R20—Exploratory Factor analysis and principal component analysis in R

Colleen F. Moore Feb 2015 cfmoore@wisc.edu
Prof Emerita, University of Wisconsin—Madison
Affiliate Professor, Montana State University, Bozeman

In R there are several ways to do exploratory factor and principal components analysis.
Best reference, and developer of the 'psych' package: William Revelle, see links inside R in documentation on the 'psych' package. Ch 6 of his forthcoming book is highly recommended.
Also very good, Michael Friendly's page, not specific to R: http://www.psych.yorku.ca/lab/psy6140/fa/facplan.htm
My handout here is not intended to be a lecture handout, but a relatively quick reference for 'how to' in R.

Contents of this handout:


```
> library(psych)  ## bring the psych package into R memory, for a lot of what is done below
```

I. Preliminaries (and how to do them) before diving into principal components or factor analysis

A. Test to see if your correlation matrix differs significantly from the identity matrix. You don't want to be fitting just error. See section IV.A.1. below.
B. Do you have a reasonable set of measures, or do some items not belong in this analysis? Find the squared multiple correlations (smc) of each variable with the others. Inspect for low values, read the items that have low smc values, and decide whether to remove them. See section IV.A.2 below. If you are constructing a new scale, you will want to remove items after fitting a model also.
C. Look for outliers using Mahalanobis distances (D2):

```
> outlier(asiq, plot=T, bad=10, na.rm=T)  # in psych package
```

In a large sample, ask it to flag more bad values than in a smaller sample. Also, Mahalanobis distances are supposed to be distributed as a chi-squared distribution, with df = number of variables going into the distance calculation. Can get some idea about how far out of your distribution the outliers are by looking at the p-values of the chisq distribution. For my 8 variable example below as follows:

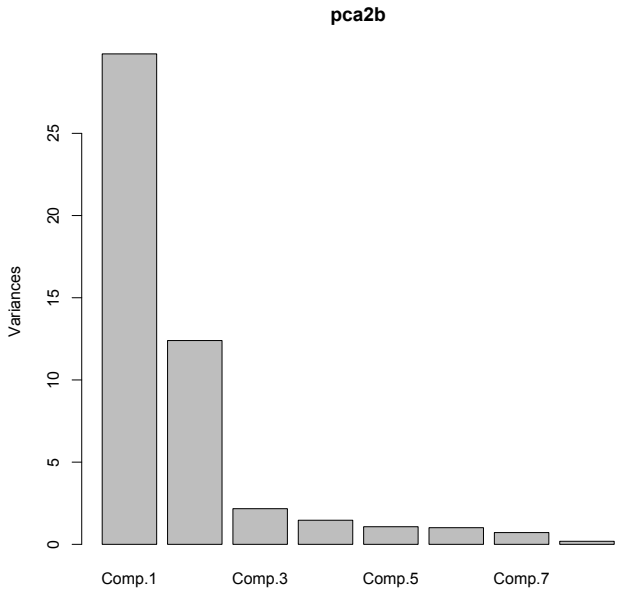
```
> qchisq(.01, 8, lower.tail=F)  # p=.01, df=8, we want the upper tail
[1] 20.09024
## this says that if an outlier has a distance over 20, it is in the upper 1% of distribution.
> pchisq(15, 8, lower.tail=F)  ## this gives you the prob of a given chisq value
[1] 0.05914546
```

