Modelling

Linear Regression

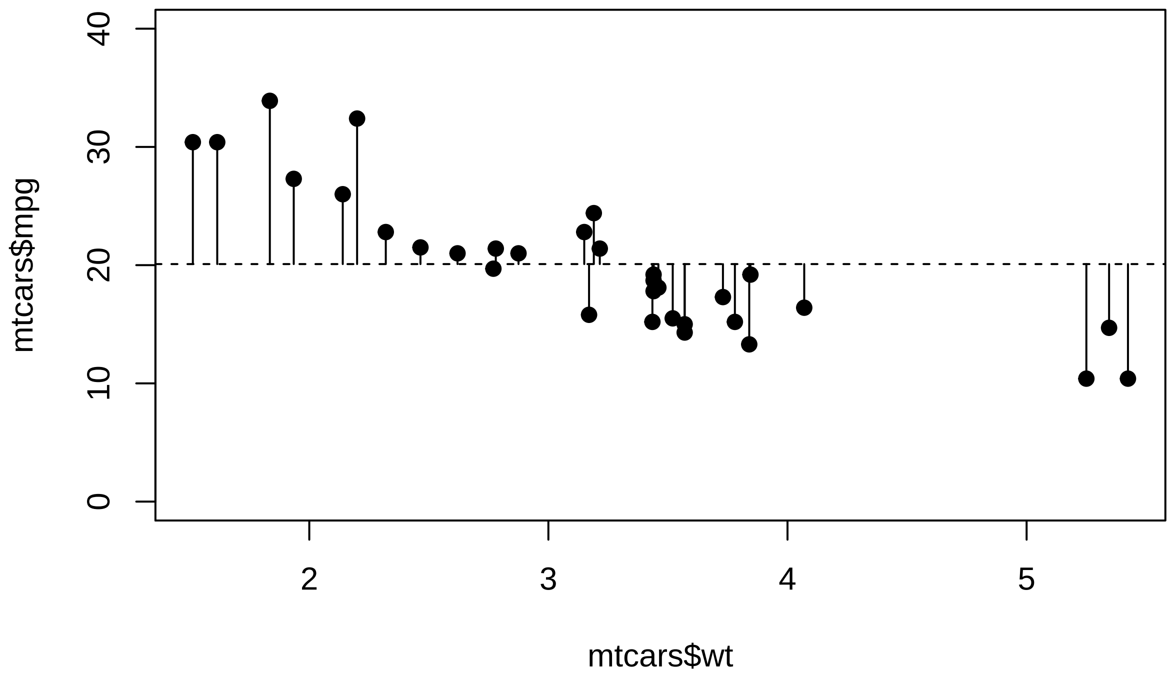
We'll start this lecture based on one of the more popular forms of modeling called Linear Regression wherein we predict the value of some outcome (like MPG from mtcars) using predictor variables from the mtcars data frame. We get an equation like Y = a + Bx + e where "e" represent some error terms, "a" is an intercept and "B"" represents a regression coefficient.

Sometimes the equation is written like Y = B0 + B1*x1 + B2*x2 . . . + B3*x3 to indicate the use of multiple predictor variables from the data set. The x values are variables from the data set and the B values are coefficients computed from solving the "least squares equation".

Background

With respect to mtcars we could plot the MPG variable as a function of a car's weight. This looks close to being a linear relationship. Close enough that we could try creating a linear model perhaps. As a first cut model we could simply estimate the MPG of any new observations by using the existing mean of the MPG from the current data. Visually this looks like:

```
plot(mtcars$wt,mtcars$mpg,pch=19,ylim=c(0,40))

abline(h=mean(mtcars$mpg),lty=2)

mpgmn <- mean(mtcars$mpg)

segments(mtcars[,'wt'],mpgmn,mtcars[,'wt'],mtcars[,'mpg'])
```



```r
denom <- sum( (mtcars$mpg - mean(mtcars$mpg))^2 )
```

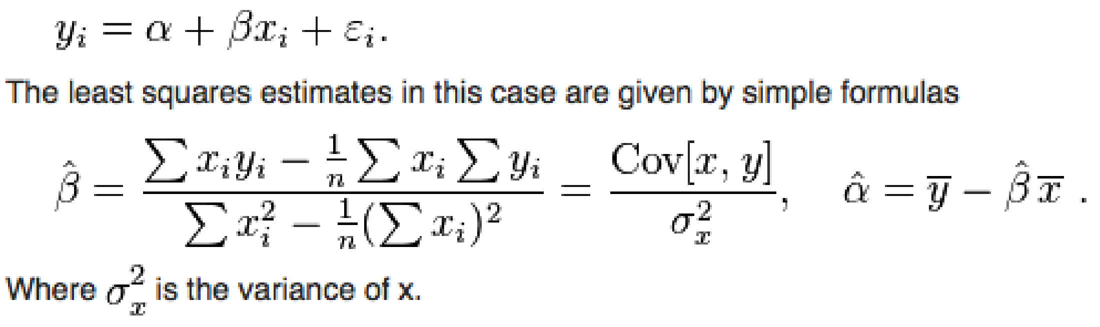
Then we could take the sum of the squares of the differences between the points and the baseline which is one measure of variation. The problem with this is that for every incoming new observation we will predict it's respective MPG as the mean of the existing data. Not only is this likely not to be a good model it is boring. We might try to come up with our own line designed to "fit" the data. The goal would be to try to find a line that minimizes the distance between the points and the line we come up with. We know that because it is a straight line it can't possibly "touch" all the points. (We could try a quadratic equation but it coould have similar problems).

The following is an attempt at "eyeballing" and appropriate line or two although I'm not using any guiding mathematical ideas except perhaps to guess that the line is of the form y = -x or some variation thereof.

See https://en.wikipedia.org/wiki/Ordinary_least_squares

See http://setosa.io/ev/ordinary-least-squares-regression/

In reality the "best" line is he one that satifies the "Ordinary Least Squares" equation.



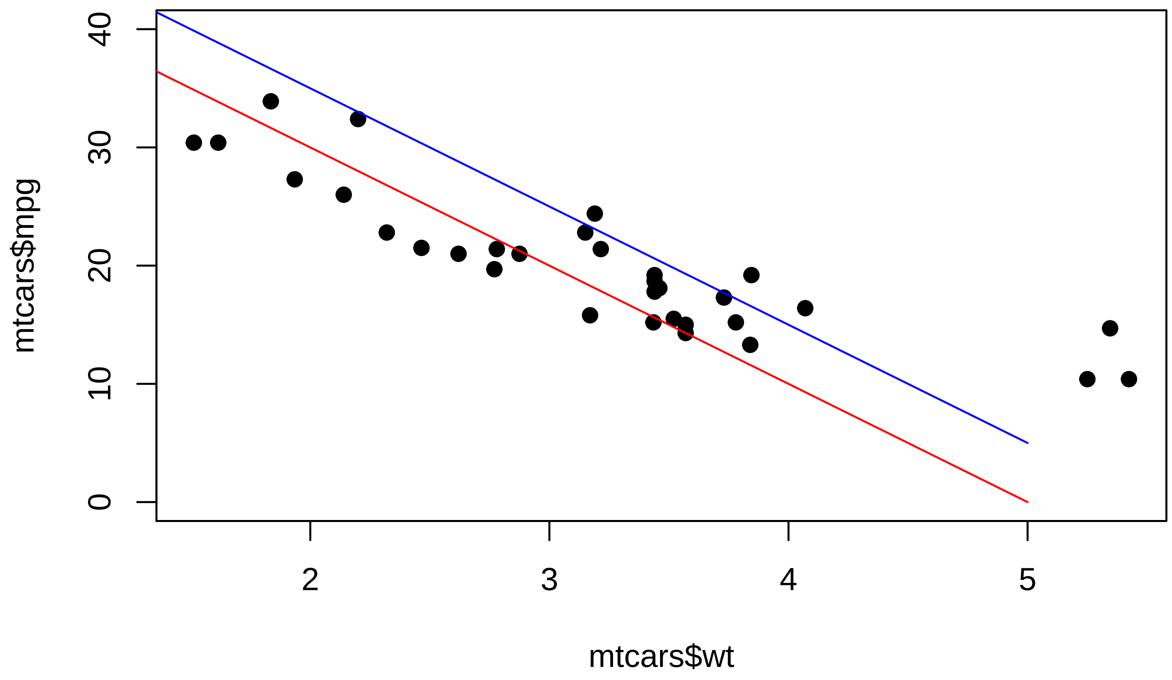
$$y_i = \alpha + \beta x_i + \varepsilon_i .$$

The least squares estimates in this case are given by simple formulas

$$\hat{\beta} = \frac{\sum x_i y_i - \frac{1}{n}\sum x_i \sum y_i}{\sum x_i^2 - \frac{1}{n}(\sum x_i)^2} = \frac{\mathrm{Cov}[x,y]}{\sigma_x^2}, \quad \hat{\alpha} = \overline{y} - \hat{\beta}\,\overline{x} .$$

Where σ_x^2 is the variance of x.

```r
plot(mtcars$wt,mtcars$mpg,pch=19,ylim=c(0,40))

lines(1:5,seq(40,0,-10),col="red")

lines(1:5,seq(45,0,-10),col="blue")
```



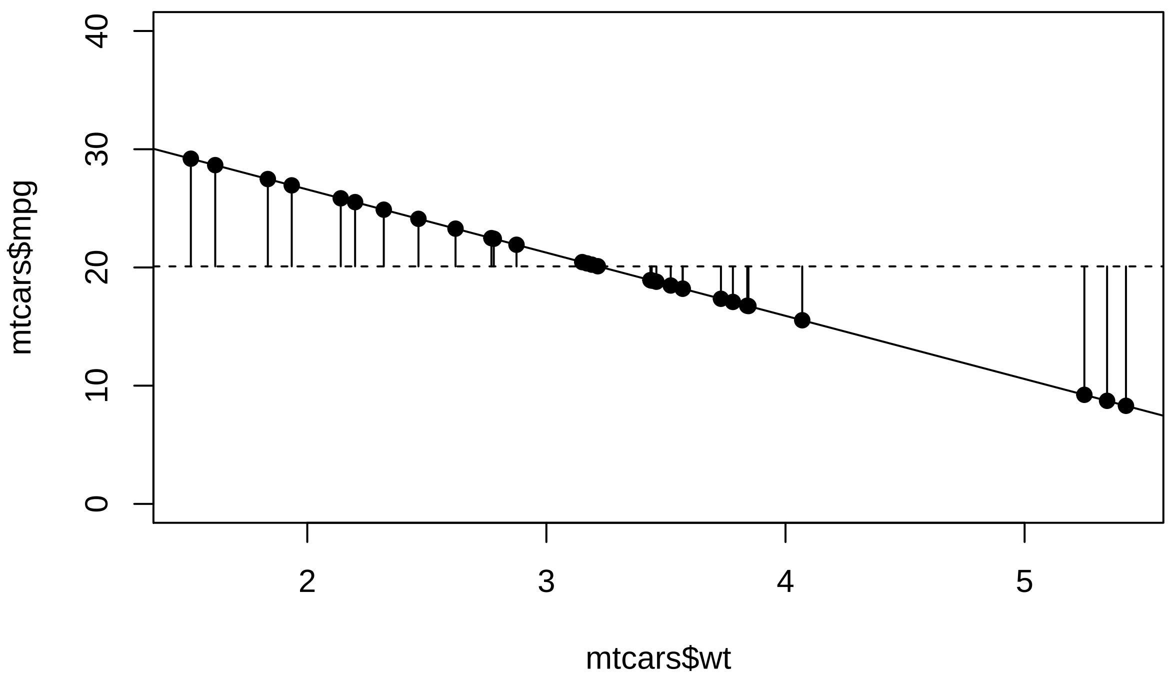
Even though we have now found the theoretical "best" line through these points we should compare the measure of variation between this line and the baseline.

```
mylm <- lm(mpg ~ wt, data=mtcars)

summary(mylm)
```

```
##
## Call:
## lm(formula = mpg ~ wt, data = mtcars)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -4.5432 -2.3647 -0.1252  1.4096  6.8727
##
## Coefficients:
##             Estimate Std. Error t value Pr(>|t|)
## (Intercept)  37.2851     1.8776  19.858  < 2e-16 ***
## wt           -5.3445     0.5591  -9.559 1.29e-10 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 3.046 on 30 degrees of freedom
## Multiple R-squared:  0.7528, Adjusted R-squared:  0.7446
## F-statistic: 91.38 on 1 and 30 DF,  p-value: 1.294e-10
```

```
plot(mtcars$wt,mtcars$mpg,pch=19,type="n",ylim=c(0,40))
points(mtcars$wt,mylm$fitted.values,pch=19)

abline(mylm)

abline(h=mean(mtcars$mpg),lty=2)
```

```
segments(mtcars[,'wt'],mean(mtcars$mpg),mtcars[,'wt'],mylm$fitted.values)
```



```
numerator <- sum( (mylm$fitted.values - mean(mtcars$mpg))^2 )
```

Once we do this we can look at the ratio of the numerator to denominator and this gives us what is known as the R squared that is many times used to judge the quality of a model. In reality we usually use something called the adjusted R-squared value but the R-squared concept is an important concept in general.

```
(r.squared <- numerator/denom)
```

```
## [1] 0.7528328
```

```
summary(mylm)$r.squared
```

```
## [1] 0.7528328
```

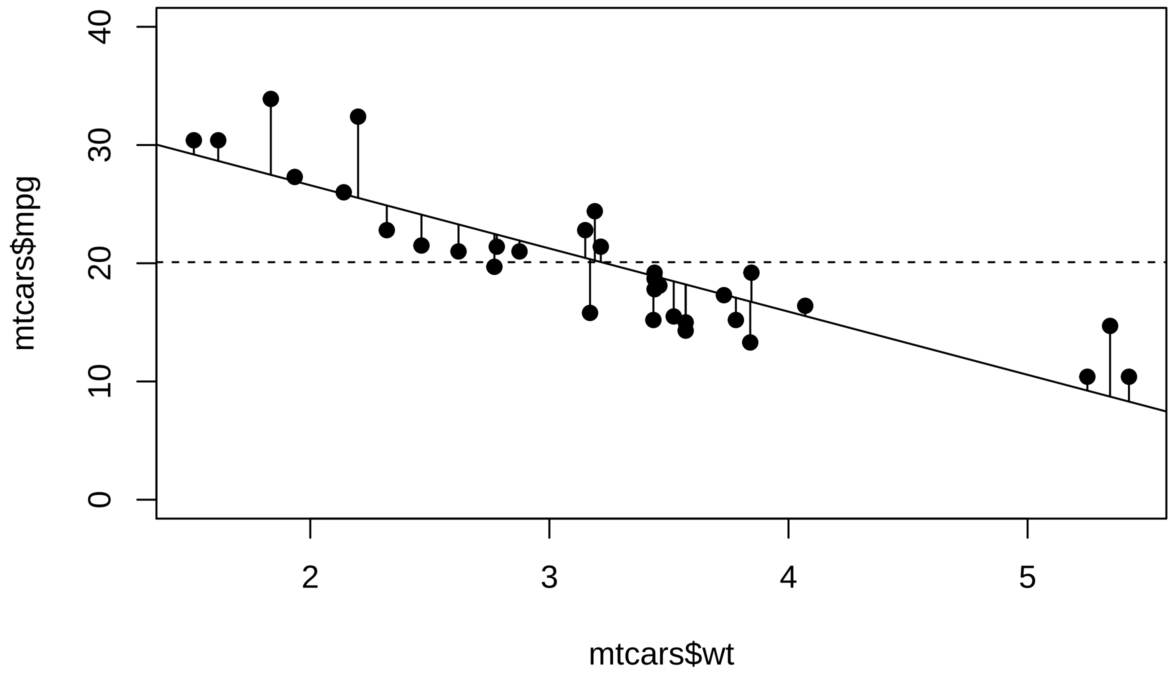
So let's build a model that predicts MPG using Wt as a predictor variable. We'll start out with just one predictor variable.

```
mylm <- lm(mpg ~ wt, data=mtcars)

summary(mylm)
```

```
##
## Call:
## lm(formula = mpg ~ wt, data = mtcars)
##
## Residuals:
```

```
##     Min      1Q  Median      3Q     Max
## -4.5432 -2.3647 -0.1252  1.4096  6.8727
##
## Coefficients:
##             Estimate Std. Error t value Pr(>|t|)
## (Intercept)  37.2851     1.8776  19.858  < 2e-16 ***
## wt           -5.3445     0.5591  -9.559 1.29e-10 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 3.046 on 30 degrees of freedom
## Multiple R-squared:  0.7528, Adjusted R-squared:  0.7446
## F-statistic: 91.38 on 1 and 30 DF,  p-value: 1.294e-10
```

```r
plot(mtcars$wt,mtcars$mpg,pch=19,ylim=c(0,40))

abline(h=mean(mtcars$mpg),lty=2)

abline(mylm)

segments(mtcars[,'wt'],mylm$fitted.values,mtcars[,'wt'],mtcars[,'mpg'])
```



Evaluating our Model

Significance of the predictors

We see a couple of things in our summary. That the Intercept and the predictor are highly significant. This means that it appears that wt is a pretty solid predictor of MPG. We also look at the Adjusted R-squared value. Generally speaking the higher the better although this will depend on the type of data being modeled. For example with human reaction data the R squares can be low but the model can still be effective. So when weight increases we can expect a reduction in MPG. That makes sense according to what we already know.

