Problem Set 8

Prateek Kumar

Contents

Problem 1: Orthogonal Designs

```
training <- rep(c(1,1,1,0,0,0,-1,-1,-1),each=3)
training

##  [1]  1  1  1  1  1  1  1  1  1  0  0  0  0  0  0  0  0  0 -1 -1 -1 -1 -1
## [24] -1 -1 -1 -1

feedback <- rep(c(1,0,-1,1,0,-1,1,0,-1),each=3)
feedback

##  [1]  1  1  1  0  0  0 -1 -1 -1  1  1  1  0  0  0 -1 -1 -1  1  1  1  0  0
## [24]  0 -1 -1 -1
```

We have two 3-level independent variables:

- training (no training, standard training, interactive training)
- feedback (no feedback, feedback on errors, feedback on correct)

Correlation between training and feedback

```
cor(training, feedback)

## [1] 0
```

Dot product between training and feedback

```
training %*% feedback

##      [,1]
## [1,]    0
```

We can verify that both of them are uncorrelated and orthogonal since their correlation and dot product is zero.

Create one that is orthogonal to the two existing IVs, and verify that it is orthogonal

```
group <- rep(c(1,-1,0,-1,0,1,0,1,-1),each=3)
group

##  [1]  1  1  1 -1 -1 -1  0  0  0 -1 -1 -1  0  0  0  1  1  1  0  0  0  1  1
## [24]  1 -1 -1 -1
```

Here the above IV "group" is both uncorrelated and orthogonal to both training and feedback, which we verified below

Correlation between training and group

```
cor(training, group)

## [1] 0
```

Dot product between training and group

```
training %*% group

##      [,1]
## [1,]    0
```

Its orthogonal and uncorrelated with training

Correlation between feedback and group

```
cor(feedback, group)

## [1] 0
```

Dot product between feedback and group

```
feedback %*% group

##      [,1]
## [1,]    0
```

Its orthogonal and uncorrelated with feedback as well.

Which of the following grouping variables are both orthogonal and uncorrelated with both feedback and train?

```
set.seed(100)
group1 <- 1:27
group2 <- rep(-1:1, 9)
group3 <- 1:27-mean(1:27)
group4 <- rep(1:3, 9)
group5 <- rep(-1:1,each=9)
group6 <- runif(27)-.5
group7 <- rep(1:9,each=3)
```

We will calculate the correlation and dot product of each group IV with training and feedback

Firstly, calculating with training:

Correlation between training and group1

```
cor(training, group1)
```

```
## [1] -0.9434564
```

Dot product between training and group1

```
training %*% group1
```

```
##      [,1]
## [1,] -162
```

Correlation between training and group2

```
cor(training, group2)
```

```
## [1] 0
```

Dot product between training and group2

```
training %*% group2
```

```
##      [,1]
## [1,]    0
```

Correlation between training and group3

```
cor(training, group3)
```

```
## [1] -0.9434564
```

Dot product between training and group3

```
training %*% group3
```

```
##      [,1]
## [1,] -162
```

Correlation between training and group4

```
cor(training, group4)
```

```
## [1] 0
```

Dot product between training and group4

```
training %*% group4
```

```
##      [,1]
## [1,]    0
```

Correlation between training and group5

```
cor(training, group5)
```

```
## [1] -1
```

Dot product between training and group5

```
training %*% group5
```

```
##      [,1]
## [1,]  -18
```

Correlation between training and group6

```
cor(training, group6)

## [1] -0.2257106
```

Dot product between training and group6

```
training %*% group6

##             [,1]
## [1,] -1.089779
```

Correlation between training and group7

```
cor(training, group7)

## [1] -0.9486833
```

Dot product between training and group7

```
training %*% group7

##      [,1]
## [1,]  -54
```

We see that only group2 and group4 are uncorrelated and orthogonal with training because the correlation and dot product is zero in this case

Now calculating with feedback

Correlation between feedback and group1

```
cor(feedback, group1)

## [1] -0.3144855
```

Dot product between feedback and group1

```
feedback %*% group1

##      [,1]
## [1,]  -54
```

Correlation between feedback and group2

```
cor(feedback, group2)

## [1] 0
```

Dot product between feedback and group2

```
feedback %*% group2

##      [,1]
## [1,]    0
```

Correlation between feedback and group3

```
cor(feedback, group3)

## [1] -0.3144855
```

Dot product between feedback and group3

```
feedback %*% group3

##      [,1]
## [1,]  -54
```

Correlation between feedback and group4

```
cor(feedback, group4)

## [1] 0
```

Dot product between feedback and group4

```
feedback %*% group4

##      [,1]
## [1,]    0
```

Correlation between feedback and group5

```
cor(feedback, group5)

## [1] 0
```

Dot product between feedback and group5

```
feedback %*% group5

##      [,1]
## [1,]    0
```

Correlation between feedback and group6

```
cor(feedback, group6)

## [1] 0.0121159
```

Dot product between feedback and group6

```
feedback %*% group6

##            [,1]
## [1,] 0.05849814
```

Correlation between feedback and group7

```
cor(feedback, group7)

## [1] -0.3162278
```

Dot product between feedback and group7

```
feedback %*% group7
```

```
##      [,1]
## [1,]  -18
```

We see that group2, group4 and group5 are uncorrelated and orthogonal with feedback because the correlation and dot product is zero in this case

So we conclude from the above results that only group2 and group4 are uncorrelated and orthogonal with both training and feedback.

Problem 2: Regression with orthogonal and uncorrelated predictors

```
data <- data.frame(buyerid = group7,
                   timeframe=group4)

set.seed(103)

buyerbase <- runif(9)
timebase <- c(1,2,2.5)
agebase  <- 20+runif(9)* 40

data$age <- round(agebase[data$buyerid])
data$purchase <- round( 25 + buyerbase[data$buyerid] * 50 +
                          data$age * .5+
                          timebase[data$timeframe] * 7 +
                          rnorm(27)*4,2)
```

Creating the dataset from above code

Do the coefficients differ across models (including intercept)?

