Multiple Regression
Introduction to Statistics Using R (Psychology 9041B)

Paul Gribble

Winter, 2016

1 Correlation, Regression & Multiple Regression

1.1 Bivariate correlation

The Pearson product-moment correlation coefficient (typically represented by r), assesses the nature and strength of the linear relationship between two continuous variables X and Y.

$$r = \frac{\sum (X - \bar{X})(Y - \bar{Y})}{\sqrt{\sum (X - \bar{X})^2 \sum (Y - \bar{Y})^2}} \tag{1}$$

The quantity r^2 represents the proportion of variance shared by the two variables.

To compute r using `R` is simple using the function `cor()`:

```
> X <- c(1,3,4,3,5,4,4,6,7,8)
> Y <- c(3,2,5,6,5,8,7,6,9,10)
> r <- cor(X,Y)
> r
```

```
[1] 0.7798415
```

We can perform significance tests on r. The null hypothesis is that $r = 0$. Note that the alternate hypothesis is simply that $r \neq 0$, not that r is large. The significance test of r simply reports the probability of getting a value of r as extreme as the one observed, if we were to randomly draw two samples from two populations that were not in fact correlated ($r_{pop1,pop2} = 0$).

We can perform an F-test with $df = (1, N - 2)$:

$$F = \frac{r^2(N-2)}{1-r^2} \tag{2}$$

For the `R` example above:

```
> N <- length(X)
> Fobs <- (r*r*(N-2)) / (1-(r*r))
> pobs <- 1 - pf(Fobs, 1, N-2)
> Fobs
```

```
[1] 12.41612

> pobs

[1] 0.007804102
```

In this case $p < 0.05$ so we reject the null hypothesis that $r = 0$, and conclude simply that there is a reliable non-zero correlation between X and Y.

There is an even faster way of performing the significance tests, on the entire correlation matrix, using the `rcorr()` function, which is part of the `Hmisc` package. You will have to first install the package (once only) using the `install.packages()` function, then tell `R` to use the package using the `library()` function. The `rcorr()` function requires a matrix as input, so we use the `as.matrix()` function to convert the data frame `bdata` to a matrix.

```
> # install.packages("Hmisc") # you only need to do this once on a given computer
> library(Hmisc) # you will need to do this once within a given R session
> fname <- "http://www.gribblelab.org/stats/data/bball.csv"
> bdata <- read.table(fname, sep=",", header=TRUE)
> summary(bdata)

     GAMES            PPM               MPG              HGT       
 Min.   : 8.00   Min.   :0.1600   Min.   : 4.35   Min.   :160.0  
 1st Qu.:51.00   1st Qu.:0.3300   1st Qu.:17.00   1st Qu.:185.0  
 Median :73.00   Median :0.4200   Median :24.28   Median :191.0  
 Mean   :64.15   Mean   :0.4236   Mean   :24.30   Mean   :190.2  
 3rd Qu.:79.00   3rd Qu.:0.4900   3rd Qu.:33.54   3rd Qu.:196.0  
 Max.   :82.00   Max.   :0.8300   Max.   :40.71   Max.   :210.0  
      FGP             AGE             FTP       
 Min.   :34.80   Min.   :22.00   Min.   :37.50  
 1st Qu.:42.80   1st Qu.:25.00   1st Qu.:74.20  
 Median :45.30   Median :27.00   Median :79.30  
 Mean   :45.13   Mean   :27.53   Mean   :77.86  
 3rd Qu.:47.80   3rd Qu.:30.00   3rd Qu.:84.00  
 Max.   :59.50   Max.   :37.00   Max.   :94.80  

> rcorr(as.matrix(bdata))

      GAMES   PPM   MPG   HGT   FGP   AGE   FTP
GAMES  1.00 -0.06  0.52 -0.17  0.19  0.16  0.31
PPM   -0.06  1.00  0.36  0.21  0.41 -0.04  0.17
MPG    0.52  0.36  1.00 -0.01  0.34  0.18  0.39
HGT   -0.17  0.21 -0.01  1.00 -0.11  0.07 -0.06
FGP    0.19  0.41  0.34 -0.11  1.00  0.11  0.28
AGE    0.16 -0.04  0.18  0.07  0.11  1.00  0.25
FTP    0.31  0.17  0.39 -0.06  0.28  0.25  1.00
```

```
n= 105


P
      GAMES  PPM    MPG    HGT    FGP    AGE    FTP   
GAMES        0.5444 0.0000 0.0799 0.0489 0.1138 0.0013
PPM   0.5444        0.0002 0.0289 0.0000 0.6544 0.0915
MPG   0.0000 0.0002        0.9158 0.0004 0.0659 0.0000
HGT   0.0799 0.0289 0.9158        0.2726 0.4782 0.5342
FGP   0.0489 0.0000 0.0004 0.2726        0.2711 0.0040
AGE   0.1138 0.6544 0.0659 0.4782 0.2711        0.0110
FTP   0.0013 0.0915 0.0000 0.5342 0.0040 0.0110       
```