R-Squared

Let's go back to R-squared which gives us a measure of how much variation is explained by this model which in this case is 75%. That's not bad actually. We look at the Adjusted R-squared value which attempts to tell us how well our model would generalize. In this case it's slightly lower but it's still good. Now here is the thing - whenever we add in new predictors the R squared will go up.

F Test

Also note that the F-statistic gives us something to consider. It is reporting the results of an ANOVA (Analysis of Variance). This result tells us that an F value this large has a 1.294e-10 probability of occurring assuming that the null hypothesis was true.

The F-test of the overall significance is a specific form of the F-test. It compares a model with no predictors to the model that you specify so the F-test can evaluate a number of predictors simultaneously. A regression model that contains no predictors is also known as an intercept-only model. The hypotheses for the F-test of the overall significance are as follows:

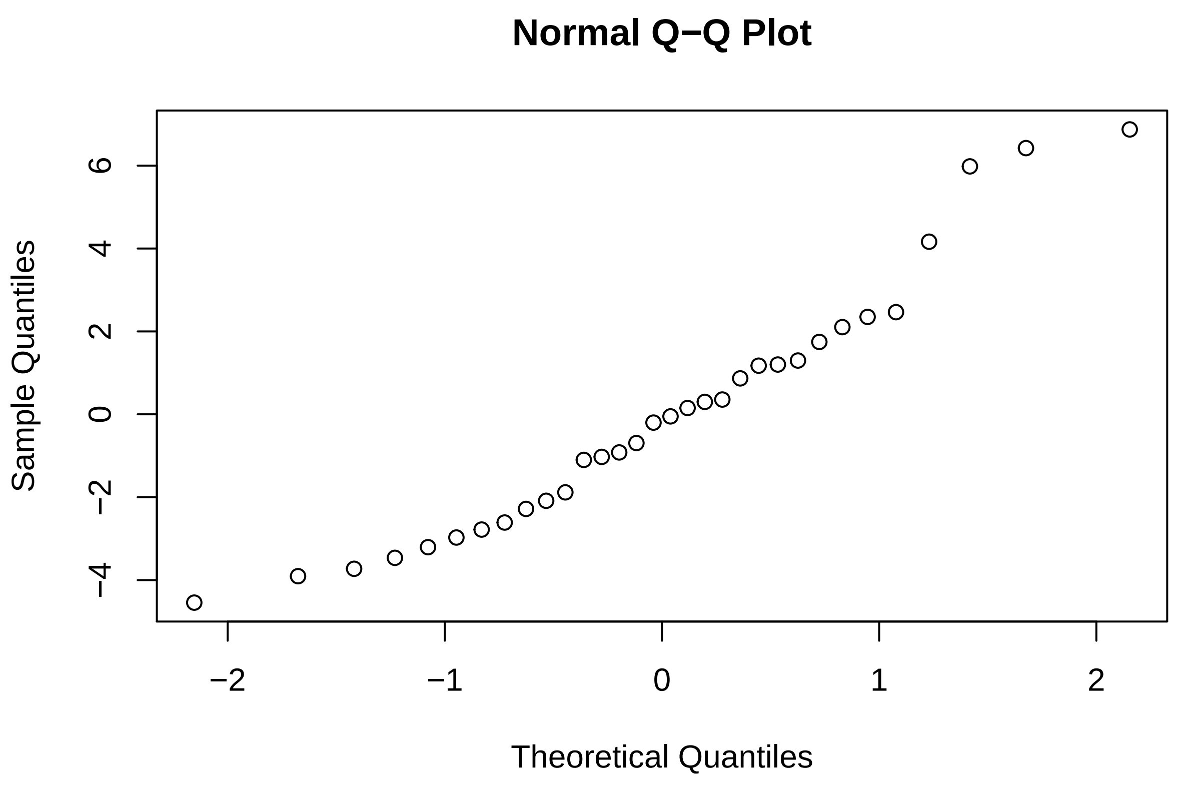
```
Null hypothesis: The fit of the intercept-only model and your model are equal.
Alternative hypothesis: The fit of the intercept-only model is significantly reduced compared to your mo
```

So in our case we know our model is better than something with no predictors. Wonderful !

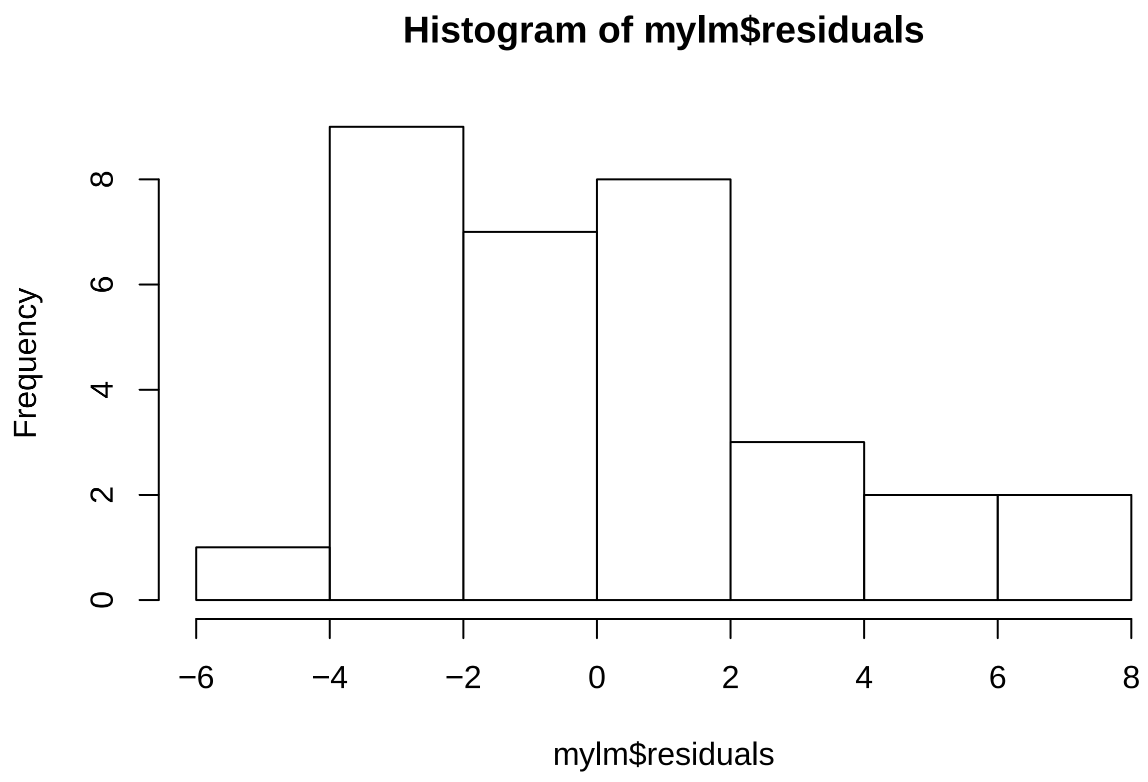
Dispersion of the residuals

The residuals should be normally distributed. There are a couple of ways to test this:

```r
# The resulting plot should be a straight diagonal line

qqnorm(mylm$residuals)
```



```r
hist(mylm$residuals)   # Should resemble a normal distribution
```



```r
shapiro.test(mylm$residuals)
```

```
##
##  Shapiro-Wilk normality test
##
## data:  mylm$residuals
## W = 0.94508, p-value = 0.1044
```

Improving the Model

Well the idea is to add variables to the model in a way that improves the R-squared AND results in coefficients that are statistically significant AND has a big F value and a low p-value AND has normally dsitributed residuals. Whew ! In reality there are other things to look at but that's enough for now. Anway, one simple way to proceed is look at the correlation matrix of the data. We probably should have done this before we created our first model but we knew that a car's weight probably impacts it's eventual MPG.

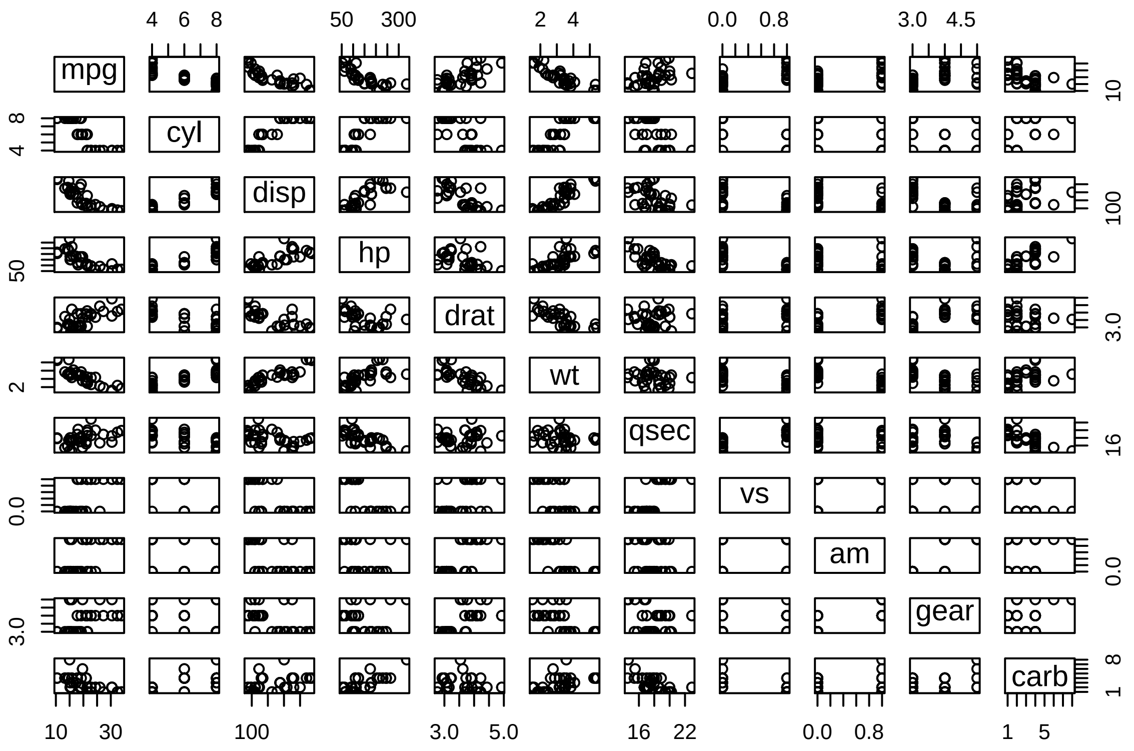
```r
mylm.1 <- lm(mpg ~ wt + cyl , data=mtcars)

summary(mylm)
```

```
##
## Call:
## lm(formula = mpg ~ wt, data = mtcars)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -4.5432 -2.3647 -0.1252  1.4096  6.8727
##
## Coefficients:
##             Estimate Std. Error t value Pr(>|t|)
## (Intercept)  37.2851     1.8776  19.858  < 2e-16 ***
## wt           -5.3445     0.5591  -9.559 1.29e-10 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 3.046 on 30 degrees of freedom
## Multiple R-squared:  0.7528, Adjusted R-squared:  0.7446
## F-statistic: 91.38 on 1 and 30 DF,  p-value: 1.294e-10
```

```r
round(cor(mtcars),2)
```

```
##        mpg   cyl  disp    hp  drat    wt  qsec    vs    am  gear  carb
## mpg   1.00 -0.85 -0.85 -0.78  0.68 -0.87  0.42  0.66  0.60  0.48 -0.55
## cyl  -0.85  1.00  0.90  0.83 -0.70  0.78 -0.59 -0.81 -0.52 -0.49  0.53
## disp -0.85  0.90  1.00  0.79 -0.71  0.89 -0.43 -0.71 -0.59 -0.56  0.39
## hp   -0.78  0.83  0.79  1.00 -0.45  0.66 -0.71 -0.72 -0.24 -0.13  0.75
## drat  0.68 -0.70 -0.71 -0.45  1.00 -0.71  0.09  0.44  0.71  0.70 -0.09
## wt   -0.87  0.78  0.89  0.66 -0.71  1.00 -0.17 -0.55 -0.69 -0.58  0.43
## qsec  0.42 -0.59 -0.43 -0.71  0.09 -0.17  1.00  0.74 -0.23 -0.21 -0.66
## vs    0.66 -0.81 -0.71 -0.72  0.44 -0.55  0.74  1.00  0.17  0.21 -0.57
## am    0.60 -0.52 -0.59 -0.24  0.71 -0.69 -0.23  0.17  1.00  0.79  0.06
## gear  0.48 -0.49 -0.56 -0.13  0.70 -0.58 -0.21  0.21  0.79  1.00  0.27
## carb -0.55  0.53  0.39  0.75 -0.09  0.43 -0.66 -0.57  0.06  0.27  1.00
```

```r
pairs(mtcars)
```



So we pick variables that are correlated with MPG although if any of the additional variables are also strongly correlated with Wt then that might not necessarily improve the model. There is a concept called “colinearity” wherein one variable can seemingly represent a second variable thus there is no need to include it. So we look for other variables correlated to weight but not necessarily so strongly correlated.

```r
mylm.2 <- lm(mpg ~ wt + drat , data=mtcars)

summary(mylm.2)
```

```
##
## Call:
## lm(formula = mpg ~ wt + drat, data = mtcars)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -5.4159 -2.0452  0.0136  1.7704  6.7466
##
## Coefficients:
##             Estimate Std. Error t value Pr(>|t|)
## (Intercept)   30.290      7.318   4.139 0.000274 ***
## wt            -4.783      0.797  -6.001 1.59e-06 ***
## drat           1.442      1.459   0.989 0.330854
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

```
##
## Residual standard error: 3.047 on 29 degrees of freedom
## Multiple R-squared:  0.7609, Adjusted R-squared:  0.7444
## F-statistic: 46.14 on 2 and 29 DF,  p-value: 9.761e-10
```

Nice try but it turns out that drat isn't significant nor does it do much to improve the Adjusted R-squared. What about horse power ? That does in fact present an interesting model. So we can keep doing this - adding in values that is to improve the model assuming that it's possible.