Creating a regression model to predict purchase amount by timeframe and age and checking the coefficients

```
lm(purchase ~ timeframe+age, data = data)$coef

## (Intercept)   timeframe         age
##  46.3220369   5.3900000   0.3200048

summary(lm(purchase ~ timeframe+age, data = data))

##
## Call:
## lm(formula = purchase ~ timeframe + age, data = data)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -15.382  -9.357  -3.122   9.808  19.608
##
## Coefficients:
##             Estimate Std. Error t value Pr(>|t|)
## (Intercept)  46.3220     8.9654   5.167 2.73e-05 ***
## timeframe     5.3900     2.5544   2.110   0.0455 *
## age           0.3200     0.1849   1.731   0.0963 .
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 10.84 on 24 degrees of freedom
## Multiple R-squared:  0.2369, Adjusted R-squared:  0.1733
## F-statistic: 3.724 on 2 and 24 DF,  p-value: 0.03902
```

Creating a regression model to predict purchase amount by timeframe only and checking the coefficients

```
lm(purchase ~ timeframe, data = data)$coef

## (Intercept)   timeframe
##    58.55333     5.39000

summary(lm(purchase ~ timeframe, data = data))

##
## Call:
## lm(formula = purchase ~ timeframe, data = data)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -17.693  -6.978  -2.393   6.537  24.017
##
## Coefficients:
##             Estimate Std. Error t value Pr(>|t|)
## (Intercept)   58.553      5.734  10.211  2.1e-10 ***
## timeframe      5.390      2.654   2.031   0.0531 .
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 11.26 on 25 degrees of freedom
## Multiple R-squared:  0.1416, Adjusted R-squared:  0.1072
## F-statistic: 4.123 on 1 and 25 DF,  p-value: 0.05307
```

Creating a regression model to predict purchase amount by age only and checking the coefficients

```
lm(purchase ~ age, data = data)$coef

## (Intercept)         age
##  57.1020369   0.3200048

summary(lm(purchase ~ age, data = data))

##
## Call:
## lm(formula = purchase ~ age, data = data)
##
## Residuals:
##      Min       1Q   Median       3Q      Max
## -16.2322  -9.1772  -0.9822   7.1678  23.6577
##
## Coefficients:
##             Estimate Std. Error t value Pr(>|t|)
## (Intercept)  57.1020     7.8597   7.265  1.3e-07 ***
## age           0.3200     0.1972   1.623    0.117
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 11.56 on 25 degrees of freedom
## Multiple R-squared:  0.09528,    Adjusted R-squared:  0.05909
## F-statistic: 2.633 on 1 and 25 DF,  p-value: 0.1172
```

We see from the above results that the β values are same when we calculate the regression with both predictors together and individually (timeframe and age) i.e. 5.39 and 0.32 respectively but the intercept values differ in all the 3 cases.

Does the coefficient differ across models with the orthogonal predictors?

Now transforming the predictors to orthogonal

```
age1 <- data$age-mean(data$age)
tf1 <- data$timeframe-mean(data$timeframe)
```

Checking the coefficients with both the new predictors together

```
lm(purchase ~ tf1+age1, data = data)$coef

## (Intercept)         tf1        age1
##  69.3333333   5.3900000   0.3200048

summary(lm(purchase ~ tf1+age1, data = data))

##
## Call:
## lm(formula = purchase ~ tf1 + age1, data = data)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -15.382  -9.357  -3.122   9.808  19.608
##
## Coefficients:
##             Estimate Std. Error t value Pr(>|t|)
## (Intercept)  69.3333     2.0857  33.243   <2e-16 ***
## tf1           5.3900     2.5544   2.110   0.0455 *
## age1          0.3200     0.1849   1.731   0.0963 .
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 10.84 on 24 degrees of freedom
## Multiple R-squared:  0.2369, Adjusted R-squared:  0.1733
## F-statistic: 3.724 on 2 and 24 DF,  p-value: 0.03902
```

Creating a regression model to predict purchase amount by orthogonal timeframe only and checking the coefficients

```
lm(purchase ~ tf1, data = data)$coef
```

```
## (Intercept)         tf1
##    69.33333     5.39000
```

```
summary(lm(purchase ~ tf1, data = data))
```

```
##
## Call:
## lm(formula = purchase ~ tf1, data = data)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -17.693  -6.978  -2.393   6.537  24.017
##
## Coefficients:
##             Estimate Std. Error t value Pr(>|t|)
## (Intercept)   69.333      2.167  31.990   <2e-16 ***
## tf1            5.390      2.654   2.031   0.0531 .
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 11.26 on 25 degrees of freedom
## Multiple R-squared:  0.1416, Adjusted R-squared:  0.1072
## F-statistic: 4.123 on 1 and 25 DF,  p-value: 0.05307
```

Creating a regression model to predict purchase amount by orthogonal age only and checking the coefficients

```
lm(purchase ~ age1, data = data)$coef
```

```
## (Intercept)        age1
##  69.3333333   0.3200048
```

```
summary(lm(purchase ~ age1, data = data))
```

```
##
## Call:
## lm(formula = purchase ~ age1, data = data)
##
## Residuals:
##      Min       1Q   Median       3Q      Max
## -16.2322  -9.1772  -0.9822   7.1678  23.6577
##
## Coefficients:
##             Estimate Std. Error t value Pr(>|t|)
## (Intercept)  69.3333     2.2250  31.161   <2e-16 ***
## age1          0.3200     0.1972   1.623    0.117
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 11.56 on 25 degrees of freedom
## Multiple R-squared:  0.09528,    Adjusted R-squared:  0.05909
## F-statistic: 2.633 on 1 and 25 DF,  p-value: 0.1172
```

From the above results we see that even the intercepts remain unchanged along with the β values i.e. 69.33 for intercept and 5.39 and 0.32 for the orthogonal timeframe and age respectively, we can see the β values are same in both the cases. This happens because the predictors have been centered, and are both uncorrelated and orthogonal.

So, the advantage of having orthogonal predictors in a regression is that it won't matter if we add or remove any - our estimates won't change. Thus, when possible, using orthogonal predictors in regression make for stable predictors. This can usually only be achieved through designing an experiment - we are unlikely to get this by sampling all variables at random from a population. But even if we have some dependence between predictors, we may be able to achieve more stable predictors by centering the variables, and so that can be a good practice regardless, and it will often make for more interpretable models, as it fits a model around the center of the data

When the first observation is missing

Removing the first observation

```
data2 <- data[-1,]
```

Checking the coefficients with the new dataset

Checking the coefficients with both the predictors together

```
lm(purchase ~ timeframe+age, data = data2)$coef

## (Intercept)   timeframe         age
##  47.3474132   5.1137764   0.3124496

summary(lm(purchase ~ timeframe+age, data = data2))

##
## Call:
## lm(formula = purchase ~ timeframe + age, data = data2)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -15.345  -9.955  -3.176   9.959  19.528
##
## Coefficients:
##             Estimate Std. Error t value Pr(>|t|)
## (Intercept)  47.3474     9.4319   5.020 4.44e-05 ***
## timeframe     5.1138     2.6781   1.909   0.0688 .
## age           0.3124     0.1889   1.654   0.1117
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 11.03 on 23 degrees of freedom
## Multiple R-squared:  0.2135, Adjusted R-squared:  0.1451
## F-statistic: 3.121 on 2 and 23 DF,  p-value: 0.06321
```