Note that if N is large, we can achieve a low probability on the significance test of r, even with a small value of r. For example we might observe a value of $r = 0.20$, but because we have $N = 100$, we end up with $Fobs = 4.08$ and $p < 0.05$. In this case $r^2 = 0.04$, which means X and Y only share 4 % of their variance in common ... but the significance test shows the correlation is reliably non-zero. It doesn't mean it's large, it's just not zero. Be careful about the use of the word "significant" — it's doesn't mean a correlation is significant in the everyday use of the word, rather it means only that the correlation is not zero.

1.2 Bivariate regression

In bivariate regression, we have two continuous variables X and Y. The variable Y is considered to be dependent on X. The goal is to characterize the (linear) relationship between X and Y, and in particular, to predict a value of Y given a value of X. For example if Y is a person's weight, and X is a person's height, then we model the (linear) relationship between these two variables using the following equation:

$$\hat{Y_i} = \beta_0 + \beta_1 X_i \tag{3}$$

where $\hat{Y_i}$ is the estimated value of Y for the ith subject, β_0 is the intercept (the value of Y when X is zero), and β_1 is the slope of the relationship between X and Y — the predicted increase in Y given a unit increase in X. Note that this is the equation for a straight line with intercept β_0 and slope β_1.

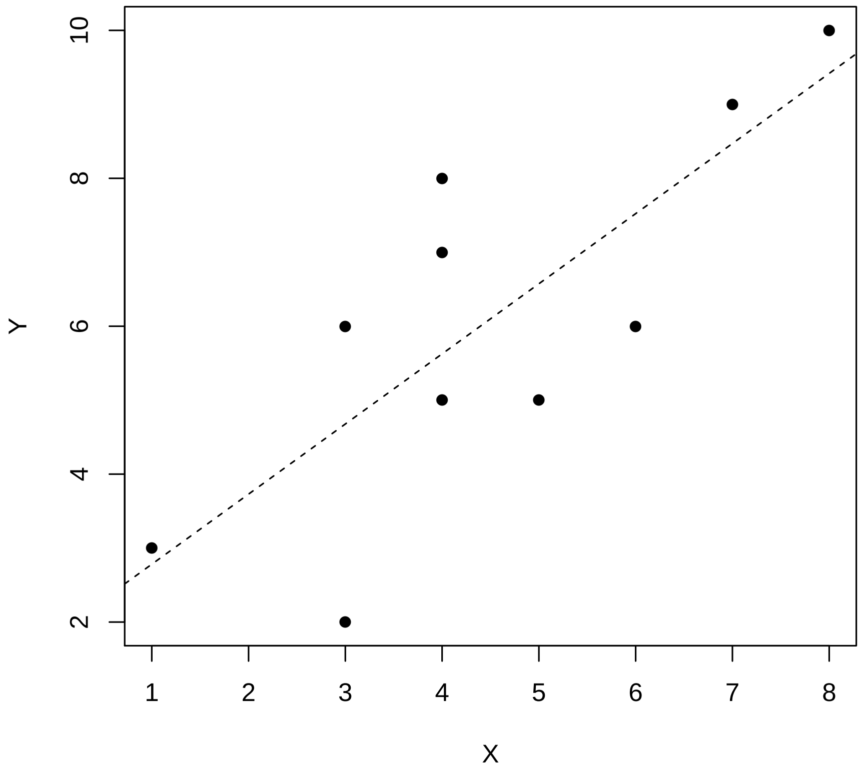
For bivariate regression we can estimate β_0 and β_1 easily using equations that correspond to least-squares estimates:

$$\begin{aligned} \beta_1 &= \frac{\sum (X - \bar{X})(Y - \bar{Y})}{\sum (X - \bar{X})^2} && (4) \\ \beta_0 &= \bar{Y} - \beta_1 \bar{X} && (5) \end{aligned}$$

This gives values of β_0 and β_1 that minimize the sum of squared deviations of the estimated values of Y (the line of best fit) and the observed values of Y (the data).