II. Principal components

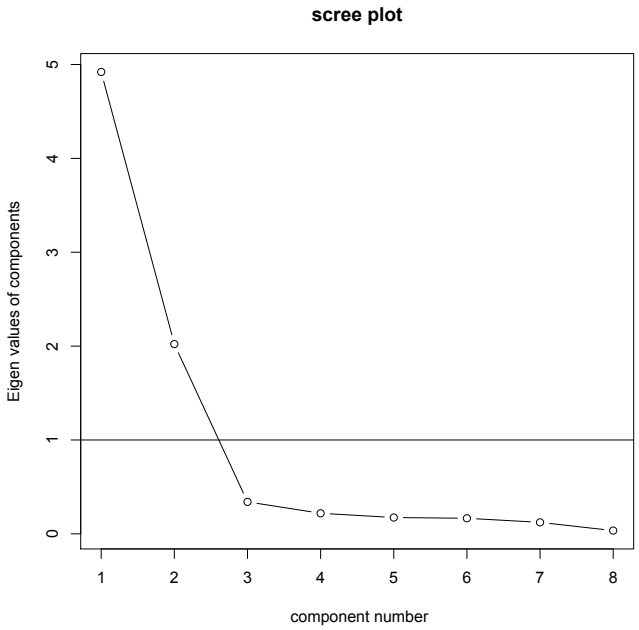
There are at least two ways to calculate principal components in R.
A. princomp — does principal components, yields eigenvalues.
Minimal output, can't control # of components??

```
> pca2b=princomp(mat2,factors=2) ## data are in mat2
> summary(pca2b)
Importance of components:
```

```
                          Comp.1    Comp.2     Comp.3     Comp.4     Comp.5     Comp.6
Standard deviation     5.4619755 3.5208939 1.47284209 1.21102164 1.03527649 1.00654155
Proportion of Variance 0.6106818 0.2537589 0.04440459 0.03002059 0.02193957 0.02073857
Cumulative Proportion  0.6106818 0.8644407 0.90884532 0.93886592 0.96080549 0.98154406
                           Comp.7      Comp.8
Standard deviation     0.84603547 0.431089184
Proportion of Variance 0.01465186 0.003804081
Cumulative Proportion  0.99619592 1.000000000

## look at scree plot, there are 2 ways to do this, note difference in scaling, the
output immediately below is the square of the sd of the components shown above, the
plot on the next page has the sd's themselves.
> screeplot(pca2b);
```



```
## From inside the 'psych' package can also make scree plot. Notice different scaling
from above. One is the square root of the other.
> VSS.scree(mat2)
```



B. Can calculate principal components using 'principal' in 'psych' package

```
> pca2=principal(mat2, nfactors=2, rotate="varimax", scores=F)
> pca2  ## get the output from R
Principal Components Analysis
Call: principal(r = mat2, nfactors = 2, rotate = "varimax", scores = F)
Standardized loadings (pattern matrix) based upon correlation matrix
                              RC1  RC2   h2    u2
Researchers_announce         0.16 0.90 0.84 0.157
Researchers_communicate_quickly 0.05 0.93 0.87 0.134
Researchers_pos_contribute   0.39 0.77 0.75 0.251
Researchers_available        0.25 0.88 0.84 0.156
Influence_worthwhile         0.92 0.23 0.90 0.097
Influence_benefits_community 0.92 0.18 0.89 0.113
Influence_important_topic    0.96 0.22 0.96 0.039
Influence_healthcare         0.93 0.14 0.89 0.110

                       RC1  RC2
SS loadings           3.74 3.21
Proportion Var        0.47 0.40
Cumulative Var        0.47 0.87
Proportion Explained  0.54 0.46
Cumulative Proportion 0.54 1.00

Test of the hypothesis that 2 components are sufficient.

The degrees of freedom for the null model are  28  and the objective function was  9.3
The degrees of freedom for the model are 13  and the objective function was  0.7
The total number of observations was  70  with MLE Chi Square =  44.89  with prob <
2.2e-05
```

```
Fit based upon off diagonal values = 1

## get more output
> pca2$loadings

Loadings:
                               RC1   RC2
Researchers_announce           0.157 0.905
Researchers_communicate_quickly       0.929
Researchers_pos_contribute     0.392 0.772
Researchers_available          0.254 0.883
Influence_worthwhile           0.923 0.225
Influence_benefits_community   0.924 0.183
Influence_important_topic      0.956 0.216
Influence_healthcare           0.933 0.137

                 RC1   RC2
SS loadings    3.736 3.207
Proportion Var 0.467 0.401
Cumulative Var 0.467 0.868

## See documentation for how to get residuals, scores, and other rotations. Notice
that asking for the loadings stored, the 'principal' program in 'psych' package omits
loadings below a low cutoff value.
```

