Chapter 2. Linear Regression – The Blocking and Tackling of Machine Learning

Univariate linear regression

The model for this can be written as, Y = B0 + B1x + e.

The least squares approach chooses the model parameters that minimize the Residual Sum of Squares (RSS) of the predicted y values versus the actual Y values.

For a simple example, let's say we have the actual values of Y1 and Y2 equal to 10 and 20 respectively, along with the predictions of y1 and y2 as 12 and 18. To calculate RSS, we add the squared differences RSS = (Y1 – y1)2 + (Y2 – y2)2, which, with simple substitution, yields (10 – 12)2 + (20 – 18)2 = 8.

```
data("anscombe")

attach(anscombe)

anscombe
```

```
##    x1 x2 x3 x4    y1   y2    y3    y4
## 1  10 10 10  8  8.04 9.14  7.46  6.58
## 2   8  8  8  8  6.95 8.14  6.77  5.76
## 3  13 13 13  8  7.58 8.74 12.74  7.71
## 4   9  9  9  8  8.81 8.77  7.11  8.84
## 5  11 11 11  8  8.33 9.26  7.81  8.47
## 6  14 14 14  8  9.96 8.10  8.84  7.04
```

```
## 7   6  6  6  8  7.24 6.13  6.08  5.25
## 8   4  4  4 19  4.26 3.10  5.39 12.50
## 9  12 12 12  8 10.84 9.13  8.15  5.56
## 10  7  7  7  8  4.82 7.26  6.42  7.91
## 11  5  5  5  8  5.68 4.74  5.73  6.89
```

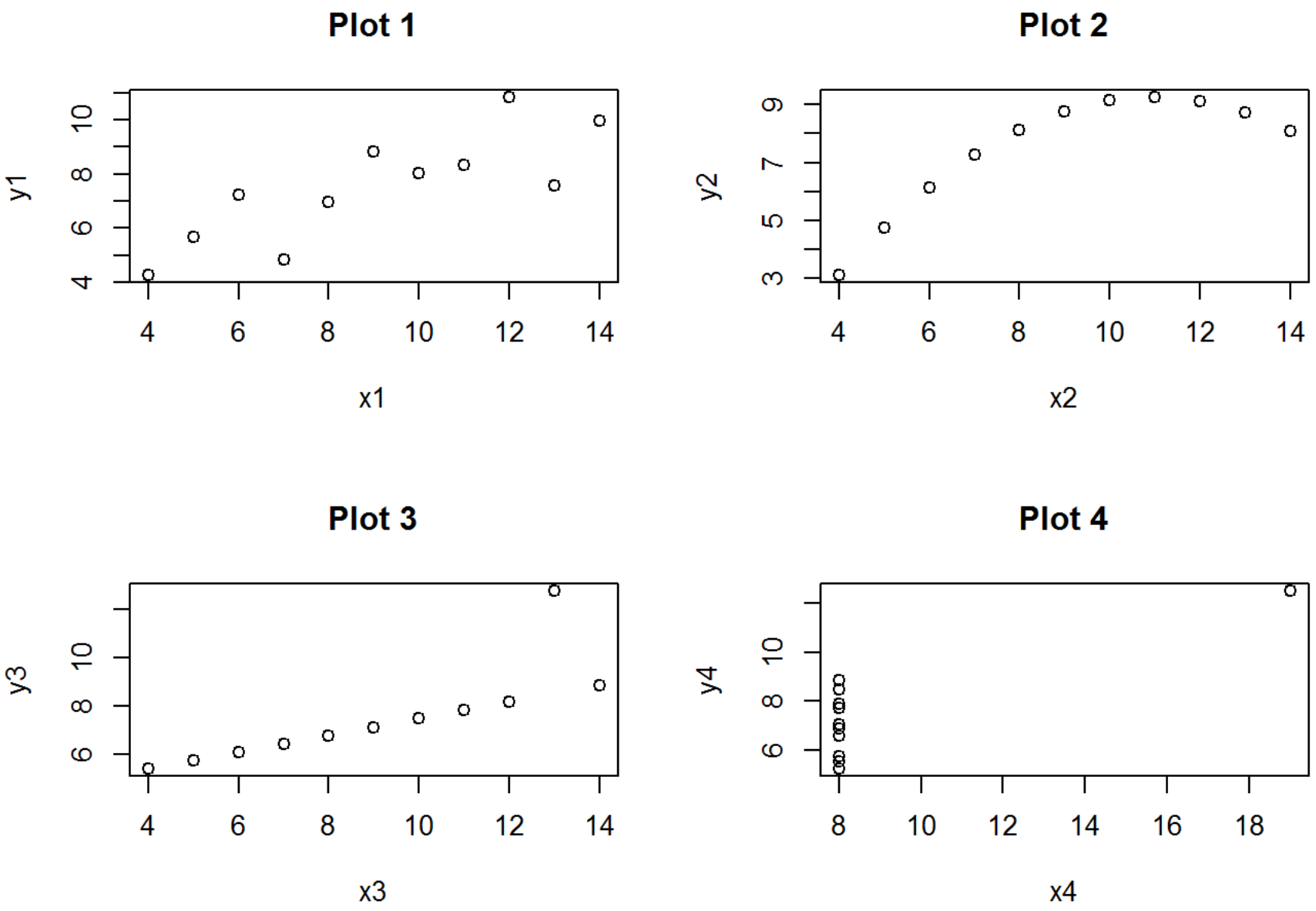
```
cor(x1, y1) ##Correlation of x1 and y1
```

```
## [1] 0.8164205
```

```
cor(x2, y2) ##Correlation of x2 and y2
```

```
## [1] 0.8162365
```

```
par(mfrow=c(2,2)) #create a 2x2 grid for plotting

plot(x1, y1, main="Plot 1")

plot(x2, y2, main="Plot 2")

plot(x3, y3, main="Plot 3")

plot(x4, y4, main="Plot 4")
```



As we can see, Plot 1 appears to have a true linear relationship, Plot 2 is curvilinear, Plot 3 has a dangerous outlier, and Plot 4 is driven by the one outlier. There you have it, a cautionary tale of sorts.

Business understanding

Let’s Analyze another example. Snake Dataset

```
require(alr3)
```

```
## Loading required package: alr3
```

```
## Loading required package: car
```

```
data("snake")

dim(snake)
```

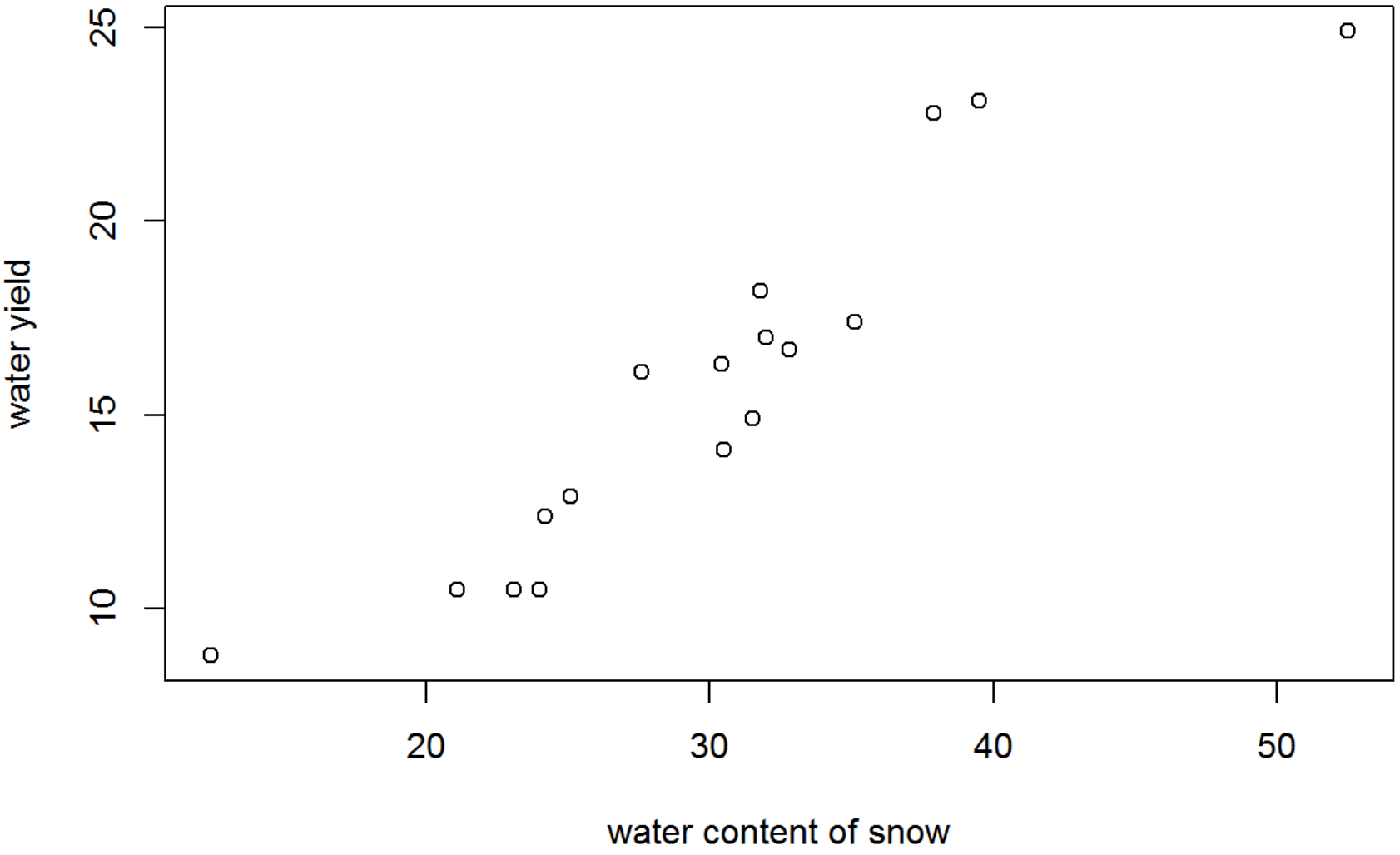
```
## [1] 17  2
```

```
names(snake) <- c("Content", "Yield")

attach(snake)

head(snake)
```

```
##   Content Yield
## 1    23.1  10.5
## 2    32.8  16.7
## 3    31.8  18.2
## 4    32.0  17.0
```

```
## 5    30.4  16.3
## 6    24.0  10.5
```

```
par(mfrow=c(1,1)) #create a 1x1 grid for plotting

plot(Content, Yield, xlab="water content of snow", ylab="water yield")
```