Creating a regression model to predict purchase amount by timeframe only and checking the coefficients

```
lm(purchase ~ timeframe, data = data2)$coef
```

```
## (Intercept)   timeframe
##   59.556327    5.013878
```

```
summary(lm(purchase ~ timeframe, data = data2))
```

```
##
## Call:
## lm(formula = purchase ~ timeframe, data = data2)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -17.568  -7.054  -0.656   6.130  23.766
##
## Coefficients:
##             Estimate Std. Error t value Pr(>|t|)
## (Intercept)   59.556      6.079   9.797 7.33e-10 ***
## timeframe      5.014      2.773   1.808   0.0831 .
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 11.42 on 24 degrees of freedom
## Multiple R-squared:  0.1199, Adjusted R-squared:  0.08325
## F-statistic:  3.27 on 1 and 24 DF,  p-value: 0.08309
```

Creating a regression model to predict purchase amount by age only and checking the coefficients

```
lm(purchase ~ age, data = data2)$coef
```

```
## (Intercept)         age
##  58.0842835   0.3043129
```

```
summary(lm(purchase ~ timeframe, data = data2))
```

```
##
## Call:
## lm(formula = purchase ~ timeframe, data = data2)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -17.568  -7.054  -0.656   6.130  23.766
##
## Coefficients:
##             Estimate Std. Error t value Pr(>|t|)
## (Intercept)   59.556      6.079   9.797 7.33e-10 ***
## timeframe      5.014      2.773   1.808   0.0831 .
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 11.42 on 24 degrees of freedom
## Multiple R-squared:  0.1199, Adjusted R-squared:  0.08325
## F-statistic:  3.27 on 1 and 24 DF,  p-value: 0.08309
```

We see that on removing the first row we get the different results. Now we can see that our β values are different when we calculate the regression with both timeframe and age together and when we calculate the regression with both timeframe and age individually.

Now transforming the predictors to orthogonal

```
age2 <- data2$age-mean(data2$age)
tf2 <- data2$timeframe-mean(data2$timeframe)
```

Checking the coefficients with both the new predictors together

```
lm(purchase ~ tf2+age2, data = data2)$coef

## (Intercept)         tf2        age2
##  69.7769231   5.1137764   0.3124496

summary(lm(purchase ~ tf2+age2, data = data2))

##
## Call:
## lm(formula = purchase ~ tf2 + age2, data = data2)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -15.345  -9.955  -3.176   9.959  19.528
##
## Coefficients:
##             Estimate Std. Error t value Pr(>|t|)
## (Intercept)  69.7769     2.1625  32.266   <2e-16 ***
## tf2           5.1138     2.6781   1.909   0.0688 .
## age2          0.3124     0.1889   1.654   0.1117
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 11.03 on 23 degrees of freedom
## Multiple R-squared:  0.2135, Adjusted R-squared:  0.1451
## F-statistic: 3.121 on 2 and 23 DF,  p-value: 0.06321
```

Creating a regression model to predict purchase amount by orthogonal timeframe only and checking the coefficients

```
lm(purchase ~ tf2, data = data2)$coef

## (Intercept)         tf2
##   69.776923    5.013878

summary(lm(purchase ~ tf2, data = data2))

##
## Call:
## lm(formula = purchase ~ tf2, data = data2)
##
## Residuals:
```

```
##     Min      1Q  Median      3Q     Max
## -17.568  -7.054  -0.656   6.130  23.766
##
## Coefficients:
##             Estimate Std. Error t value Pr(>|t|)
## (Intercept)   69.777      2.239  31.160   <2e-16 ***
## tf2            5.014      2.773   1.808   0.0831 .
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 11.42 on 24 degrees of freedom
## Multiple R-squared:  0.1199, Adjusted R-squared:  0.08325
## F-statistic:  3.27 on 1 and 24 DF,  p-value: 0.08309
```

Creating a regression model to predict purchase amount by orthogonal age only and checking the coefficients

```r
lm(purchase ~ age2, data = data2)$coef
```

```
## (Intercept)        age2
##  69.7769231   0.3043129
```

```r
summary(lm(purchase ~ age2, data = data2))
```

```
##
## Call:
## lm(formula = purchase ~ age2, data = data2)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -16.728  -7.999  -1.274   7.571  23.491
##
## Coefficients:
##             Estimate Std. Error t value Pr(>|t|)
## (Intercept)  69.7769     2.2786  30.622   <2e-16 ***
## age2          0.3043     0.1990   1.529    0.139
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 11.62 on 24 degrees of freedom
## Multiple R-squared:  0.08877,    Adjusted R-squared:  0.0508
## F-statistic: 2.338 on 1 and 24 DF,  p-value: 0.1393
```

From the above results we see that only the intercepts remain unchanged whereas the β values differ. The reason of this can be fetched from the very beginning of the question which is the creation of the dataframe. We see that in the "data" dataframe purchase is a dependent variable which is created from the independent variables age and timeframe. The regression algorithm sees the underlying data points, so the regression algorithm treats them as more important to "get right." (Basically, we can think of each occurrence of a data point as pulling the regression line towards it with the same force--so if we have two data points at a given spot, they will pull the line towards them twice as hard.)

This happens here removing the data points creates a void in that location and hence the value changes. Moreover, since purchase is created from age and timeframe so its correlated with both of them. Now, when we remove an entry the correlation value changes hence changing the values of coefficients.

Problem 3. Modeling a time series.

A. Loess regression

Creating the dataframe "down"

```
down <- read.csv("DownloadData.csv")
down$Month <- as.factor(substr(down$Date,6,8))
down$MonthNumber <- 1:nrow(down)
```

Plotting Date vs Downloads

```
plot(as.numeric(down$Date),down$Downloads,xaxt="n",bty="n",pch=21,cex=.5,type="p",las=1,
     ylab="Downloads",xlab="Year")
axis(1,0:12*12,2006:2018,las=3,cex.axis=.95)
```