III. "Common factors" or true factor analysis

A. Can use fa in psych package

```
> paf2=fa(mat2,nfactors=2,rotate="varimax",SMC=T,symmetric=T, fm="pa")
> paf2   ## ask for results
Factor Analysis using method =  pa
Call: fa(r = mat2, nfactors = 2, rotate = "varimax", SMC = T, symmetric = T,
    fm = "pa")
Standardized loadings (pattern matrix) based upon correlation matrix
                               PA1  PA2   h2    u2 com
Researchers_announce           0.16 0.87 0.79 0.212 1.1
Researchers_communicate_quickly 0.06 0.90 0.82 0.178 1.0
Researchers_pos_contribute     0.38 0.72 0.67 0.334 1.5
Researchers_available          0.26 0.86 0.80 0.203 1.2
Influence_worthwhile           0.90 0.23 0.87 0.131 1.1
Influence_benefits_community   0.90 0.19 0.84 0.160 1.1
Influence_important_topic      0.97 0.22 0.98 0.017 1.1
Influence_healthcare           0.90 0.15 0.84 0.160 1.1

                       PA1  PA2
SS loadings           3.62 2.99
Proportion Var        0.45 0.37
Cumulative Var        0.45 0.83
Proportion Explained  0.55 0.45
Cumulative Proportion 0.55 1.00

Mean item complexity =  1.1
Test of the hypothesis that 2 factors are sufficient.

The degrees of freedom for the null model are  28  and the objective function was  9.3
with Chi Square of  609.43
The degrees of freedom for the model are 13  and the objective function was  0.44

The root mean square of the residuals (RMSR) is  0.02
The df corrected root mean square of the residuals is  0.02
```

```
The harmonic number of observations is  70 with the empirical chi square  1.13  with
prob <  1
The total number of observations was  70  with MLE Chi Square =  28.15  with prob <
0.0086

Tucker Lewis Index of factoring reliability =  0.943
RMSEA index =  0.139  and the 90 % confidence intervals are  0.062 0.195
BIC =  -27.08
Fit based upon off diagonal values = 1
Measures of factor score adequacy
                                                 PA1  PA2
Correlation of scores with factors              0.99 0.96
Multiple R square of scores with factors        0.99 0.92
Minimum correlation of possible factor scores   0.98 0.85

## See documentation for other options for both rotation and factoring methods.
```

B. Another option: factanal, which does maximum likelihood factor analysis

```
> mlf2=factanal(mat2, factors=2, rotation="varimax");
> mlf2; ## get R to show results

Call:
factanal(x = mat2, factors = 2, rotation = "varimax")

Uniquenesses:
           Researchers_announce Researchers_communicate_quickly
                          0.198                          0.172
     Researchers_pos_contribute           Researchers_available
                          0.348                          0.221
           Influence_worthwhile    Influence_benefits_community
                          0.169                          0.147
      Influence_important_topic             Influence_healthcare
                          0.010                          0.138

Loadings:
                               Factor1 Factor2
Researchers_announce           0.147   0.883
Researchers_communicate_quickly        0.908
Researchers_pos_contribute     0.364   0.721
Researchers_available          0.248   0.847
Influence_worthwhile           0.878   0.245
Influence_benefits_community   0.902   0.197
Influence_important_topic      0.968   0.228
Influence_healthcare           0.915   0.155

               Factor1 Factor2
SS loadings      3.580   3.016
Proportion Var   0.447   0.377
Cumulative Var   0.447   0.824

Test of the hypothesis that 2 factors are sufficient.
The chi square statistic is 25.15 on 13 degrees of freedom.
The p-value is 0.0221

## See documentation for estimating factor scores, etc
```

IV. Other nifty things related to principal components or factor analysis **in psych package**

A. **Bartlett's test for a correlation matrix** (is it identity matrix + error). You shouldn't do factor analysis on a random matrix. Also known as Bartlett's test of sphericity. You want the Bartlett test to have a small p-value.

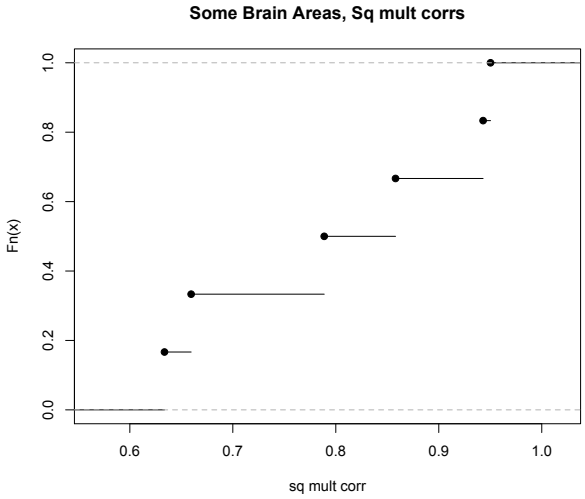
```
> cortest.bartlett(D2AxS[,26:31])  ## example using columns 26:31 of my data
R was not square, finding R from data
$chisq
[1] 75.47375

$p.value
[1] 4.648804e-10

$df
[1] 15

> cortest.mat(D2AxS[,26:31]); ## also calculates Bartlett's test
Bartlett's test of is R = I
Tests of correlation matrices
Call:cortest.mat(R1 = D2AxS[, 26:31])
 Chi Square value 75.47  with df =  15   with probability < 4.6e-10
Warning message:
In cortest.mat(D2AxS[, 26:31]) :
  R1 matrix was not square, correlations found> cortest.jennrich   compares
matrices

> cortest.normal;  ## differs but can use this to compare pairs of matrices,
which is interesting to do if you have two samples tested on the same
variables
```