In `R` we use the `lm()` function to fit a linear model to continuous data:

```
> m1 <- lm(Y ~ X)
> summary(m1)

Call:
lm(formula = Y ~ X)

Residuals:
   Min     1Q Median     3Q    Max
-2.678 -1.298  0.374  1.137  2.374

Coefficients:
            Estimate Std. Error t value Pr(>|t|)
(Intercept)   1.8338     1.3208   1.388   0.2025
X             0.9481     0.2691   3.524   0.0078 **
---
Signif. codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1

Residual standard error: 1.669 on 8 degrees of freedom
Multiple R-squared:  0.6082,        Adjusted R-squared:  0.5592
F-statistic: 12.42 on 1 and 8 DF,  p-value: 0.007804

> plot(X,Y,pch=16)
> abline(m1,lty=2)
```



We can read from the output, the estimates of β_0 and β_1 (in the column of the output labelled `Estimate` — `(Intercept)` is β_0 and `X` is β_1. We also get a t-test on each parameter. These are significance tests of the null hypothesis that the population value of the parameter of interest is zero.

To assess how well our model fits the data, we can compute the "standard error of estimate", which gives a measure of the typical prediction error, in the same units of Y:

$$se = \sqrt{\frac{\sum (Y - \hat{Y})^2}{N - 2}} \tag{6}$$

In `R` it's a simple matter of reading it from the output of `summary.lm()` — in the above example, $se = 1.669$.

We can also get confidence intervals on the predicted values of Y:

```
> ypred <- predict(m1, data.frame(X=3.0), interval="prediction", level=0.95)
> ypred

       fit       lwr      upr
1 4.677922 0.5344288 8.821415
```

Obviously the standard error of the estimate as well as the confidence interval depends on sample size N.

Another measure of fit is r^2, which gives the proportion of variance in Y accounted for by variation in X.

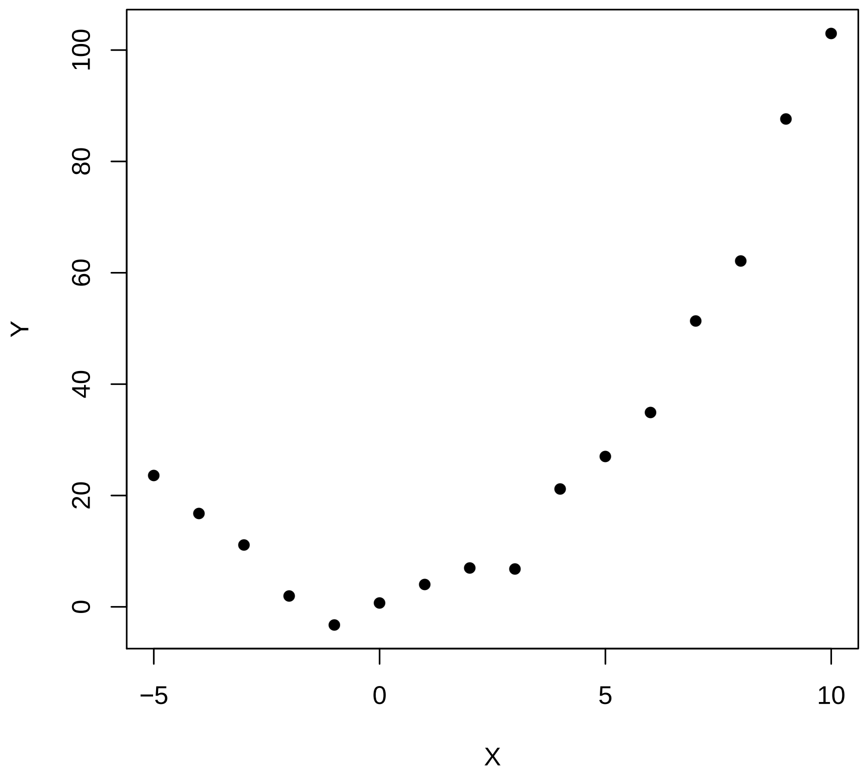
$$r^2 = \frac{\sum(\hat{Y} - \bar{Y})^2}{\sum(Y - \bar{Y})^2} \tag{7}$$

In `R` we can read off the value of r^2 directly from the output of `summary.lm()`.