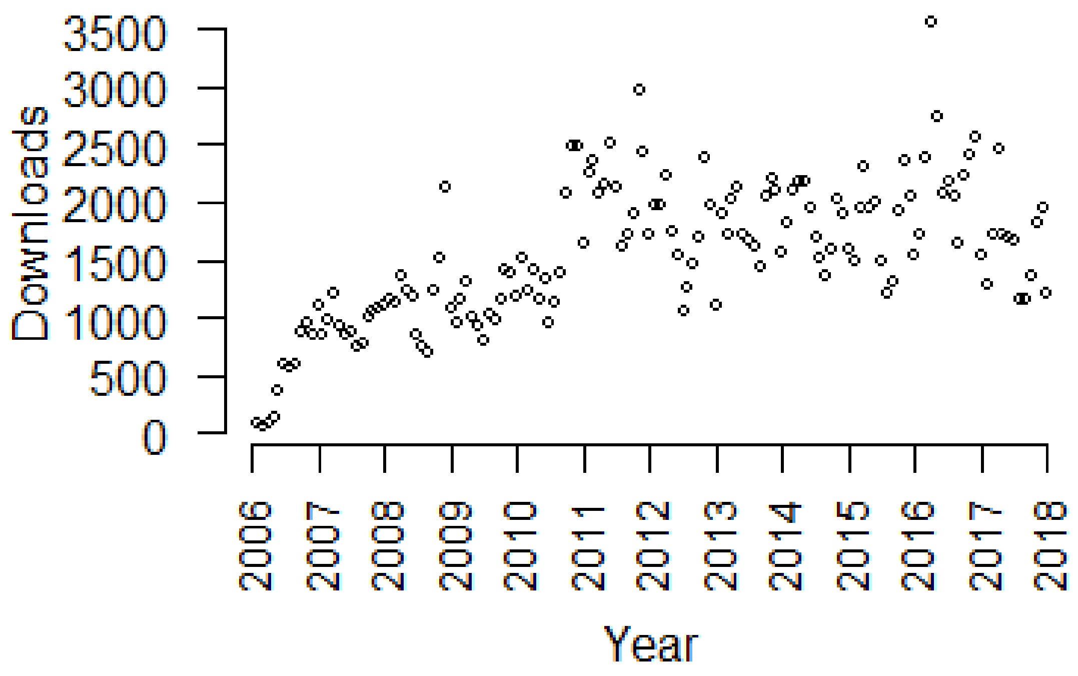

Figure 1: Download vs Date scatterplot

Plotting the lowess regression lines with different f values

```
plot(as.numeric(down$Date),down$Downloads,xaxt="n",bty="n",pch=21,cex=.5,type="p",las=1,
     ylab="Downloads",xlab="Year")
axis(1,0:12*12,2006:2018,las=3,cex.axis=.95)
lines(lowess(down$Downloads~down$MonthNumber, f=0.10), col="red")
lines(lowess(down$Downloads~down$MonthNumber, f=0.25), col="green")
lines(lowess(down$Downloads~down$MonthNumber, f=0.50), col="blue")
lines(lowess(down$Downloads~down$MonthNumber, f=0.01), col="purple")
lines(lowess(down$Downloads~down$MonthNumber, f=0.05), col="gold")
```

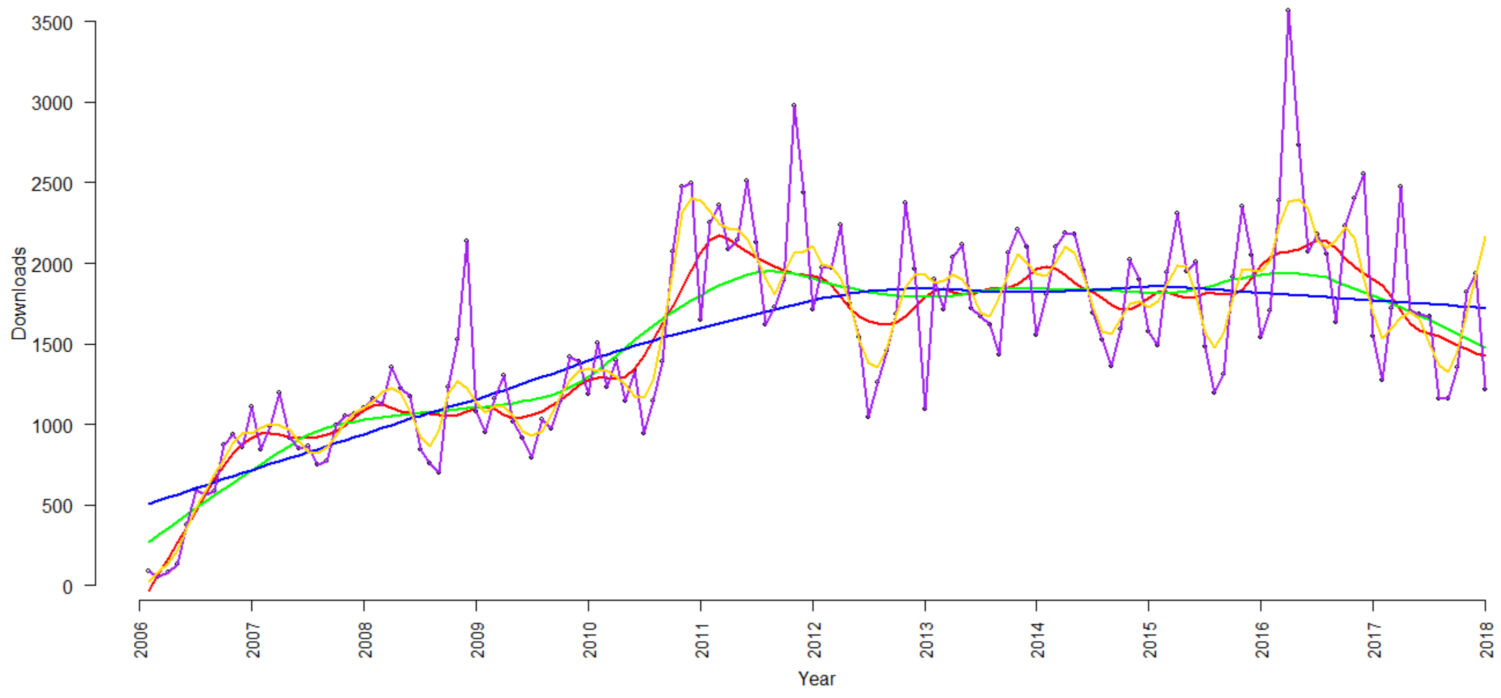

Figure 2: Lowess regression with different f values

Firstly, I plotted the regression lines with the lowess function where I changed the f values. In the above plot we see that the purple line is overfitting the data points which has f=0.01 and our best estimate here is the red line with f=0.1 so we see here that when f value is low the model is over fitted and when its high the model is under fitted so we have to choose a value which fits the model best which here in this case is the red line, f=0.1