```
mylm.3 <- lm(mpg ~ wt + hp , data=mtcars)

summary(mylm.3)
```

```
##
## Call:
## lm(formula = mpg ~ wt + hp, data = mtcars)
##
## Residuals:
##    Min     1Q Median     3Q    Max
## -3.941 -1.600 -0.182  1.050  5.854
##
## Coefficients:
##             Estimate Std. Error t value Pr(>|t|)
## (Intercept) 37.22727    1.59879  23.285  < 2e-16 ***
## wt          -3.87783    0.63273  -6.129 1.12e-06 ***
## hp          -0.03177    0.00903  -3.519  0.00145 **
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 2.593 on 29 degrees of freedom
## Multiple R-squared:  0.8268, Adjusted R-squared:  0.8148
## F-statistic: 69.21 on 2 and 29 DF,  p-value: 9.109e-12
```

Step Wise Regression

But there is an easier way although you might lose your "statistician's card" for relying upon stepwise regression. We could use it in combination with something called the AIC (Akaike Information Criterion) to systematically evaluate different modesl which in turn are combinations of predictors from the data.

We go "forwards" or "backwards" which means we can start with one predictor and build into a model with multiple predictors. Or we can start with many predictors and move down to one. For each model we compute the AIC and at the end of the process we pick the model that has the "best" AIC. Note that the computed AIC is not absolute in any sense. It's value is useful only when compared to other models. There are other approaches to doing a systematic evaluation of predictors such as using "Forced Entry" or "Hierarchical Regression". But for now we'll check out Stepwise to show some other models. With regard to our data:

```
# We start with an initial model that includes all predictors.

mylm <- lm(mpg ~ ., data=mtcars)

steps <- step(mylm, direction="backward")
```

```
## Start:  AIC=70.9
## mpg ~ cyl + disp + hp + drat + wt + qsec + vs + am + gear + carb
##
##        Df Sum of Sq    RSS    AIC
## - cyl   1    0.0799 147.57 68.915
## - vs    1    0.1601 147.66 68.932
## - carb  1    0.4067 147.90 68.986
## - gear  1    1.3531 148.85 69.190
## - drat  1    1.6270 149.12 69.249
## - disp  1    3.9167 151.41 69.736
## - hp    1    6.8399 154.33 70.348
## - qsec  1    8.8641 156.36 70.765
## <none>              147.49 70.898
## - am    1   10.5467 158.04 71.108
## - wt    1   27.0144 174.51 74.280
##
## Step:  AIC=68.92
## mpg ~ disp + hp + drat + wt + qsec + vs + am + gear + carb
##
##        Df Sum of Sq    RSS    AIC
## - vs    1    0.2685 147.84 66.973
## - carb  1    0.5201 148.09 67.028
## - gear  1    1.8211 149.40 67.308
## - drat  1    1.9826 149.56 67.342
## - disp  1    3.9009 151.47 67.750
## - hp    1    7.3632 154.94 68.473
## <none>              147.57 68.915
## - qsec  1   10.0933 157.67 69.032
## - am    1   11.8359 159.41 69.384
## - wt    1   27.0280 174.60 72.297
##
## Step:  AIC=66.97
## mpg ~ disp + hp + drat + wt + qsec + am + gear + carb
##
##        Df Sum of Sq    RSS    AIC
## - carb  1    0.6855 148.53 65.121
## - gear  1    2.1437 149.99 65.434
## - drat  1    2.2139 150.06 65.449
## - disp  1    3.6467 151.49 65.753
## - hp    1    7.1060 154.95 66.475
## <none>              147.84 66.973
## - am    1   11.5694 159.41 67.384
## - qsec  1   15.6830 163.53 68.200
## - wt    1   27.3799 175.22 70.410
##
## Step:  AIC=65.12
## mpg ~ disp + hp + drat + wt + qsec + am + gear
##
##        Df Sum of Sq    RSS    AIC
## - gear  1     1.565 150.09 63.457
## - drat  1     1.932 150.46 63.535
## <none>              148.53 65.121
## - disp  1    10.110 158.64 65.229
## - am    1    12.323 160.85 65.672
## - hp    1    14.826 163.35 66.166
## - qsec  1    26.408 174.94 68.358
## - wt    1    69.127 217.66 75.350
##
## Step:  AIC=63.46
## mpg ~ disp + hp + drat + wt + qsec + am
##
##        Df Sum of Sq    RSS    AIC
## - drat  1     3.345 153.44 62.162
## - disp  1     8.545 158.64 63.229
## <none>              150.09 63.457
## - hp    1    13.285 163.38 64.171
## - am    1    20.036 170.13 65.466
## - qsec  1    25.574 175.67 66.491
## - wt    1    67.572 217.66 73.351
##
## Step:  AIC=62.16
## mpg ~ disp + hp + wt + qsec + am
##
##        Df Sum of Sq    RSS    AIC
## - disp  1     6.629 160.07 61.515
## <none>              153.44 62.162
## - hp    1    12.572 166.01 62.682
## - qsec  1    26.470 179.91 65.255
## - am    1    32.198 185.63 66.258
## - wt    1    69.043 222.48 72.051
##
## Step:  AIC=61.52
## mpg ~ hp + wt + qsec + am
##
##        Df Sum of Sq    RSS    AIC
## - hp    1     9.219 169.29 61.307
## <none>              160.07 61.515
## - qsec  1    20.225 180.29 63.323
## - am    1    25.993 186.06 64.331
## - wt    1    78.494 238.56 72.284
##
## Step:  AIC=61.31
## mpg ~ wt + qsec + am
##
##        Df Sum of Sq    RSS    AIC
## <none>              169.29 61.307
## - am    1    26.178 195.46 63.908
## - qsec  1   109.034 278.32 75.217
## - wt    1   183.347 352.63 82.790
```

```r
summary(steps)
```

```
##
## Call:
## lm(formula = mpg ~ wt + qsec + am, data = mtcars)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -3.4811 -1.5555 -0.7257  1.4110  4.6610
##
## Coefficients:
##             Estimate Std. Error t value Pr(>|t|)
## (Intercept)   9.6178     6.9596   1.382 0.177915
## wt           -3.9165     0.7112  -5.507 6.95e-06 ***
## qsec          1.2259     0.2887   4.247 0.000216 ***
## am            2.9358     1.4109   2.081 0.046716 *
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 2.459 on 28 degrees of freedom
## Multiple R-squared:  0.8497, Adjusted R-squared:  0.8336
## F-statistic: 52.75 on 3 and 28 DF,  p-value: 1.21e-11
```

Note that evaluating the output of any regression requires careful consideration of a number of things. These are just some of them.

Training vs Test

Well we used the entire data frame for this but we should have held out some of the records. We don't have many observations in this data frame as it is but let's hold out say 25 percent which equates to 8 records.

```r
set.seed(123)
test <- sample(1:32,8,F)
testcars  <- mtcars[test,]
traincars <- mtcars[-test,]

model <- lm(mpg~.,data=traincars)
steps <- step(model, direction="backward")
```

```
## Start:  AIC=61.99
## mpg ~ cyl + disp + hp + drat + wt + qsec + vs + am + gear + carb
##
##        Df Sum of Sq    RSS    AIC
## - cyl   1    0.0002 127.02 59.991
## - vs    1    0.1529 127.17 60.020
## - carb  1    0.4849 127.50 60.082
## - gear  1    0.7297 127.75 60.128
## - disp  1    1.2579 128.28 60.227
## - drat  1    1.7109 128.73 60.312
## - hp    1    5.8885 132.91 61.078
## - qsec  1    8.6748 135.69 61.576
## - am    1   10.0919 137.11 61.826
## <none>              127.02 61.991
## - wt    1   20.6609 147.68 63.608
##
## Step:  AIC=59.99
## mpg ~ disp + hp + drat + wt + qsec + vs + am + gear + carb
##
##        Df Sum of Sq    RSS    AIC
## - vs    1    0.1862 127.21 58.026
## - carb  1    0.4855 127.50 58.082
## - gear  1    0.7372 127.76 58.130
## - disp  1    1.4408 128.46 58.262
## - drat  1    1.7852 128.81 58.326
## - hp    1    6.0566 133.08 59.109
## - qsec  1    9.4553 136.47 59.714
## - am    1   10.8499 137.87 59.958
## <none>              127.02 59.991
## - wt    1   20.8248 147.84 61.635
##
## Step:  AIC=58.03
## mpg ~ disp + hp + drat + wt + qsec + am + gear + carb
##
##        Df Sum of Sq    RSS    AIC
## - carb  1    0.3298 127.54 56.088
## - gear  1    0.5866 127.79 56.137
## - drat  1    1.6625 128.87 56.338
## - disp  1    1.9571 129.16 56.393
## - hp    1    6.9317 134.14 57.300
## - qsec  1   10.2641 137.47 57.888
## <none>              127.21 58.026
## - am    1   11.5699 138.78 58.115
## - wt    1   21.0354 148.24 59.699
##
## Step:  AIC=56.09
## mpg ~ disp + hp + drat + wt + qsec + am + gear
##
##        Df Sum of Sq    RSS    AIC
## - gear  1     0.324 127.86 54.149
## - drat  1     1.452 128.99 54.360
## - disp  1     4.540 132.07 54.928
## <none>              127.54 56.088
## - hp    1    11.718 139.25 56.198
## - am    1    12.994 140.53 56.417
## - qsec  1    17.170 144.71 57.120
## - wt    1    47.077 174.61 61.628
##
## Step:  AIC=54.15
## mpg ~ disp + hp + drat + wt + qsec + am
##
##        Df Sum of Sq    RSS    AIC
## - drat  1     1.646 129.51 52.456
## - disp  1     4.320 132.18 52.947
## <none>              127.86 54.149
## - hp    1    12.649 140.51 54.413
## - qsec  1    17.454 145.31 55.220
## - am    1    20.077 147.94 55.650
## - wt    1    46.781 174.64 59.632
##
## Step:  AIC=52.46
## mpg ~ disp + hp + wt + qsec + am
##
##        Df Sum of Sq    RSS    AIC
## - disp  1     3.723 133.23 51.136
## <none>              129.51 52.456
## - hp    1    11.677 141.18 52.528
## - qsec  1    19.629 149.13 53.843
## - am    1    29.895 159.40 55.441
## - wt    1    49.784 179.29 58.263
##
## Step:  AIC=51.14
## mpg ~ hp + wt + qsec + am
##
##        Df Sum of Sq    RSS    AIC
## - hp    1     9.329 142.56 50.761
## <none>              133.23 51.136
## - qsec  1    15.919 149.15 51.845
## - am    1    26.188 159.42 53.443
## - wt    1    67.260 200.49 58.945
##
## Step:  AIC=50.76
## mpg ~ wt + qsec + am
##
##        Df Sum of Sq    RSS    AIC
## <none>              142.56 50.761
## - am    1    25.475 168.03 52.707
## - qsec  1   110.456 253.01 62.529
## - wt    1   165.233 307.79 67.233
```

```
summary(steps)
```

```
##
## Call:
## lm(formula = mpg ~ wt + qsec + am, data = traincars)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -3.6201 -1.6317 -0.5828  1.6148  4.7157
##
## Coefficients:
##             Estimate Std. Error t value Pr(>|t|)
## (Intercept)   8.1521     8.0053   1.018 0.320675
## wt           -3.8562     0.8009  -4.815 0.000105 ***
## qsec          1.2884     0.3273   3.937 0.000816 ***
## am            3.2375     1.7125   1.890 0.073271 .
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 2.67 on 20 degrees of freedom
## Multiple R-squared:  0.8547, Adjusted R-squared:  0.8329
## F-statistic: 39.21 on 3 and 20 DF,  p-value: 1.446e-08
```

The same model as before winds up being the best although the am variable is barely significant. Now let's do some prediction on the test set to see how close we came to the actual y values.

```r
predict(steps,testcars)
```

```
##          Merc 280 Pontiac Firebird        Merc 450SL         Fiat X1-9 
##          18.46419          15.29193          16.44401          28.27839 
##     Porsche 914-2     Mazda RX4 Wag       Merc 450SLC       AMC Javelin 
##          24.65341          22.23136          16.76655          17.19508
```

```r
predict(steps,testcars) - testcars$mpg
```

```
##          Merc 280 Pontiac Firebird        Merc 450SL         Fiat X1-9 
##        -0.7358089        -3.9080731        -0.8559912         0.9783949 
##     Porsche 914-2     Mazda RX4 Wag       Merc 450SLC       AMC Javelin 
##        -1.3465914         1.2313568         1.5665526         1.9950827
```

Applied Regression - MoneyBall

Note that the lots of this material has been taken from "The Analytics Edge" course on EDX with some additions and modifications. First let's read in some information relating to Professional Baseball teams between the years 1962 and 2012 although we will focus on certain years in this range.

```r
# Read in data
# Source is http://www.baseball-reference.com/ via Edx class

url <- "https://raw.githubusercontent.com/steviep42/INFO550/master/Week10/baseball.csv"
baseball <- read.csv(url,stringsAsFactors=FALSE)
str(baseball)
```

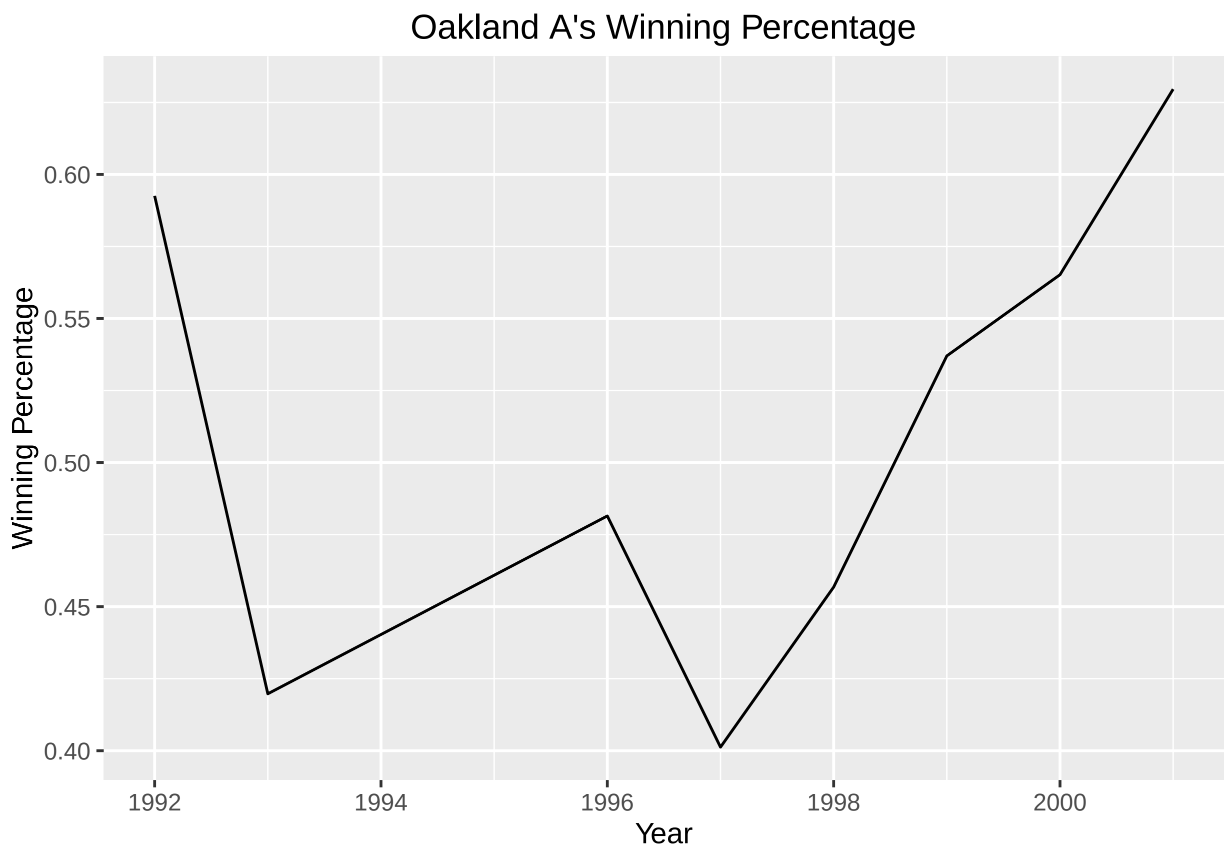
```
## 'data.frame':    1232 obs. of  15 variables:
##  $ Team        : chr  "ARI" "ATL" "BAL" "BOS" ...
##  $ League      : chr  "NL" "NL" "AL" "AL" ...
##  $ Year        : int  2012 2012 2012 2012 2012 2012 2012 2012 2012 2012 ...
##  $ RS          : int  734 700 712 734 613 748 669 667 758 726 ...
##  $ RA          : int  688 600 705 806 759 676 588 845 890 670 ...
##  $ W           : int  81 94 93 69 61 85 97 68 64 88 ...
##  $ OBP         : num  0.328 0.32 0.311 0.315 0.302 0.318 0.315 0.324 0.33 0.335 ...
##  $ SLG         : num  0.418 0.389 0.417 0.415 0.378 0.422 0.411 0.381 0.436 0.422 ...
##  $ BA          : num  0.259 0.247 0.247 0.26 0.24 0.255 0.251 0.251 0.274 0.268 ...
##  $ Playoffs    : int  0 1 1 0 0 0 1 0 0 1 ...
##  $ RankSeason  : int  NA 4 5 NA NA NA 2 NA NA 6 ...
##  $ RankPlayoffs: int  NA 5 4 NA NA NA 4 NA NA 2 ...
##  $ G           : int  162 162 162 162 162 162 162 162 162 162 ...
##  $ OOBP        : num  0.317 0.306 0.315 0.331 0.335 0.319 0.305 0.336 0.357 0.314 ...
##  $ OSLG        : num  0.415 0.378 0.403 0.428 0.424 0.405 0.39 0.43 0.47 0.402 ...
```

Some key column meanings - some are self-explanatory like Team, Year RS - Runs Scored, RA - Runs Allowed, W - Wins for the year, G - Number of games played. Let's focus on the Okaland A's subject of movie called "Moneyball". They got new owners in 1995 who imposed strong budget cuts and limits. How do you field a competitive team when you can't afford to pay anyone known to be good by standard performance measures ?