1.3 Linear regression with non-linear terms

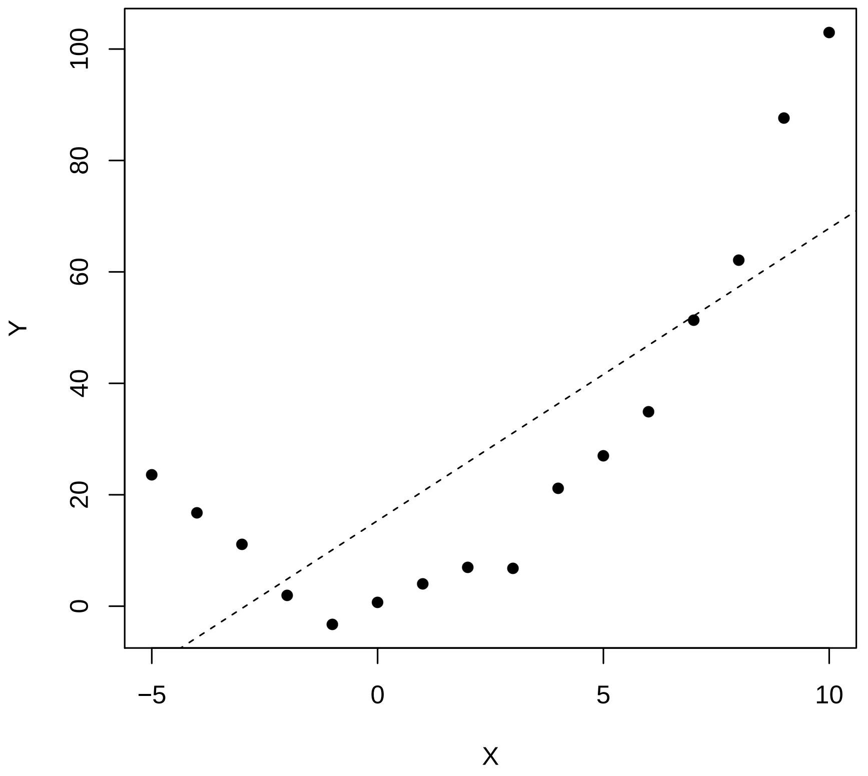
We can include non-linear *terms* in a linear model. This may be appropriate in cases when you are trying to model some forms of non-linear relationships, e.g. that shown below:

```
> X <- c(-5,-4,-3,-2,-1,0,1,2,3,4,5,6,7,8,9,10)
> Y <- X^2 + rnorm(16,0,3)
> plot(X,Y,pch=16)
```



A linear fit would produce the following:

```
> m2 <- lm(Y ~ X)
> abline(m2, lty=2)
```



We can attempt to fit the data using an equation of the form:

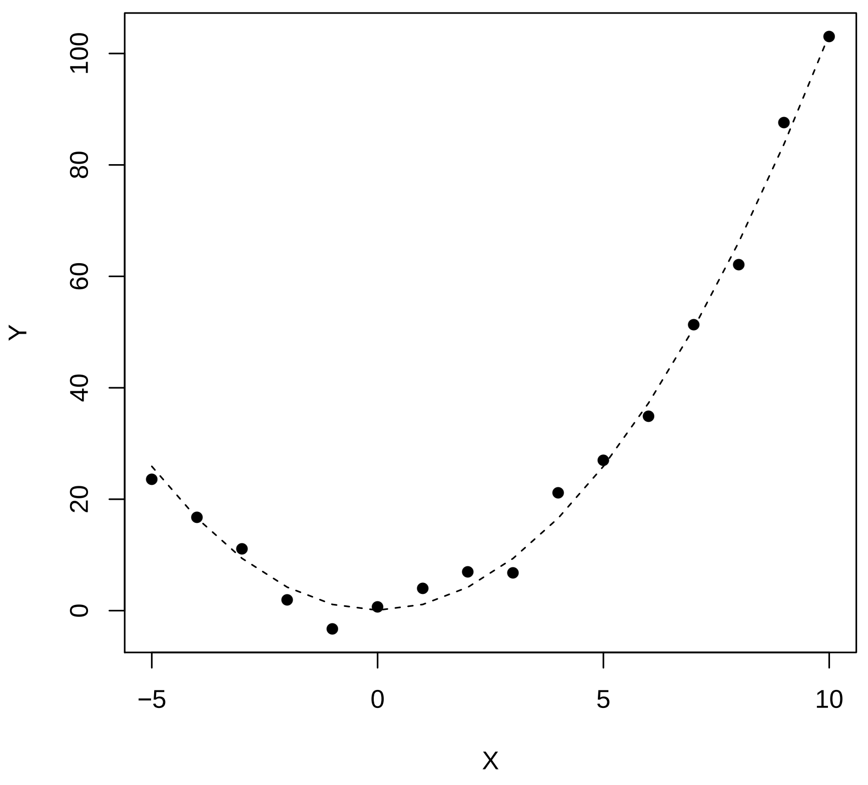
$$Y = \beta_0 + \beta_1 X^2 \tag{8}$$

In this case the model equation is still linear *in betas* even though we are predicting based on X^2 instead of X.

To do this in R simply create a new variable representing X^2:

```
> Xsquared <- X*X
> m3 <- lm(Y ~ Xsquared)
> summary(m3)

Call:
lm(formula = Y ~ Xsquared)

Residuals:
    Min      1Q  Median      3Q     Max
```

```
-4.3621 -2.3101  0.3834  1.9751  4.5874

Coefficients:
            Estimate Std. Error t value Pr(>|t|)
(Intercept)  0.07866    0.97716   0.081    0.937
Xsquared     1.03292    0.02410  42.867 2.97e-16 ***
---
Signif. codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1

Residual standard error: 2.873 on 14 degrees of freedom
Multiple R-squared:  0.9924,        Adjusted R-squared:  0.9919
F-statistic:  1838 on 1 and 14 DF,  p-value: 2.97e-16

> yfit <- predict(m3,data.frame(X=Xsquared))
> lines(X,yfit,lty=2)
```



You can see we have achieved a much better fit.

