CONCEPTS beyond linearity: Splines

Table of Contents

[1. Executive Summary 2](#_Toc468048398)

[2. About Data 2](#_Toc468048399)

[2.1 Data Exploration 3](#_Toc468048400)

[3. Linear Regression 4](#_Toc468048401)

[4. Polynomial Models 5](#_Toc468048402)

[5. Stepwise Function 7](#_Toc468048403)

[6. Regression Splines 8](#_Toc468048404)

[7. Smoothing Splines 9](#_Toc468048405)

[8. Local Regression 10](#_Toc468048406)

[8.1 Comparing Local Regression Splines vs Linear Regression: 13](#_Toc468048407)

[9. Generalized Additive Models 14](#_Toc468048408)

[9.1 GAM using Natural splines 14](#_Toc468048409)

[9.2 GAM using Smoothening splines 17](#_Toc468048410)

[9.3 GAM using Local Regression 19](#_Toc468048411)

[10. References 21](#_Toc468048412)

[11. Appendix 22](#_Toc468048413)

# Executive Summary

Linear models are simple to describe and implement, and have advantages over other modern approaches in terms of interpretation and inference. But they do come with certain limitations in terms of predictive power. This is mainly because linearity assumption is always almost an approximation and may not be a real-time problem solving one. Splines is an improvement to such model which relaxes the assumption of linearity and while still attempting to maintain as much interpretability possible compared to linear models. We explored polynomial regression, Step functions (piecewise linear functions), splines, local regression and generalized additive models to move on to the world beyond linearity.

# About Data

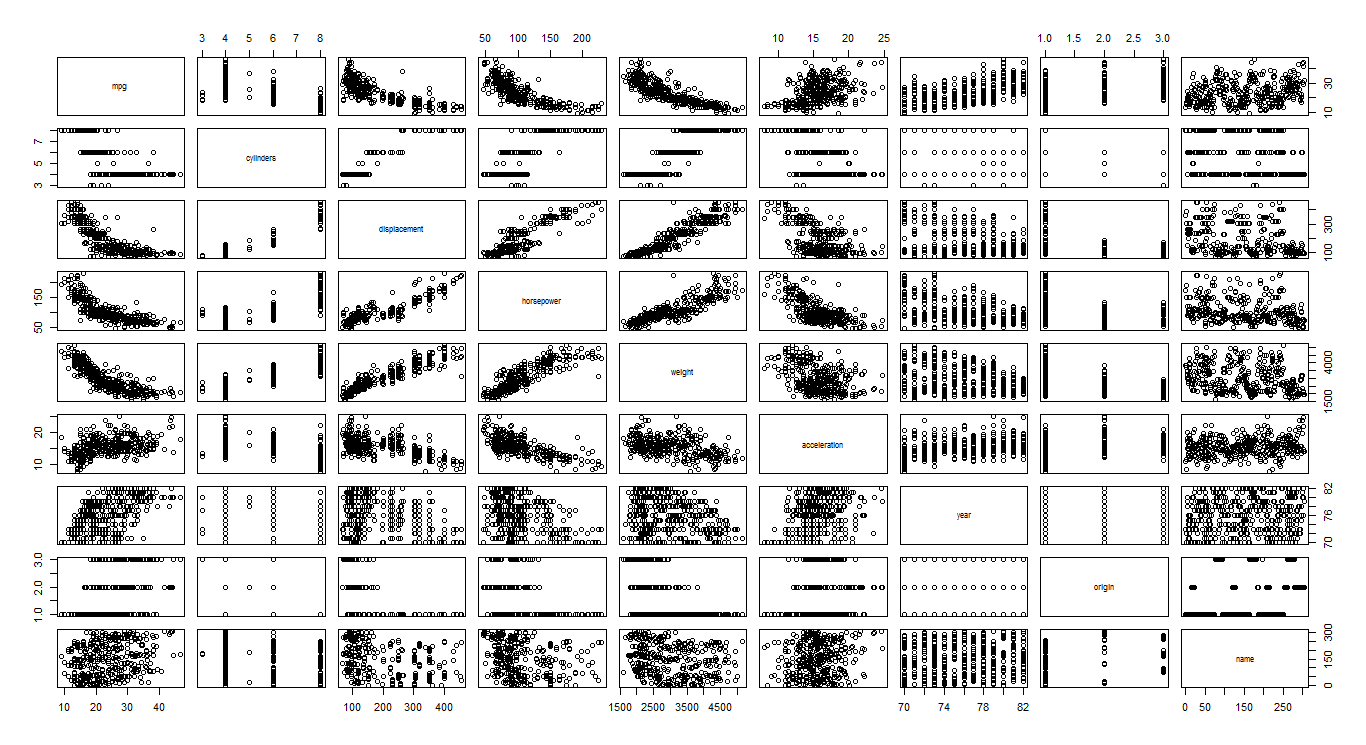
We selected ‘Auto’ data set from ISLR package in R to explore the concept of non-linear curve fitting or the splines.