```r
library(dplyr)
```

```
##
## Attaching package: 'dplyr'

## The following objects are masked from 'package:stats':
##
##     filter, lag

## The following objects are masked from 'package:base':
##
##     intersect, setdiff, setequal, union
```

```r
library(ggplot2)

moneyball <- filter(baseball, Year < 2002)

# Some win loss history

moneyball %>% filter(Team=="OAK",Year >= 1992) %>%
  mutate(percent_win=W/G) %>%
  ggplot(aes(x=Year,y=percent_win)) + geom_line() +
  ggtitle("Oakland A's Winning Percentage") + ylab("Winning Percentage")
```



```
# Let's build a table

moneyball %>% filter(Team=="OAK",Year >= 1992) %>%
  mutate(percent=round(W/G,2)) %>% group_by(Year) %>%
  summarize(percent)
```

```
## Source: local data frame [8 x 2]
##
##    Year percent
##   (int)   (dbl)
## 1  1992    0.59
## 2  1993    0.42
## 3  1996    0.48
## 4  1997    0.40
## 5  1998    0.46
## 6  1999    0.54
## 7  2000    0.57
## 8  2001    0.63
```

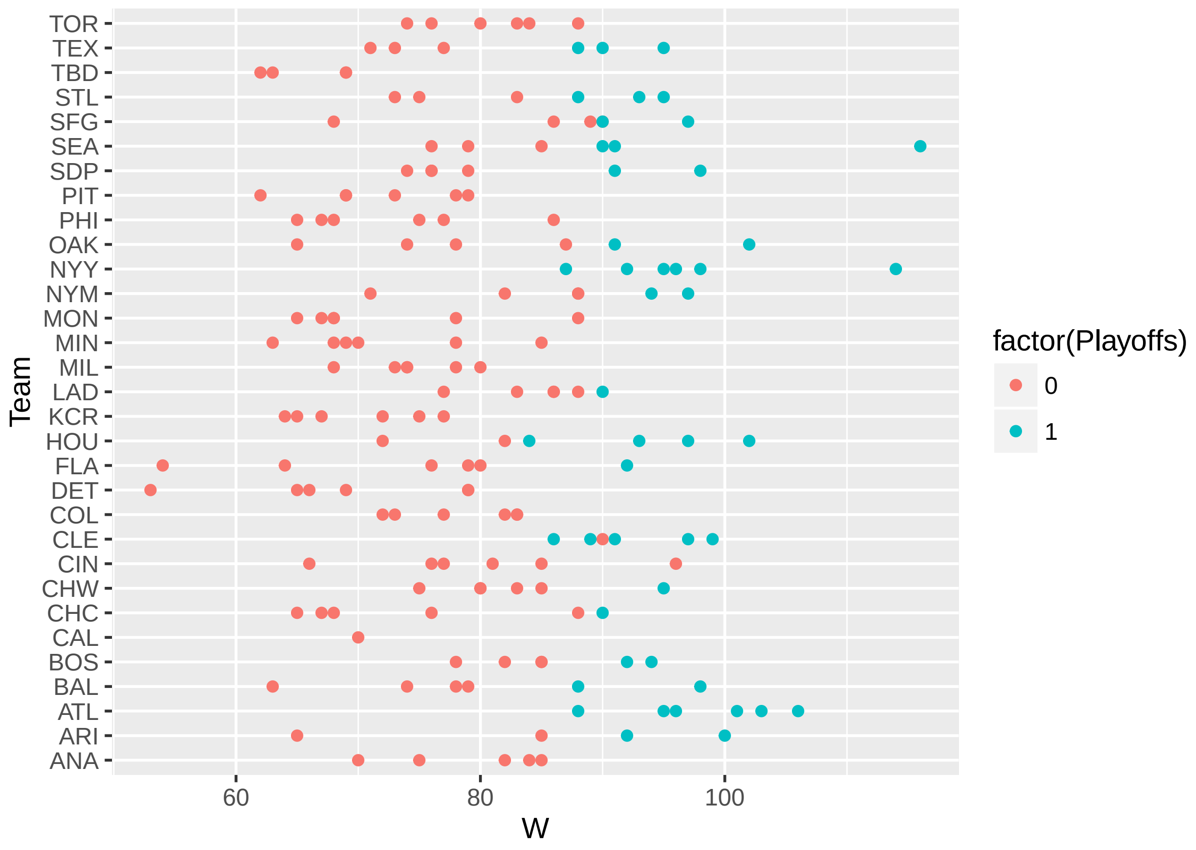
```
# Let's do some salary comparison for the year 1999

moneyball %>%
   filter(Year==1999, Team == "OAK" | Team == "NYY" | Team == "BOS") %>%
   select(Team,W)
```

```
##   Team  W
## 1  BOS 94
## 2  NYY 98
## 3  OAK 87
```

Okay looking at http://www.stlsports.com/articles/ja.mlbsalaries.html We see that NYY spent 91 million, BOS spent 72 million, and OAK spent 25 mil But Oakland still won lots of games. Perhaps some skills are overvalued and some undervalued in the scouting process ?

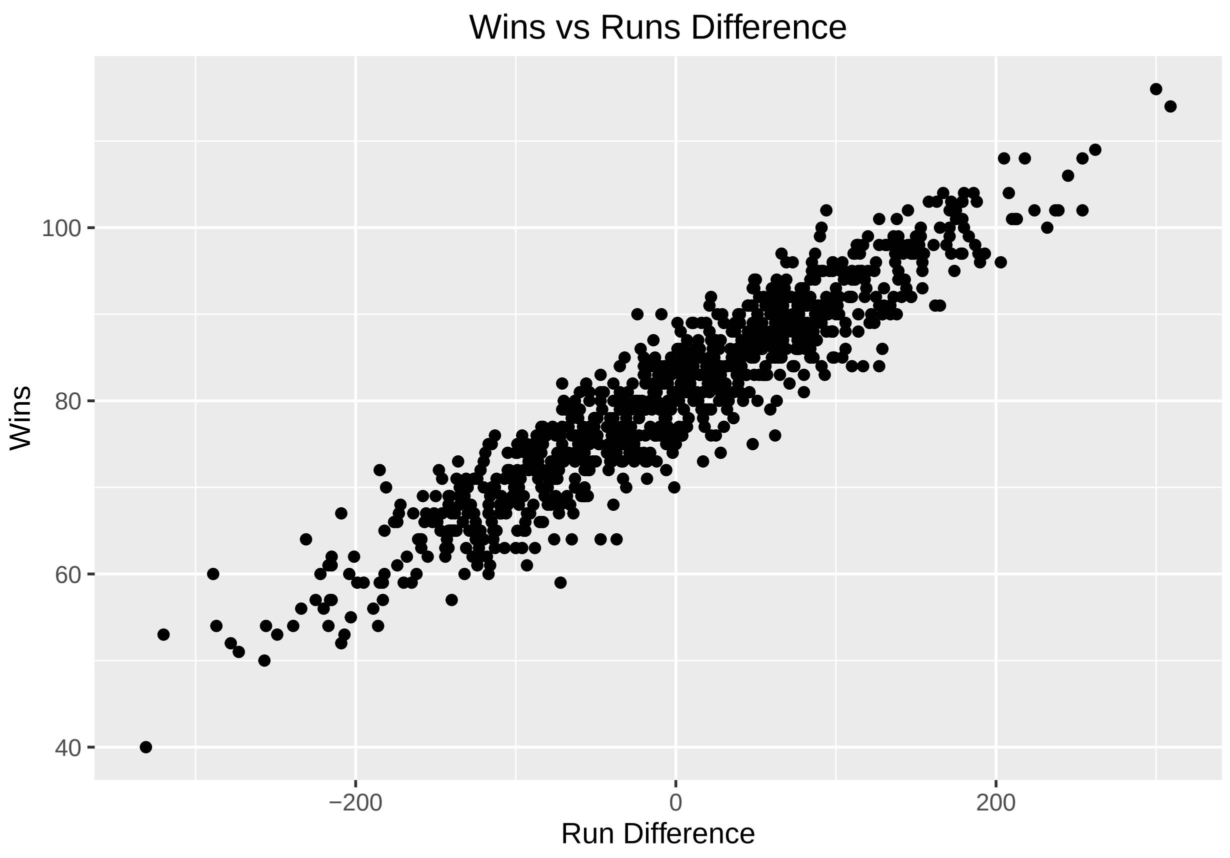
We would like to predict if a team can make the playoffs by knowing how many games they won in a season. How many wins would it take to make it to playoffs? 95 ?

```
moneyball %>% filter(Year >= 1996) %>%
  ggplot(aes(x=W,y=Team,col=factor(Playoffs))) + geom_point()
```



Use Linear Regression to determine how many games they will win using the diff between runs scored and runs allowed. So compute this difference and save in a new column.

```
moneyball <- mutate(moneyball,RD = RS - RA)

# Scatterplot to check for linear relationship
ggplot(moneyball,aes(x=RD,y=W)) +
  geom_point() + ggtitle("Wins vs Runs Difference") +
  xlab("Run Difference") + ylab("Wins")
```



```r
# Strong linear relationship between these two variables
# Regression model to predict wins

WinsReg <- lm(W ~ RD, data=moneyball)
summary(WinsReg)
```

```
##
## Call:
## lm(formula = W ~ RD, data = moneyball)
##
## Residuals:
##      Min       1Q   Median       3Q      Max
## -14.2662  -2.6509   0.1234   2.9364  11.6570
##
## Coefficients:
##              Estimate Std. Error t value Pr(>|t|)
## (Intercept) 80.881375   0.131157  616.67   <2e-16 ***
## RD           0.105766   0.001297   81.55   <2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 3.939 on 900 degrees of freedom
## Multiple R-squared:  0.8808, Adjusted R-squared:  0.8807
## F-statistic:  6651 on 1 and 900 DF,  p-value: < 2.2e-16
```

So the RD coefficient is strongly significant and also has a high R-squared value. We could now ask some questions like given that we know that we need to win at least 95 games to make the playoffs how large does the run difference need to be to accomplish that ? This can be solved using some basic algerbra.

```
Wins = 80.8814 + 0.1068*RD

Wins >= 95

80.8814 + 0.1068*RD >= 95

RD >= (95 - 80.8814)/0.1068 = 133.4
```

So a team needs to score at least 134 runs than they allow. This next leads us to how to predict the number runs a team will score. For this we turn to batting statistics to make that determination.

As a second consideration we also want to predict how many runs a team will allow based on pitching and fielding stats.

Let's build a regression model to predict runs scored by a team - How does a team score runs ? Hitting the ball, walking, getting on base for starters. There are some metrics in the data frame that can help us with these. For example the variables for On Base Percentage and Slugging Percentage (how far a batter gets around bases), as well as Batting Average are there. Note that the professional scouts, at least up until the mid 90s, though that Batting Average along was the single most important deteriminant of run scoring.

```
RunsReg <- lm(RS ~ OBP + SLG + BA, data=moneyball)
summary(RunsReg)
```

```
##
## Call:
## lm(formula = RS ~ OBP + SLG + BA, data = moneyball)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -70.941 -17.247  -0.621  16.754  90.998
##
## Coefficients:
##             Estimate Std. Error t value Pr(>|t|)
## (Intercept)  -788.46      19.70 -40.029  < 2e-16 ***
## OBP          2917.42     110.47  26.410  < 2e-16 ***
## SLG          1637.93      45.99  35.612  < 2e-16 ***
## BA           -368.97     130.58  -2.826  0.00482 **
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 24.69 on 898 degrees of freedom
## Multiple R-squared:  0.9302, Adjusted R-squared:   0.93
## F-statistic:  3989 on 3 and 898 DF,  p-value: < 2.2e-16
```

```
# We might have some multicollinearity here

cor(moneyball[,7:9])
```

```
##           OBP       SLG        BA
## OBP 1.0000000 0.8061539 0.8540549
## SLG 0.8061539 1.0000000 0.8140681
## BA  0.8540549 0.8140681 1.0000000
```

```r
RunsReg <- lm(RS ~ OBP + SLG, data=moneyball)
summary(RunsReg)
```

```
##
## Call:
## lm(formula = RS ~ OBP + SLG, data = moneyball)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -70.838 -17.174  -1.108  16.770  90.036
##
## Coefficients:
##             Estimate Std. Error t value Pr(>|t|)
## (Intercept)  -804.63      18.92  -42.53   <2e-16 ***
## OBP          2737.77      90.68   30.19   <2e-16 ***
## SLG          1584.91      42.16   37.60   <2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 24.79 on 899 degrees of freedom
## Multiple R-squared:  0.9296, Adjusted R-squared:  0.9294
## F-statistic:  5934 on 2 and 899 DF,  p-value: < 2.2e-16
```

```r
# So maybe BA is overvalued
```

So it turns out that we have the information for an opposing teams On Base Percentage and Slugging Percentage so it becomes easy for us to come up with a regression equation that predict's the number of runs they will score against us (in this case "we" are the Oakland Atheltics).

```r
RunsAllowed <- lm(RA ~ OOBP + OSLG, data=moneyball)

summary(RunsAllowed)
```

```
##
## Call:
## lm(formula = RA ~ OOBP + OSLG, data = moneyball)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -82.397 -15.178  -0.129  17.679  60.955
##
## Coefficients:
##             Estimate Std. Error t value Pr(>|t|)
## (Intercept)  -837.38      60.26 -13.897  < 2e-16 ***
## OOBP         2913.60     291.97   9.979 4.46e-16 ***
## OSLG         1514.29     175.43   8.632 2.55e-13 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 25.67 on 87 degrees of freedom
##   (812 observations deleted due to missingness)
## Multiple R-squared:  0.9073, Adjusted R-squared:  0.9052
## F-statistic: 425.8 on 2 and 87 DF,  p-value: < 2.2e-16
```

Predicting the number of Runs scored in 2002

What if we wanted to predict the number of runs for the 2002 season ? Well we have information that allowed us to create a regression equation to predict runs scored. Now, we have the data in our data set for the 2002 season but we filtered it out. Here is what we could do - we could go to a site like http://espn.go.com/mlb/stats/team/_/stat/batting/year/2001 and come up with an average OBP and SLP for the team and plug that into our equation.

