Regression Homework

William Revelle
Northwestern University
Prepared as part of course on psychometric theory (Psychology 405)
and as a supplement to the Short Guide to R for psychologists

April 11, 2017

Contents

1 Introduction

A standard problem in psychology is to predict a dependent variable as a function of multiple independent variables. This is, of course, the problem of multiple regression. R does this as one case of the standard linear model.

$$model = Y \sim X$$

or

$$\hat{\mathbf{Y}} = \beta_{y.x}\mathbf{X} + \epsilon$$

Both Y and X can be matrices, in which case as many multiple regressions will be done as the number of variables in Y.

Consider the following (artificial) data set of ability tests, motivation tests, and performance measures. The Ability tests are (simulated) GRE Verbal, Quantitative, and Advanced, the motivation tests measure Need for Achievement and Anxiety, the Performance measures are graduate GPA, rated performance on the Prelims, and quality of the Masters Thesis.

2 Getting the data

The data are read from a remote file and stored in a data.frame called dataset. The first variable in dataset is the ID number. This column can be dropped for all columns of the dataset.

We can do this directly from R or we can read it indirectly by opening the file (https://personality-project.org/r/datasets/psychometrics.prob2.txt in our browser, copying the file to the clipboard and then reading the clipboard.

```
library(psych). #always do this when you start
fn="https://personality-project.org/r/datasets/psychometrics.prob2.txt"
dataset =read.table(datafilename, header=TRUE)   #read the data file
#or
dataset <- read.file(fn). #using the psych package read.file function
#or we can try another approach
library(psych)
dataset <- read.clipboard()
names(dataset)                                  #what are the variables?
dataset=dataset[,-1]                            #get rid of the ID
names(dataset)                                  #check the names again
```

```
names(dataset)
[1] "ID"     "GREV"   "GREQ"   "GREA"   "Ach"    "Anx"    "Prelim" "GPA"    "MA"
> names(dataset) #we dropped variable 1
[1] "GREV"   "GREQ"   "GREA"   "Ach"    "Anx"    "Prelim" "GPA"    "MA"
```

3 Descriptive statistics including graphic displays

It is important when doing regressions to actually look at the data. First we can find the correlations and display them two different ways.

```
> describe(dataset)                                 #get the basic descriptive statistics

       vars    n   mean     sd median trimmed    mad   min     max  range
skew kurtosis   se
ID        1 1000 500.50 288.82 500.50  500.50 370.65   1.0 1000.00 999.00
 0.00     -1.20 9.13
GREV      2 1000 499.77 106.11 497.50  498.75 106.01 138.0  873.00 735.00
 0.09     -0.07 3.36
GREQ      3 1000 500.53 103.85 498.00  498.51 105.26 191.0  914.00 723.00
 0.22      0.08 3.28
GREA      4 1000 498.13 100.45 495.00  498.67  99.33 207.0  848.00 641.00 -0.02
-0.06 3.18
Ach       5 1000  49.93   9.84  50.00   49.88  10.38  16.0   79.00  63.00
 0.00      0.02 0.31
Anx       6 1000  50.32   9.91  50.00   50.43  10.38  14.0   78.00  64.00 -0.14
 0.14 0.31
Prelim    7 1000  10.03   1.06  10.00   10.02   1.48   7.0   13.00   6.00 -0.02
-0.01 0.03
GPA       8 1000   4.00   0.50   4.02    4.01   0.53   2.5    5.38   2.88 -0.07
-0.29 0.02
MA        9 1000   3.00   0.49   3.00    3.00   0.44   1.4    4.50   3.10 -0.07
-0.09 0.02

> round(cor(dataset),2)                             #find the correlation matrix, round to 2 decimals

          ID  GREV GREQ  GREA   Ach   Anx Prelim   GPA    MA
ID      1.00 -0.01 0.00 -0.01  0.00 -0.01   0.02  0.00 -0.01
GREV   -0.01  1.00 0.73  0.64  0.01  0.01   0.43  0.42  0.32
GREQ    0.00  0.73 1.00  0.60  0.01  0.01   0.38  0.37  0.29
GREA   -0.01  0.64 0.60  1.00  0.45 -0.39   0.57  0.52  0.45
Ach     0.00  0.01 0.01  0.45  1.00 -0.56   0.30  0.28  0.26
Anx    -0.01  0.01 0.01 -0.39 -0.56  1.00  -0.23 -0.22 -0.22
Prelim  0.02  0.43 0.38  0.57  0.30 -0.23   1.00  0.42  0.36
GPA     0.00  0.42 0.37  0.52  0.28 -0.22   0.42  1.00  0.31
MA     -0.01  0.32 0.29  0.45  0.26 -0.22   0.36  0.31  1.00

> # or
> #lowerCor(dataset). which rounds it and shows it neatly (see below)
> z.data=scale(dataset)                             #convert to standardized scores
> z.df=data.frame(z.data)                           #convert to a data.frame for regressions
```