Auto data set has information about Gas Mileage, Horsepower, displacement and other information for 392 vehicles. In short it has 392 records and 9 variables. Variables are:

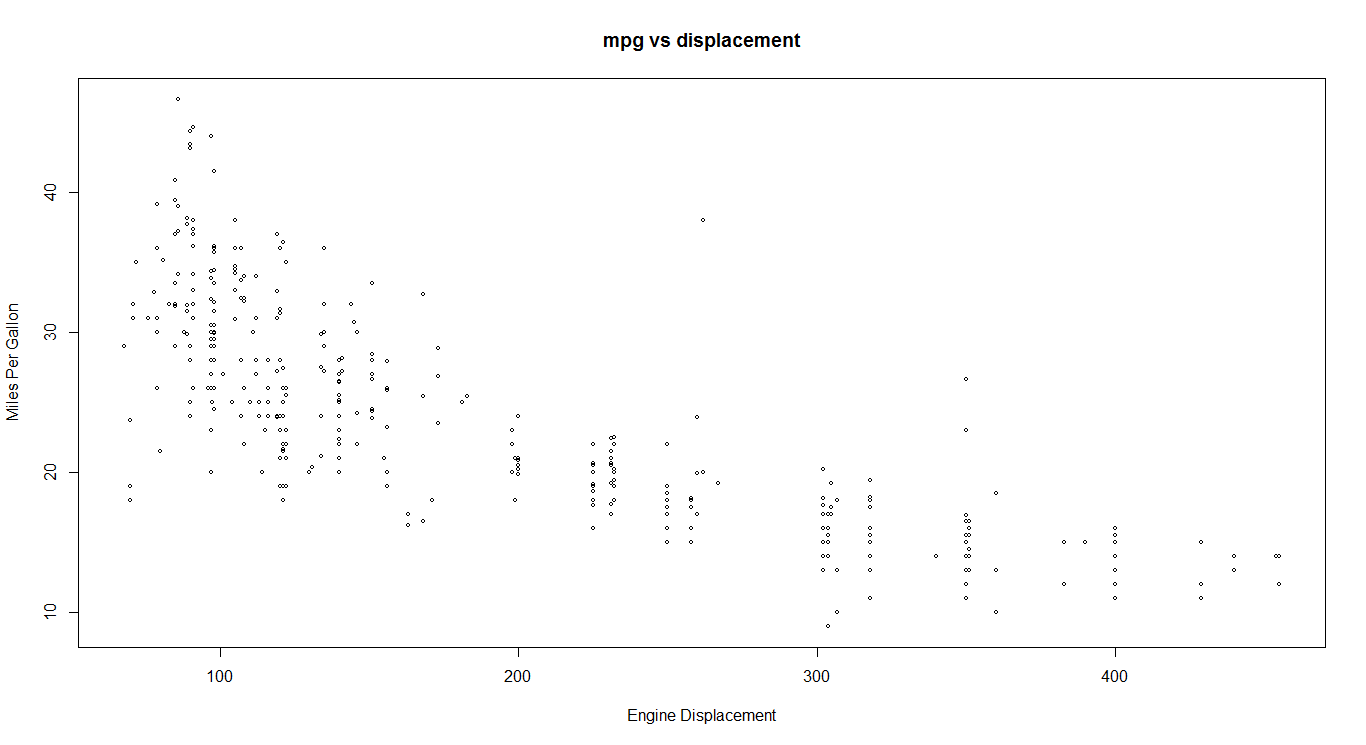
* Mpg 🡪 miles per gallon
* Cylinders 🡪 Number of cylinders between 4 and 8
* Displacement 🡪 Engine displacement (cu. inches)
* Horsepower 🡪 Engine horsepower
* Weight 🡪 Vehicle weight (lbs.)
* Acceleration 🡪 Time to accelerate from 0 to 60 mph (sec.)
* Year Model 🡪 year (modulo 100)
* Origin 🡪 Origin of car (1. American, 2. European, 3. Japanese)
* Name 🡪 Vehicle name