B. Get the **squared multiple correlations** of each variable with all the others. Look at these to see if you should throw out some variables. Some say use a .30 (about 10% shared variance) criterion, but it is just a rule of thumb.

```
> smc1= smc(D2AxS[,26:31]); ## a few columns of my data again
   subnig       vta      acmb      amyg        bs      caud
0.7887606 0.6336429 0.9430341 0.8579995 0.6595345 0.9502663
```

When you use this, it is important to look at the content of the items and to think. You can also plot the cumulative distribution function of the squared multiple correlations and look at it to get a feel for whether some items don't correlate very well with the others.

```
> plot.ecdf(smc1, main="Some Brain Areas, Sq mult corrs", xlab="sq mult
corr")
```

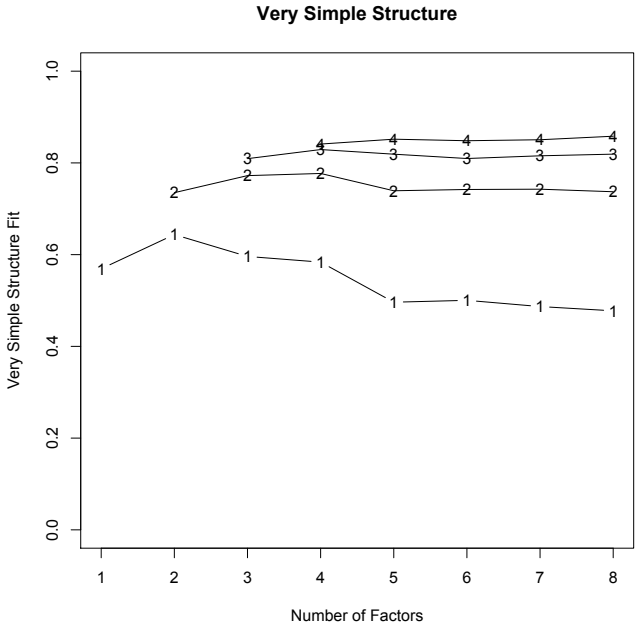



```
> abline(v=.3, lty=2)  ## puts a vertical line at .30 to aid eyeball, which
wasn't necessary in this case. Some recommend removing items that have smc
less than .3 with other items. Can position it wherever you like.
```

C. **VSS, or Very Simple structure**, an aid to choosing number of factors.
VSS gives a lot of information, not just the VSS fit, which is also plotted. Complexity is the number of factors that each variable loads on. Choose the number of factors based on the maximum value of the VSS. Note, this gives a different answer for the data example below from the parallel analysis in the next section. This also included Velicer's MAP function, which should be minimized.