1.4 Multiple regression

In multiple regression the idea is to predict a dependent variable Y based on a linear combination of independent variables X_1 - X_k, where $k > 1$:

$$\hat{Y} = \beta_0 + \beta_1 X_1 + \beta_2 X_2 + \cdots + \beta_k X_k \tag{9}$$

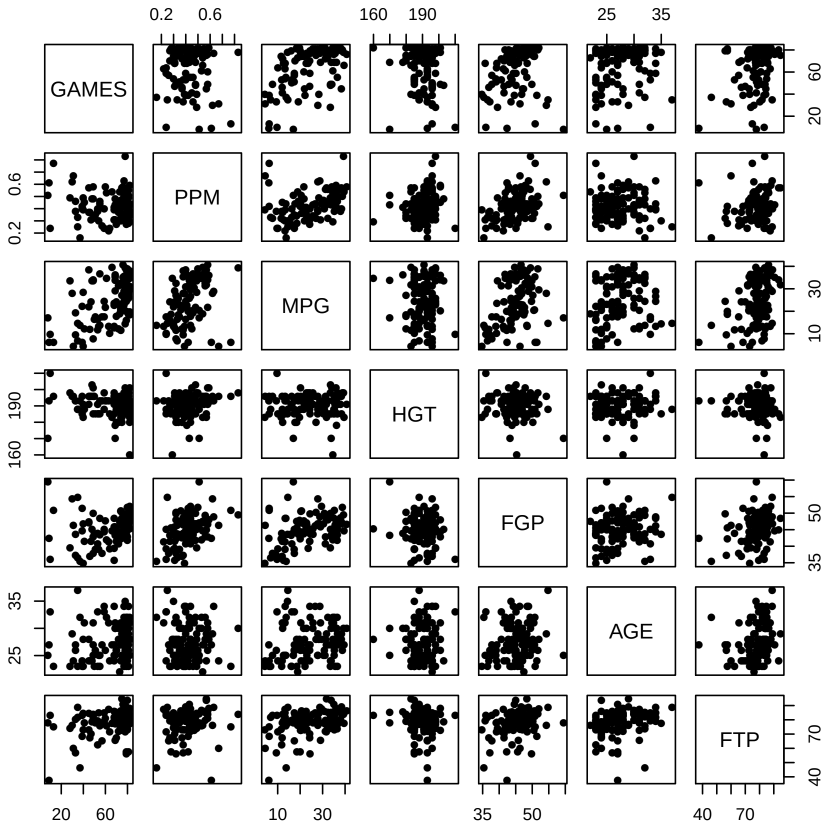
The coefficients are computed according to a least-squares criterion, in order to minimize the sum of squared differences between the predicted values of Y, $\hat{Y}$, and the observed values of Y.

For the purposes of illustration, we'll work with a dataset containing measures on 7 variables from 105 basketball players:

`GAMES`	# games played in previous season
`PPM`	average points scored per minute
`MPG`	average minutes played per game
`HGT`	height of player (centimetres)
`FGP`	field-goal percentage (% successful shots from 3-point line)
`AGE`	age of player (years)
`FTP`	percentage of successful penalty free throws

After loading the data into a data frame, we can use the `pairs()` function to quickly visualize scatterplots of all variables plotted against each other:

```
> fname <- "http://www.gribblelab.org/stats/data/bball.csv"
> bdata <- read.table(fname,sep=",",header=TRUE)
> pairs(bdata, pch=16)
```



For the purpose of this demonstration, let's assume we want to develop a model that predicts a players free throw percentage `FTP` based on the other 6 variables. We can ask several questions of the data:

- What is the best single predictor of `FTP`?
- What is the best set of variables (best model) to predict `FTP`?
- Does a certain variable add significantly to the predictive power of the model?

To determine the single best predictor, we can simply compute the correlation coefficient r for each variable paired with `FTP`, and pick the one with the highest (absolute) value. First we'll compute a correlation matrix, i.e. the correlation of every variable with every other variable; then we'll pick the row of interest (the correlations of `FTP` with the other 6 variables); then we'll throw out the entry corresponding to the correlation of `FTP` with itself, since that will be equal to 1:

```
> r <- cor(bdata)
> r <- r[7,-7]
> r
```

```
     GAMES         PPM         MPG         HGT         FGP         AGE
 0.31067275  0.16552229  0.39141431 -0.06133401  0.27855246  0.24735655
```

In this case the variable most correlated with `FTP` is `MPG`, the average number of minutes player per game.