Now plotting the loess regression lines with different span values

```
plot(as.numeric(down$Date),down$Downloads,xaxt="n",bty="n",pch=21,cex=.5,type="p",las=1,
     ylab="Downloads",xlab="Year")
axis(1,0:12*12,2006:2018,las=3,cex.axis=.95)

lmodel <- loess(down$Downloads~down$MonthNumber)
xs <- 0:50000/100
points(xs,predict(lmodel,xs),type="l",col="red",lwd=2)
summary(lmodel)
```

```
## Call:
## loess(formula = down$Downloads ~ down$MonthNumber)
##
## Number of Observations: 144
## Equivalent Number of Parameters: 4.34
## Residual Standard Error: 405.6
## Trace of smoother matrix: 4.73  (exact)
##
## Control settings:
##   span     :  0.75
##   degree   :  2
##   family   :  gaussian
##    surface  :  interpolate      cell = 0.2
##    normalize:  TRUE
##   parametric:  FALSE
##  drop.square:  FALSE
```

```r
print(paste("lmodel R^2:", cor(down$Downloads,predict(lmodel))^2))
```

```
## [1] "lmodel R^2: 0.567812171532501"
```

We here have span = 0.75 and R^2 = 0.56 and RSE = 405.6

```r
lmodel2 <- loess(down$Downloads~down$MonthNumber,span=.2,degree = 2)
points(xs,predict(lmodel2,xs),type="l",col="green",lwd=2)
summary(lmodel2)
```

```
## Call:
## loess(formula = down$Downloads ~ down$MonthNumber, span = 0.2,
##     degree = 2)
##
## Number of Observations: 144
## Equivalent Number of Parameters: 14.98
## Residual Standard Error: 351.1
## Trace of smoother matrix: 16.57  (exact)
##
## Control settings:
##    span     :  0.2
##    degree   :  2
##    family   :  gaussian
##    surface  :  interpolate      cell = 0.2
##    normalize:  TRUE
##   parametric:  FALSE
##  drop.square:  FALSE
```

```r
print(paste("lmodel2 R^2:", cor(down$Downloads,predict(lmodel2))^2))
```

```
## [1] "lmodel2 R^2: 0.706344969156763"
```

We here have span = 0.2 and R^2 = 0.706 and RSE = 351.1

```r
lmodel3 <- loess(down$Downloads~down$MonthNumber,span=.3)
points(xs,predict(lmodel3,xs),type="l",col="navy",lwd=2)
summary(lmodel3)
```

```
## Call:
## loess(formula = down$Downloads ~ down$MonthNumber, span = 0.3)
##
## Number of Observations: 144
## Equivalent Number of Parameters: 10.19
## Residual Standard Error: 358.9
## Trace of smoother matrix: 11.26  (exact)
```

```
##
## Control settings:
##   span     :  0.3
##   degree   :  2
##   family   :  gaussian
##   surface  :  interpolate      cell = 0.2
##   normalize:  TRUE
##  parametric:  FALSE
## drop.square:  FALSE
```

```
print(paste("lmodel3 R^2:", cor(down$Downloads,predict(lmodel3))^2))
```

```
## [1] "lmodel3 R^2: 0.678958327747172"
```

We here have span = 0.3 and R^2 = 0.678 and RSE = 358.9

```
lmodel4 <- loess(down$Downloads~down$MonthNumber,span=.4)
points(xs,predict(lmodel4,xs),type="l",col="gold",lwd=2)
```

```
summary(lmodel4)
```

```
## Call:
## loess(formula = down$Downloads ~ down$MonthNumber, span = 0.4)
##
## Number of Observations: 144
## Equivalent Number of Parameters: 7.71
## Residual Standard Error: 365.4
## Trace of smoother matrix: 8.5  (exact)
##
## Control settings:
##   span     :  0.4
##   degree   :  2
##   family   :  gaussian
##   surface  :  interpolate      cell = 0.2
##   normalize:  TRUE
##  parametric:  FALSE
## drop.square:  FALSE
```

```
print(paste("lmodel4 R^2:", cor(down$Downloads,predict(lmodel4))^2))
```

```
## [1] "lmodel4 R^2: 0.660427690680401"
```

We here have span = 0.4 and R^2 = 0.66 and RSE = 365.4

We see from the above results that the R^2 value is highest for lmodel2 where R^2 = 0.706. So the most reasonable span parameter is 0.2