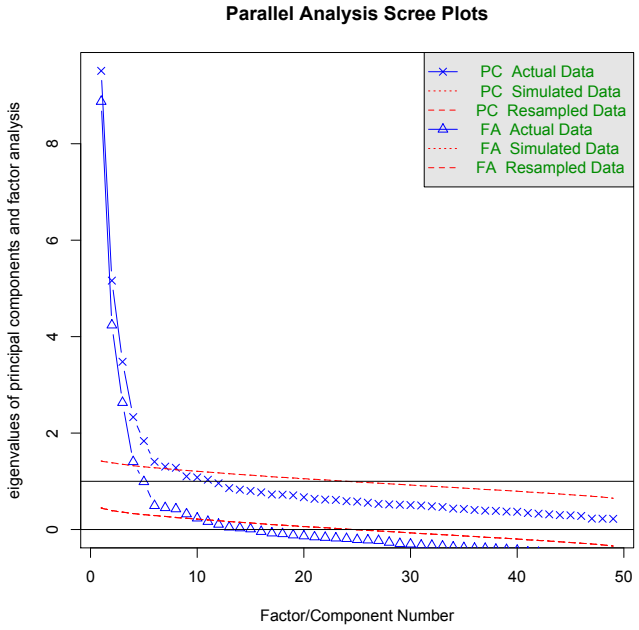
```
> VSS(asiq, n=8, rotate="varimax", fm="pa", plot=T)
# different data set from earlier in this handout

Very Simple Structure
Call: vss(x = x, n = n, rotate = rotate, diagonal = diagonal, fm = fm,
    n.obs = n.obs, plot = plot, title = title)
VSS complexity 1 achieves a maximimum of 0.64  with  2  factors
VSS complexity 2 achieves a maximimum of 0.78  with  4  factors

The Velicer MAP achieves a minimum of 0.01  with  5  factors
BIC achieves a minimum of  -3054.04  with  8  factors
Sample Size adjusted BIC achieves a minimum of  -474.89  with  8  factors

Statistics by number of factors
  vss1 vss2    map  dof chisq     prob sqresid  fit RMSEA   BIC SABIC complex eChisq
1 0.57 0.00 0.0207 1127 13383  0.0e+00      68 0.57 0.099  5451  9031     1.0  38321
2 0.64 0.74 0.0130 1079  8950  0.0e+00      42 0.74 0.081  1356  4783     1.3  16478
3 0.60 0.77 0.0087 1032  6313  0.0e+00      30 0.81 0.068  -950  2328     1.5   7918
4 0.58 0.78 0.0073  986  4948  0.0e+00      25 0.84 0.060 -1991  1141     1.7   4972
5 0.50 0.74 0.0069  941  4068  0.0e+00      22 0.86 0.055 -2555   434     2.0   3454
6 0.50 0.74 0.0073  897  3613 2.6e-321      20 0.87 0.052 -2700   149     2.1   2854
7 0.49 0.74 0.0076  854  3071 7.2e-247      19 0.88 0.048 -2940  -227     2.1   2267
8 0.48 0.74 0.0076  812  2661 5.8e-195      17 0.89 0.045 -3054  -475
2.1   1753
   SRMR eCRMS  eBIC
1 0.120 0.122 30389
2 0.078 0.082  8884
3 0.054 0.058   654
4 0.043 0.047 -1967
5 0.036 0.040 -3169
6 0.033 0.037 -3459
```

```
7 0.029 0.034 -3743
8 0.026 0.031 -3962
```



D. **Parallel analysis.** Choose the number of factors by simulating a random data set, and choosing the point where the eigenvalues of the real data fall below the simulated data.

```
> pfa3=fa.parallel(asiq, fm="minres",fa="both")
Parallel analysis suggests that the number of factors =  10  and the number
of components =  8
```



E. **Compare factor solutions.** Some writers say to "Factor the data by several different analytical procedures and hold sacred only those factors that appear across all the procedures used." (Gorsuch, Factor Analysis,p. 330, 1983).

```
> pca1=principal(asiq, nfactors=5, rotate="promax",scores=F)
> paf1=fa(asiq,nfactors=5,rotate="varimax",SMC=T,symmetric=T, fm="pa")
> factor.congruence(pca1,paf1); # compare princ comp and factor analysis
      PA1   PA3   PA2   PA4   PA5
PC3 -0.39 -0.98  0.12  0.02 -0.05
PC1  0.94  0.36 -0.09  0.00  0.24
PC2 -0.08 -0.03  0.97 -0.13  0.07
PC4  0.06 -0.04 -0.23  0.98  0.18
PC5  0.14 -0.04  0.01 -0.06  0.94
```

For this example, models are fit with 5 principal components or with 5 factors, and different rotations are applied. I have highlighted the diagonal elements, because the components/factors are not ordered the same.

Another way to do this is to use the solution from one set of data and apply it to another (for example, a random half of the sample).

```
> predict.psych; (see documentation in R)
```

F. **Compare factor solutions by applying one analysis to another data set.** See documentation in psych package. Example with 2 data sets with 8 variables, make 2 principal components.

```
## get pca from survey 1, apply to survey 2, then make correlations

> pca1=principal(survey1items, nfactors=2, rotate="varimax", scores=T);
> predpca2=predict(pca1, survey2items, survey1items);
> pca2=principal(survey2items, nfactors=2, rotate="varimax", scores=T);
> round(cor(pca2$scores,predpca2, use="pairwise.complete.obs"),2);

     RC1  RC2
RC1 0.16 0.98
RC2 0.99 0.17

### works in reverse too-- predict survey1 from survey2 pca
> predpca1=predict(pca2,survey1items, survey2items);
> round(cor(pca1$scores, predpca1, use="pairwise.complete.obs"),3);
       RC1    RC2
RC1 -0.178  0.992
RC2  0.979 -0.124
```