## 2.1 Data Exploration

To explain the concepts of non-linear or non-parametric regression, we should select one(mostly) or more variables which distinguishes sharp line of linearity and non-linearity towards dependent variable. We selected ‘mpg’ as dependent variable and saw how other independent variables behave respectively. Using pairs(Auto) function in R, we see that ‘displacement’, ‘horsepower’ and ‘weight’ show this property of non-linearity among them with respect to ‘mpg’. We decided to go ahead with the ‘displacement’ in the subsequent sections to explain the concepts.



mpg vs displacement:

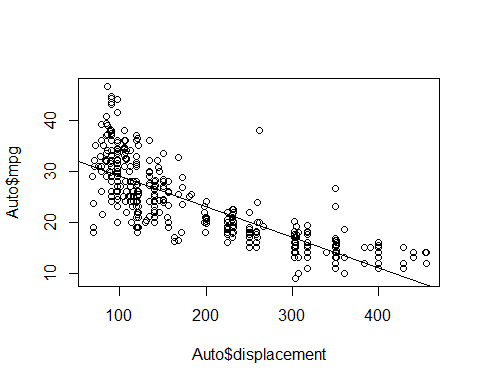


We see that as engine displacement increases, miles per gallon go on to reduce. For engines with less displacement, mileage of the vehicle is very high and vice versa.

# Linear Regression

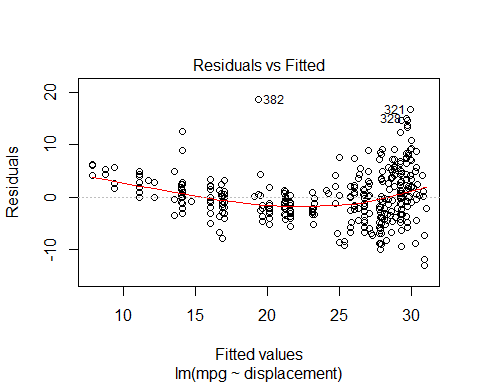
Fitting a linear regression to see how it fits the above data (mpg vs displacement)

#The below command gives us the plot of displacement and mpg and a corresponding linear fit   
plot(Auto$displacement, Auto$mpg)  
abline(lm(mpg~displacement))



*Figure 1: Linear fit for displacement vs mpg*

plot(lm(mpg~displacement))



*Figure 2: Residual vs filled plot*

The residual plot reveals that the assumption of homoscedasticity is violated. There is something beyond linearity here.

# Polynomial Models

Once we found out that the relation between predictors and dependent variable is nonlinear, one option available is to fit a polynomial model. A polynomial model is an extension to the linear model with extra predictors which are the raised powers (higher degree) of the original predictors.