We can perform significance tests on each correlation coefficient; the code involving `round()` is simply to show the values with three decimal places of precision — it's easier to read.

```
> N <- nrow(bdata)
> # compute F statistic and then p value
> F <- ((r^2)*(N-2))/(1-(r^2))
> p <- 1-pf(F,1,N-2)
> p
```

```
       GAMES          PPM          MPG          HGT          FGP          AGE
1.257318e-03 9.151557e-02 3.647017e-05 5.342385e-01 4.010633e-03 1.095996e-02
```

```
> round(p*1000)/1000
```

```
GAMES   PPM   MPG   HGT   FGP   AGE
0.001 0.092 0.000 0.534 0.004 0.011
```

We can see that with an alpha level set at 0.05, two variables, `HGT` and `PPM` are not significantly correlated with `FTP`, but the others are. Once again, be careful: this simply means that for `GAMES`, `MPG`, `FGP` and `AGE`, the correlation is not zero — it doesn't mean it's large.

Now it's time to develop a model — in other words, what's the best model to predict `FTP`? Certainly a model that contains all variables will have the lowest prediction error ... but our goal should be to balance prediction precision with parsimony — in other words, we want the minimum number of variables necessary to achieve a reasonable predictive power. We could include all available variables, but some of them will not be accounting for a significant unique portion of the total variance in the dependent variable Y. Another way of thinking about this is, with each additional independent variable X included in the model, we lower the prediction error, but at a cost of one degree of freedom.

There are several methods for developing the best model, taking account of the tradeoff between model precision and model parsimony. A good method to use is called stepwise regression. The procedure is a mixture of *add*ing and *drop*ping IVs, and proceeds as follows:

1. Start with an empty model, with no independent variables (IVs).

2. Check to see if any of the available IVs significantly predict the dependent variable (DV).

3. If no, stop. If yes, **add** the best one, and go to step 4.

4. Check to see if any of the remaining IVs account for a *unique* proportion of variance in Y, above and beyond what's already accounted for by the IVs already in the model.

5. If no, stop. If yes, **add** the best one and go to step 6.
6. Check that each IV currently in the model still accounts for a significant unique proportion of variance in Y, above and beyond what's accounted for by the other variables in the model, and **drop** any variables that don't.
7. go to step 4

The stepwise procedure is designed to ensure that at the end of the procedure you end up with a model in which all IVs included in the model account for a significant unique proportion of variance in Y, above and beyond what's accounted for by the other variables in the model.

In `R` the `step()` function performs stepwise regression. The `step()` function requires us to pass as parameters a starting model (in our case it will be an empty model containing no IVs), a list containing `lower` and `upper` parameters corresponding to the minimum and maximum models to be tested, and a `direction` parameter that we will pass `"both"`, to signify that we want `step()` to both add and drop IVs as necessary.

```
> m0 <- lm(FTP ~ 1, data=bdata)
> mall <- lm(FTP ~ GAMES + MPG + HGT + PPM + AGE + FGP, data=bdata)
> mbest <- step(m0, list(lower=m0, upper=mall), direction="both")

Start:  AIC=466.49
FTP ~ 1

        Df Sum of Sq    RSS    AIC
+ MPG    1   1341.70 7415.8 451.03
+ GAMES  1    845.26 7912.3 457.83
+ FGP    1    679.51 8078.0 460.01
+ AGE    1    535.83 8221.7 461.86
+ PPM    1    239.94 8517.6 465.57
<none>               8757.5 466.49
+ HGT    1     32.94 8724.6 468.09

Step:  AIC=451.03
FTP ~ MPG

        Df Sum of Sq    RSS    AIC
+ AGE    1    283.08 7132.8 448.94
+ FGP    1    209.93 7205.9 450.01
<none>               7415.8 451.03
+ GAMES  1    135.11 7280.7 451.10
+ HGT    1     28.71 7387.1 452.62
+ PPM    1      6.83 7409.0 452.93
- MPG    1   1341.70 8757.5 466.49
```

```
Step:  AIC=448.94
FTP ~ MPG + AGE

        Df Sum of Sq    RSS    AIC
+ FGP    1    186.27 6946.5 448.16
<none>               7132.8 448.94
+ GAMES  1    108.73 7024.0 449.33
+ HGT    1     43.62 7089.1 450.30
+ PPM    1     21.43 7111.3 450.63
- AGE    1    283.08 7415.8 451.03
- MPG    1   1088.95 8221.7 461.86

Step:  AIC=448.16
FTP ~ MPG + AGE + FGP

        Df Sum of Sq    RSS    AIC
<none>               6946.5 448.16
+ GAMES  1    104.52 6842.0 448.57
- FGP    1    186.27 7132.8 448.94
+ HGT    1     25.65 6920.8 449.77
- AGE    1    259.42 7205.9 450.01
+ PPM    1      0.01 6946.5 450.16
- MPG    1    713.63 7660.1 456.43

> summary(mbest)

Call:
lm(formula = FTP ~ MPG + AGE + FGP, data = bdata)

Residuals:
    Min      1Q  Median      3Q     Max
-34.013  -3.349   1.519   4.704  14.636

Coefficients:
            Estimate Std. Error t value Pr(>|t|)
(Intercept) 43.12593   10.34628   4.168 6.49e-05 ***
MPG          0.28590    0.08876   3.221  0.00172 **
AGE          0.47727    0.24575   1.942  0.05490 .
FGP          0.32438    0.19711   1.646  0.10294
---
Signif. codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1

Residual standard error: 8.293 on 101 degrees of freedom
Multiple R-squared:  0.2068,	Adjusted R-squared:  0.1832
F-statistic: 8.777 on 3 and 101 DF,  p-value: 3.157e-05
```