G. **Sort the factors by loading** size, makes it easier to think through.

```
> fa.sort(faresults)  ## where 'faresults' has the results of a factor analysis
> fa.organize (faresults)  ## leaves items in original order
```

H. ..

I. **Kaiser-Meyer-Olin** test of "sampling adequacy". Some say don't extract factors if this is below .50. The higher the better.

```
> KMO(D2AxS[, 26:31])  ## a few columns of a small data set
Kaiser-Meyer-Olkin factor adequacy
Call: KMO(r = D2AxS[, 26:31])
Overall MSA =  0.55
MSA for each item =
subnig    vta   acmb   amyg     bs   caud
  0.58   0.49   0.57   0.58   0.47   0.56
```

J. Calculate Cronbach's alpha(see Revelle's documentation for other methods that are less entrenched but perhaps better)

First make a matrix with the items in your scale. Then use 'alpha'.

```
> library(psych); ## just a reminder to you to activate the 'psych' package
> fac1=data.frame(R_understandable_language, R_friendly, R_available,
      R_announceresults, R_reportresults, R_sigcontribution_community,
      R_sigcontribution_personal) # put the variables in a data frame

> alphafac1=alpha(fac1,keys=NULL, cumulative=F,na.rm=T)
> alphafac1  ## get the results from R

Reliability analysis
Call: alpha(x = fac1, keys = NULL, cumulative = F, na.rm = T)

  raw_alpha std.alpha G6(smc) average_r S/N   ase mean  sd
       0.9       0.9    0.92      0.56 8.8 0.034  6.5 1.9

 lower alpha upper     95% confidence boundaries
0.83 0.9 0.96

 Reliability if an item is dropped:
                           raw_alpha std.alpha G6(smc) average_r S/N alpha se
R_understandable_language       0.89      0.89    0.91      0.58 8.4    0.038
R_friendly                      0.89      0.89    0.90      0.58 8.2    0.038
R_available                     0.88      0.88    0.91      0.56 7.5    0.040
R_announceresults               0.87      0.87    0.89      0.53 6.8    0.041
R_reportresults                 0.87      0.87    0.89      0.54 6.9    0.041
R_sigcontribution_community     0.87      0.88    0.90      0.54 7.1    0.041
R_sigcontribution_personal      0.89      0.89    0.91      0.58 8.2    0.039

 Item statistics
                             n    r r.cor r.drop mean  sd
R_understandable_language  112 0.72  0.66   0.60  6.7 2.3
R_friendly                 113 0.73  0.68   0.61  7.5 2.1
R_available                114 0.79  0.75   0.70  6.4 2.3
R_announceresults          112 0.86  0.86   0.80  6.3 2.4
R_reportresults            115 0.85  0.85   0.79  6.3 2.5
R_sigcontribution_community 114 0.83  0.81   0.78  6.5 2.7
R_sigcontribution_personal  114 0.74  0.68   0.64  5.8 2.9
```

V. Nifty stuff (Not inside 'psych' package)

Sometimes we want to split a large sample in order to cross validate a factor solution.

K. Code to split a large enough data set randomly into 2 groups (won't be exactly equal, but fiddle around until the split is close to equal)

```
> x=as.matrix(sample(c(0,1),1139, replace=T))
# the data sample has 1139 observations, so create a variable, x, with 1139
randomly sampled 1's and 0's.
> mean(x) # find the mean to see how close to an equal split it was
[1] 0.4978051  ## can re-do the split until we get one that is about 50-50
> newdat=cbind(x,asiq); ## column bind the new variable with original data
> ncol(newdat)  ## original data had 49 columns, checking that now I have 50
[1] 50
> newdat1=subset(newdat, x==1); ## now I extract the cases with the 1's
## the '==' means logically true
> nrow(newdat1);
[1] 567   ## there are 567 observations in the data set labeled 1.
> newdat0=subset(newdat, x==0); ## extract the cases with the 0's
> nrow(newdat0); ## check the number of observations
```

```
[1] 572    ## there are 572 observations in the data set labeled 0.
> library(MASS)  ## the write.matrix function is in MASS package
> write.matrix(newdat0,file="TA0data.txt",sep=" ")
# save the results for the cases with 0's  Instead of using a blank as the
separator you can use a comma to create a csv file

> write.matrix(newdat1, file="TA1data.txt",sep=" ")
# save the results for the cases with 1's.
```

L. **Code to do what**
M. **more nifty code maybe**

1. (blah blah to be continued… perhaps)
2.