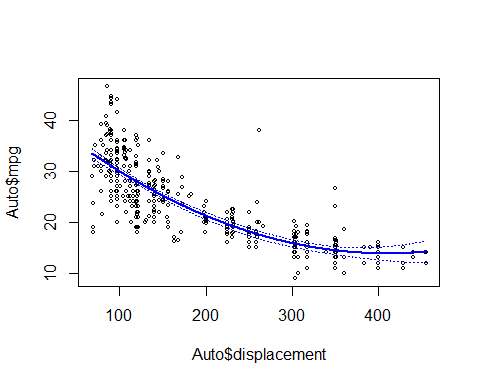
yi = β0 + β1xi + β2xi2 + β3xi3 + ... + βdxid + Ei

This is called as polynomial regression.

Model2=lm(mpg ~ poly(displacement ,2) ,data=Auto) #polynomial of 2 degree  
(summary (Model2))

## lm(formula = mpg ~ poly(displacement, 2), data = Auto)  
##   
## Residuals:  
## Min 1Q Median 3Q Max   
## -15.2165 -2.2404 -0.2508 2.1094 20.5158   
##   
## Coefficients:  
## Estimate Std. Error t value Pr(>|t|)   
## (Intercept) 23.4459 0.2205 106.343 < 2e-16 \*\*\*  
## poly(displacement, 2)1 -124.2585 4.3652 -28.466 < 2e-16 \*\*\*  
## poly(displacement, 2)2 31.0895 4.3652 7.122 5.17e-12 \*\*\*  
## ---  
## Signif. codes: 0 '\*\*\*' 0.001 '\*\*' 0.01 '\*' 0.05 '.' 0.1 ' ' 1  
##   
## Residual standard error: 4.365 on 389 degrees of freedom  
## Multiple R-squared: 0.6888, Adjusted R-squared: 0.6872   
## F-statistic: 430.5 on 2 and 389 DF, p-value: < 2.2e-16

allparam(Model2)

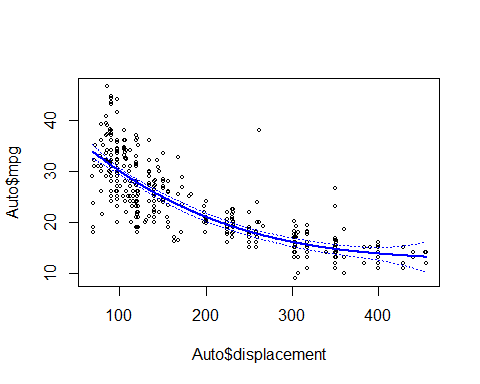


*Figure 3: Polynomial function of degree 2*

Model3=lm(mpg ~ poly(displacement ,3) ,data=Auto) #polynomial of 2 degree  
(summary (Model3))

##   
## Call:  
## lm(formula = mpg ~ poly(displacement, 3), data = Auto)  
##   
## Residuals:  
## Min 1Q Median 3Q Max   
## -15.6791 -2.3900 -0.2987 2.1156 20.4528   
##   
## Coefficients:  
## Estimate Std. Error t value Pr(>|t|)   
## (Intercept) 23.4459 0.2205 106.350 < 2e-16 \*\*\*  
## poly(displacement, 3)1 -124.2585 4.3649 -28.468 < 2e-16 \*\*\*  
## poly(displacement, 3)2 31.0895 4.3649 7.123 5.18e-12 \*\*\*  
## poly(displacement, 3)3 -4.4655 4.3649 -1.023 0.307   
## ---  
## Signif. codes: 0 '\*\*\*' 0.001 '\*\*' 0.01 '\*' 0.05 '.' 0.1 ' ' 1  
##   
## Residual standard error: 4.365 on 388 degrees of freedom  
## Multiple R-squared: 0.6896, Adjusted R-squared: 0.6872   
## F-statistic: 287.4 on 3 and 388 DF, p-value: < 2.2e-16

allparam(Model3)



*Figure 4: Polynomial function of degree 3*

anova(Model1,Model2,Model3)