We can also use the `lowerCor` function from the *psych* package to find the correlations, round to 2 decimals, and show it in lower triangular form. We do this on the standardized data just to make the point that correlations are just standardized covariances and do not vary with a transformation such as standardization.

```
> lowerCor(z.df)

       ID    GREV  GREQ  GREA  Ach   Anx   Prelm GPA   MA
ID      1.00
GREV   -0.01  1.00
GREQ    0.00  0.73  1.00
GREA   -0.01  0.64  0.60  1.00
```

```
Ach      0.00  0.01  0.01  0.45  1.00
Anx     -0.01  0.01  0.01 -0.39 -0.56  1.00
Prelim   0.02  0.43  0.38  0.57  0.30 -0.23  1.00
GPA      0.00  0.42  0.37  0.52  0.28 -0.22  0.42  1.00
MA      -0.01  0.32  0.29  0.45  0.26 -0.22  0.36  0.31  1.00
```

3.1 Graph the scatter plots

Scatter Plot matrices are very useful ways of showing many relations at once (Figure 1).

```
> pairs.panels(z.df,pch='.', gap=0) #by setting the gap, we make a slightly bigger figure.
```

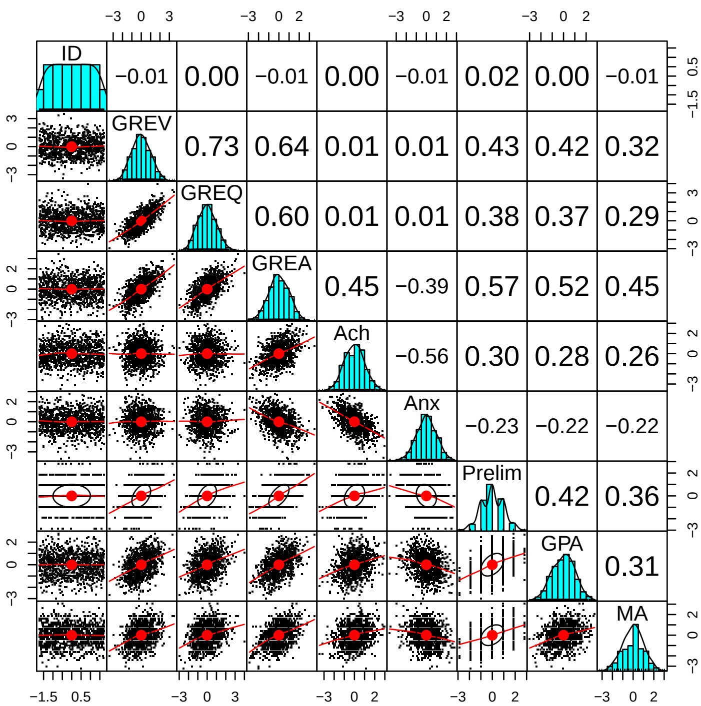

Figure 1: A pairs panels graphic. Because we have many cases, it is better to use a very small (.) plot character (pch='.')

4 Regressions from the raw data

There are a number of analyses that can be done. The most basic is to note from the correlation matrix that the first 3 variables seem to be good predictors of the prelim performance.

What is the multiple correlation of Prelim performance on these three measures of ability? We test this by using the *lm* (linear model) function and then asking for summary statistics.