There are potential two outliers that might drive linear to be a non linear relationship

To perform a linear regression in R, one uses the lm()

```r
yield.fit = lm(Yield~Content)

summary(yield.fit)
```

```
##
## Call:
## lm(formula = Yield ~ Content)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -2.1793 -1.5149 -0.3624  1.6276  3.1973
##
## Coefficients:
##             Estimate Std. Error t value Pr(>|t|)
## (Intercept)  0.72538    1.54882   0.468    0.646
## Content      0.49808    0.04952  10.058 4.63e-08 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 1.743 on 15 degrees of freedom
## Multiple R-squared:  0.8709, Adjusted R-squared:  0.8623
## F-statistic: 101.2 on 1 and 15 DF,  p-value: 4.632e-08
```

Yield = 0.72538 + 0.49808 * content

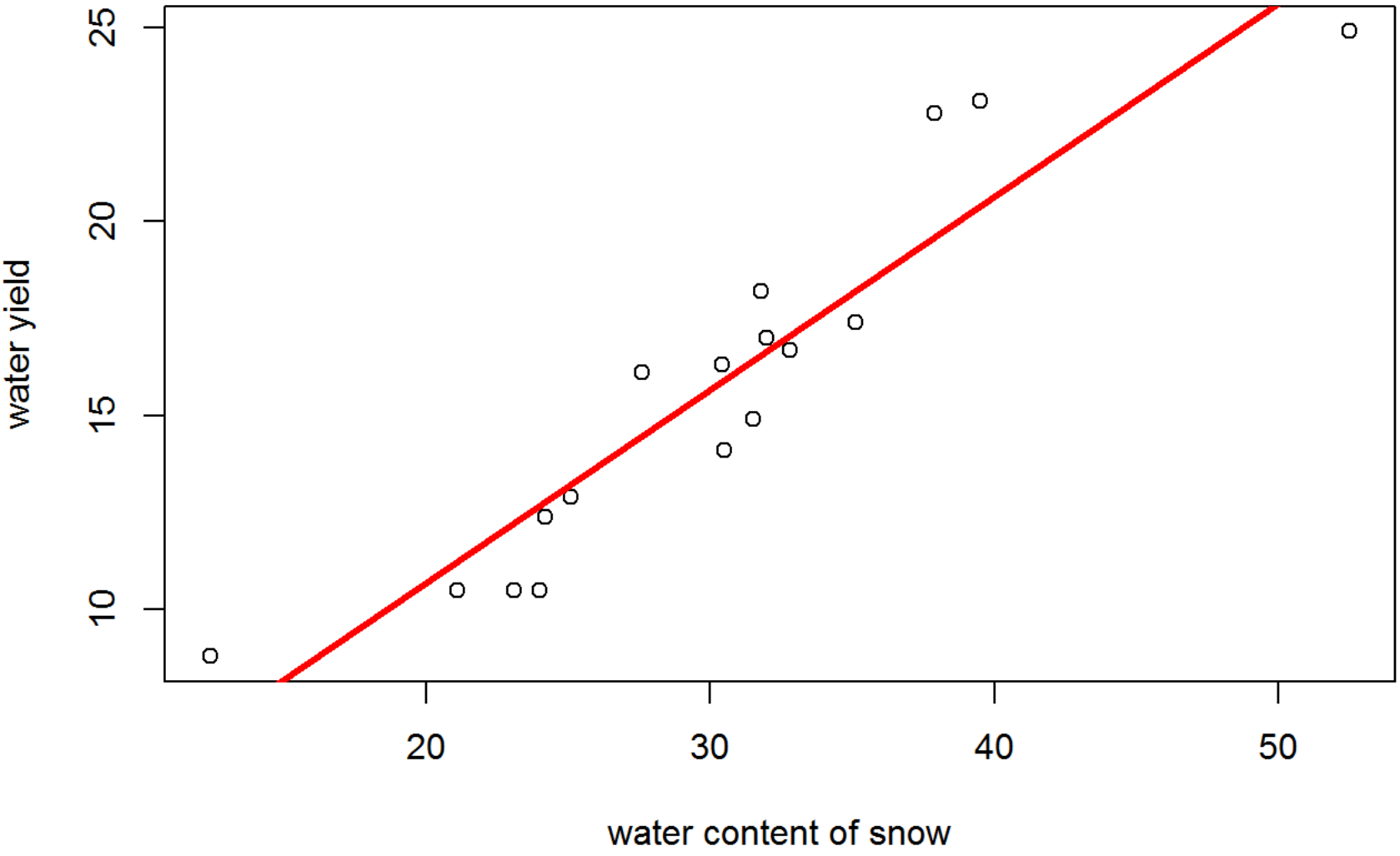
We can observe Summary yield “Adjusted R Squared” and “Multiple R Squared”. The Adjusted R-squared will be covered under the multivariate regression topic. We see that Multiple R Squared = 0.8709, which ranges from 0 and 1 and measures the strength of the association between X and Y.

The interpretation in this case is that 87 percent of the variation in the water yield can be explained by the water content of snow

On a side note, R-squared is nothing more than the correlation coefficient of [X, Y] squared.

We can add the best fit line as follows

```
plot(Content, Yield, xlab="water content of snow", ylab="water yield")

abline(yield.fit, lwd=3, col="red")
```



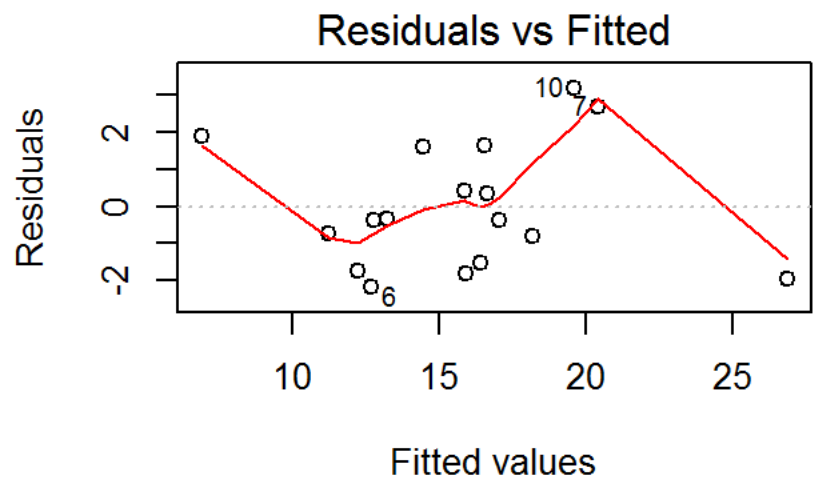
Assumptions in Linear Regression

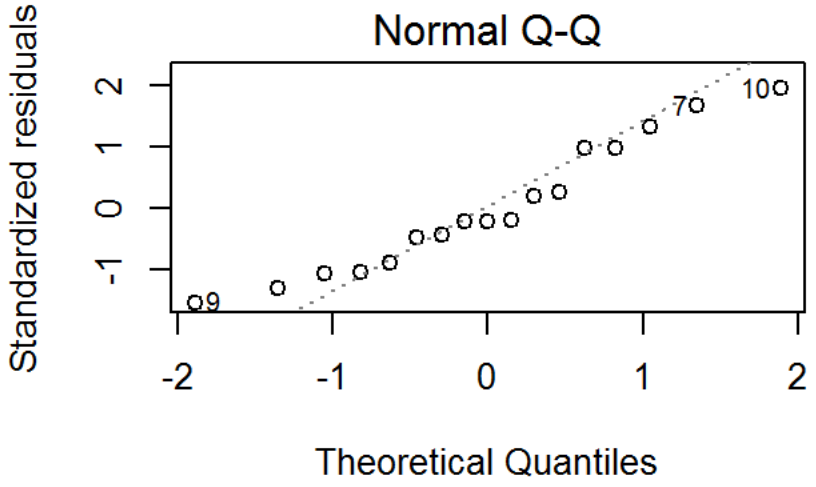
- Linearity: This is a linear relationship between the predictor and the response variables. If this relationship is not clearly present, transformations (log, polynomial, exponent and so on) of the X or Y may solve the problem.