The `step()` function outputs each step in the stepwise regression. We end up in this case with a linear model containing `MPG + AGE + FGP`.

Note that in the `step()` function, the question of whether an IV accounts for a unique proportion of variance, above and beyond the other variables in the model, is assessed using the Akaike Information Criterion (AIC). This is simply a measure of the goodness of fit of a statistical model, one that is grounded in the concept of entropy. It essentially provides a relative measure of the information lost when a given model is used to describe data, and accounts for the tradeoff between bias and variance in model construction (in other words, precision vs complexity of the model). Lower values of AIC correspond to a better model.

Note that the default test in SPSS does not use AIC, but instead performs an F-test. The idea is the same; compare two models, one without the IV of interest, and one with the IV of interest, and perform an F-test to see if the additional variance accounted for is "worth" giving up one degree of freedom. We can perform tests like this in `R` using the `add1()` and `drop1()` functions. The `add1()` function performs F-tests for adding each variable, individually, to a given model; the `drop1()` function performs F-tests for dropping each variable, individually, from a given model.

1.5 Stepwise Regression using F tests, by hand

Let's perform a stepwise regression manually, using F-tests:

```
> m0 <- lm(FTP ~ 1,data=bdata)
> mall <- lm(FTP ~ GAMES + MPG + HGT + PPM + AGE + FGP, data=bdata)
> # start with nothing in the model - which IV to add?
> add1(m0, formula(mall), test="F")

Single term additions

Model:
FTP ~ 1
       Df Sum of Sq    RSS    AIC F value    Pr(>F)
<none>              8757.5 466.49
GAMES   1    845.26 7912.3 457.83 11.0033  0.001257 **
MPG     1   1341.70 7415.8 451.03 18.6351 3.647e-05 ***
HGT     1     32.94 8724.6 468.09  0.3889  0.534239
PPM     1    239.94 8517.6 465.57  2.9014  0.091516 .
AGE     1    535.83 8221.7 461.86  6.7128  0.010960 *
FGP     1    679.51 8078.0 460.01  8.6642  0.004011 **
---
Signif. codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1

> # check the output of the add1() tests, and add the best single one
> mbest <- lm(FTP ~ 1 + MPG, data=bdata)
> # which to add next?
> add1(mbest, formula(mall), test="F")
```

```
Single term additions

Model:
FTP ~ 1 + MPG
       Df Sum of Sq    RSS    AIC F value  Pr(>F)
<none>              7415.8 451.03
GAMES   1   135.106 7280.7 451.10  1.8928 0.17190
HGT     1    28.706 7387.1 452.62  0.3964 0.53038
PPM     1     6.830 7409.0 452.93  0.0940 0.75975
AGE     1   283.083 7132.8 448.94  4.0481 0.04686 *
FGP     1   209.928 7205.9 450.01  2.9715 0.08777 .
---
Signif. codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1

> # check the output of the add1() tests, and add the best single one
> mbest <- lm(FTP ~ 1 + MPG + AGE, data=bdata)
> # now do a check for unique variance, drop those who don't contribute
> drop1(mbest,test="F")

Single term deletions

Model:
FTP ~ 1 + MPG + AGE
       Df Sum of Sq    RSS    AIC F value    Pr(>F)
<none>              7132.8 448.94
MPG     1   1088.95 8221.7 461.86 15.5722 0.0001461 ***
AGE     1    283.08 7415.8 451.03  4.0481 0.0468567 *
---
Signif. codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1

> # all still contribute unique variance so we keep them all in the model
> # which one to add next?
> add1(mbest, formula(mall), test="F")

Single term additions

Model:
FTP ~ 1 + MPG + AGE
       Df Sum of Sq    RSS    AIC F value Pr(>F)
<none>              7132.8 448.94
GAMES   1   108.732 7024.0 449.33  1.5635 0.2140
HGT     1    43.617 7089.1 450.30  0.6214 0.4324
PPM     1    21.430 7111.3 450.63  0.3044 0.5824
FGP     1   186.268 6946.5 448.16  2.7083 0.1029

> # none are significant any more, we stop
> # (FGP would be the next best one, but p = 0.1029, not < 0.05)
```