At the data level, we can work with the raw data matrix X, or convert these to deviation scores (X.dev) by subtracting the means from all elements of X. At the raw data level we have

$$_m\hat{Y}_1 =_m X_{nn}\beta_1 +_m \epsilon_1 \tag{1}$$

and we can solve for $_n\beta_1$ by pre multiplying by $_nX'_m$ (thus making the matrix on the right side of the equation into a square matrix so that we can multiply through by an inverse.)

$$_nX'_{mm}\hat{Y}_1 =_n X'_{mm}X_{nn}\beta_1 +_m \epsilon_1 \tag{2}$$

and then solving for beta by pre multiplying both sides of the equation by $(XX')^{-1}$

$$\beta = (XX')^{-1}X'Y \tag{3}$$

These beta weights will be the weights with no intercept. Compare this solution to the one using the lm function with the intercept removed.

```
> mod.1 <- lm(Prelim ~ GREV + GREQ + GREA, data=dataset)

Call:
lm(formula = Prelim ~ GREV + GREQ + GREA, data = dataset)

Coefficients:
(Intercept)         GREV         GREQ         GREA
  6.8398644    0.0009607    0.0001318    0.0052978

> summary(mod.1)

Call:
lm(formula = Prelim ~ GREV + GREQ + GREA, data = dataset)

Residuals:
```

```
     Min        1Q    Median        3Q       Max
-2.95824  -0.59584  -0.00721   0.56142   2.99086

Coefficients:
             Estimate Std. Error t value Pr(>|t|)
(Intercept) 6.8398644  0.1539609  44.426   <2e-16 ***
GREV        0.0009607  0.0004063   2.364   0.0182 *
GREQ        0.0001318  0.0003969   0.332   0.7399
GREA        0.0052978  0.0003661  14.470   <2e-16 ***
---
Signif. codes:  0 âĂŸ***âĂŹ 0.001 âĂŸ**âĂŹ 0.01 âĂŸ*âĂŹ 0.05 âĂŸ.âĂŹ 0.1 âĂ

Residual standard error: 0.8647 on 996 degrees of freedom
Multiple R-squared:  0.3341,    Adjusted R-squared:  0.3321
F-statistic: 166.6 on 3 and 996 DF,  p-value: < 2.2e-16

> mod.2 <- lm(MA ~ GREV + GREQ + GREA, data=dataset)

Call:
lm(formula = MA ~ GREV + GREQ + GREA, data = dataset)

Coefficients:
(Intercept)         GREV         GREQ         GREA
  1.844e+00    2.555e-04   -1.986e-05    2.076e-03
```

What about looking at how long it takes to get a MA? Use the lm function again.

```
> mod.2 <- lm(Prelim ~ GREV + GREQ + GREA, data=dataset)

Call:
lm(formula = Prelim ~ GREV + GREQ + GREA, data = dataset)

Coefficients:
(Intercept)         GREV         GREQ         GREA
  6.8398644    0.0009607    0.0001318    0.0052978

> summary(mod.1)

Call:
lm(formula = Prelim ~ GREV + GREQ + GREA, data = dataset)
```

```
Residuals:
     Min       1Q   Median       3Q      Max
-2.95824 -0.59584 -0.00721  0.56142  2.99086

Coefficients:
             Estimate Std. Error t value Pr(>|t|)
(Intercept) 6.8398644  0.1539609  44.426   <2e-16 ***
GREV        0.0009607  0.0004063   2.364   0.0182 *
GREQ        0.0001318  0.0003969   0.332   0.7399
GREA        0.0052978  0.0003661  14.470   <2e-16 ***
---
Signif. codes:  0 âĂŸ***âĂŹ 0.001 âĂŸ**âĂŹ 0.01 âĂŸ*âĂŹ 0.05 âĂŸ.âĂŹ 0.1 âĂ

Residual standard error: 0.8647 on 996 degrees of freedom
Multiple R-squared:  0.3341,    Adjusted R-squared:  0.3321
F-statistic: 166.6 on 3 and 996 DF,  p-value: < 2.2e-16
```

5 Forming sets of variables, and then doing linear regression

Given that there are many different predictors that seem to work, we could work our way through with all of them, or we could find sets variables of ability, motivation, and performance and examine how these sets work. We use the `with` to do this operation.