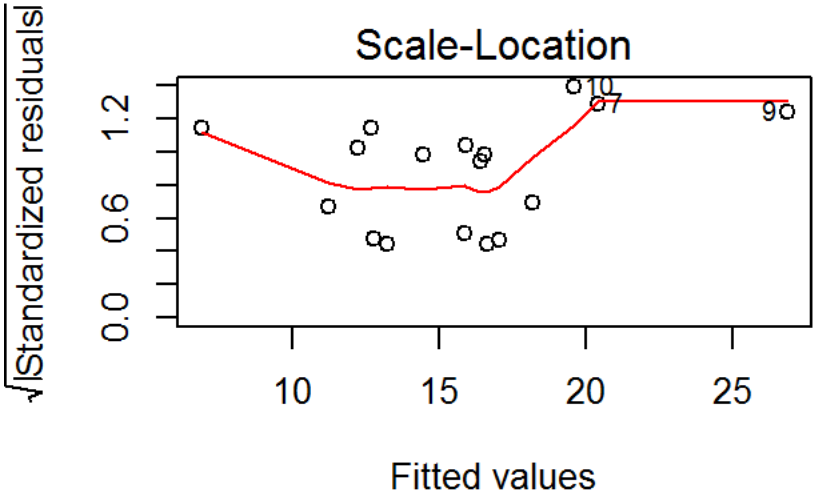
- Non-correlation of errors: If the errors are correlated, you run the risk of creating a poorly specified model.
- Homoscedasticity: The variance of the errors is constant across the different values of inputs. Violations of this assumption can create biased coefficient estimates
- No collinearity: There should be no correlation between the features. This, again, can lead to biased estimates.
- Presence of outliers: Outliers can severely skew the estimation and, ideally, must be removed prior to fitting a model using linear regression; this again can lead to a biased estimate.

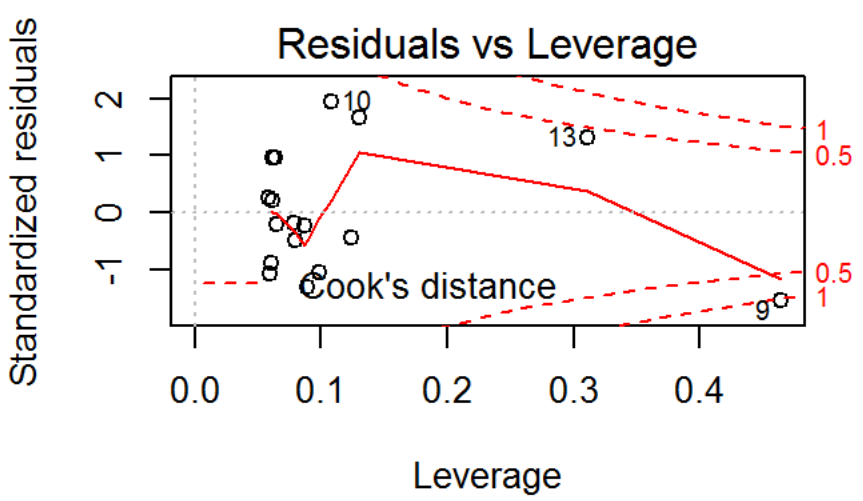
Best way to check the assumptions are is by producing plots.

```
par(mfrow=c(2,2))

plot(yield.fit)
```









The two plots on the left allow us to examine the homoscedasticity of errors and nonlinearity

Common shapes that violates homoscedasticity are when errors appear

- u-shaped

- inverted u-shaped
- cluster close together on the left side of the plot and become wider as the fitted values increase (a funnel shape)

Nothing of that sort appears in our model (we also have only 17 obs)

Normal Q-Q plot to determine if the residuals are normally distributed

The outliers (observations 7, 9, and 10), may be causing a violation of the assumption

The Residuals vs Leverage plot can tell us what observations, if any, are unduly influencing the model; in other words, if there are any outliers we should be concerned about

The statistic is Cook's distance or Cook's D, and it is generally accepted that a value greater than one should be worthy of further inspection

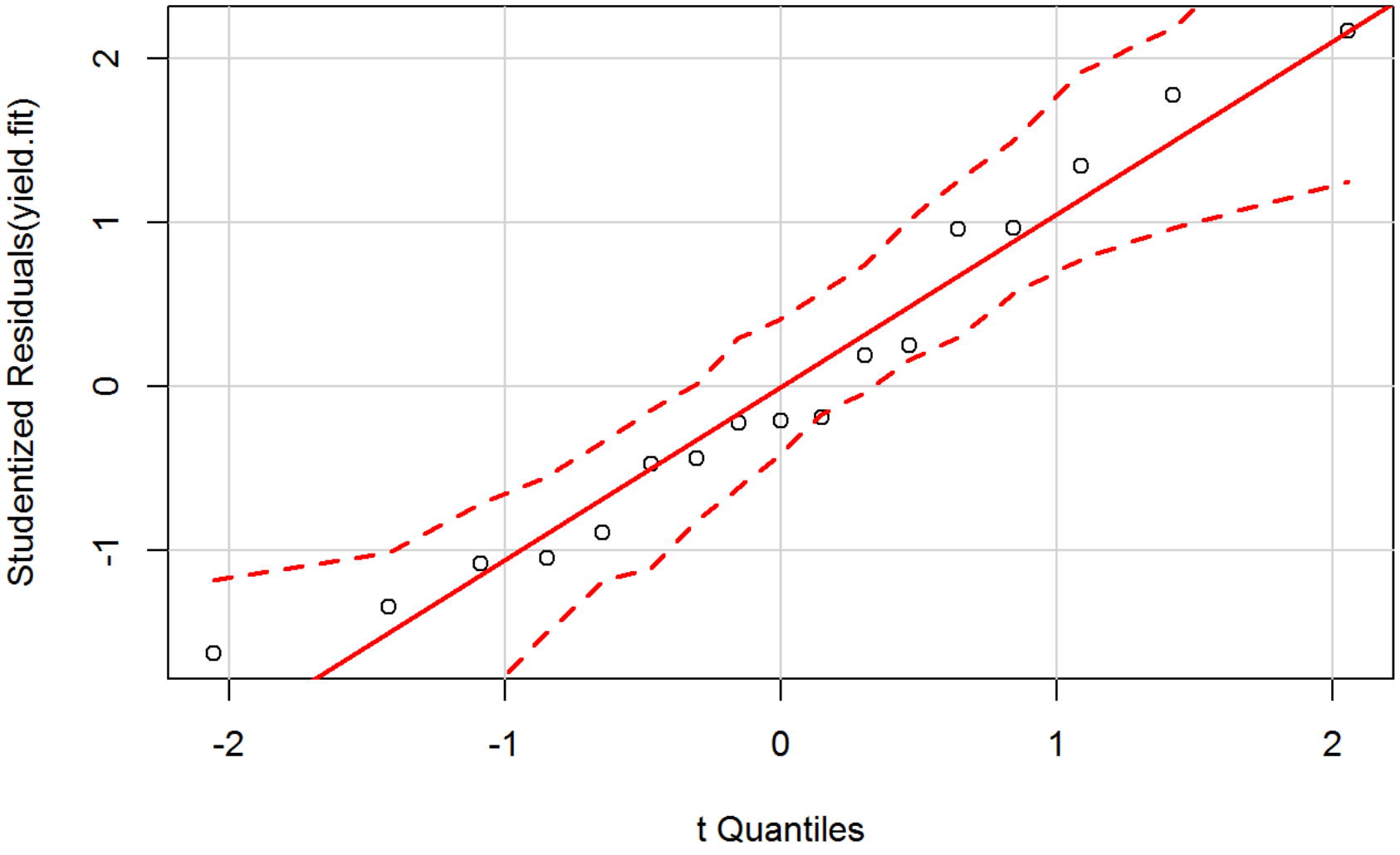
The easy way out would be to simply delete the observation, in this case number 9, and redo the model.

If we just delete observation 9, then maybe observations 10 and 13 would fall outside the band of greater than 1

However, a better option may be to transform the predictor and/or the response variables

R does not provide confidence intervals to the default Q-Q plot, and given our concerns in looking at the base plot, we should check the confidence intervals

```
par(mfrow=c(1,1))
qqPlot(yield.fit)
```



According to the plot, the residuals are normally distributed. I think this can give us some confidence to select the model with all the observations. Clear rationale and judgment would be needed to attempt other models. If we could clearly reject the assumption of normally distributed errors, then we would probably have to examine the variable transformations and/or observation deletion.

Multivariate linear regression

Y = B0 + B1x1 +…Bnxn + e, where the predictor variables (features) can be from 1 to n.

Data understanding and preparation

```
data("water")

str(water)
```

```
## 'data.frame':    43 obs. of  8 variables:
##  $ Year   : int  1948 1949 1950 1951 1952 1953 1954 1955 1956 1957 ...
##  $ APMAM  : num  9.13 5.28 4.2 4.6 7.15 9.7 5.02 6.7 10.5 9.1 ...
##  $ APSAB  : num  3.58 4.82 3.77 4.46 4.99 5.65 1.45 7.44 5.85 6.13 ...
##  $ APSLAKE: num  3.91 5.2 3.67 3.93 4.88 4.91 1.77 6.51 3.38 4.08 ...
##  $ OPBPC  : num  4.1 7.55 9.52 11.14 16.34 ...
##  $ OPRC   : num  7.43 11.11 12.2 15.15 20.05 ...
##  $ OPSLAKE: num  6.47 10.26 11.35 11.13 22.81 ...
##  $ BSAAM  : int  54235 67567 66161 68094 107080 67594 65356 67909 92715 70024 ...
```

Excluding the Year

```
socal.water = water[ ,-1] #new dataframe with the deletion of column 1

head(socal.water)
```

```
##   APMAM APSAB APSLAKE OPBPC  OPRC OPSLAKE  BSAAM
## 1  9.13  3.58    3.91  4.10  7.43    6.47  54235
## 2  5.28  4.82    5.20  7.55 11.11   10.26  67567
```

```
## 3  4.20  3.77    3.67  9.52 12.20   11.35  66161
## 4  4.60  4.46    3.93 11.14 15.15   11.13  68094
## 5  7.15  4.99    4.88 16.34 20.05   22.81 107080
## 6  9.70  5.65    4.91  8.88  8.15    7.41  67594
```

With all the features being quantitative, it makes sense to look at the correlation statistics and then produce a matrix of scatterplots.