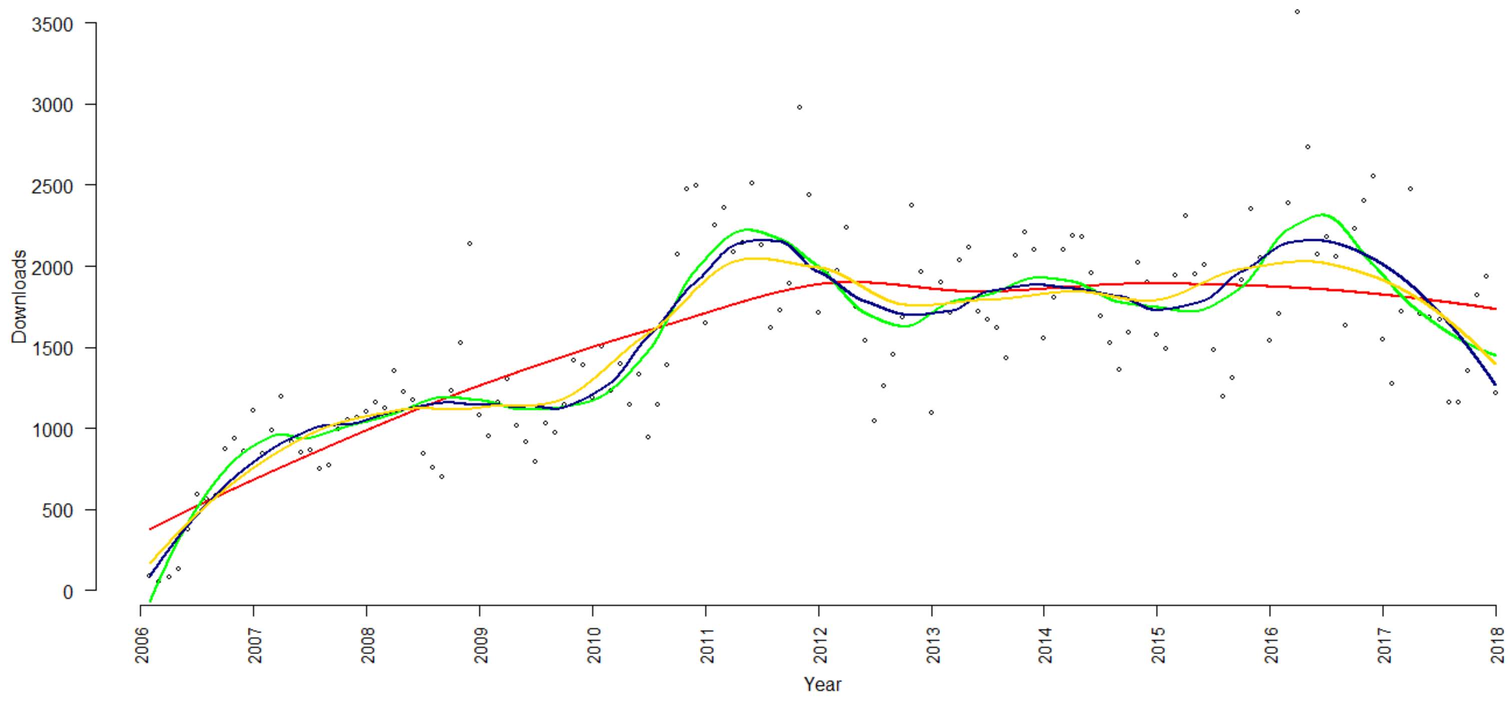

Figure 3:Loess regression lines with different span values

We plotted all the loess regression lines to the main plot and as we concluded that span = 0.2 is the best span estimate and we can validate it easily from Fig. 3 where span = 0.2 is the green line and it fits the model best as compared to other loess regression lines.

B. Polynomial regression

Plotting MonthNumber vs Downloads and drawing the loess regression line

```
down <- read.csv("DownloadData.csv")
down$Month <- as.factor(substr(down$Date,6,8))
down$MonthNumber <- 1:nrow(down)
plot(down$MonthNumber,down$Downloads,xaxt="n",bty="n",pch=21,cex=.5,type="p",las=1,
     ylab="Downloads",xlab="Year")
axis(1,0:12*12,2006:2018,las=3,cex.axis=.95)



lines(lowess(down$Downloads~down$MonthNumber))
```

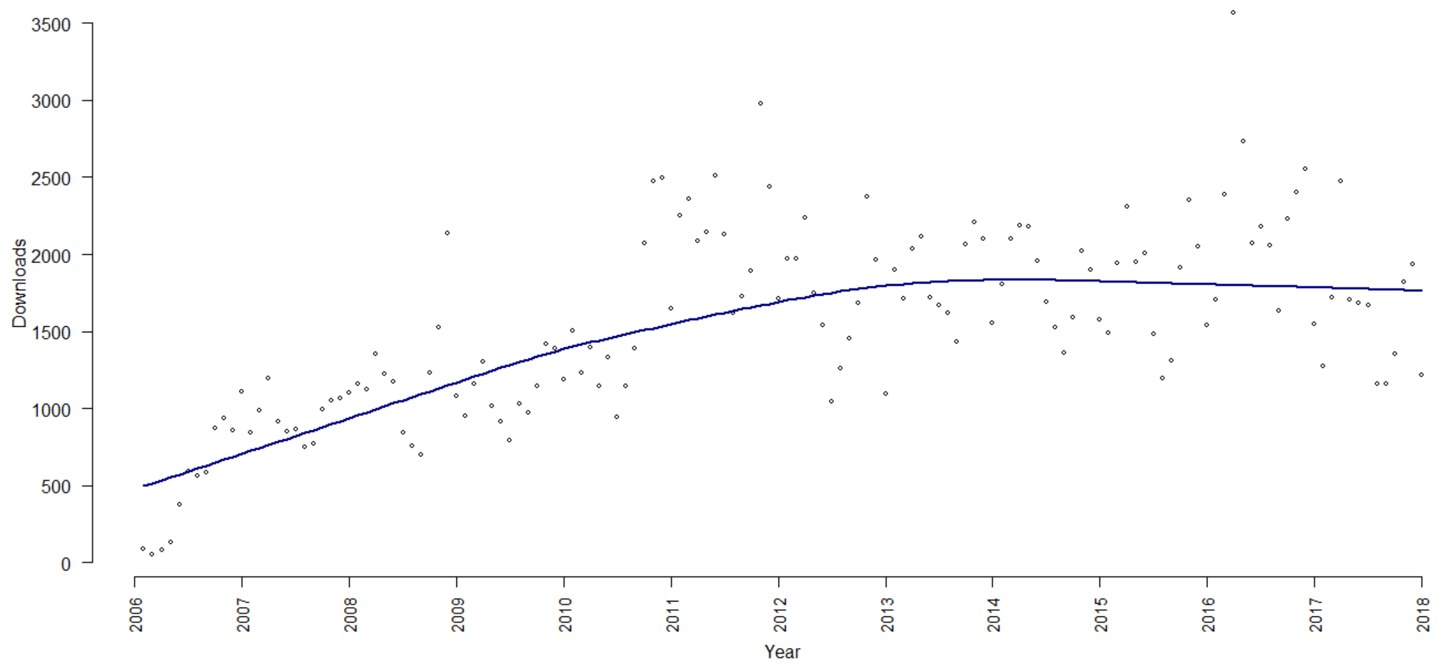

Figure 4:Plotting MonthNumber vs Downloads and drawing the loess regression line

Creating the polynomial regression with the poly() function

```
poly1 <- lm(Downloads ~ poly(MonthNumber, 1), data = down)
poly2 <- lm(Downloads ~ poly(MonthNumber, 2), data = down)
poly3 <- lm(Downloads ~ poly(MonthNumber, 3), data = down)
poly4 <- lm(Downloads ~ poly(MonthNumber, 4), data = down)
poly5 <- lm(Downloads ~ poly(MonthNumber, 5), data = down)
```

Using AIC to determine the best model

```
data.frame(model = paste ("lm" ,1:5 , sep =""),
               rbind ( extractAIC ( poly1 ),
                       extractAIC ( poly2 ),
                       extractAIC ( poly3 ),
                       extractAIC ( poly4 ),
                       extractAIC ( poly5 )))

##   model X1       X2
## 1   lm1  2 1774.614
## 2   lm2  3 1733.811
## 3   lm3  4 1735.810
## 4   lm4  5 1737.027
## 5   lm5  6 1738.086
```

We see that the AIC values decrease till lm2 and then again increases so lm2 is the best model from the results of AIC.