This forms three sets of variables and then finds the linear regression of each set of predictors with each of the set of criteria.

```
> with(z.df, {
+                     #This temporarily uses the standardized set
+ ability <- cbind(GREV,GREQ,GREA)      #form three different sets of variables
+ motivation <- cbind(Ach,Anx)
+ perform<- cbind(Prelim,GPA,MA)
+ comps.df <- data.frame(ability,motivation, perform)  #all together
+ #show the correlations
+ lowerCor(comps.df)
+ model.regress <- lm(perform~ability+motivation)
```

```
+ model.regress                                          #show the
resulting regression
+ print(model.regress, digits=3)                     #another way of
showing the result
+ summary(model.regress)                              #yet another way
of showing the output
+ #this shows each separate regression
+ }   #this is the end of the expression in the with statement
+ )   #this is the end of the with function
```

```
        GREV  GREQ  GREA   Ach    Anx   Prelm GPA   MA
GREV    1.00
GREQ    0.73  1.00
GREA    0.64  0.60  1.00
Ach     0.01  0.01  0.45  1.00
Anx     0.01  0.01 -0.39 -0.56  1.00
Prelim  0.43  0.38  0.57  0.30 -0.23  1.00
GPA     0.42  0.37  0.52  0.28 -0.22  0.42  1.00
MA      0.32  0.29  0.45  0.26 -0.22  0.36  0.31  1.00

Call:
lm(formula = perform ~ ability + motivation)

Coefficients:
               Prelim     GPA        MA
(Intercept)    -7.08e-16  -5.81e-17  -6.33e-16
abilityGREV     1.40e-01   1.99e-01   1.04e-01
abilityGREQ     3.94e-02   5.10e-02   2.56e-02
abilityGREA     4.04e-01   2.86e-01   3.11e-01
motivationAch   1.14e-01   1.19e-01   9.62e-02
motivationAnx  -8.50e-03  -4.70e-02  -4.57e-02

Response Prelim :

Call:
lm(formula = Prelim ~ ability + motivation)

Residuals:
     Min       1Q    Median      3Q     Max
-2.85962 -0.55321 -0.00403 0.50350 2.71849
```

```
Coefficients:
                 Estimate Std. Error t value Pr(>|t|)
(Intercept)    -7.080e-16  2.570e-02   0.000 1.000000
abilityGREV     1.399e-01  4.259e-02   3.283 0.001061 **
abilityGREQ     3.944e-02  3.954e-02   0.998 0.318722
abilityGREA     4.041e-01  4.539e-02   8.903  < 2e-16 ***
motivationAch   1.143e-01  3.441e-02   3.323 0.000925 ***
motivationAnx  -8.500e-03  3.239e-02  -0.262 0.793050
---
Signif. codes:  0 âĂŸ***âĂŹ 0.001 âĂŸ**âĂŹ 0.01 âĂŸ*âĂŹ 0.05 âĂŸ.âĂŹ 0.1 âĂ

Residual standard error: 0.8126 on 994 degrees of freedom
Multiple R-squared:  0.3429,    Adjusted R-squared:  0.3396
F-statistic: 103.8 on 5 and 994 DF,  p-value: < 2.2e-16


Response GPA :

Call:
lm(formula = GPA ~ ability + motivation)

Residuals:
     Min       1Q   Median       3Q      Max
-2.60408 -0.61369  0.02232  0.60416  2.53718

Coefficients:
                 Estimate Std. Error t value Pr(>|t|)
(Intercept)    -5.810e-17  2.665e-02   0.000 1.000000
abilityGREV     1.987e-01  4.418e-02   4.497 7.72e-06 ***
abilityGREQ     5.098e-02  4.101e-02   1.243 0.214094
abilityGREA     2.861e-01  4.708e-02   6.078 1.73e-09 ***
motivationAch   1.190e-01  3.569e-02   3.334 0.000886 ***
motivationAnx  -4.703e-02  3.360e-02  -1.400 0.161883
---
Signif. codes:  0 âĂŸ***âĂŹ 0.001 âĂŸ**âĂŹ 0.01 âĂŸ*âĂŹ 0.05 âĂŸ.âĂŹ 0.1 âĂ

Residual standard error: 0.8429 on 994 degrees of freedom
Multiple R-squared:  0.2931,    Adjusted R-squared:  0.2895
```

```
F-statistic: 82.43 on 5 and 994 DF,  p-value: < 2.2e-16


Response MA :

Call:
lm(formula = MA ~ ability + motivation)

Residuals:
     Min       1Q   Median       3Q      Max
-2.66414 -0.61142  0.02422  0.61993  2.87736

Coefficients:
                 Estimate Std. Error t value Pr(>|t|)
(Intercept)    -6.328e-16  2.804e-02   0.000   1.0000
abilityGREV     1.041e-01  4.648e-02   2.240   0.0253 *
abilityGREQ     2.555e-02  4.314e-02   0.592   0.5537
abilityGREA     3.111e-01  4.952e-02   6.282 4.98e-10 ***
motivationAch   9.621e-02  3.754e-02   2.563   0.0105 *
motivationAnx  -4.569e-02  3.534e-02  -1.293   0.1964
---
Signif. codes:  0 âĂŸ***âĂŹ 0.001 âĂŸ**âĂŹ 0.01 âĂŸ*âĂŹ 0.05 âĂŸ.âĂŹ 0.1 âĂ

Residual standard error: 0.8867 on 994 degrees of freedom
Multiple R-squared:  0.2177,    Adjusted R-squared:  0.2138
F-statistic: 55.32 on 5 and 994 DF,  p-value: < 2.2e-16
```