**## Analysis of Variance Table**  
## Model 1: mpg ~ poly(displacement)  
## Model 2: mpg ~ poly(displacement, 2)  
## Model 3: mpg ~ poly(displacement, 3)  
## Res.Df RSS Df Sum of Sq F Pr(>F)   
## 1 390 8378.8   
## 2 389 7412.3 1 966.56 50.7317 5.18e-12 \*\*\*  
## 3 388 7392.3 1 19.94 1.0466 0.3069   
## ---  
## Signif. codes: 0 '\*\*\*' 0.001 '\*\*' 0.01 '\*' 0.05 '.' 0.1 ' ' 1

The anova function reveals that quadratic with degree 3 is sufficient There are quite a few disadvantages when a polynomial of degree > 3 is fit on the data. It may take unusual shapes and model becomes complex. Hence a stepwise function has been fit.

# Stepwise Function

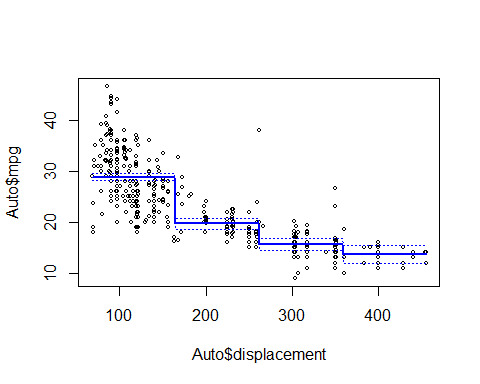
Step functions cut the range of a variable into ‘K’ distinct regions in order to produce a qualitative variable. This has the effect of fitting a piecewise constant function. In Auto dataset, stepwise function cuts the region into 4 parts and fits a piece wise constant function for each of these regions. The region can be divided into any number of parts. The region is divided automatically at 165,262,358,455.

table(cut(displacement ,4)) # cut the region in to 4 parts  
## (67.6,165] (165,262] (262,358] (358,455]   
## 213 78 72 29

Stepwise<-lm(mpg~cut(displacement,4))  
summary(Stepwise)

##   
## Call:  
## lm(formula = mpg ~ cut(displacement, 4))  
## Residuals:  
## Min 1Q Median 3Q Max   
## -12.6099 -2.8099 -0.6897 2.3093 22.4250   
##   
## Coefficients:  
## Estimate Std. Error t value Pr(>|t|)   
## (Intercept) 28.8099 0.3367 85.56 <2e-16 \*\*\*  
## cut(displacement, 4)(165,262] -9.1188 0.6504 -14.02 <2e-16 \*\*\*  
## cut(displacement, 4)(262,358] -13.2349 0.6699 -19.76 <2e-16 \*\*\*  
## cut(displacement, 4)(358,455] -15.1202 0.9727 -15.54 <2e-16 \*\*\*  
## ---  
## Signif. codes: 0 '\*\*\*' 0.001 '\*\*' 0.01 '\*' 0.05 '.' 0.1 ' ' 1  
##   
## Residual standard error: 4.914 on 388 degrees of freedom  
## Multiple R-squared: 0.6066, Adjusted R-squared: 0.6036   
## F-statistic: 199.4 on 3 and 388 DF, p-value: < 2.2e-16

allparam(Stepwise)



*Figure 5: Piece wise constant function*

# Regression Splines

Instead of fitting a high-degree polynomial over the entire range of X, piecewise polynomial regression splines involve fitting separate low-degree polynomials over different regions of X.

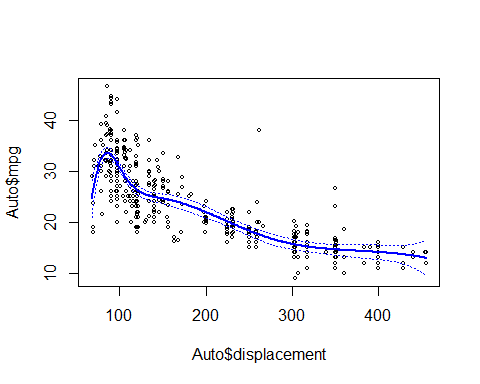
The equations are of the type, yi =b01 + b11xi + b21xi2 + b31xi3 + Ei