```
summary(RunsReg)
```

```
##
## Call:
## lm(formula = RS ~ OBP + SLG, data = moneyball)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -70.838 -17.174  -1.108  16.770  90.036
##
## Coefficients:
##             Estimate Std. Error t value Pr(>|t|)
## (Intercept)  -804.63      18.92  -42.53   <2e-16 ***
## OBP          2737.77      90.68   30.19   <2e-16 ***
## SLG          1584.91      42.16   37.60   <2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 24.79 on 899 degrees of freedom
## Multiple R-squared:  0.9296, Adjusted R-squared:  0.9294
## F-statistic:  5934 on 2 and 899 DF,  p-value: < 2.2e-16
```

```
# Runs = -804.63 + 2737.77*OBP + 1584.91*SLG
# Runs = -804.63 + 2737.77*0.345 + 1584.91*0.439 = 835.67 runs
```

We could also use our regression equation that we generated for predicting the number of runs that would be scored against us. We would look for pitching statistics for the Oakland As from here http://espn.go.com/mlb/stats/team/_/stat/pitching/year/2001/type/expanded to get the Opposing OBP and SLG and plug that into our equation.

```
summary(RunsAllowed)
```

```
##
## Call:
## lm(formula = RA ~ OOBP + OSLG, data = moneyball)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -82.397 -15.178  -0.129  17.679  60.955
##
## Coefficients:
##             Estimate Std. Error t value Pr(>|t|)
## (Intercept)  -837.38      60.26 -13.897  < 2e-16 ***
## OOBP         2913.60     291.97   9.979 4.46e-16 ***
## OSLG         1514.29     175.43   8.632 2.55e-13 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 25.67 on 87 degrees of freedom
##   (812 observations deleted due to missingness)
## Multiple R-squared:  0.9073, Adjusted R-squared:  0.9052
## F-statistic: 425.8 on 2 and 87 DF,  p-value: < 2.2e-16
```

```
# RA = -837.38 + 2913.6*OOBP + 1514.29*OSLG
# RA = -837.38 + 2913.6*0.308 + 1514.29*0.380 = 635.439
```

```
Now remember that the number of Wins in a season had a regression equation of

Wins = 80.8814 + 0.1068*RD

So let's compute this for the 2002 season

Wins = 80.8814 + 0.1068*RD

Wins = 80.8814 + 0.1068*(835.67-635.439)

Wins = 102.2661

baseball %>% filter(Year==2002 & Team=="OAK") %>% head
```

Practice

Fetch the following file that has information on musical recording artists and their sales figures. Work with this data to come up with at least 4 different models involving different predictors and combinations thereof to predict sales. The data set isn't complicated so it should be easy to get a handle on the data. Do things look at correlations. Do some pairs plotting to see what variables might have linear relationships. Then create some models.

```
url <- "https://raw.githubusercontent.com/steviep42/INFO550/master/Week10/album_sales_2.csv"
sales <- read.csv(url,sep="\t",header=TRUE)
```

Clustering

Here we will consider a couple of approaches known as 1) Hierarchical and 2) K-Means Clustering. We'll consider Hierachical Clustering first. This takes data and puts each data point / observation into it's own cluster. The individual points/clusters are merged into pairs of clusters until there is a single cluster. This is what Hierachical Clustering does. To be specific the recursive algorithm is:

- Find the two closest points in the dataset
- Link these points and consider them as a single point
- The process starts again, now using the the new dataset that contains the new point.

Distance

To do this requires us to measure the distance between points. The aim is that the measured distances between observations of the same cluster are as small as possible and the distances between clusters are as large as possible. There are a number of methods to compute distances such as:

- Euclidean
- Maximum
- Manhattan
- Canberra
- Binary
- Perason
- Correlation
- Spearman

One of the simplest is to use the "euclidean distance". Here is how we might compute the distance between two vectors.

```
v1 <- c(0,1,1,0,0,1,1,0,1)

v2 <- c(0,1,1,0,0,1,1,0,1)

v3 <- c(0,1,1,0,0,1,1,0,0)

 sqrt(sum((v1-rev(v2))^2))
```

```
## [1] 2.44949
```

```
# These are identical so there should not be any distance

dist(rbind(v1,v2))
```

```
##    v1
## v2  0
```

```
dist(rbind(v1,v3))
```

```
##    v1
## v3  1
```

```
dist(rbind(v1,rev(v2)))
```

```
##       v1
##  2.44949
```

```
dist(rbind(rbind(v1,v2,v3)))
```

```
##    v1 v2
## v2  0
## v3  1  1
```

Note that if the vectors are not on the same scale then we need to standardize the data - subtract each vector element from it's mean and then divide by the standard deviation. Anyway R has a function called **dist** that supports many distance methods:

```
dist(x, method = "euclidean", diag = FALSE, upper = FALSE, p = 2)

the distance measure to be used. This must be one of "euclidean", "maximum",
"manhattan", "canberra", "binary" or "minkowski". Any unambiguous substring can be given.
```

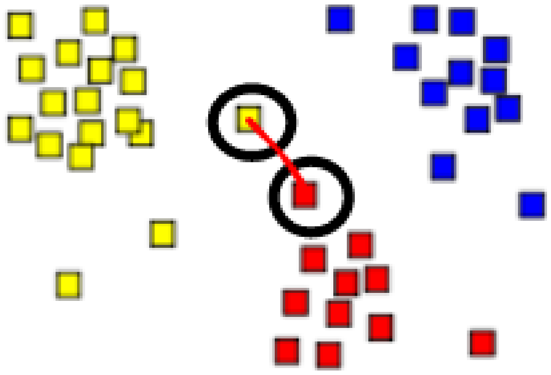
Linkage

Tightly coupled with the idea of computing distance is the **linkage method** which is necessary for calculating the inter-cluster distances.

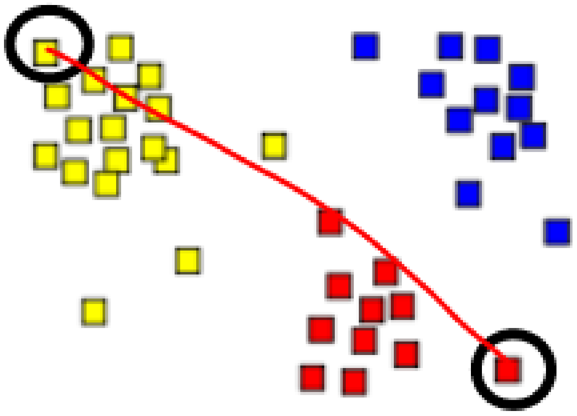
LINKAGE METHOD	DEFINITION
Single Linkage	The distance between two clusters is the minimum distance between an observation in one cluster and an observation in the other cluster. A good choice when clusters are obviously separated
Complete Linkage	The distance between two clusters is the maximum distance between an observation in one cluster and an observation in the other cluster. It can be sensitive to outliers
Average Linkage	The distance between two clusters is the mean distance between an observation in one cluster and an observation in the other cluster
Centroid Linkage	The distance between two clusters is the distance between the cluster centroids or means
Median Linkage	The distance between two clusters is the median distance between an observation in one cluster and an observation in the other cluster. It reduces the effect of outliers
Ward Linkage	The distance between two clusters is the sum of the squared deviations from points to centroids. Try to minimize the within-cluster sum of squares. It can be sensitive to outliers

Here are some graphic examples of distances and linkages between some example clusters. First we have:

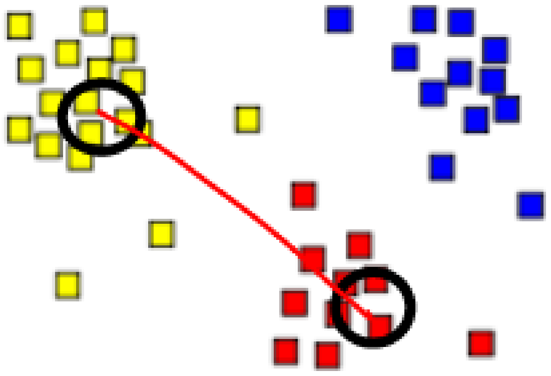
Minimum Distance



Maximum Distance

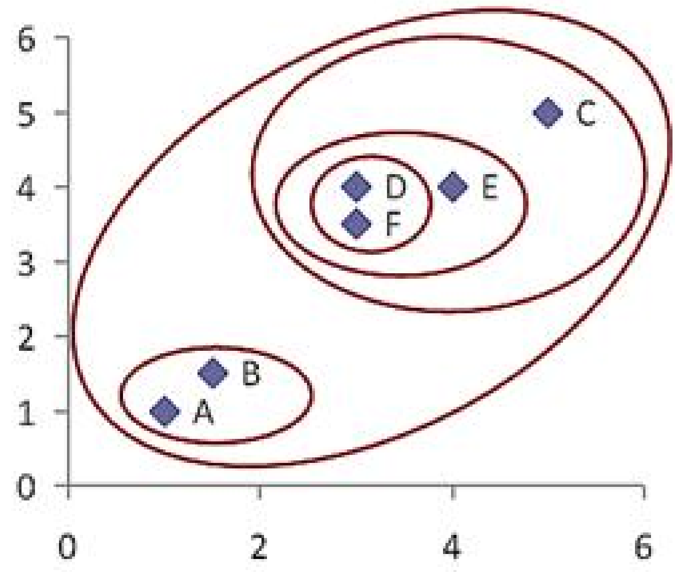


Centroid



Hierarchical Clustering

Here is a graphic representation of what happens during HC:



Clustering the Iris Data

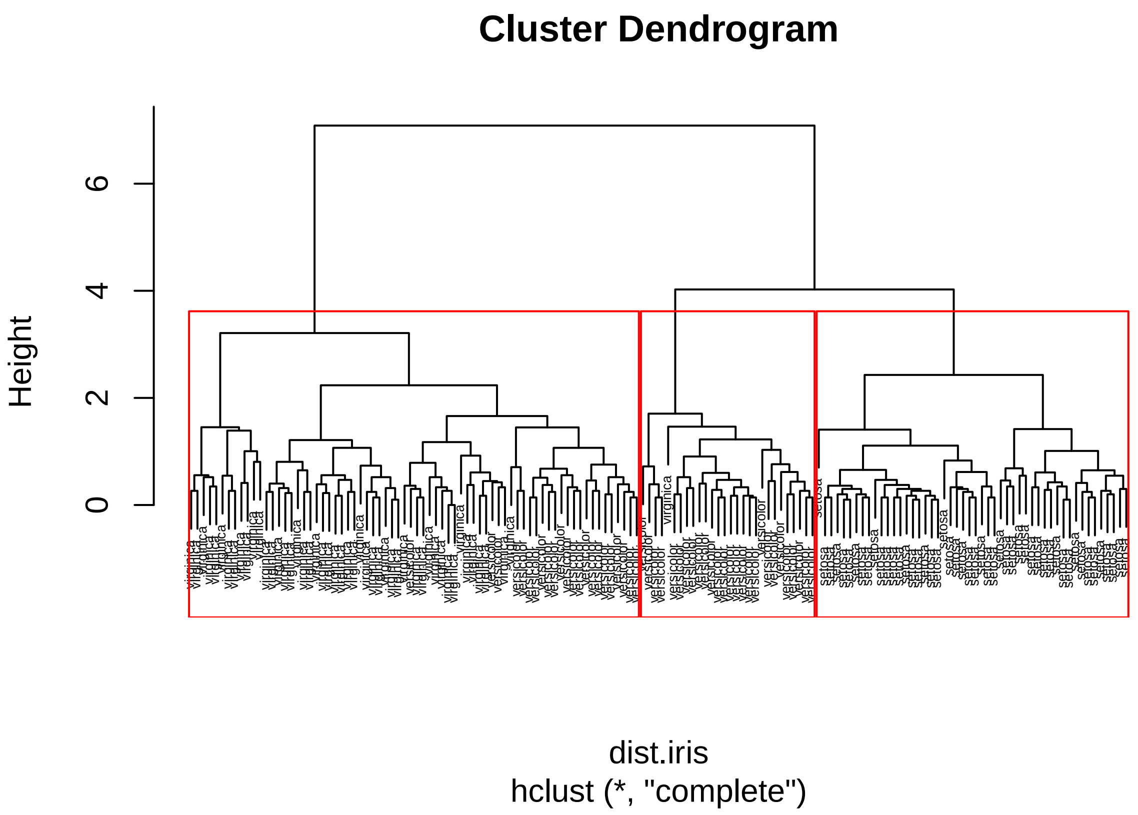
Let's look at the built-in iris data that has three Species. We can group the 150 rows into clusters using Hieratchical clustering. We'll create a distance matrix. To do this we'll need to exclude the text that describes the Species since it is a label.

```
data(iris)

str(iris)
```

```
## 'data.frame':    150 obs. of  5 variables:
##  $ Sepal.Length: num  5.1 4.9 4.7 4.6 5 5.4 4.6 5 4.4 4.9 ...
##  $ Sepal.Width : num  3.5 3 3.2 3.1 3.6 3.9 3.4 3.4 2.9 3.1 ...
##  $ Petal.Length: num  1.4 1.4 1.3 1.5 1.4 1.7 1.4 1.5 1.4 1.5 ...
##  $ Petal.Width : num  0.2 0.2 0.2 0.2 0.2 0.4 0.3 0.2 0.2 0.1 ...
##  $ Species     : Factor w/ 3 levels "setosa","versicolor",..: 1 1 1 1 1 1 1 1 1 1 ...
```

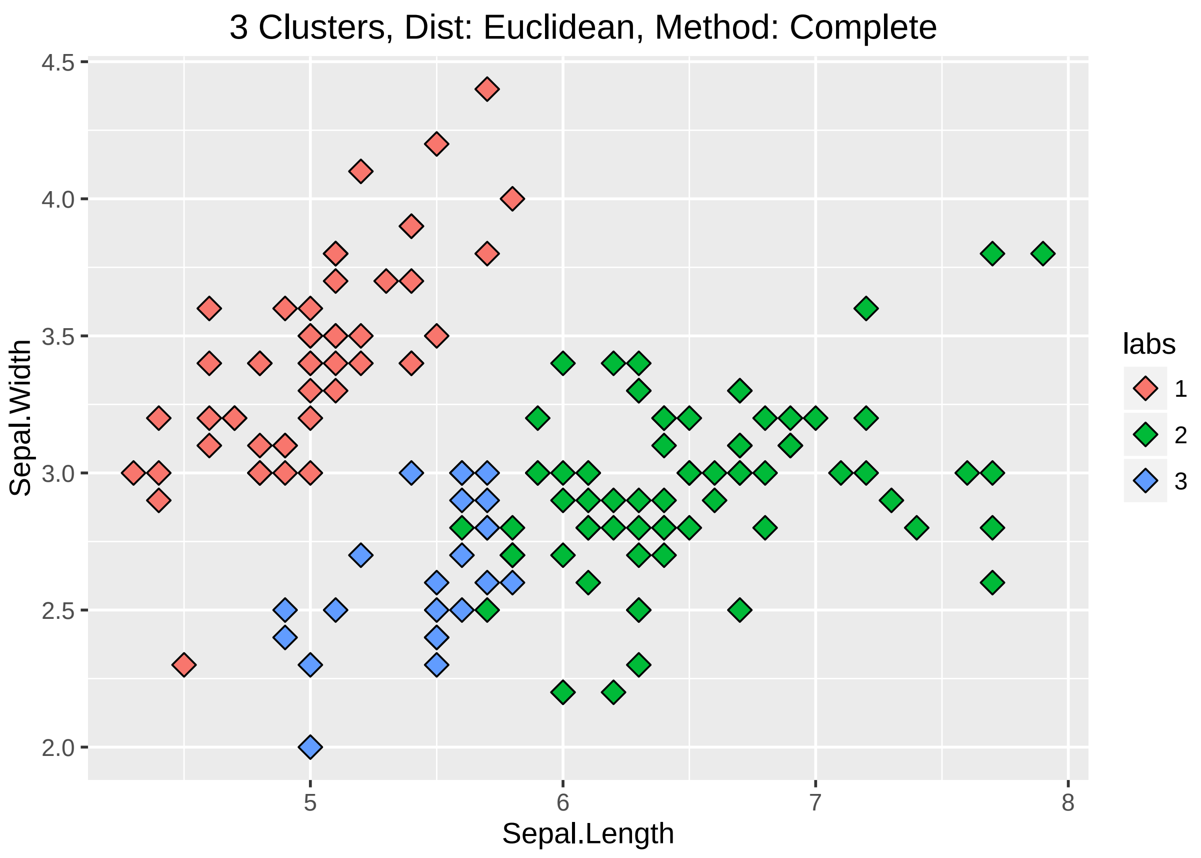
```
dist.iris <- dist(iris[,-5])   # Default method is euclidean

# Now we pass the distance matrix to the hclust function

iris_hclust <- hclust(dist.iris)

# Clustered objects have their own special plot function that results
# in a Dendrogram

plot(iris_hclust, label=iris$Species, cex=0.4)

# There is another special plot that given a number of proposed clusters
# will draw a corresponding number of rectangles
```

```
rect.hclust(iris_hclust, k=3, border="red")
```



So one way to pick the number of clusters is by "drawing" a horizontal line across the vertical lines of the "dendrogram". The longer the vertical lines then the greater the difference between the clusters. The idea is to draw a vertical line in such a way as to cross over a number of vertical lines with some "wiggle room". We can do this visually or we could use the **cutree** function to do this.