5.1 Using `setCor`

This can also be done using the `setCor` function. We use the `summary` function to just give the important values.

```
> summary(setCor(y=c("Prelim","GPA","MA"),x=c("GREV","GREQ","GREA","Ach","Anx"),data=z.df))

Multiple Regression from raw data
setCor(y = c("Prelim", "GPA", "MA"), x = c("GREV", "GREQ", "GREA",
    "Ach", "Anx"), data = z.df)

Multiple Regression from matrix input

Beta weights
     Prelim   GPA    MA
GREV 0.1399 0.199 0.104
```

```
GREQ   0.0394   0.051   0.026
GREA   0.4041   0.286   0.311
Ach    0.1143   0.119   0.096
Anx   -0.0085  -0.047  -0.046

Multiple R
Prelim    GPA     MA
  0.59   0.54   0.47

Multiple R2
Prelim    GPA     MA
  0.34   0.29   0.22

Cohen's set correlation R2
[1] 0.5

Squared Canonical Correlations
[1] 0.4943 0.0036 0.0017
[1] 0.494343922 0.003621246 0.001671701
```

=setCor will work on raw data, or from the correlation matrix.

```
> R <- lowerCor(z.df)#find the correlations

        ID     GREV   GREQ   GREA   Ach    Anx    Prelm  GPA
MA
ID       1.00
GREV    -0.01   1.00
GREQ     0.00   0.73   1.00
GREA    -0.01   0.64   0.60   1.00
Ach      0.00   0.01   0.01   0.45   1.00
Anx     -0.01   0.01   0.01  -0.39  -0.56   1.00
Prelim   0.02   0.43   0.38   0.57   0.30  -0.23   1.00
GPA      0.00   0.42   0.37   0.52   0.28  -0.22   0.42   1.00
MA      -0.01   0.32   0.29   0.45   0.26  -0.22   0.36   0.31
1.00

>  summary(setCor(y=c("Prelim","GPA","MA"),x=c("GREV","GREQ","GREA","Ach",

Multiple Regression from matrix input
setCor(y = c("Prelim", "GPA", "MA"), x = c("GREV", "GREQ", "GREA",
    "Ach", "Anx"), data = R)

Multiple Regression from matrix input

Beta weights
       Prelim    GPA     MA
```

```
GREV    0.1399   0.199   0.104
GREQ    0.0394   0.051   0.026
GREA    0.4041   0.286   0.311
Ach     0.1143   0.119   0.096
Anx    -0.0085  -0.047  -0.046

Multiple R
Prelim    GPA     MA
  0.59    0.54   0.47

Multiple R2
Prelim    GPA     MA
  0.34    0.29   0.22

Cohen's set correlation R2
[1] 0.5

Squared Canonical Correlations
[1] 0.4943 0.0036 0.0017
[1] 0.494343922 0.003621246 0.001671701
```

Compare this with the previous analysis.