Now, checking with the results of BIC

```
extractBIC <- function (model)
{
  extractAIC (model ,k= log ( length ( model $ residuals )))
}

data.frame( model = paste ("lm" ,1:5 , sep =""),
            rbind ( extractBIC ( poly1 ),
                    extractBIC ( poly2 ),
                    extractBIC ( poly3 ),
                    extractBIC ( poly4 ),
                    extractBIC ( poly5 )))

##   model X1       X2
## 1   lm1  2 1780.554
## 2   lm2  3 1742.721
## 3   lm3  4 1747.690
## 4   lm4  5 1751.876
## 5   lm5  6 1755.905
```

We see here as well that the BIC values decrease till lm2 and then again increases so lm2 is the best model from the results of BIC as well.

```
summary(poly2)

##
## Call:
## lm(formula = Downloads ~ poly(MonthNumber, 2), data = down)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -807.14 -299.35   -2.19  202.95 1685.16
##
## Coefficients:
##                       Estimate Std. Error t value Pr(>|t|)
## (Intercept)            1541.67      33.95  45.406  < 2e-16 ***
## poly(MonthNumber, 2)1  4616.69     407.43  11.331  < 2e-16 ***
## poly(MonthNumber, 2)2 -2846.36     407.43  -6.986 1.03e-10 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 407.4 on 141 degrees of freedom
## Multiple R-squared:  0.5569, Adjusted R-squared:  0.5506
## F-statistic:  88.6 on 2 and 141 DF,  p-value: < 2.2e-16



pred<-predict(poly2)

plot(down$MonthNumber,down$Downloads)
lines(down$MonthNumber,y=pred,col="red")
```

Comparing both the results

```
plot(down$MonthNumber,down$Downloads,xlab = "Month Number",ylab = "Downloads")

lmodel2 <- loess(down$Downloads~down$MonthNumber,span=.2,degree = 2)
points(xs,predict(lmodel2,xs),type="l",col="green",lwd=2)

summary(lmodel2)
print(paste("lmodel2 R^2:", cor(down$Downloads,predict(lmodel2))^2))

## [1] "lmodel2 R^2: 0.706344969156763"


poly2 <- lm(Downloads ~ poly(MonthNumber, 2), data = down)
summary(poly2)
pred<-predict(poly2)
lines(down$MonthNumber,y=pred,col="red",lwd=2)
```

```
## Call:
## lm(formula = Downloads ~ poly(MonthNumber, 2), data = down)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -807.14 -299.35   -2.19  202.95 1685.16
##
## Coefficients:
##                       Estimate Std. Error t value Pr(>|t|)
## (Intercept)            1541.67      33.95  45.406  < 2e-16 ***
## poly(MonthNumber, 2)1  4616.69     407.43  11.331  < 2e-16 ***
## poly(MonthNumber, 2)2 -2846.36     407.43  -6.986 1.03e-10 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 407.4 on 141 degrees of freedom
## Multiple R-squared:  0.5569,       Adjusted R-squared:  0.5506
## F-statistic:  88.6 on 2 and 141 DF,  p-value: < 2.2e-16
```

We here have span = 0.2 and R^2 = 0.706 and RSE = 351.1 for loess regression whereas we have the R^2 = 0.5506 and RSE = 407.4 for poly() so we can conclude that the loess regression line fits the model better because it has the greater R^2 value.

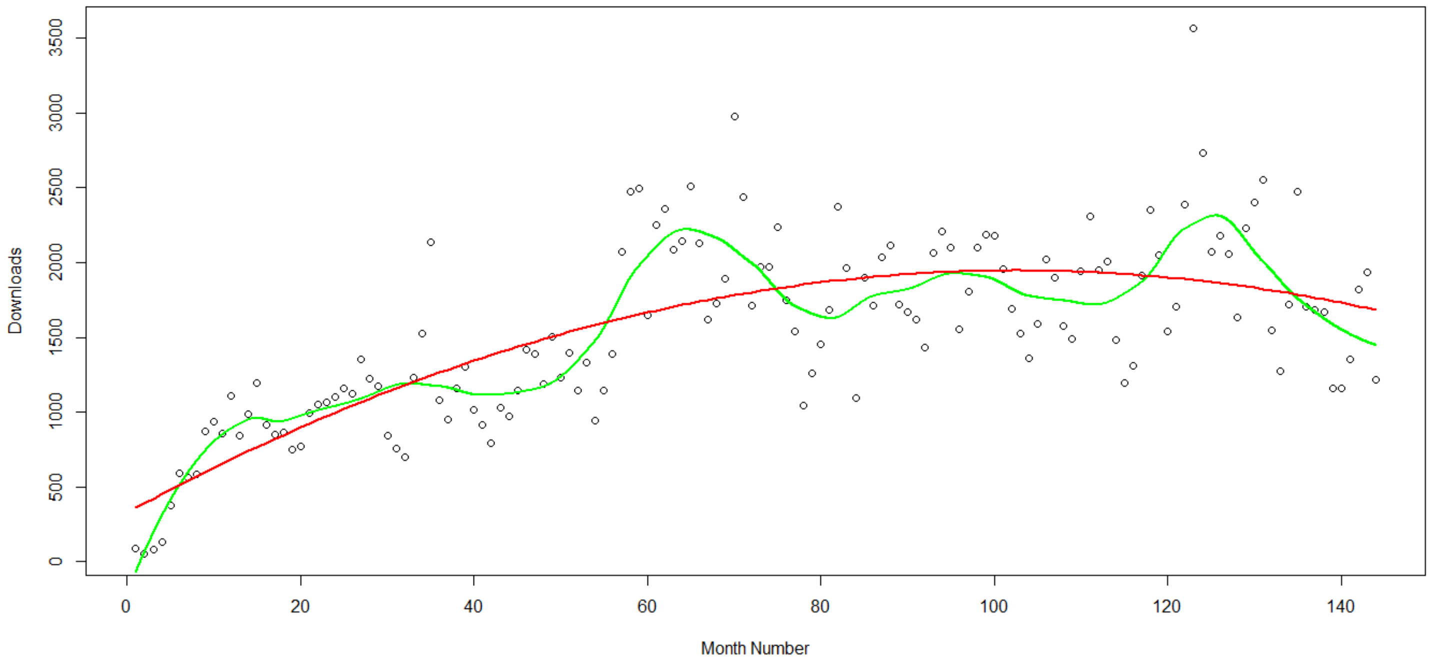

Figure 5: Comparing loess and poly()

We can verify our conclusion from the above plot the loess regression line(green) fits the model better than ploy()(red).