```
library(ggplot2)

iris_3 <- cutree(iris_hclust, k=3)

# iris_3 <-  cutree(iris_hclust, h=3.5)

labs <- factor(iris_3,labels=1:3)

ggplot(iris,aes(x=Sepal.Length,y=Sepal.Width,fill=labs)) +
       geom_point(pch=23,size=3.0) +
       ggtitle("3 Clusters, Dist: Euclidean, Method: Complete ")
```



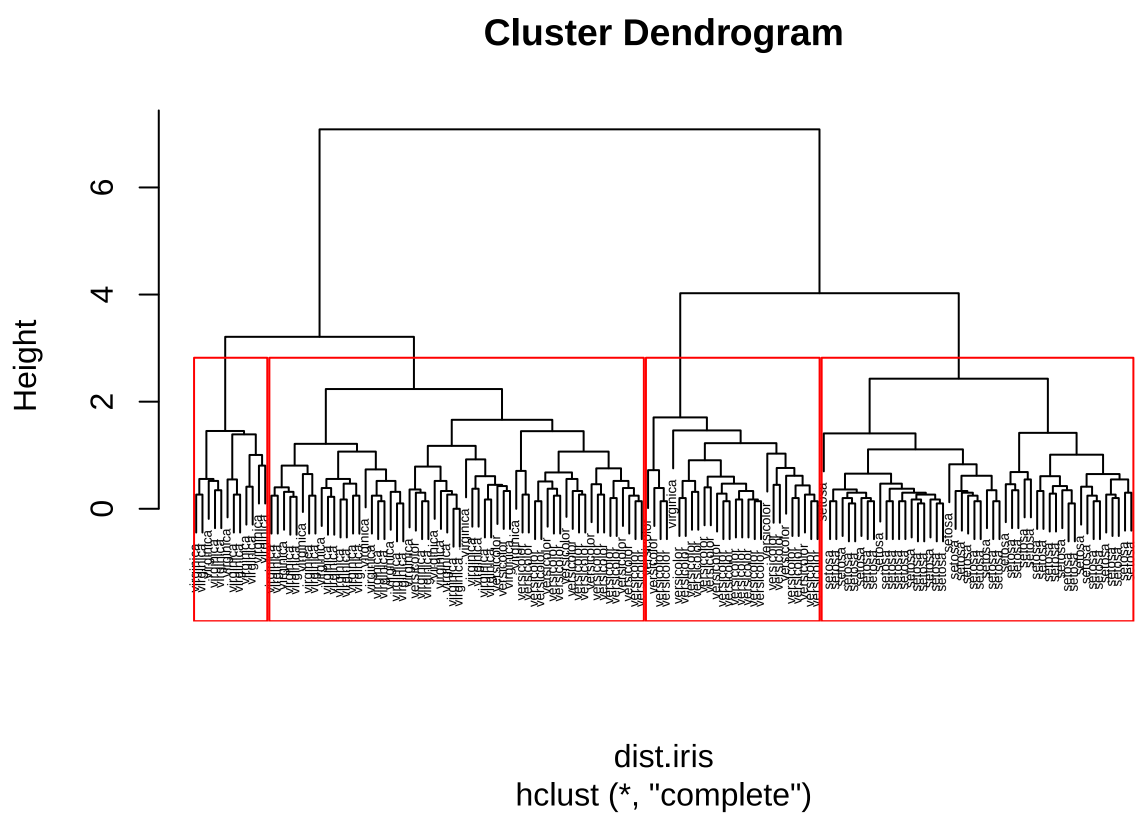
Note that Hierarchical Clustering is not a classification method but since we created 3 clusters let's see how close the clusters match the actual Species given in the iris data frame. It seems like it got all of the Setosas correct, but it got only 23 of the Versicolors correct, and only one of the Virginicas correct. But again, this isn't a classification method.

```
table(cluster=iris_3,species=iris$Species)
```

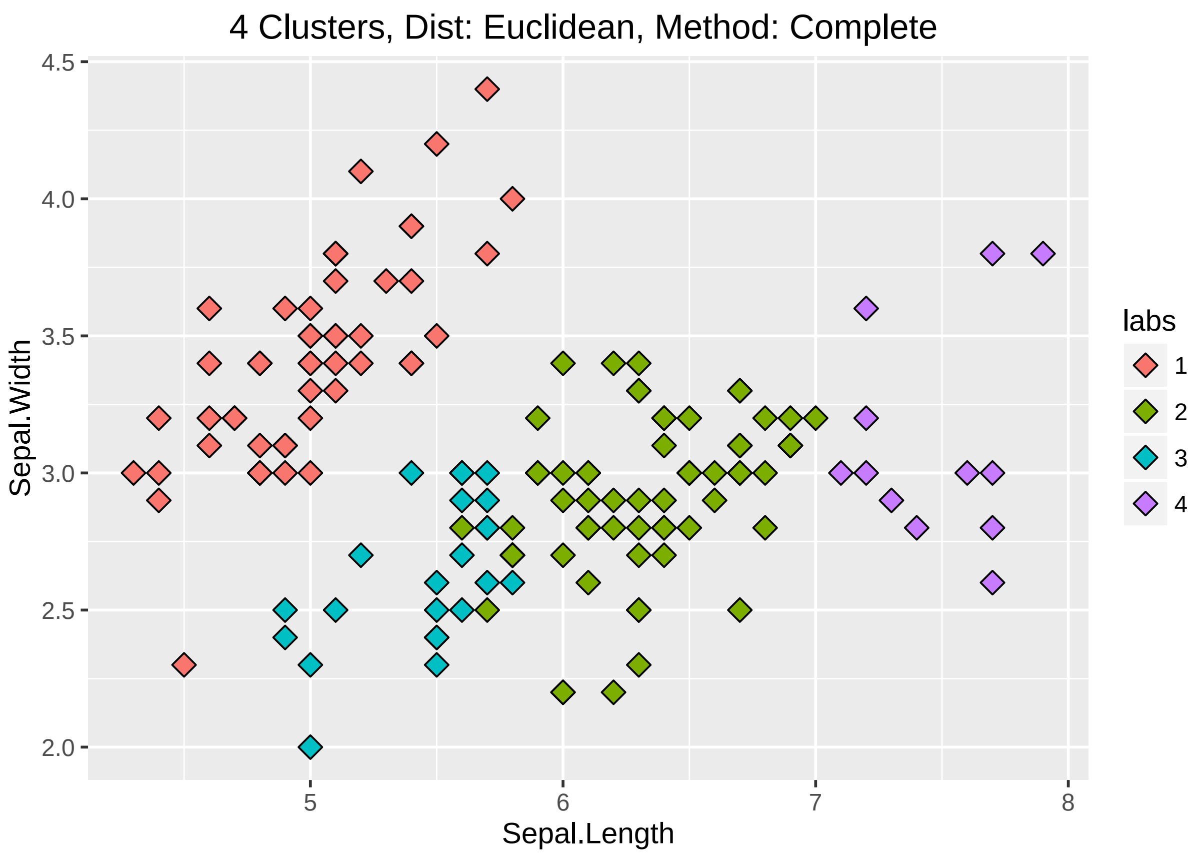
```
##        species
## cluster setosa versicolor virginica
##       1     50          0         0
##       2      0         23        49
##       3      0         27         1
```

We could try a different number of clusters based on the dendrogram.

```
plot(iris_hclust, label=iris$Species, cex=0.4)

rect.hclust(iris_hclust, k=4, border="red")
```



```
iris_4 <- cutree(iris_hclust, k=4)

labs <- factor(iris_4,labels=1:4)

ggplot(iris,aes(x=Sepal.Length,y=Sepal.Width,fill=labs)) +
       geom_point(pch=23,size=3.0) +
       ggtitle("4 Clusters, Dist: Euclidean, Method: Complete ")
```



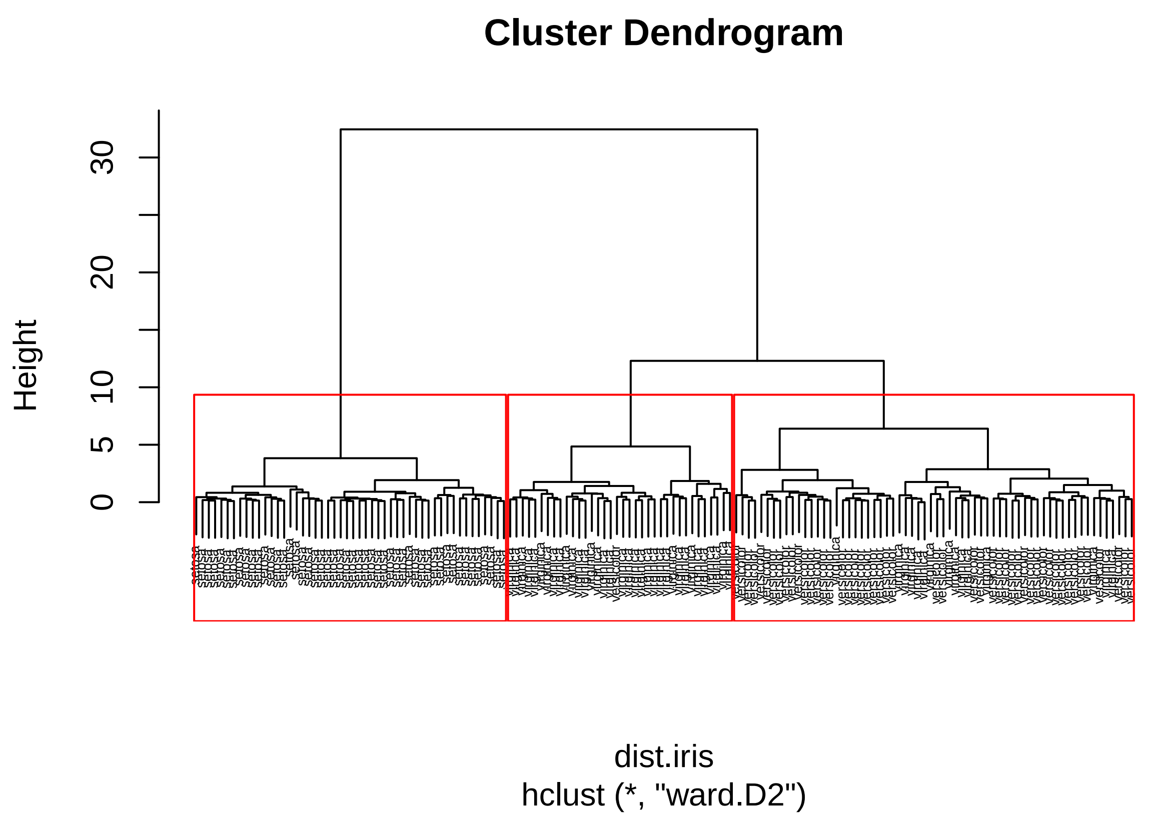
```
table(cluster=iris_4,species=iris$Species)
```

```
##        species
## cluster setosa versicolor virginica
##       1     50          0         0
##       2      0         23        37
##       3      0         27         1
##       4      0          0        12
```

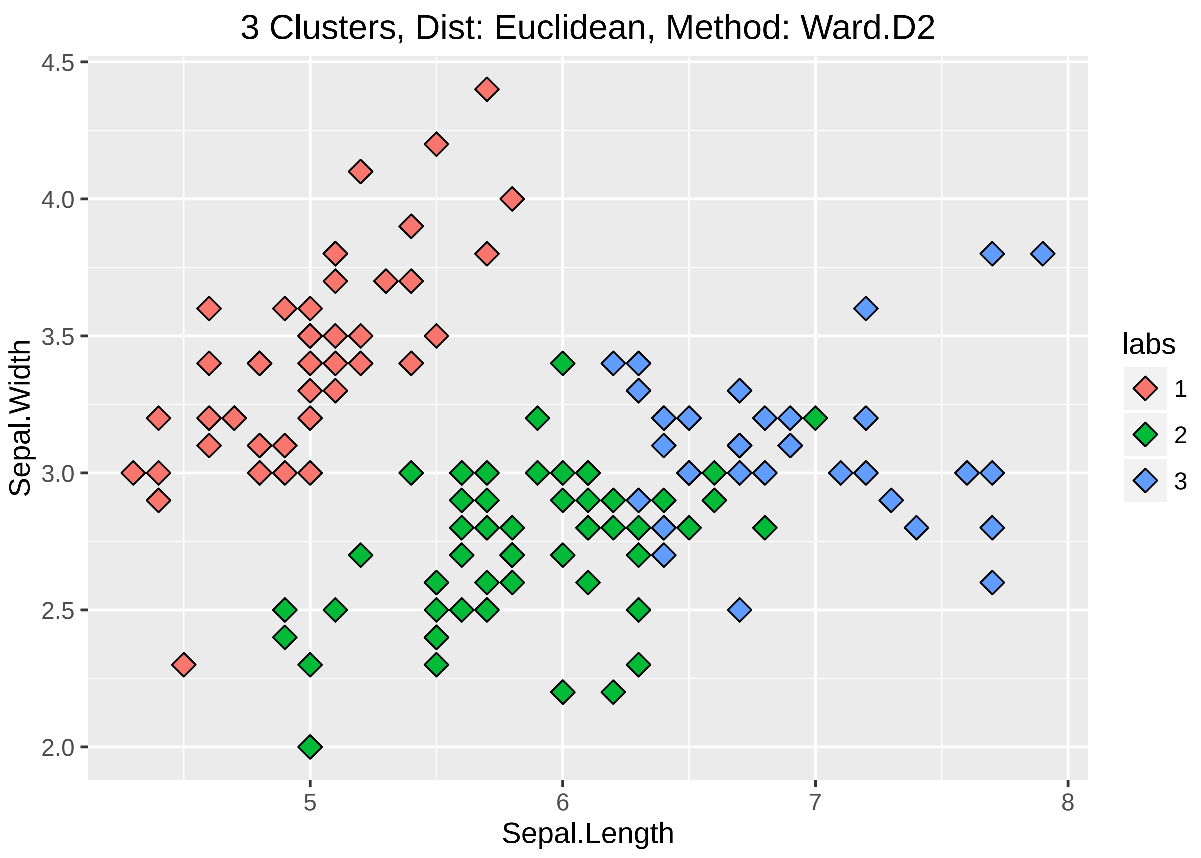
Different Linkage Method

Let's pick a different linkage method. We'll select "Ward". Again, this isn't an attempt to do classification but in this case we see a better match between the actual Species and the predicted clusters.