6 Forming Composites of variables, and then doing linear regression

Alternatively, we could find composites of the three sets of variables (linear sums) and then find their relationships.

```
>  #first form the composite data frame
> temp <- within(z.df, {
+                        #This temporarily uses the standardized set
+ ability <- cbind(GREV,GREQ,GREA)          #form three different sets of variables
+ ability <- rowSums(ability)
+ motivation <- cbind(Ach,-Anx)       #note that we reverse the sign of Anxiety
+ motivation <- rowSums(motivation)
+ perform <- cbind(Prelim,GPA,MA)
+ perform <- rowSums(perform)
```

```
+ }
+ )
> composite.df <- data.frame(temp[c('ability','motivation','perform')])
> lowerCor(composite.df)

           ablty mtvtn prfrm
ability    1.00
motivation 0.18  1.00
perform    0.63  0.38  1.00

> #now do the linear regression
> model.composite <- lm(perform~ability + motivation, data=composite.df)
> summary(model.composite)

Call:
lm(formula = perform ~ ability + motivation, data = composite.df)

Residuals:
    Min      1Q  Median      3Q     Max
-5.8496 -1.1433 -0.0521  1.1866  4.8591

Coefficients:
              Estimate Std. Error t value Pr(>|t|)
(Intercept) -9.726e-16  5.271e-02    0.00        1
ability      4.984e-01  2.037e-02   24.47   <2e-16 ***
motivation   3.515e-01  3.039e-02   11.56   <2e-16 ***
---
Signif. codes:  0 âĂŸ***âĂŹ 0.001 âĂŸ**âĂŹ 0.01 âĂŸ*âĂŹ 0.05 âĂŸ.âĂŹ 0.1 âĂ

Residual standard error: 1.667 on 997 degrees of freedom
Multiple R-squared:  0.4641,    Adjusted R-squared:  0.463
F-statistic: 431.7 on 2 and 997 DF,  p-value: < 2.2e-16
```

6.1 Unit weight the predictors vs optimal weight

Compare what happens if we find the linear sum of our five predictors to predict one outcome, versus the multiple R. Note that we sum the first four and then subtract Anx from the total. We believe that Anx should be negatively weighted.

Compare the next two outputs. One was just adding up the first 4 predictors and subtracting anxiety. Weighting them all equally. The other is using optimal weights, and doesn't actually do much better.

```
> all <- rowSums(z.df[2:5] ) - z.df[,6]
> mod.all <- lm(z.df$Prelim ~ all)
> summary(mod.all)

Call:
lm(formula = z.df$Prelim ~ all)

Residuals:
     Min       1Q   Median       3Q      Max
-2.92331 -0.54784 -0.02613  0.54270  3.03636

Coefficients:
             Estimate Std. Error t value Pr(>|t|)
(Intercept) -6.950e-16  2.623e-02    0.00        1
all          1.633e-01  7.664e-03   21.31   <2e-16 ***
---
Signif. codes:  0 âĂŸ***âĂŹ 0.001 âĂŸ**âĂŹ 0.01 âĂŸ*âĂŹ 0.05 âĂŸ.âĂŹ 0.1 âĂ

Residual standard error: 0.8294 on 998 degrees of freedom
Multiple R-squared:  0.3128,     Adjusted R-squared:  0.3121
F-statistic: 454.2 on 1 and 998 DF,  p-value: < 2.2e-16

> #vs optimal linear model
> mod.linear <- lm(Prelim ~ GREV + GREQ  + GREA + Ach + Anx,data=z.df)
> summary(mod.linear)

Call:
lm(formula = Prelim ~ GREV + GREQ + GREA + Ach + Anx, data = z.df)

Residuals:
     Min       1Q   Median       3Q      Max
-2.85962 -0.55321 -0.00403  0.50350  2.71849

Coefficients:
             Estimate Std. Error t value Pr(>|t|)
(Intercept) -7.080e-16  2.570e-02   0.000 1.000000
GREV         1.399e-01  4.259e-02   3.283 0.001061 **
```

```
GREQ           3.944e-02   3.954e-02    0.998  0.318722
GREA           4.041e-01   4.539e-02    8.903   < 2e-16 ***
Ach            1.143e-01   3.441e-02    3.323  0.000925 ***
Anx           -8.500e-03   3.239e-02   -0.262  0.793050
---
Signif. codes:  0 âĂŸ***âĂŹ 0.001 âĂŸ**âĂŹ 0.01 âĂŸ*âĂŹ 0.05 âĂŸ.âĂŹ 0.1 âĂ

Residual standard error: 0.8126 on 994 degrees of freedom
Multiple R-squared:  0.3429,    Adjusted R-squared:  0.3396
F-statistic: 103.8 on 5 and 994 DF,  p-value: < 2.2e-16
```

