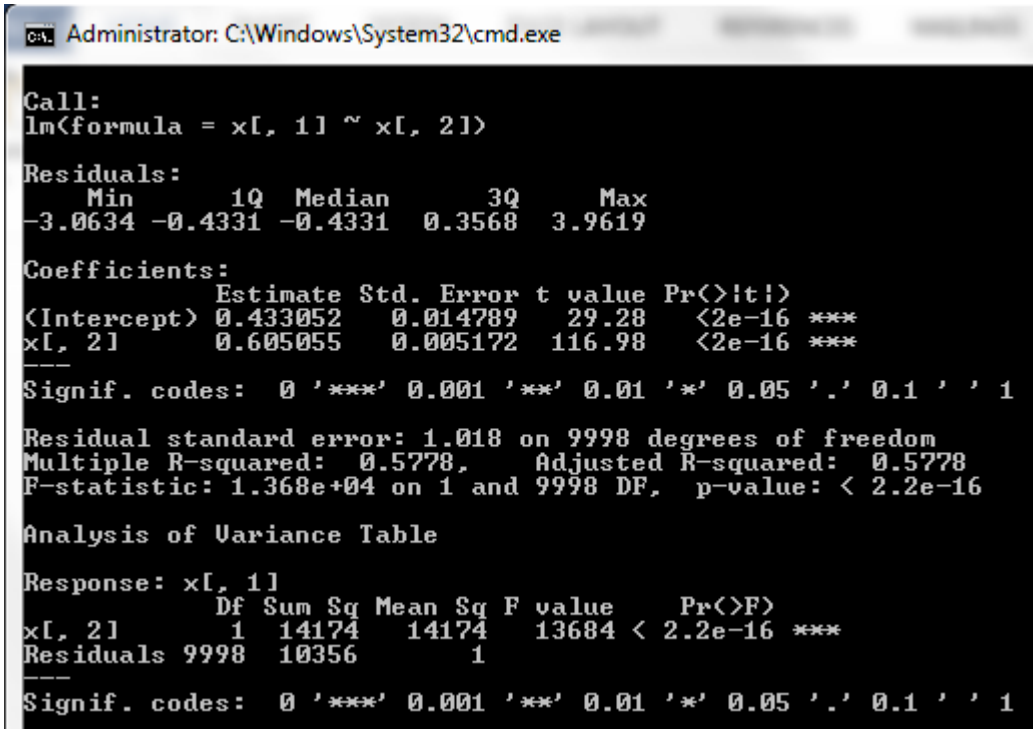
File Edit Format View Help

```
# Anova

x <- read.table('clipboard', header = TRUE, row.names = 1, sep = '\t', dec = '.')
mod=lm(x[,1]~x[,2])
summary(mod)
anova(mod)
```

Since the parameter for anova() is a linear model, run on the same table as above, as



The output is


Administrator: C:\Windows\System32\cmd.exe
Call:
lm(formula = x[, 1] ~ x[, 2])
Residuals:
Min 1Q Median 3Q Max
-3.0634 -0.4331 -0.4331 0.3568 3.9619
Coefficients:
Estimate Std. Error t value Pr(>|t|)
(Intercept) 0.433052 0.014789 29.28 <2e-16 ***
x[, 2] 0.605055 0.005172 116.98 <2e-16 ***

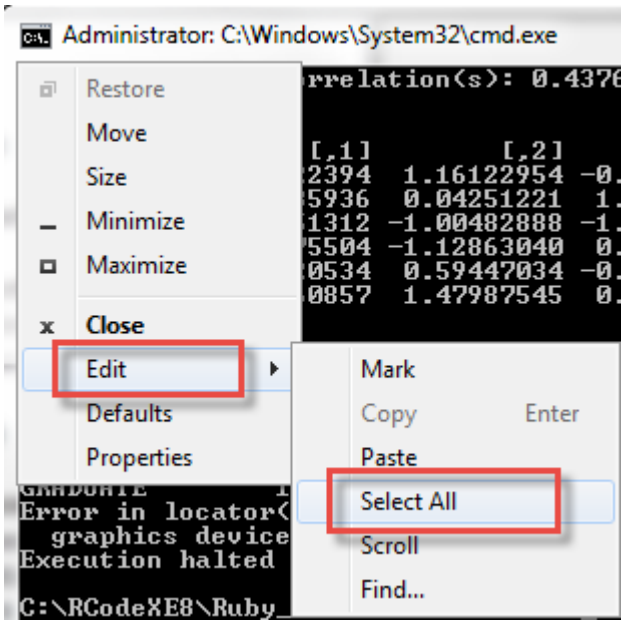
Signif. codes: 0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
Residual standard error: 1.018 on 9998 degrees of freedom
Multiple R-squared: 0.5778, Adjusted R-squared: 0.5778
F-statistic: 1.368e+04 on 1 and 9998 DF, p-value: < 2.2e-16
Analysis of Variance Table
Response: x[, 1]
Df Sum Sq Mean Sq F value Pr(>F)
x[, 2] 1 14174 14174 13684 < 2.2e-16 ***
Residuals 9998 10356 1