C. Adding month predictor

```
fit3 <- lm(Downloads ~ poly(MonthNumber,2)+as.factor(Month)+0, data = down)
pred<-predict(fit3)
```

We added month predictor to the poly function with degree 2 which was the best model as we saw the results of AIC and BIC. Also we added the 0 intercept so that each month has unique values

```
fit3

## 
## Call:
## lm(formula = Downloads ~ poly(MonthNumber, 2) + as.factor(Month) + 
##     0, data = down)
## 
## Coefficients:
## poly(MonthNumber, 2)1  poly(MonthNumber, 2)2     as.factor(Month)01  
##                  4620                  -2851                   1466  
##    as.factor(Month)02     as.factor(Month)03     as.factor(Month)04  
##                  1605                   1884                   1606  
##    as.factor(Month)05     as.factor(Month)06     as.factor(Month)07  
##                  1524                   1327                   1215  
##    as.factor(Month)08     as.factor(Month)09     as.factor(Month)10  
##                  1193                   1563                   1931  
##    as.factor(Month)11     as.factor(Month)12  
##
```

We can see that all the 12 months have a unique value.

```
summary(fit3)

##
## Call:
## lm(formula = Downloads ~ poly(MonthNumber, 2) + as.factor(Month) +
##     0, data = down)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -684.79 -226.71  -53.08  180.43 1342.22
##
## Coefficients:
##                        Estimate Std. Error t value Pr(>|t|)
## poly(MonthNumber, 2)1  4620.13     338.96   13.63  < 2e-16 ***
## poly(MonthNumber, 2)2 -2851.11     337.80   -8.44 5.36e-14 ***
## as.factor(Month)01     1466.43      97.58   15.03  < 2e-16 ***
## as.factor(Month)02     1605.46      97.56   16.46  < 2e-16 ***
## as.factor(Month)03     1884.47      97.54   19.32  < 2e-16 ***
## as.factor(Month)04     1606.12      97.53   16.47  < 2e-16 ***
## as.factor(Month)05     1523.82      97.52   15.63  < 2e-16 ***
## as.factor(Month)06     1327.17      97.51   13.61  < 2e-16 ***
## as.factor(Month)07     1215.49      97.51   12.46  < 2e-16 ***
## as.factor(Month)08     1193.20      97.52   12.24  < 2e-16 ***
## as.factor(Month)09     1562.97      97.53   16.03  < 2e-16 ***
## as.factor(Month)10     1931.13      97.54   19.80  < 2e-16 ***
## as.factor(Month)11     1867.27      97.56   19.14  < 2e-16 ***
## as.factor(Month)12     1316.46      97.58   13.49  < 2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 337.8 on 130 degrees of freedom
## Multiple R-squared:  0.9625, Adjusted R-squared:  0.9584
## F-statistic


plot(down$MonthNumber,down$Downloads)
lines(down$MonthNumber,y=pred,col="red")

lmodel2 <- loess(down$Downloads~down$MonthNumber,span=.2,degree = 2)
points(xs,predict(lmodel2,xs),type="l",col="green",lwd=2)

poly2 <- lm(Downloads ~ poly(MonthNumber, 2), data = down)
pred<-predict(poly2)
lines(down$MonthNumber,y=pred,col="navy",lwd=2)
```

Models	Loess regression	Poly()	+0 intercept
R^2	0.706	0.5506	0.9584

We see here that the R^2 value is highest for our model with +0 intercept so we can say that it best fits our model.

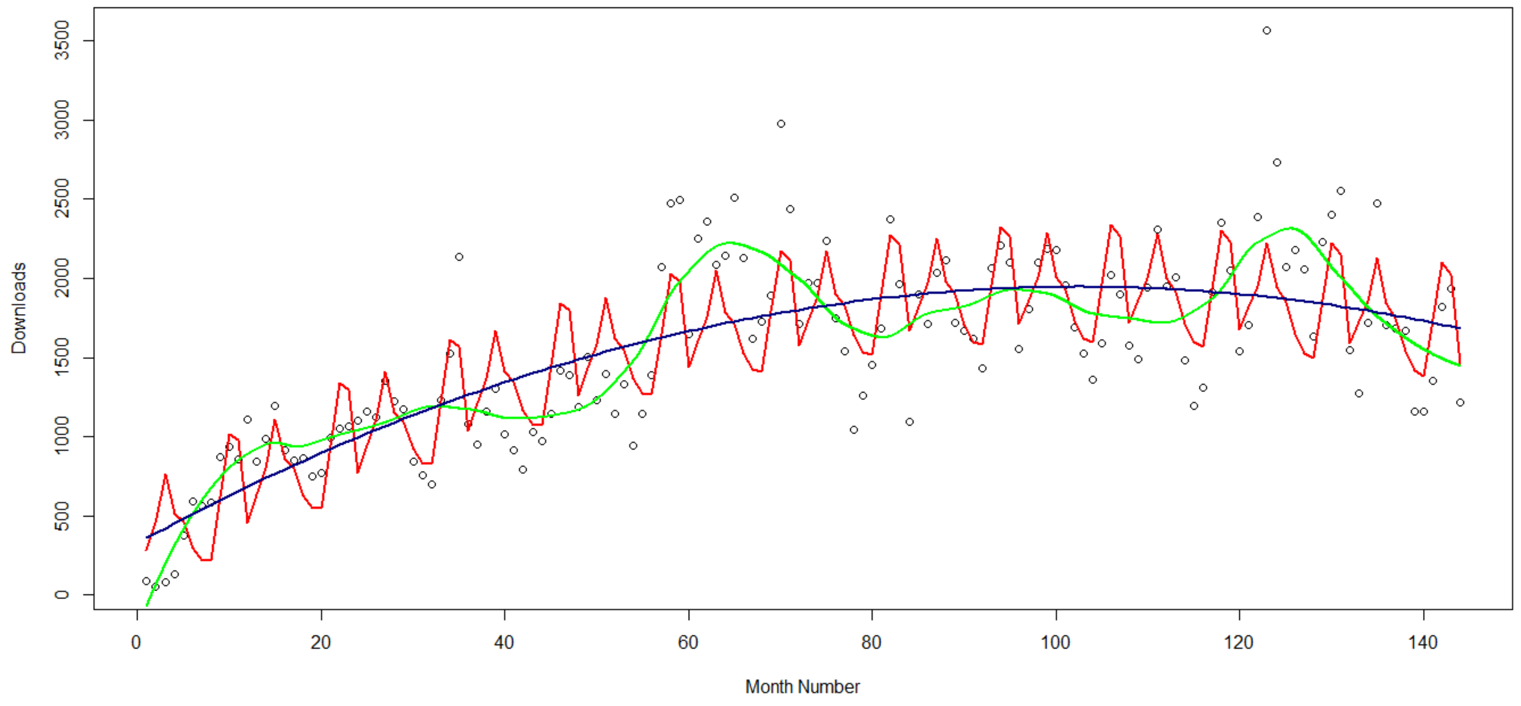

Figure 6: Comparison of all 3 best models

On comparing loess(green), poly() (blue) and +0 intercept(red) we see that this model is fitting the values better than the other 2 models.