Note how both models fit almost equally well. What would happen if we took a random sample of the data, took the beta weights from the first sample, and applied them to the second sample? This is known as cross validation.

6.2 Validating our model from one subsample to another

```
>  set.seed(42)
> s1 <- sample(1:1000,500)
>  mod1 <- lm(Prelim ~ GREV + GREQ  + GREA + Ach + Anx,data=z.df[s1,])
>  z.m <- as.matrix(z.df)
>  coeff <- as.matrix(mod1$coefficients)
> mod.deriv <- z.m[s1,1:5] %*% coeff[-1] #the derivation sample
> mod.valid <- z.m[-s1,1:5] %*% coeff[-1] #the validation sample
> round( cor(mod.deriv,z.m[s1,]),2)  #derivation

       ID GREV GREQ GREA   Ach   Anx Prelim  GPA   MA
[1,] 0.34 0.72 0.92 0.66 -0.01 -0.01   0.42 0.41 0.28

> round(cor(mod.valid,z.m[-s1,]),2) #validation

       ID GREV GREQ GREA  Ach   Anx Prelim  GPA   MA
[1,] 0.36 0.72 0.93 0.63 0.05 -0.06    0.4 0.34 0.32
```

Compare this to unit weighting the data

```
> #the first data set
> all <- rowSums(z.df[s1,2:5] ) - z.df[s1,6]
>  mod.all <- lm(z.df[s1,]$Prelim ~ all)
>  summary(mod.all)

Call:
lm(formula = z.df[s1, ]$Prelim ~ all)

Residuals:
     Min       1Q   Median       3Q      Max
-2.95245 -0.50845  0.00002  0.53840  2.38406
```

```
Coefficients:
             Estimate Std. Error t value Pr(>|t|)
(Intercept)   0.02633    0.03724   0.707     0.48
all           0.17046    0.01098  15.530   <2e-16 ***
---
Signif. codes:  0 âĂŸ***âĂŹ 0.001 âĂŸ**âĂŹ 0.01 âĂŸ*âĂŹ 0.05 âĂŸ.âĂŹ 0.1 âĂŸ âĂŹ 1

Residual standard error: 0.8324 on 498 degrees of freedom
Multiple R-squared:  0.3263,    Adjusted R-squared:  0.3249
F-statistic: 241.2 on 1 and 498 DF,  p-value: < 2.2e-16

> #the replication data set
> all <- rowSums(z.df[-s1,2:5] ) - z.df[-s1,6]
> mod.all <- lm(z.df[-s1,]$Prelim ~ all)
>  summary(mod.all)

Call:
lm(formula = z.df[-s1, ]$Prelim ~ all)

Residuals:
     Min       1Q   Median       3Q      Max
-2.03455 -0.54645 -0.04724  0.53789  3.05249

Coefficients:
             Estimate Std. Error t value Pr(>|t|)
(Intercept) -0.02506    0.03698  -0.678    0.498
all          0.15684    0.01071  14.640   <2e-16 ***
---
Signif. codes:  0 âĂŸ***âĂŹ 0.001 âĂŸ**âĂŹ 0.01 âĂŸ*âĂŹ 0.05 âĂŸ.âĂŹ 0.1 âĂŸ âĂŹ 1

Residual standard error: 0.8266 on 498 degrees of freedom
Multiple R-squared:  0.3009,    Adjusted R-squared:  0.2995
F-statistic: 214.3 on 1 and 498 DF,  p-value: < 2.2e-16

>
```