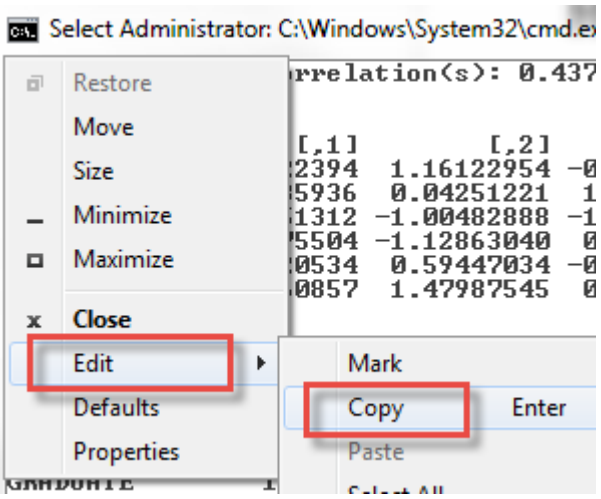
Signif. codes: 0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

HANDY HINTS

If you have many windows open, occasionally the Run RScript form can end up behind a window. Bringing Ruby to the front should restore visibility.

The Command window content can be selected and copied from here:





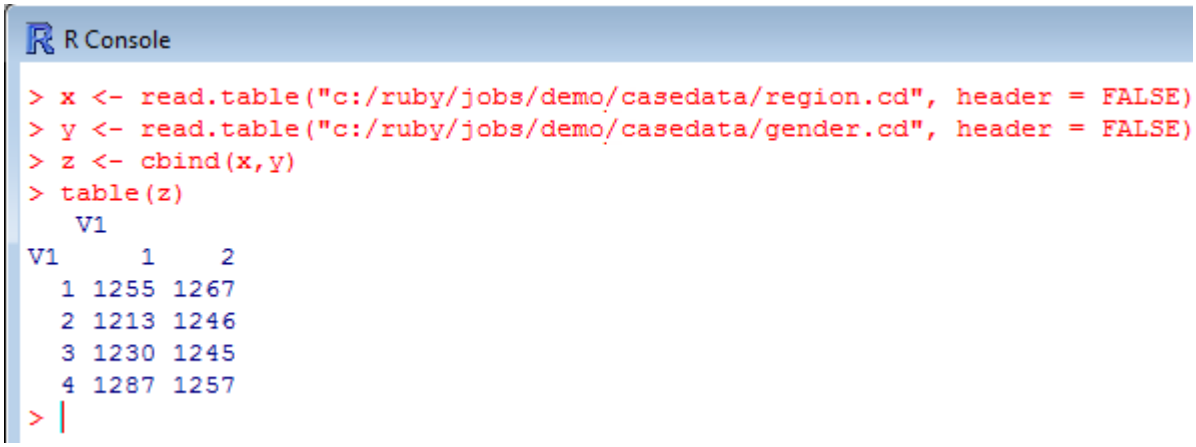
If editing and testing or interacting with an R procedure script (for example, commenting in/out the various plot types), then you can use any editor you like, but remember to save the file before starting Run_RScripts.vbs.

For procedures which write PDFs or other files, make sure you close the output files before running again, or else you will get Windows errors. Save anything you want to keep under your own name.

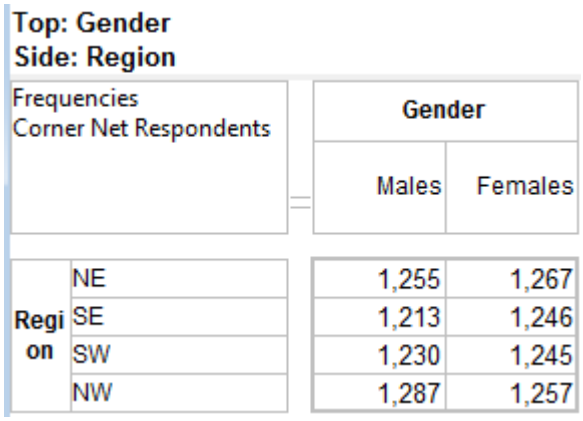
MISCELLANEOUS

You can read Ruby variable case data directly into R like this:

```
x <- read.table("c:/ruby/jobs/demo/casedata/region.cd", header = FALSE)
y <- read.table("c:/ruby/jobs/demo/casedata/gender.cd", header = FALSE)
z <- cbind(x,y)
table(z)
```



The Ruby equivalent is



Uninstall statCONN (if you have it)

The first release of this document relied on statCONN, a third party COM interface to R. Unfortunately, statCONN is now licence-limited, with free usage for non-commercial purposes only.

There were also other problems with the statCONN approach: cumbersome, a difficult installation procedure, slow, hard to edit the COM calls, and so on.

These various issues have been addressed by redesigning the Ruby-R interface to call Rscript.exe directly. The only disadvantage (compared to statCONN) is that the path to Rscript.exe needs to be known up front, and entered manually to the master VBS script.

This document replaces all previous versions.

[end of document]