Note that sometimes people use a different alpha level for adding IVs versus dropping IVs — e.g. the default in SPSS is to use $p < 0.05$ for entering a variable into the model, but $p > 0.10$ for dropping a variable from the model.

We can then, as before with bivariate regression, generate a prediction for a value of Y given values of X_1 - $X)k$. Let's say we have a new basketball player, and we want to predict, what will his free throw percentage be, given that we know values on 6 other IVs:

```
> newRecruit <- data.frame(GAMES=64, PPM=0.4, MPG=20, HGT=200, FGP=60, AGE=23)
> newFTP <- predict(mbest, newRecruit, se.fit=TRUE, interval="prediction", level=0.95)
> newFTP

$fit
      fit      lwr      upr
1 74.1619 57.35022 90.97358

$se.fit
[1] 1.382007

$df
[1] 102

$residual.scale
[1] 8.362355
```

We see that the predicted `FTP` is 74.1619, and that the 95 % confidence interval on the prediction is 57.35022 - 90.97358.

1.5.1 Stepwise Regression using F tests instead of AIC

We can ask the `step()` function in `R` to use F tests instead of the AIC criterion to add and drop terms in the stepwise regression, like this:

```
> step(m0, list(lower=m0, upper=mall), direction="both", test="F")

Start:  AIC=466.49
FTP ~ 1

        Df Sum of Sq    RSS    AIC F value    Pr(>F)
+ MPG    1   1341.70 7415.8 451.03 18.6351 3.647e-05 ***
+ GAMES  1    845.26 7912.3 457.83 11.0033  0.001257 **
+ FGP    1    679.51 8078.0 460.01  8.6642  0.004011 **
+ AGE    1    535.83 8221.7 461.86  6.7128  0.010960 *
+ PPM    1    239.94 8517.6 465.57  2.9014  0.091516 .
<none>                  8757.5 466.49
+ HGT    1     32.94 8724.6 468.09  0.3889  0.534239
---
```

```
Signif. codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1

Step:  AIC=451.03
FTP ~ MPG

        Df Sum of Sq    RSS    AIC F value    Pr(>F)
+ AGE    1    283.08 7132.8 448.94  4.0481   0.04686 *
+ FGP    1    209.93 7205.9 450.01  2.9715   0.08777 .
<none>               7415.8 451.03
+ GAMES  1    135.11 7280.7 451.10  1.8928   0.17190
+ HGT    1     28.71 7387.1 452.62  0.3964   0.53038
+ PPM    1      6.83 7409.0 452.93  0.0940   0.75975
- MPG    1   1341.70 8757.5 466.49 18.6351 3.647e-05 ***
---
Signif. codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1

Step:  AIC=448.94
FTP ~ MPG + AGE

        Df Sum of Sq    RSS    AIC F value    Pr(>F)
+ FGP    1    186.27 6946.5 448.16  2.7083 0.1029364
<none>               7132.8 448.94
+ GAMES  1    108.73 7024.0 449.33  1.5635 0.2140460
+ HGT    1     43.62 7089.1 450.30  0.6214 0.4323649
+ PPM    1     21.43 7111.3 450.63  0.3044 0.5823754
- AGE    1    283.08 7415.8 451.03  4.0481 0.0468567 *
- MPG    1   1088.95 8221.7 461.86 15.5722 0.0001461 ***
---
Signif. codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1

Step:  AIC=448.16
FTP ~ MPG + AGE + FGP

        Df Sum of Sq    RSS    AIC F value   Pr(>F)
<none>               6946.5 448.16
+ GAMES  1    104.52 6842.0 448.57  1.5276 0.219374
- FGP    1    186.27 7132.8 448.94  2.7083 0.102936
+ HGT    1     25.65 6920.8 449.77  0.3706 0.544061
- AGE    1    259.42 7205.9 450.01  3.7719 0.054905 .
+ PPM    1      0.01 6946.5 450.16  0.0001 0.992058
- MPG    1    713.63 7660.1 456.43 10.3759 0.001718 **
---
Signif. codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1

Call:
```

```
lm(formula = FTP ~ MPG + AGE + FGP, data = bdata)

Coefficients:
(Intercept)          MPG          AGE          FGP
    43.1259       0.2859       0.4773       0.3244
```