6.3 Validating our model from one subsample to another. Take a smaller sample

```
>  set.seed(17)
> s1 <- sample(1:1000,500)
>  mod1 <- lm(Prelim ~ GREV + GREQ  + GREA + Ach + Anx,data=z.df[s1[1:100],])
>  z.m <- as.matrix(z.df)
>  coeff <- as.matrix(mod1$coefficients)
> ss <- 200
> mod.deriv <- z.m[s1[1:ss],1:5] %*% coeff[-1] #the derivation sample
> mod.valid <- z.m[s1[(ss+1):(2*ss)],1:5] %*% coeff[-1] #the validation sample
> round( cor(mod.deriv,z.m[s1[1:ss],]),2)   #derivation

       ID GREV GREQ GREA  Ach   Anx Prelim  GPA   MA
[1,] 0.51 0.64 0.74 0.85 0.38 -0.31   0.39 0.41 0.33

> round(cor(mod.valid,z.m[s1[(ss+1):(2*ss)],]),2) #validation
```

```
        ID GREV GREQ GREA   Ach     Anx  Prelim   GPA    MA
[1,]  0.46  0.67  0.79  0.87  0.31  -0.23    0.49  0.45  0.37
```

Compare this to unit weighting the data. Although subtle, the unit weighted actually has a higher cross validated multiple R when contrasted to the multiple regression.

```
> #the first data set
> all <- rowSums(z.m[s1[1:ss],2:5] ) - z.df[s1[1:ss],6]
> mod.all <- lm(z.m[s1[1:ss],"Prelim"] ~ all)
>  summary(mod.all)

Call:
lm(formula = z.m[s1[1:ss], "Prelim"] ~ all)

Residuals:
     Min        1Q    Median        3Q       Max
-1.87949  -0.50561  -0.04541   0.48038   2.67093

Coefficients:
            Estimate Std. Error t value Pr(>|t|)
(Intercept) -0.01217    0.05641  -0.216    0.829
all          0.12995    0.01805   7.200 1.22e-11 ***
---
Signif. codes:  0 âĂŸ***âĂŹ 0.001 âĂŸ**âĂŹ 0.01 âĂŸ*âĂŹ 0.05 âĂŸ.âĂŹ 0.1 âĂŸ âĂŹ 1

Residual standard error: 0.7976 on 198 degrees of freedom
Multiple R-squared:  0.2075,    Adjusted R-squared:  0.2035
F-statistic: 51.85 on 1 and 198 DF,  p-value: 1.221e-11

> #the replication data set
> all <- rowSums(z.m[s1[(ss+1):(2*ss)],2:5]) - z.m[s1[(ss+1):(2*ss)],6]
> mod.all <- lm(z.m[s1[(ss+1):(2*ss)],'Prelim'] ~ all)
>  summary(mod.all)

Call:
lm(formula = z.m[s1[(ss + 1):(2 * ss)], "Prelim"] ~ all)

Residuals:
     Min        1Q    Median        3Q       Max
-2.39460  -0.57364   0.01889   0.55124   2.28768

Coefficients:
            Estimate Std. Error t value Pr(>|t|)
(Intercept)  0.01239    0.06064   0.204    0.838
all          0.14996    0.01793   8.364 1.07e-14 ***
---
Signif. codes:  0 âĂŸ***âĂŹ 0.001 âĂŸ**âĂŹ 0.01 âĂŸ*âĂŹ 0.05 âĂŸ.âĂŹ 0.1 âĂŸ âĂŹ 1

Residual standard error: 0.8563 on 198 degrees of freedom
Multiple R-squared:  0.2611,    Adjusted R-squared:  0.2573
F-statistic: 69.95 on 1 and 198 DF,  p-value: 1.073e-14

>
```