The regions are divided based on the knots mentioned. Here we have mentioned 3 knots which are 25,50,75 quantiles of displacement. We can also use a df option to produce splines. In general, for a cubic spline with K knots the degree of freedom = 4+K. This is further illustrated in smooth splines

quantile(displacement)

## 0% 25% 50% 75% 100%   
## 68.00 105.00 151.00 275.75 455.00

splinemodel<-lm(mpg ~ bs(displacement, knots = c(105,151,275)), data = Auto)  
allparam(splinemodel)



*Figure 6: Regression spline with knots at 105,151,275*

# Smoothing Splines

In fitting a smooth curve to a set of data, what we really want to do is find some function, say g(x), that fits the observed data well: that is, we want RSS = ∑ (yi - g(xi))2 to be small. However, there is a problem with this approach. If we don't put any constraints on g(xi), then we can always make RSS zero simply by choosing g such that it interpolates all of the yi. Such a function would woefully overfit the data-it would be far too flexible. What we really want is a function g that makes RSS small, but that is also smooth. How could we ensure that g is smooth? There are a number of ways to do this. A natural approach is to find the function g that minimizes

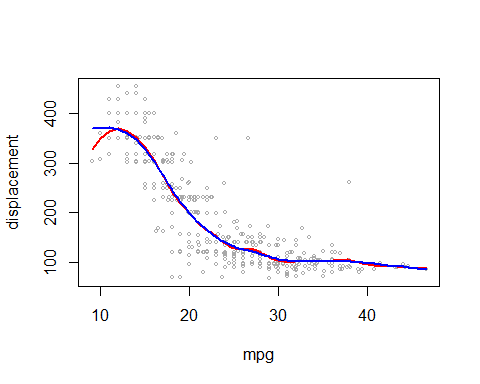
∑ (yi - g(xi))2 + λ ʃg’’(t)2dt where λ is a non-negative tuning parameter

The smooth model uses 16 degrees of freedom to train the data. However, a more sophisticated approach uses Leave one out cross validation to automatically select the degrees of freedom.

smooth<-smooth.spline(mpg,displacement , df = 15)  
smooth2=smooth.spline (mpg ,displacement ,cv=TRUE)

## Warning in smooth.spline(mpg, displacement, cv = TRUE): cross-validation  
## with non-unique 'x' values seems doubtful

plot(mpg,displacement ,cex =.5,col=" darkgrey ")  
lines(smooth ,col="red",lwd =2)  
lines(smooth2,col="blue",lwd=2)



*Figure 7: Smooth splines with 1. Degrees of freedom and 2. Cross validation*

The resultant graph shows that the second model gives only 8 degrees of freedom but the first one gives 15 degrees of freedom. However, the fit looks similar.

# Local Regression

Local regression is an extension to Smoothing splines, which involves computing the fit at a target point using observations in its neighbourhood. It assumes constant variance and estimates the fit using least squares criterion. In local regression, a fraction (s=k/n) of training points closest to the target observation are selected and are assigned with weights. The weights are assigned such that observations closest to the target get more weight and these weights tends to 0 as we go away further from the target. Using these weights, a least squares regression is fitted which tend to reduce the residual sum of squares.

In fitting a local regression, there are many assumptions to be made like degree of the polynomial, weight functions etc. but most important is defining the span.