```
iris_hclust <- hclust(dist.iris, method="ward.D2")

plot(iris_hclust, label=iris$Species, cex=0.4)

rect.hclust(iris_hclust, k=3, border="red")
```



```r
iris_3 <- cutree(iris_hclust, k=3)

labs <- factor(iris_3,labels=1:3)

ggplot(iris,aes(x=Sepal.Length,y=Sepal.Width,fill=labs)) +
       geom_point(pch=23,size=3.0) +
       ggtitle("3 Clusters, Dist: Euclidean, Method: Ward.D2")
```



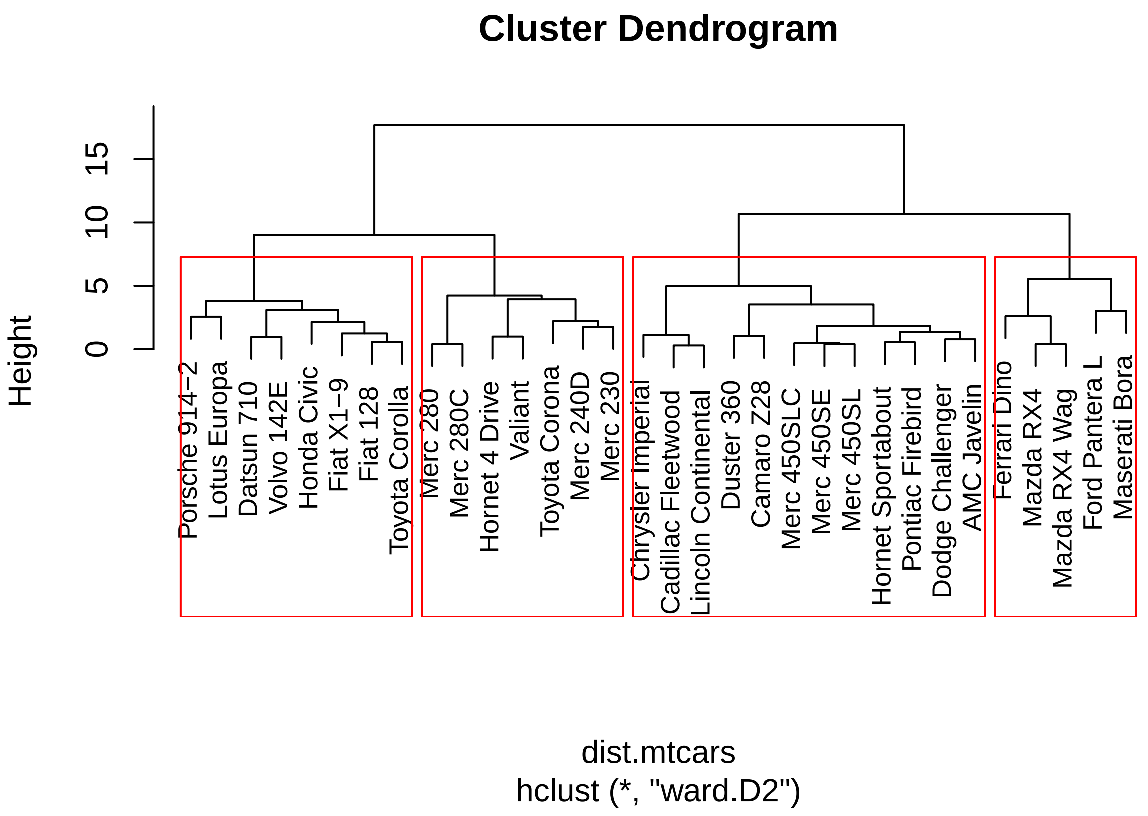
```
table(cluster=iris_3,species=iris$Species)
```

```
##        species
## cluster setosa versicolor virginica
##       1     50          0         0
##       2      0         49        15
##       3      0          1        35
```

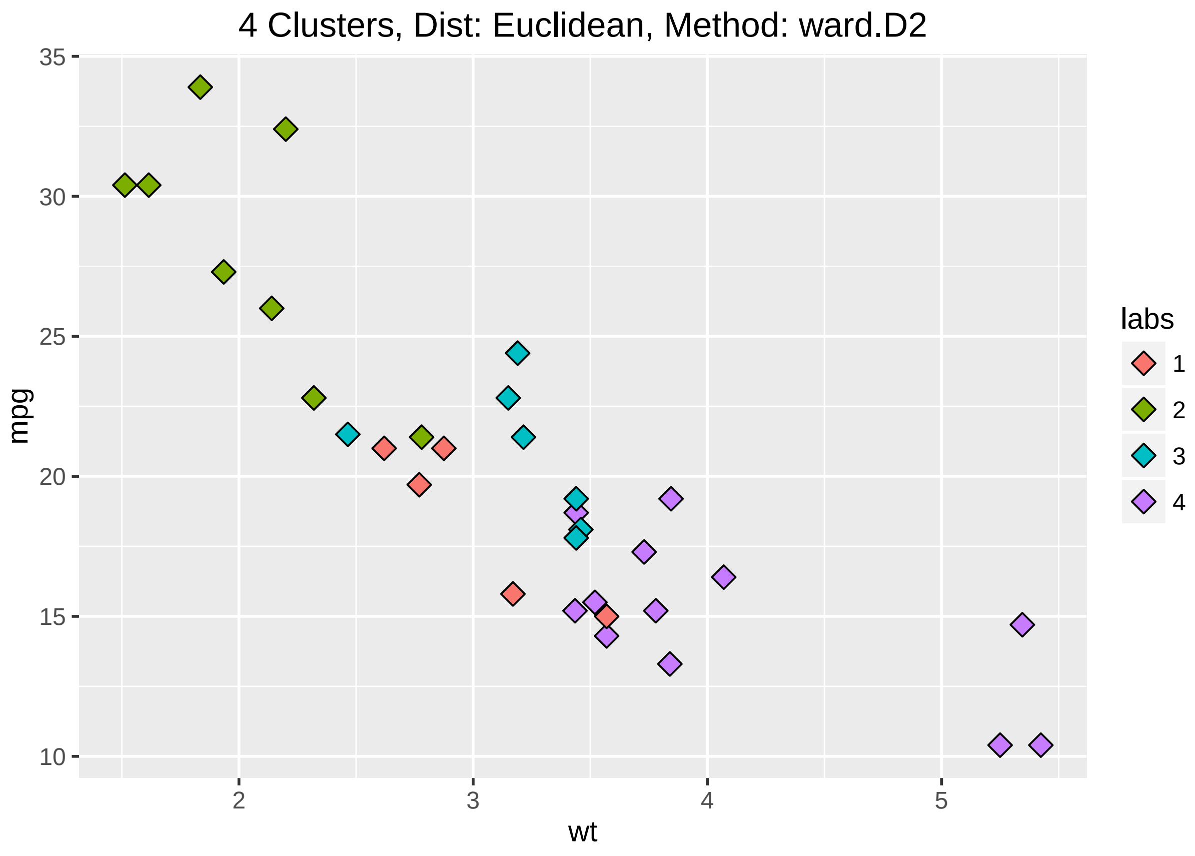
Looking at the mtcars data

We could look at the mtcars data frame and attempt to cluster those cars therein. We have a difference here in that the columns are not all measured on the same scale so we'll need to scale the data.

```
data(mtcars)

scaled.mtcars <- scale(mtcars)

dist.mtcars <- dist(scaled.mtcars)

mtcars_clust <- hclust(dist.mtcars,method="ward.D2")

plot(mtcars_clust,cex=0.8)

# Based on the plot let's try 4 clusters

rect.hclust(mtcars_clust, k=4, border="red")
```



```
mtcars_4 <- cutree(mtcars_clust, k=4)

labs <- factor(mtcars_4,labels=1:4)

ggplot(mtcars,aes(x=wt,y=mpg,fill=labs)) +
       geom_point(pch=23,size=3.0) +
       ggtitle("4 Clusters, Dist: Euclidean, Method: ward.D2")
```



```
table(cluster=mtcars_4,species=mtcars$cyl)
```

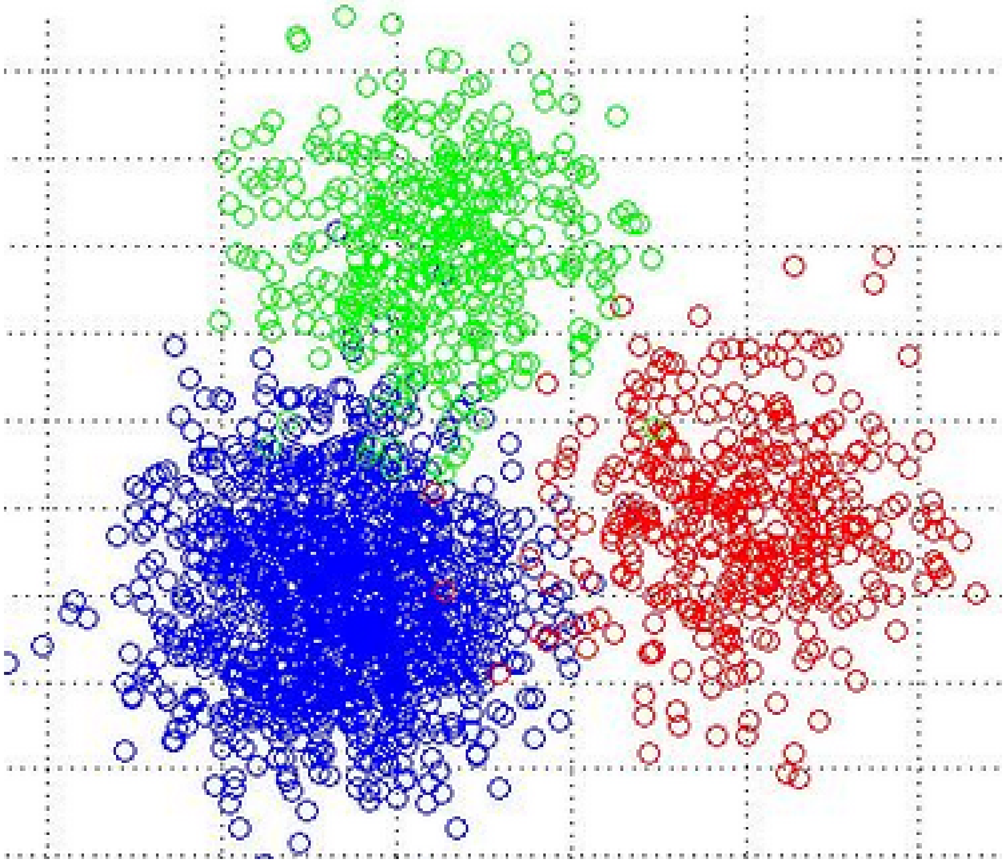
```
##        species
## cluster  4  6  8
##       1  0  3  2
##       2  8  0  0
##       3  3  4  0
##       4  0  0 12
```

K-Means Clustering

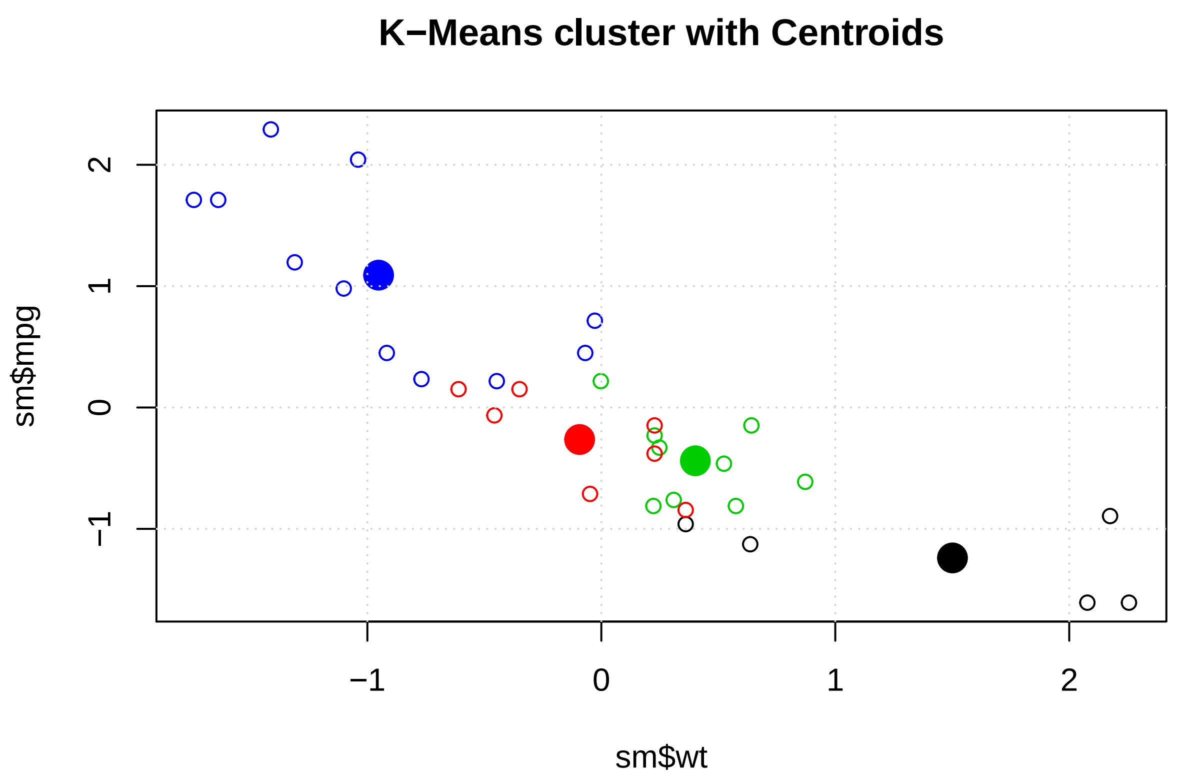
Before looking at a more interesting example let's consider a companion method to Hierarchical Clustering called K-Means cluster. K-Means is different in that we pick the number of clusters in advance. There are some helper functions that can assist us in determining the number of clusters but the idea here is if we have some intuition up front we pick he number of clusters. See http://www.di.fc.ul.pt/~jpn/r/clustering/clustering.html#k-means

1) Pick an initial set of K centroids (this can be random or any other means)
2) For each data point, assign it to the member of the closest centroid according to the given distance function
3) Adjust the centroid position as the mean of all its assigned member data points.
4) Go back to (2) until the membership isn't change and centroid position is stable.
5) Output the centroids.

Visually the result might look like:



```
set.seed(101) # So you will get the same result as me

data(mtcars)

scaled.mtcars <- scale(mtcars)

km <- kmeans(scaled.mtcars,4)
labs <- factor(km$cluster,labels=1:4)
sm <- data.frame(scaled.mtcars)
centers <- as.data.frame(km$centers)

title <- "K-Means cluster with Centroids"
plot(sm$wt,sm$mpg,col=km$cluster,main=title)
points(km$centers[,c(6,1)],col=1:4,pch=19,cex=2)
grid()
```

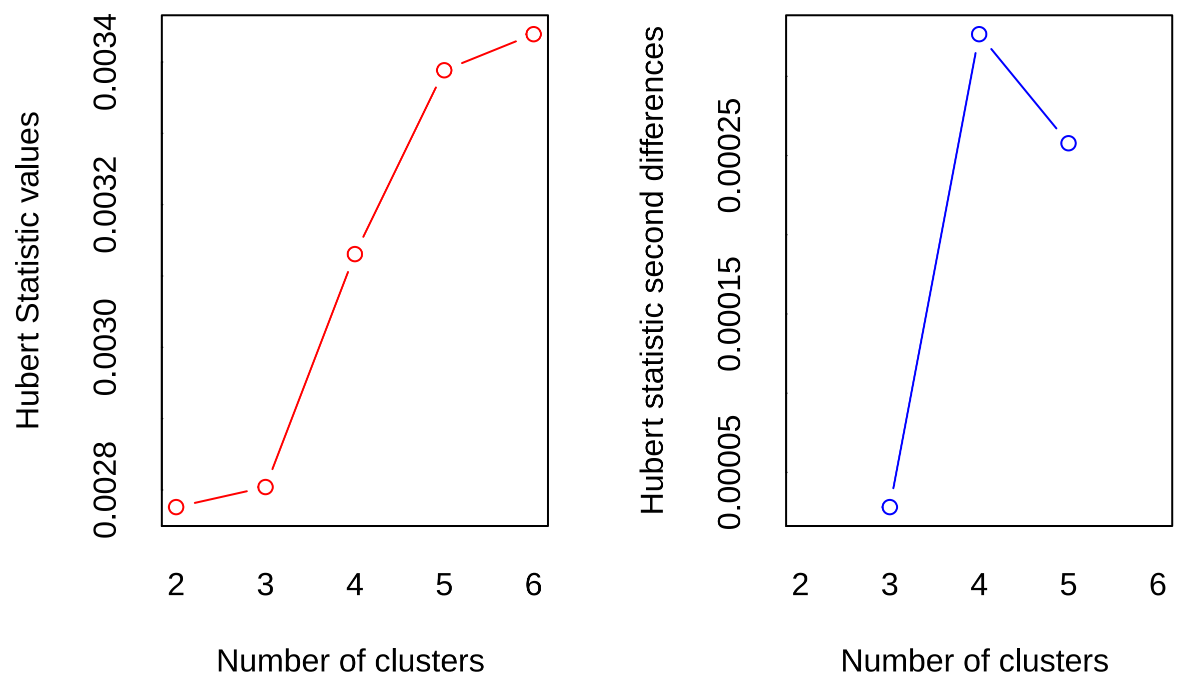


```r
table(cyl=mtcars$cyl,cluster=km$cluster)
```

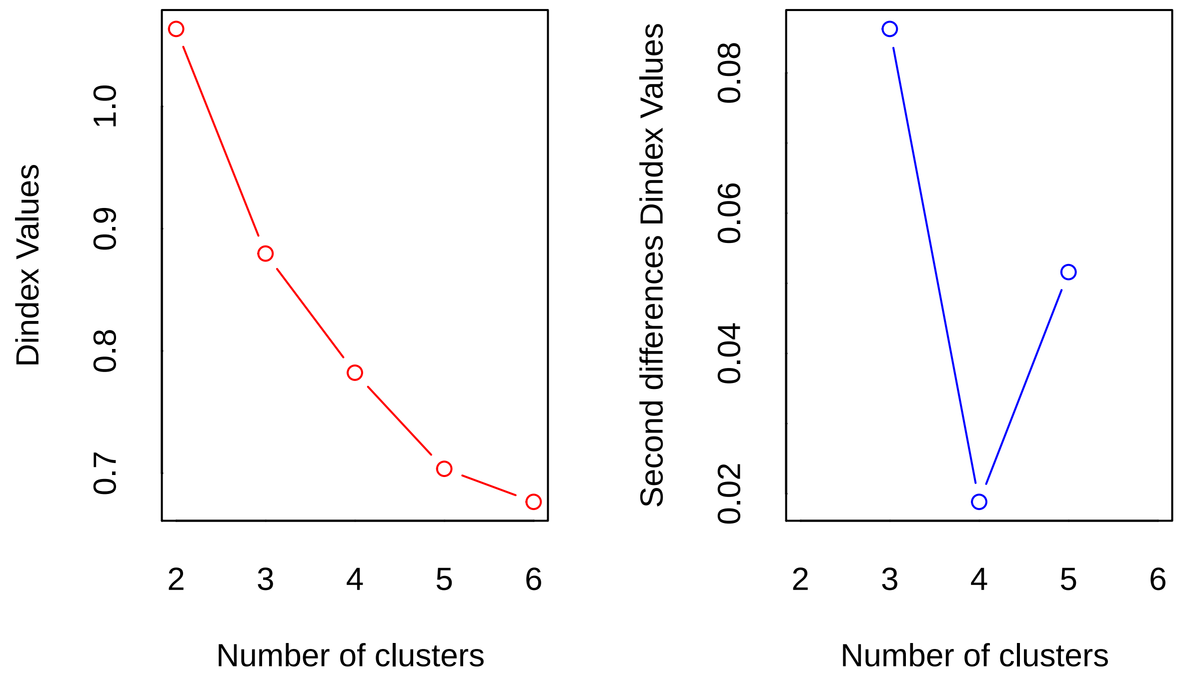
```
##    cluster
## cyl  1  2  3  4
##   4  0  0  0 11
##   6  0  5  2  0
##   8  5  2  7  0
```

There are methods to selecting the so called optimal number of clusters such as noting at what point adding another cluster doesn't contribute to an explanation of the total variance in the data set. This is called the "elbow method". There are packages that can help you with this selection process such as the NbClust package in R.

```r
library(NbClust)

scaled.mtcars <- scale(mtcars)

res <- NbClust(scaled.mtcars,distance="euclidean",
               min.nc=2,max.nc=6,method="ward.D2",
               index="silhouette")


res <- NbClust(scale(iris[,-5]),distance="euclidean",
               min.nc=2,max.nc=6,method="ward.D2",
               index="all")
```



```
## *** : The Hubert index is a graphical method of determining the number of clusters.
##                 In the plot of Hubert index, we seek a significant knee that corresponds to a
##                 significant increase of the value of the measure i.e the significant peak in Hubert
##                 index second differences plot.
##
```



```
## *** : The D index is a graphical method of determining the number of clusters.
##                 In the plot of D index, we seek a significant knee (the significant peak in Dindex
##                 second differences plot) that corresponds to a significant increase of the value of
##                  the measure.
##
## *******************************************************************
## * Among all indices:
## * 10 proposed 2 as the best number of clusters
## * 8 proposed 3 as the best number of clusters
## * 2 proposed 4 as the best number of clusters
## * 3 proposed 5 as the best number of clusters
## * 1 proposed 6 as the best number of clusters
##
##                    ***** Conclusion *****
##
## * According to the majority rule, the best number of clusters is  2
##
##
## *******************************************************************
```

Let's go to the Movies

Collaborative Filtering

	Men in Black	Apollo 13	Top Gun	Terminator
Amy	5	4	5	4
Bob	3		2	5
Carl		5	4	4
Dan	4	2		

- Given this data maybe we suggest to Carl that he watch "Men in Black" since Amy liked it and they appear to have similar preferences. This is known as **Collaborative Filtering**
- Can suggest things without understanding the underlying attributes of the movie
- There can be lots of information on each user times the total number of users ! Big Data

Content Filtering

Consider the following line of thinking. Amy liked "Men in Black". We don't necessarily know why unless she tells us or makes some comments in a forum that we monitor. But based on the information that we have we can make some recommendations using the following considerations. The movie "Men in Black":

- Was directed by Barry Sonnenfeld so maybe recommend "Get Shorty" (directed by Sonnenfeld)
- Is classified in Action, Adventure, Sci-Fi, Comedy so maybe recommend "Jurassic Park" which is similarly classfied
- Stars Will Smith so maybe recommend another Will Smith comedy move like "Hutch"
- Also stars Tommy Lee Jones so maybe recommend "Space Cowboys"

Content filtering requires little data to get started. Content-based filtering methods are based on a description of the item and maybe a profile of the user's preference

Hybrid Recommenders

Hybrid recommendation systems are also a possibility and offer a strong solutions:

- A collaborative filtering approach that finds that Amy and Carl have similar preferences
- After that we could then do content filtering where we find that "Terminator", which both Amy and Carl liked is also classified in the same set of genres as is "Starship Troopers"
- So recommend "Starship Troopers" even though neither have seen it

Movie Lens

Movies in the dataset are categorized according to 18 different genres. Each movie can belong to different genres. Can we systematically find groups of movies with similar sets of genres ?

Unsupervised learning goal is to segment the data into similar groups instead of doing predictions as to what group it belongs in. But once we observe the clusters we could then come up with some hypothesis about important variables that could be used in a classification problem.

So we put each data point into a group with "similar" values. It doesn't predict anything. This works best for large data sets. many different algorithms. we'll do hierarchical and k-means

Hierarchical Clustering

Each data point starts in its own cluster. Then HC combines the two nearest into one clust

R Markdown

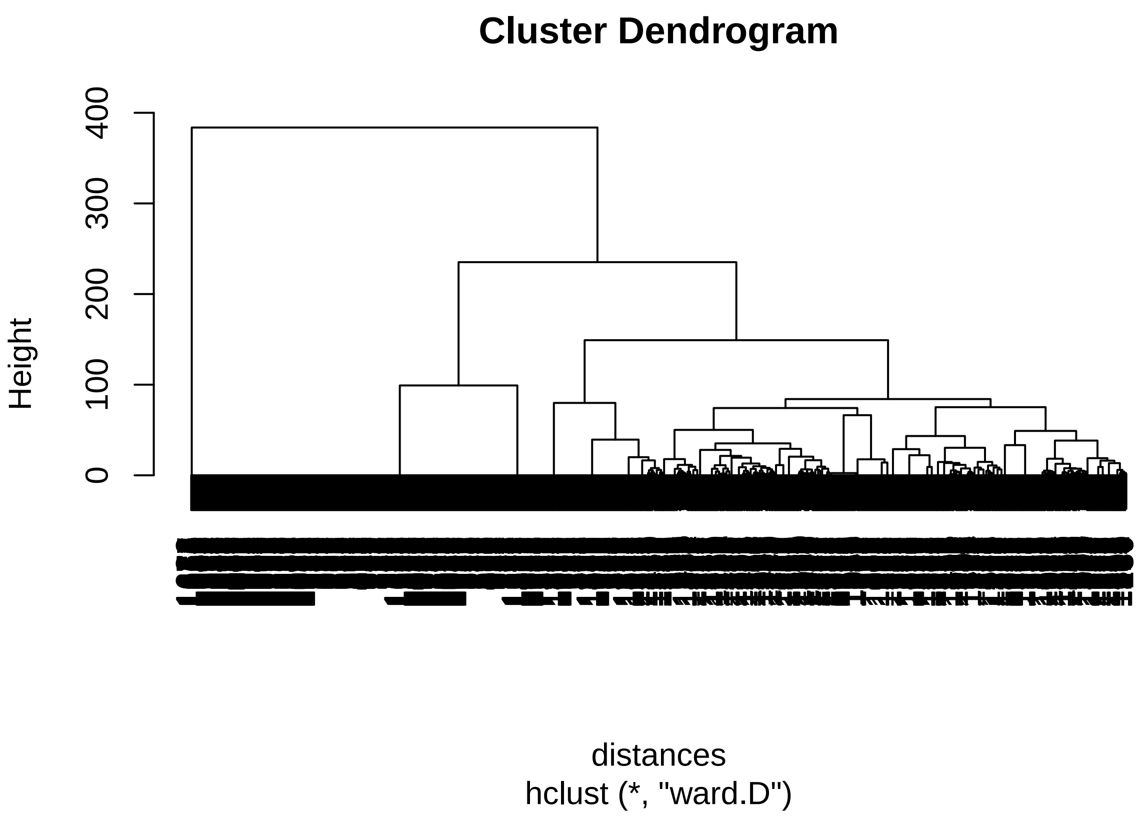
```r
suppressMessages(library(dplyr))
url <- "http://files.grouplens.org/datasets/movielens/ml-100k/u.item"

movies <- read.table(url,header=FALSE,sep="|",quote="\"",stringsAsFactors = FALSE)

mnames <- c("ID","Title","ReleaseDate","VideoReleaseDate",
             "IMDB","Unknown","Action","Adventure","Animation",
             "Childrens","Comedy","Crime","Documentary",
             "Drama","Fantasy","FilmNoir","Horror","Musical",
              "Mystery","Romance","SciFi","Thriller",
              "War","Western")


colnames(movies) <- mnames

movies <- select(movies,-ID,-ReleaseDate,-VideoReleaseDate,-IMDB) %>% unique
```

Okay how to analyze this stuff ? Let's cluster the movies by genre. Let's see how we can make recs. First we compute all distances then cluster

```r
distances <- dist(movies[,2:20],method="euclidean")
clusterMovies <- hclust(distances, method="ward")
```

```
## The "ward" method has been renamed to "ward.D"; note new "ward.D2"
```

```
plot(clusterMovies)
```



We can label each point by the cluster it belongs to

```
clusterGroups <- cutree(clusterMovies, k=10)

# Okay let's see how many Action Movies show up in each cluster - we want to compute a percentage

myt <- table(movies=movies$Action,clusters=clusterGroups)

# Okay divide the number of actual action movies (row 2) by the column sum to get the
# percentages. We now know the percentage of Action movies in each cluster.

myt[2,]/colSums(myt)
```

```
##         1         2         3         4         5         6         7
## 0.1784512 0.7839196 0.1238532 0.0000000 0.0000000 0.1015625 0.0000000
##         8         9        10
## 0.0000000 0.0000000 0.0000000
```

```
# We should do this for each genre. Another way to do all this is this way

tapply(movies$Action,clusterGroups,mean)
```

```
##         1         2         3         4         5         6         7
## 0.1784512 0.7839196 0.1238532 0.0000000 0.0000000 0.1015625 0.0000000
##         8         9        10
## 0.0000000 0.0000000 0.0000000
```

```
# note the next line is pretty advanced - it's just a quick way to avoid for loops

myclusters <- data.frame(t(sapply(3:20,function(x) tapply(movies[,x],clusterGroups,mean))))

myclusters <- data.frame(sapply(myclusters,round,2))
rownames(myclusters) <- names(movies)[3:20]
colnames(myclusters) <- paste("C",1:10,sep="")
```

Inspect the clusters for meaningful content

- Cluster 1 has lots of different movie types so it's like a Misc category.
- Cluster 2 has lots of Action, Adventure, and Sci-fi
- Cluster 3 has lots of Crime, Mystery, and Thriller.
- Cluster 4 has ONLY Drama movies
- Cluster 5 has ONLY Comedy movies
- Cluster 6 has mostly Romance movies
- Cluster 7 has lots of Comedy and Romance movies
- Cluster 8 has Documentary
- Cluster 9 has Comedy and Drama
- Cluster 10 has mostly Horror

So what if we like the movie Men in Black. What other movies might we want to see based on our clustering scheme ?

```
movies[movies$Title=="Men in Black (1997)",]
```

```
##                     Title Unknown Action Adventure Animation Childrens
## 257 Men in Black (1997)       0      1         1         0         0
##     Comedy Crime Documentary Drama Fantasy FilmNoir Horror Musical Mystery
## 257      1     0           0     0       0        0      0       0       0
##     Romance SciFi Thriller War Western
## 257       0     1        0   0       0
```

```
# So row 257 corresponds to MIB. Which cluster did 257 go into

clusterGroups[257]
```

```
## 257
##   2
```

```
cluster2 <- movies[clusterGroups == 2,]
cluster2$Title[1:20]
```

```
##  [1] "GoldenEye (1995)"
##  [2] "Bad Boys (1995)"
##  [3] "Apollo 13 (1995)"
##  [4] "Net, The (1995)"
##  [5] "Natural Born Killers (1994)"
##  [6] "Outbreak (1995)"
##  [7] "Stargate (1994)"
##  [8] "Fugitive, The (1993)"
##  [9] "Jurassic Park (1993)"
## [10] "Robert A. Heinlein's The Puppet Masters (1994)"
## [11] "Terminator 2: Judgment Day (1991)"
## [12] "Heavy Metal (1981)"
## [13] "Mystery Science Theater 3000: The Movie (1996)"
## [14] "Rock, The (1996)"
## [15] "Twister (1996)"
## [16] "Supercop (1992)"
## [17] "Die Hard (1988)"
## [18] "Lawnmower Man, The (1992)"
## [19] "Long Kiss Goodnight, The (1996)"
## [20] "Ghost and the Darkness, The (1996)"
```