The correlation coefficient or Pearson's r, is a measure of both the strength and direction of the linear relationship between two variables. The statistic will be a number between -1 and 1 where -1 is the total negative correlation and +1 is the total positive correlation.

```
water.cor <- cor(socal.water)

water.cor
```

```
##              APMAM      APSAB    APSLAKE      OPBPC      OPRC     OPSLAKE
## APMAM   1.0000000 0.82768637 0.81607595 0.12238567 0.1544155 0.10754212
## APSAB   0.8276864 1.00000000 0.90030474 0.03954211 0.1056396 0.02961175
## APSLAKE 0.8160760 0.90030474 1.00000000 0.09344773 0.1063836 0.10058669
## OPBPC   0.1223857 0.03954211 0.09344773 1.00000000 0.8647073 0.94334741
## OPRC    0.1544155 0.10563959 0.10638359 0.86470733 1.0000000 0.91914467
## OPSLAKE 0.1075421 0.02961175 0.10058669 0.94334741 0.9191447 1.00000000
## BSAAM   0.2385695 0.18329499 0.24934094 0.88574778 0.9196270 0.93843604
##              BSAAM
## APMAM   0.2385695
## APSAB   0.1832950
## APSLAKE 0.2493409
## OPBPC   0.8857478
## OPRC    0.9196270
## OPSLAKE 0.9384360
```

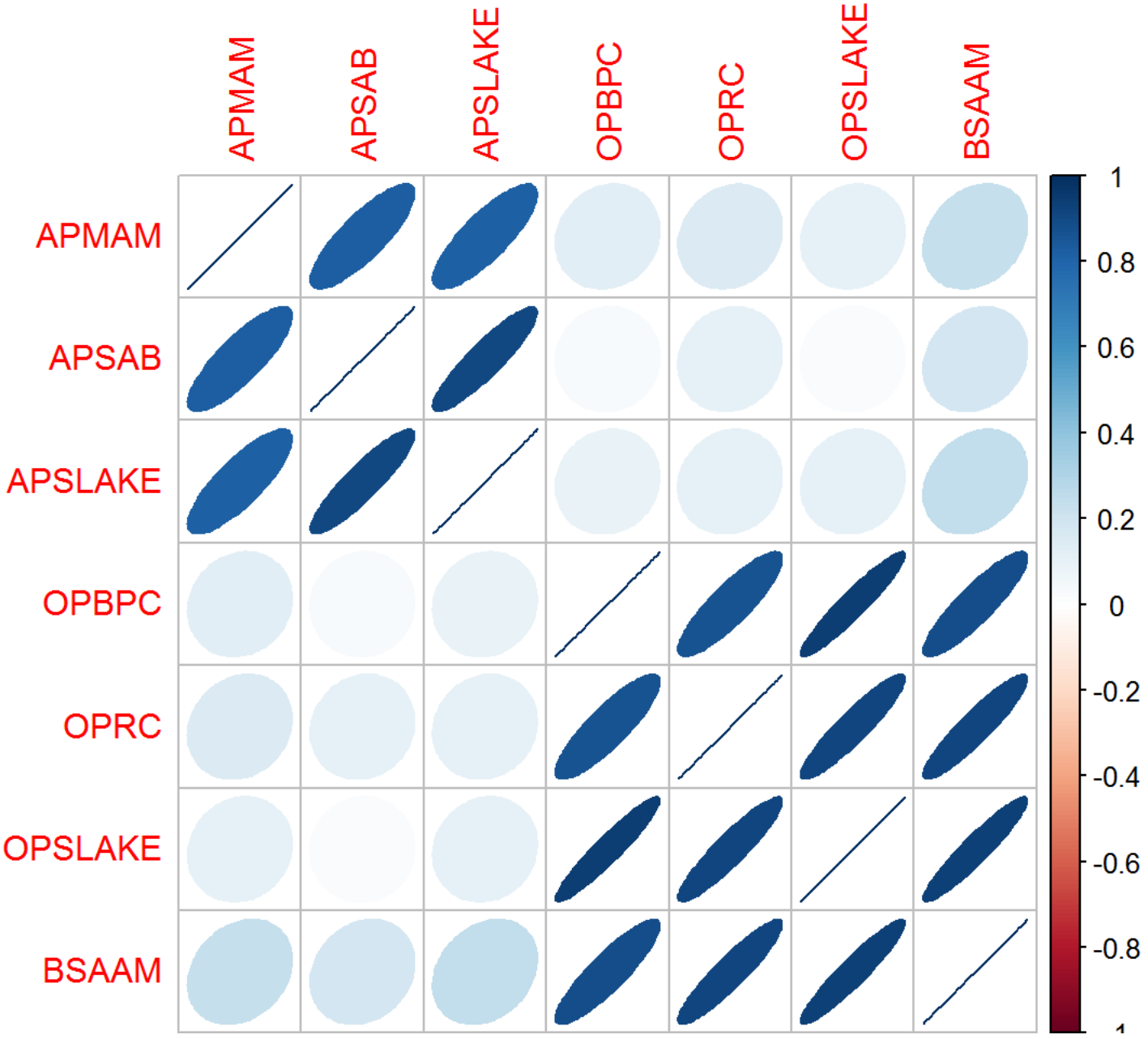
```
## BSAAM    1.0000000
```

We can examine correlation using Corrplot package

```
require(corrplot)
```

```
## Loading required package: corrplot
```

```
corrplot(water.cor, method="ellipse")
```



Modeling and evaluation

We will discuss the best subsets regression methods stepwise, using the “leaps” package.

Forward stepwise selection

This starts with a model that has zero features.

It then adds the features one at a time until all the features are added.

A selected feature is added in the process that creates a model with the lowest RSS. So in theory, the first feature selected should be the one that explains the response variable better than any of the others, and so on.

PS: It is important to note that adding a feature will always decrease RSS and increase R-squared, but will not necessarily improve the model fit and interpretability.

Backward stepwise regression

This begins with all the features in the model and removes the least useful one at a time.

A hybrid approach is available where the features are added through forward stepwise regression, but the algorithm then examines if any features that no longer improve the model fit can be removed.

Once the model is built, the analyst can examine the output and use various statistics to select the features they believe provide the best fit.

Stepwise can produce biased regression coefficients. Best subsets regression can be a satisfactory alternative to the stepwise methods for feature selection. In best subsets regression, the algorithm fits a model for all the possible feature combinations

Let’s start with Leaps package and build stepwise model

```
require(leaps)
```

```
## Loading required package: leaps
```

```
fit <- lm(BSAAM~., data=socal.water)
```

```
summary(fit)
```

```
##
## Call:
## lm(formula = BSAAM ~ ., data = socal.water)
##
## Residuals:
##    Min     1Q Median     3Q    Max
## -12690  -4936  -1424   4173  18542
##
## Coefficients:
##             Estimate Std. Error t value Pr(>|t|)
## (Intercept) 15944.67    4099.80   3.889 0.000416 ***
## APMAM         -12.77     708.89  -0.018 0.985725
## APSAB        -664.41    1522.89  -0.436 0.665237
## APSLAKE      2270.68    1341.29   1.693 0.099112 .
## OPBPC          69.70     461.69   0.151 0.880839
## OPRC         1916.45     641.36   2.988 0.005031 **
## OPSLAKE      2211.58     752.69   2.938 0.005729 **
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 7557 on 36 degrees of freedom
## Multiple R-squared:  0.9248, Adjusted R-squared:  0.9123
## F-statistic: 73.82 on 6 and 36 DF,  p-value: < 2.2e-16
```

We should check for variables that are significant.

We can see “OPRC” and “OPSLAKE” significant but “OPBPC”not significant even though highly correlated with response variable.

This is because with “OPRC” and “OPSLAKE”, “OPBPC” is not adding any statisitical significance to the model.

Creating a best subset using regsubsets() function of leaps package

```
sub.fit <- regsubsets(BSAAM~., data=socal.water)

best.summary <- summary(sub.fit)

names(best.summary)
```

```
## [1] "which"  "rsq"    "rss"    "adjr2"  "cp"     "bic"    "outmat" "obj"
```

We can use which.min() and which.max() to find min and max values.

```
which.min(best.summary$rss)
```

```
## [1] 6
```

This is obvious because 6 is max number of inputs and RSS keeps reducing by adding features and also increases R-squared. We need to effectively find relavant features.

For feature selection there are 4 statistical methods.

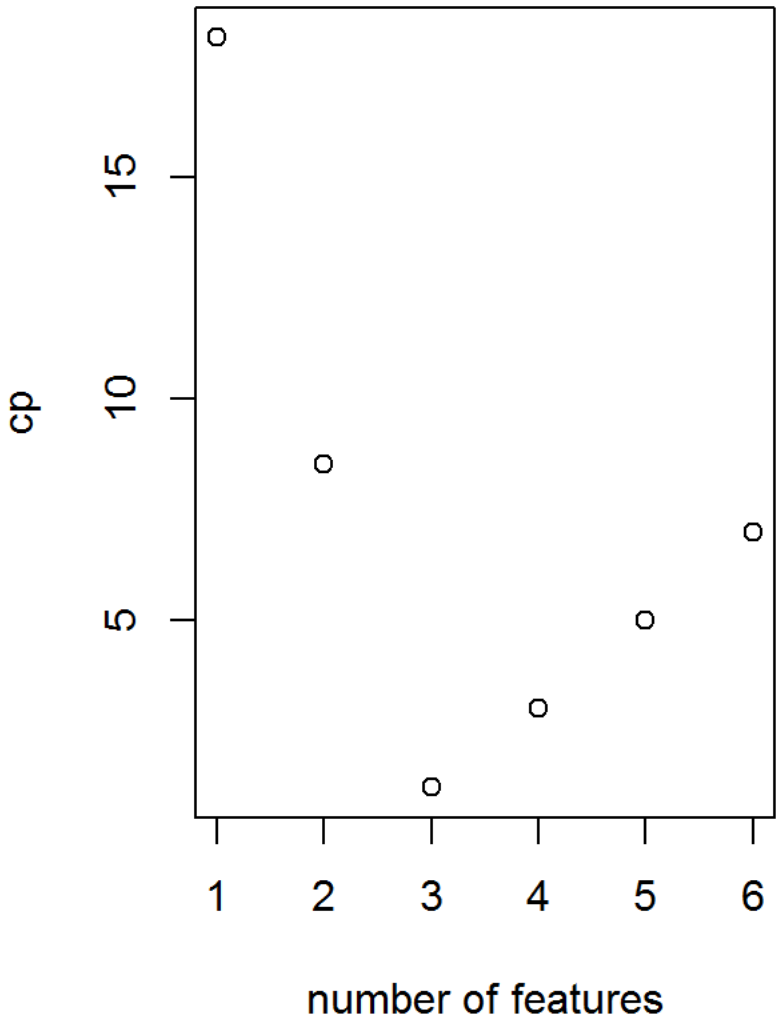
- Aikake’s Information Criterion (AIC): This should be low as possible
- Mallow’s Cp (CP): This should be low as possible
- Bayesian Information Criterion (BIC): This should be low as possible
- Adjusted R-squared: As high as possible

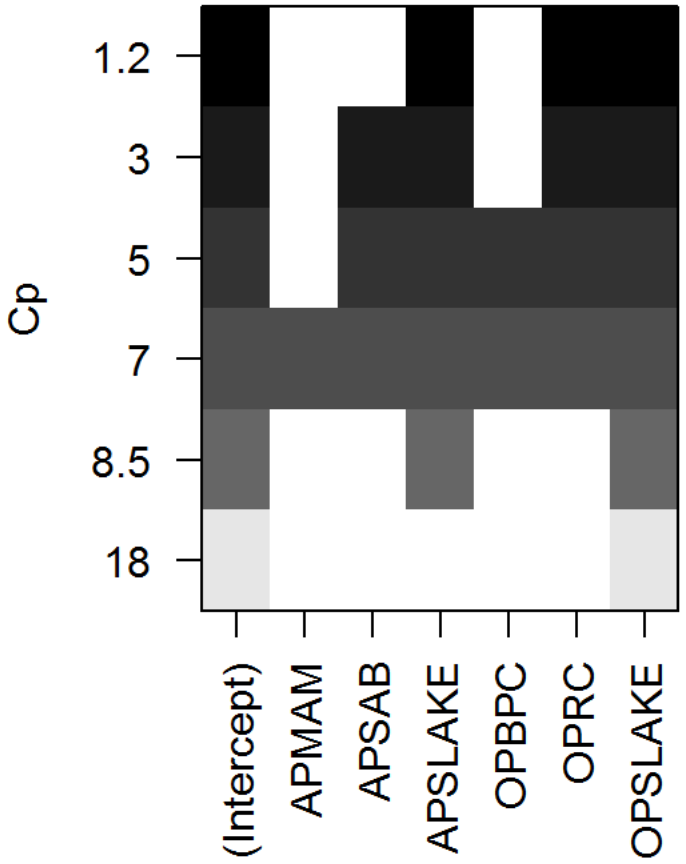
The purpose of these statistics is to create as parsimonious a model as possible, in other words, penalize model complexity.

In a linear model, AIC and Cp are proportional to each other, so we will only concern ourselves with Cp

BIC tends to select the models with fewer variables than Cp, so we will compare both. To do so, we can create and analyze two plots side by side

```
par(mfrow=c(1,2))

plot(best.summary$cp, xlab="number of features", ylab="cp")

plot(sub.fit, scale="Cp")
```





Left plot shows Cp is lowest with 3 features and right plot shows which are those plots. We can also do this using which.min() function.

```
which.min(best.summary$cp)
```

```
## [1] 3
```

```
which.max(best.summary$adjr2)
```

```
## [1] 3
```

Now we can recreate the model using the above mentioned 3 features.

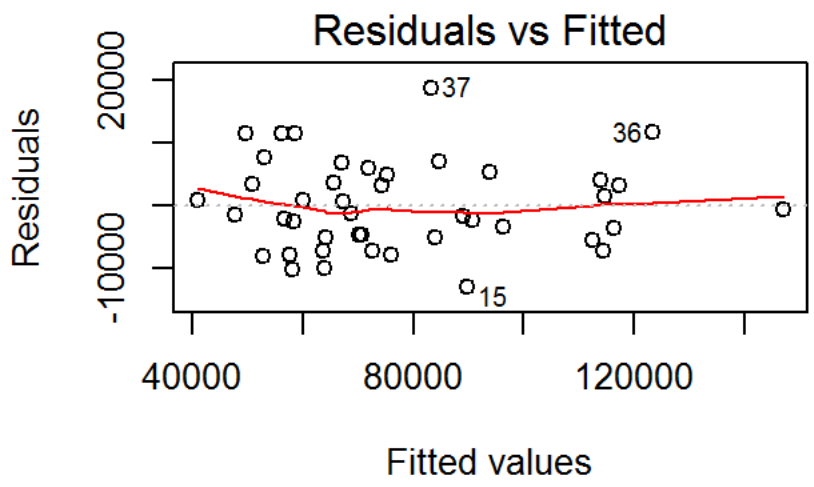
```
best.fit = lm(BSAAM~APSLAKE+OPRC+OPSLAKE, data=socal.water)

summary(best.fit)
```

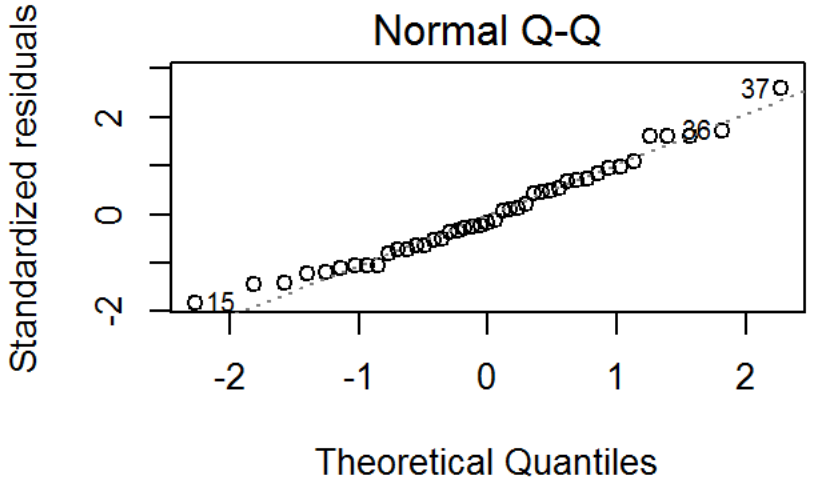
```
##
## Call:
## lm(formula = BSAAM ~ APSLAKE + OPRC + OPSLAKE, data = socal.water)
##
## Residuals:
##    Min     1Q Median     3Q    Max
## -12964  -5140  -1252   4446  18649
##
## Coefficients:
##             Estimate Std. Error t value Pr(>|t|)
## (Intercept)  15424.6     3638.4   4.239 0.000133 ***
## APSLAKE       1712.5      500.5   3.421 0.001475 **
## OPRC          1797.5      567.8   3.166 0.002998 **
## OPSLAKE       2389.8      447.1   5.346 4.19e-06 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

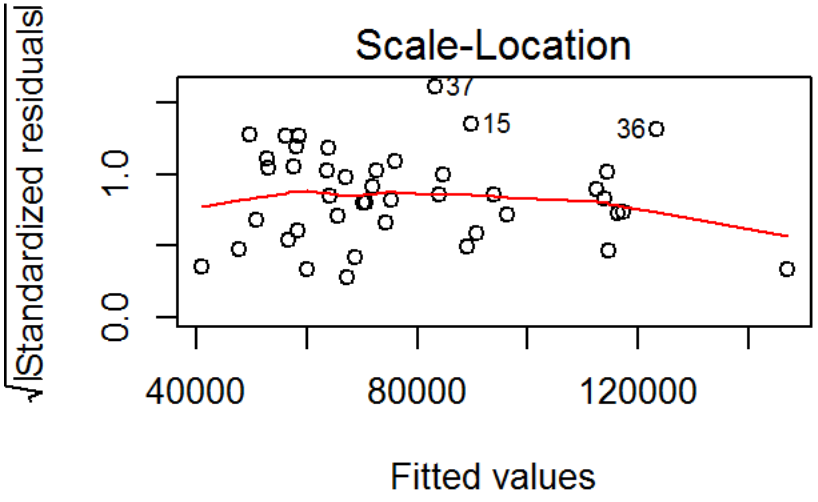
```
##
## Residual standard error: 7284 on 39 degrees of freedom
## Multiple R-squared:  0.9244, Adjusted R-squared:  0.9185
## F-statistic: 158.9 on 3 and 39 DF,  p-value: < 2.2e-16
```

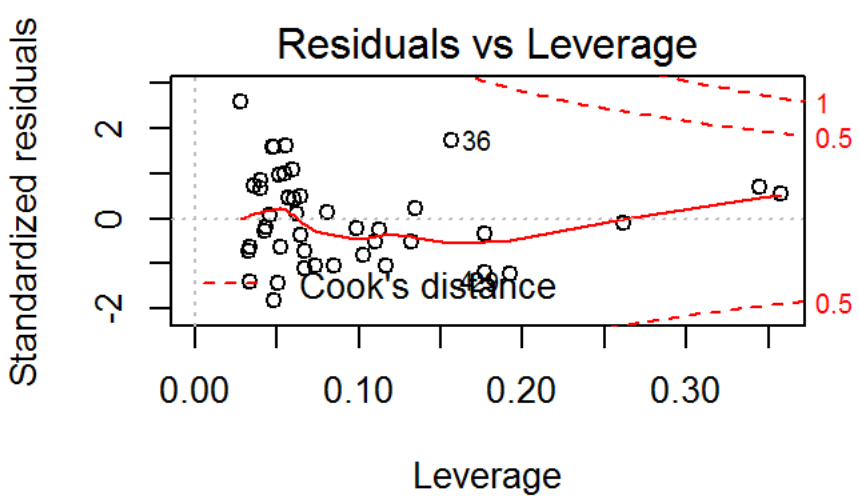
With the three-feature model, F-statistic and all the t-tests have significant p-values. Having passed the first test, we can produce our diagnostic plots:

```
par(mfrow=c(2,2))

plot(best.fit)
```









Looking at the plots, it seems safe to assume that the residuals have a constant variance and are normally distributed.

To investigate the issue of collinearity, one can call up the Variance Inflation Factor (VIF) statistic.

VIF = (Variance of model with all features)/(variance of model with itself)

VIF = 1/(1-Ri-Squared), Ri-squared - R-square of feature of interest i.

Minimum value of VIF is 1 (No Collinearity), any value > 5 or 10 needs to inspected.

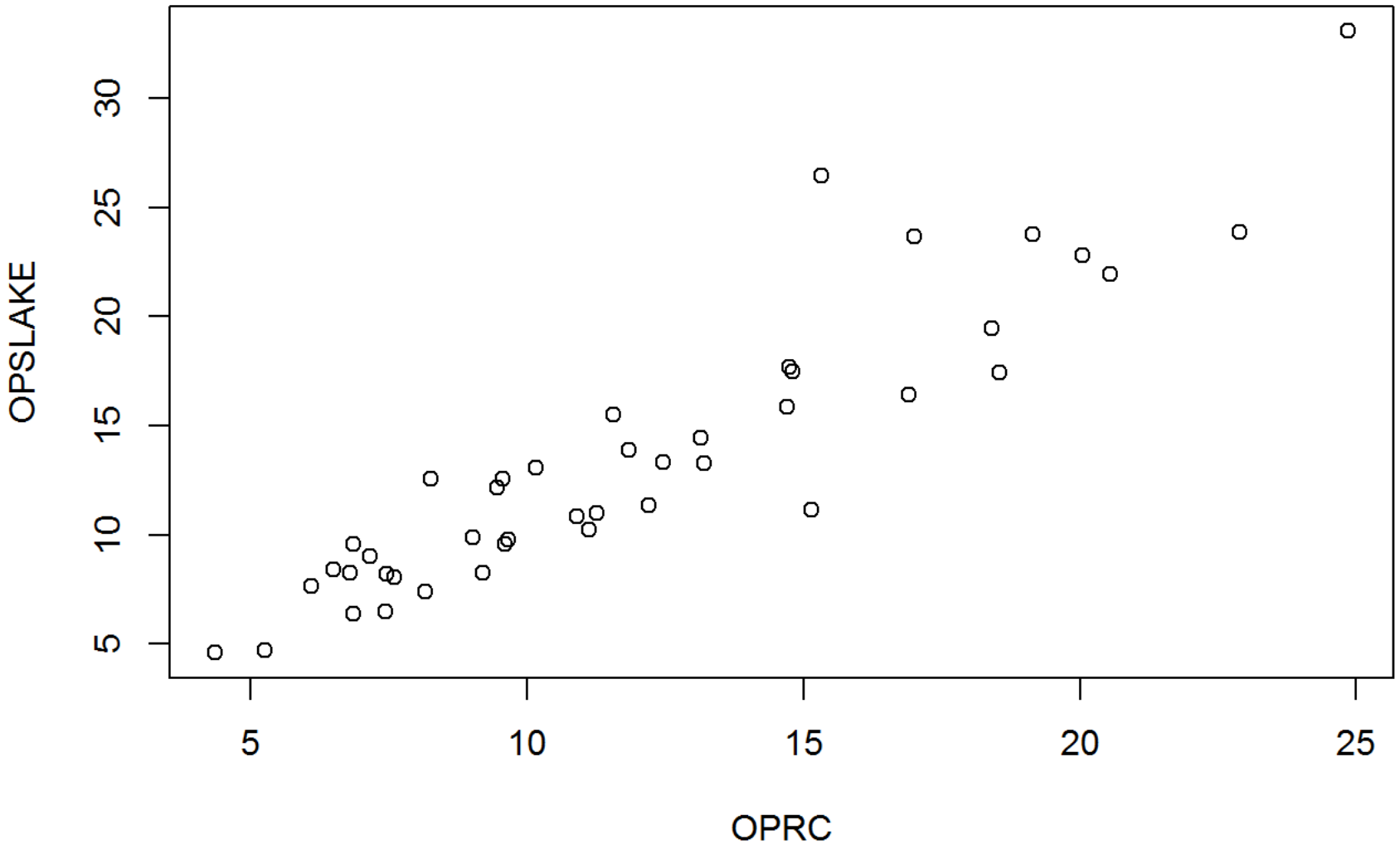
Now let's check the VIF of features of the model

```
vif(best.fit)
```

```
##  APSLAKE     OPRC  OPSLAKE
## 1.011499 6.452569 6.444748
```

"OPRC" and "OPSLAKE" was correlated when seen in cor plot. Let's again see the correlation

```
par(mfrow=c(1,1))

plot(socal.water$OPRC, socal.water$OPSLAKE, xlab="OPRC", ylab="OPSLAKE")
```



The simple solution to address collinearity is to drop the variables to remove the problem, without compromising the predictive ability.

If we look at the adjusted R-squared from the best subsets

```
best.summary$adjr2 #adjusted r-squared values
```

```
## [1] 0.8777515 0.9001619 0.9185369 0.9168706 0.9146772 0.9123079
```

We can see that the two-variable model of APSLAKE and OPSLAKE produced a value of 0.90, while adding OPRC only marginally increased it to 0.92

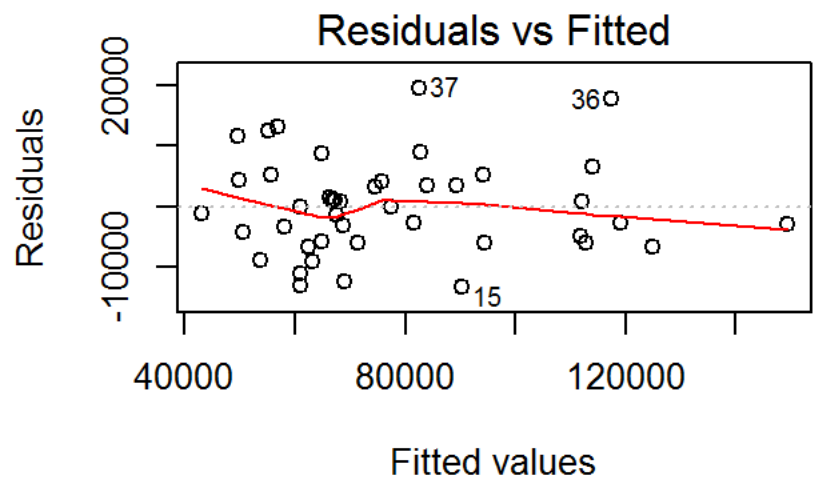
Let's have a look at the two-variable model and test its assumptions, as follows:

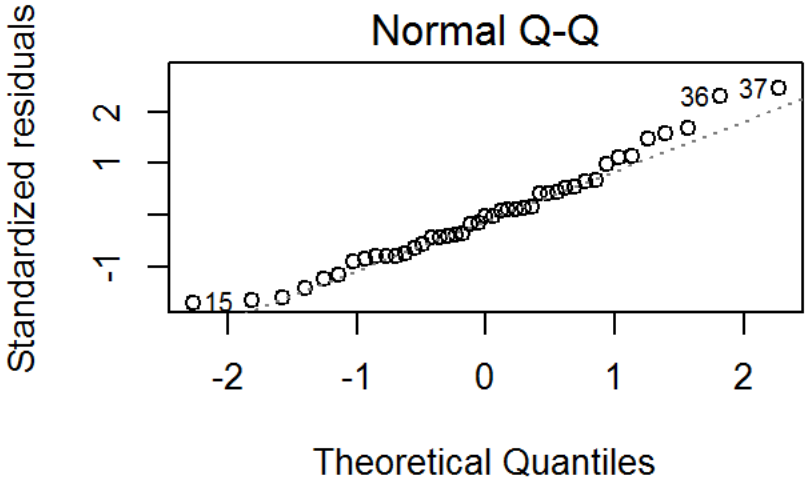
```
fit.2 = lm(BSAAM~APSLAKE+OPSLAKE, data=socal.water)

summary(fit.2)
```

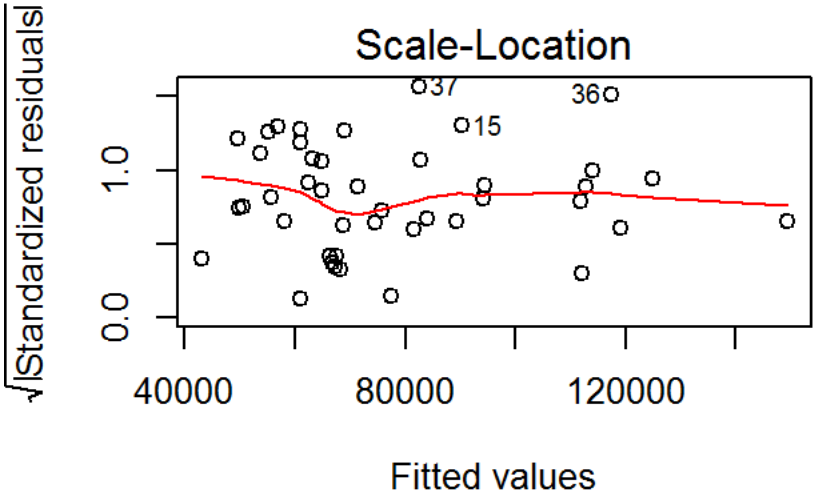
```
##
## Call:
## lm(formula = BSAAM ~ APSLAKE + OPSLAKE, data = socal.water)
##
## Residuals:
##      Min       1Q   Median       3Q      Max
## -13335.8  -5893.2   -171.8   4219.5  19500.2
##
## Coefficients:
##             Estimate Std. Error t value Pr(>|t|)
## (Intercept)  19144.9     3812.0   5.022  1.1e-05 ***
## APSLAKE       1768.8      553.7   3.194  0.00273 **
## OPSLAKE       3689.5      196.0  18.829  < 2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 8063 on 40 degrees of freedom
```

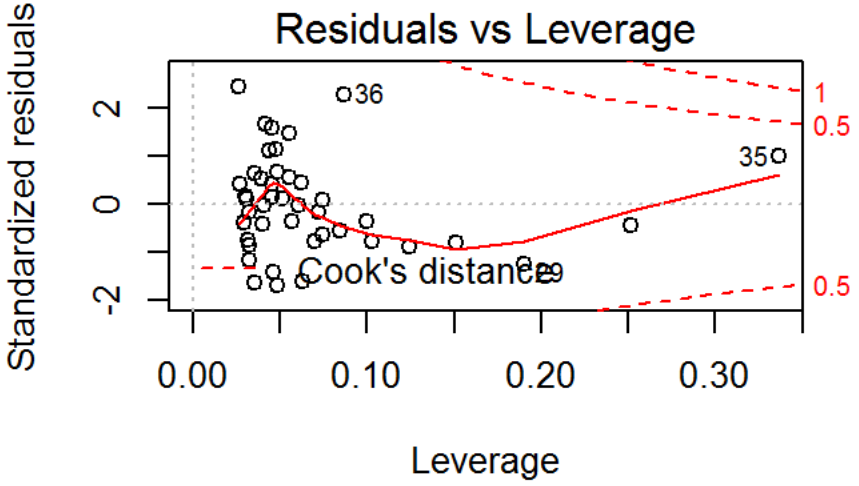
```
## Multiple R-squared:  0.9049, Adjusted R-squared:  0.9002
## F-statistic: 190.3 on 2 and 40 DF,  p-value: < 2.2e-16
```

```
par(mfrow=c(2,2))

plot(fit.2)
```









The model is significant, and the diagnostics do not seem to be a cause for concern

Checking Collinearity

```
vif(fit.2)
```

```
##  APSLAKE  OPSLAKE
## 1.010221 1.010221
```

You can formally test the assumption of the constant variance of errors in R using Breusch-Pagan (BP) test. The BP test has the null hypotheses that the error variances are zero versus the alternative of not zero

We need to load lmtest package

```
require(lmtest)
```

```
## Loading required package: lmtest
```

```
## Loading required package: zoo
```

```
##
## Attaching package: 'zoo'
```

```
## The following objects are masked from 'package:base':
##
##     as.Date, as.Date.numeric
```

```
bptest(fit.2)
```

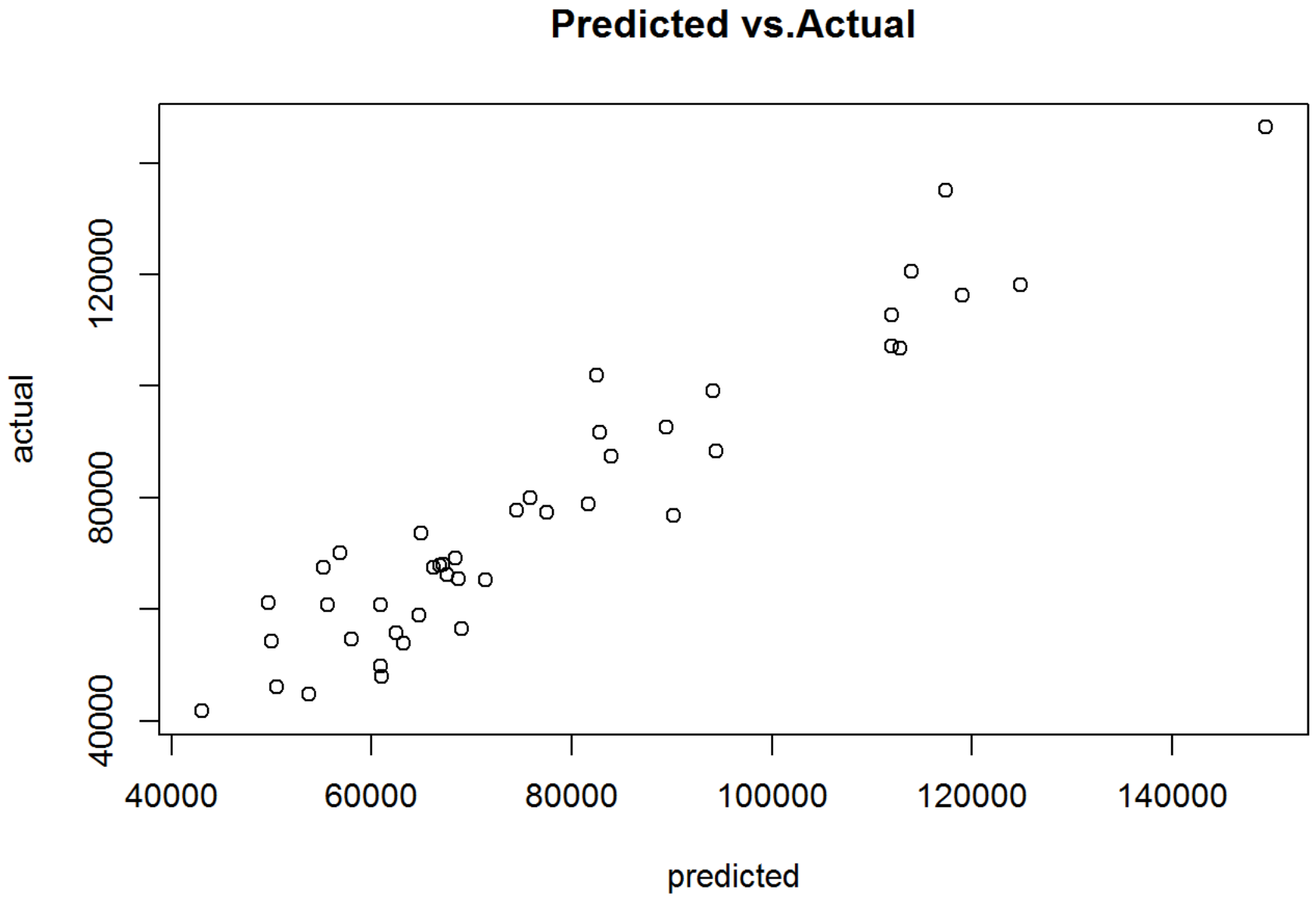
```
##
##  studentized Breusch-Pagan test
```

```
##
## data:  fit.2
## BP = 0.0046205, df = 2, p-value = 0.9977
```

We do not have evidence to reject the null that implies the error variances are zero because p-value = 0.9977. The BP = 0.0046 value in the summary of the test is the chi-squared value.

A scatterplot of the Predicted vs. Actual values can be done in base R using the fitted values from the model and the response variable values as follows:

```
par(mfrow=c(1,1))
plot(fit.2$fitted.values, socal.water$BSAAM,xlab="predicted", ylab="actual", main="Predicted vs
.Actual")
```



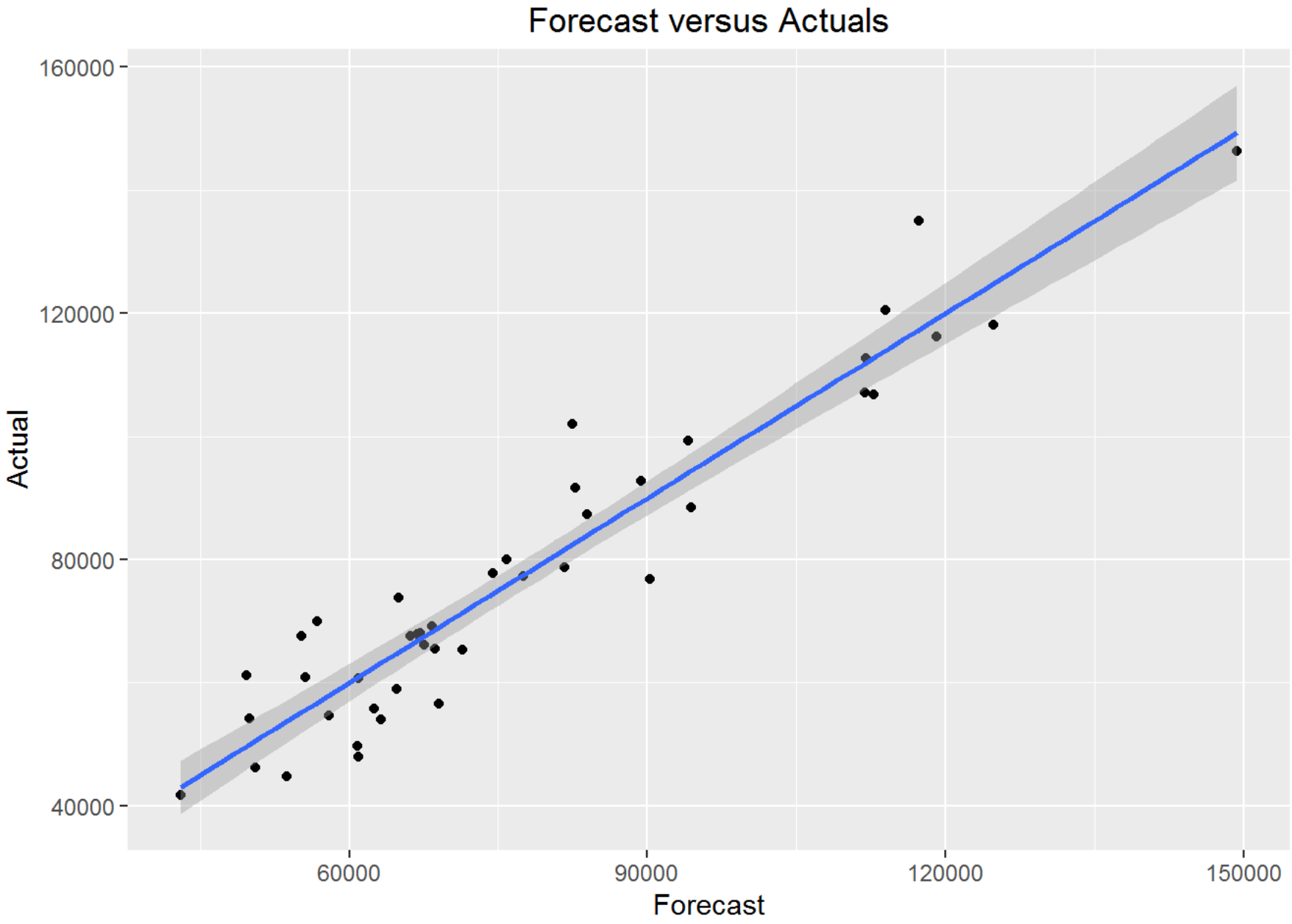
Let’s do good graphics using ggplot2 package

```
socal.water$Actual <- water$BSAAM
```

```
socal.water$Forecast <- predict(fit.2)

require(ggplot2)
```

```
## Loading required package: ggplot2
```

```
ggplot(socal.water, aes(x=Forecast, y=Actual)) +geom_point() + geom_smooth(method=lm) + labs(ti
tle = "Forecast versus Actuals")
```



Other Linear Model Considerations

- Qualitative feature

- Interaction term

Qualitative feature

A qualitative feature, also referred to as a factor, can take on two or more levels such as Male/Female or Bad/Neutral/Good

If we have a feature with two levels, say gender, we can create a dummy variable with “0” for Male and “1” for female. Y = B0 for Male and Y = B0 + B1X for Female.

But when number of features increases we will fall in dummy variable trap, which results in perfect multicollinearity.

```
require(ISLR)
```

```
## Loading required package: ISLR
```

```
data("Carseats")

str(Carseats)
```

```
## 'data.frame':    400 obs. of  11 variables:
##  $ Sales      : num  9.5 11.22 10.06 7.4 4.15 ...
##  $ CompPrice  : num  138 111 113 117 141 124 115 136 132 132 ...
##  $ Income     : num  73 48 35 100 64 113 105 81 110 113 ...
##  $ Advertising: num  11 16 10 4 3 13 0 15 0 0 ...
##  $ Population : num  276 260 269 466 340 501 45 425 108 131 ...
##  $ Price      : num  120 83 80 97 128 72 108 120 124 124 ...
##  $ ShelveLoc  : Factor w/ 3 levels "Bad","Good","Medium": 1 2 3 3 1 1 3 2 3 3 ...
##  $ Age        : num  42 65 59 55 38 78 71 67 76 76 ...
##  $ Education  : num  17 10 12 14 13 16 15 10 10 17 ...
##  $ Urban      : Factor w/ 2 levels "No","Yes": 2 2 2 2 2 1 2 2 1 1 ...
##  $ US         : Factor w/ 2 levels "No","Yes": 2 2 2 2 1 2 1 2 1 2 ...
```

Let’s only consider Advertisement and Shelveloc as the parameters.

```
sales.fit = lm(Sales~Advertising+ShelveLoc, data=Carseats)

summary(sales.fit)
```

```
##
## Call:
## lm(formula = Sales ~ Advertising + ShelveLoc, data = Carseats)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -6.6480 -1.6198 -0.0476  1.5308  6.4098
##
## Coefficients:
##                 Estimate Std. Error t value Pr(>|t|)
## (Intercept)      4.89662    0.25207  19.426  < 2e-16 ***
## Advertising      0.10071    0.01692   5.951 5.88e-09 ***
## ShelveLocGood    4.57686    0.33479  13.671  < 2e-16 ***
## ShelveLocMedium  1.75142    0.27475   6.375 5.11e-10 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 2.244 on 396 degrees of freedom
## Multiple R-squared:  0.3733, Adjusted R-squared:  0.3685
## F-statistic: 78.62 on 3 and 396 DF,  p-value: < 2.2e-16
```

If the shelving location is good, the estimate of sales is almost double than when the location is bad, given the intercept of 4.89662.

To see how R codes the indicator features, you can use the contrasts()

```
contrasts(Carseats$ShelveLoc)
```

```
##        Good Medium
## Bad       0      0
## Good      1      0
## Medium    0      1
```

Interaction term

Two features interact if the effect on the prediction of one feature depends on the value of the other feature.

Y = B0 + B1x + B2x + B1B2x + e.

```
require(MASS)
```

```
## Loading required package: MASS
```

```
##
## Attaching package: 'MASS'
```

```
## The following object is masked from 'package:alr3':
##
##     forbes
```

```
data("Boston")
```

```
str(Boston)
```

```
## 'data.frame':    506 obs. of  14 variables:
##  $ crim   : num  0.00632 0.02731 0.02729 0.03237 0.06905 ...
##  $ zn     : num  18 0 0 0 0 0 12.5 12.5 12.5 12.5 ...
##  $ indus  : num  2.31 7.07 7.07 2.18 2.18 2.18 7.87 7.87 7.87 7.87 ...
##  $ chas   : int  0 0 0 0 0 0 0 0 0 0 ...
##  $ nox    : num  0.538 0.469 0.469 0.458 0.458 0.458 0.524 0.524 0.524 0.524 ...
##  $ rm     : num  6.58 6.42 7.18 7 7.15 ...
##  $ age    : num  65.2 78.9 61.1 45.8 54.2 58.7 66.6 96.1 100 85.9 ...
##  $ dis    : num  4.09 4.97 4.97 6.06 6.06 ...
##  $ rad    : int  1 2 2 3 3 3 5 5 5 5 ...
##  $ tax    : num  296 242 242 222 222 222 311 311 311 311 ...
##  $ ptratio: num  15.3 17.8 17.8 18.7 18.7 18.7 15.2 15.2 15.2 15.2 ...
##  $ black  : num  397 397 393 395 397 ...
##  $ lstat  : num  4.98 9.14 4.03 2.94 5.33 ...
##  $ medv   : num  24 21.6 34.7 33.4 36.2 28.7 22.9 27.1 16.5 18.9 ...
```

Interacting lstat and Age:

```
value.fit = lm(medv~lstat*age, data=Boston)

summary(value.fit)
```

```
##
## Call:
## lm(formula = medv ~ lstat * age, data = Boston)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
```

```
## -15.806  -4.045  -1.333   2.085  27.552
##
## Coefficients:
##               Estimate Std. Error t value Pr(>|t|)
## (Intercept) 36.0885359  1.4698355  24.553  < 2e-16 ***
## lstat       -1.3921168  0.1674555  -8.313 8.78e-16 ***
## age         -0.0007209  0.0198792  -0.036   0.9711
## lstat:age    0.0041560  0.0018518   2.244   0.0252 *
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 6.149 on 502 degrees of freedom
## Multiple R-squared:  0.5557, Adjusted R-squared:  0.5531
## F-statistic: 209.3 on 3 and 502 DF,  p-value: < 2.2e-16
```

lstat is highly significant but age is not. But we can observe interaction term is also significant.