Span, s=k/n

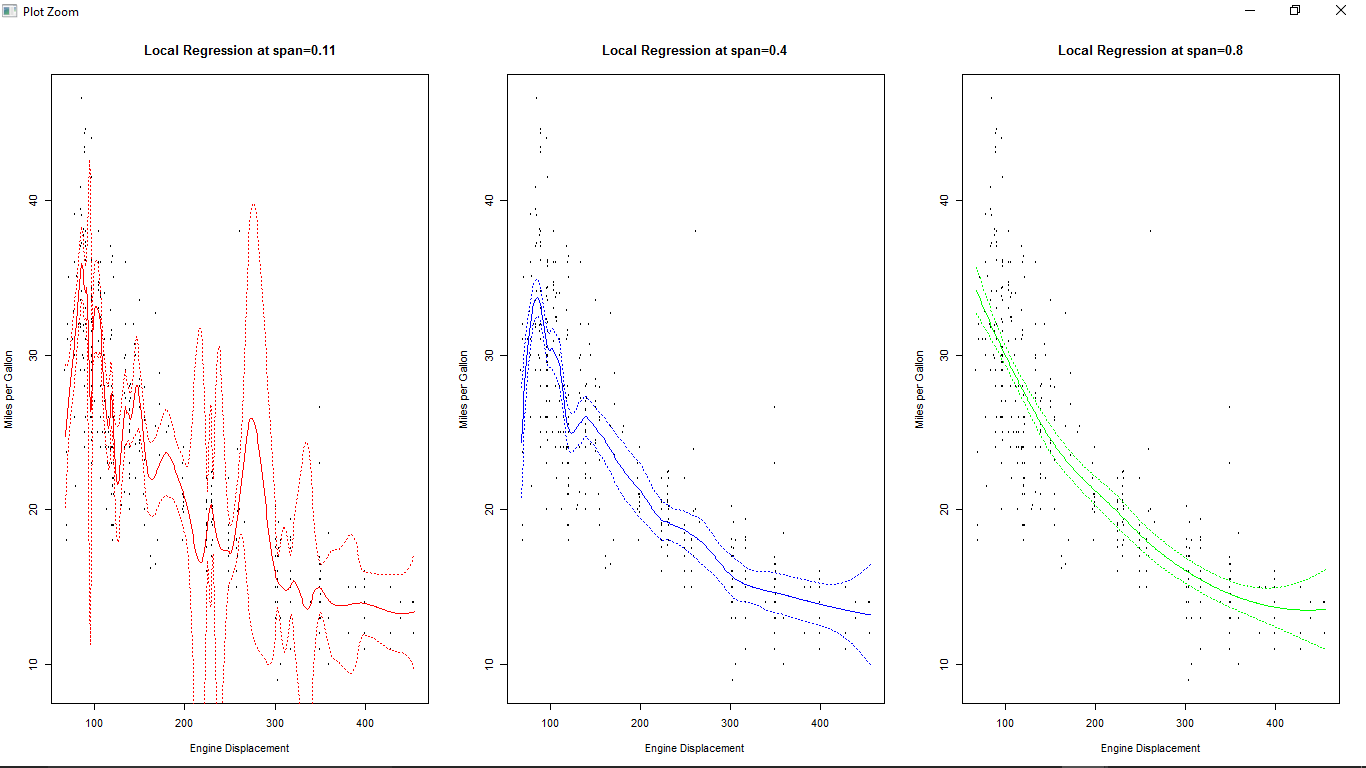
k is k nearest neighbours and n is the total number of observations.

Span controls the smoothing of the curve and flexibility of a non-linear fit. Smaller the span value, less number of observations are considered in the neighbourhood of target observation and wigglier the curve fits. As the span value increases, curve smoothens and becomes globally fit regression at span equal to 1. Ideal span value should be considered depending on the domain and the data distribution. There is a bias-variance trade-off happening while choosing span for the curve.

Local regression can be applied to data with multiple variables, however as the number of variables increases, local regression performs poorly. In general, at most 3 or 4 variables are applied with local regression to get benefit from the results.

In R, ‘loess’ function is used to fit local polynomial regression. For Auto data (from ISLR package), we considered ‘mpg’ and examined how it varied with ‘displacement’.

We tried fitting a smooth curve by varying span values to the data. Following is the plot and code for fitting a 2nd degree local regression on Auto dataset. We also included confidence intervals at 95% to the fitted curves.



localreg1=loess(Auto$mpg~Auto$displacement,degree=2,span=0.11)

localreg2=loess(Auto$mpg~Auto$displacement,degree=2,span=0.3)

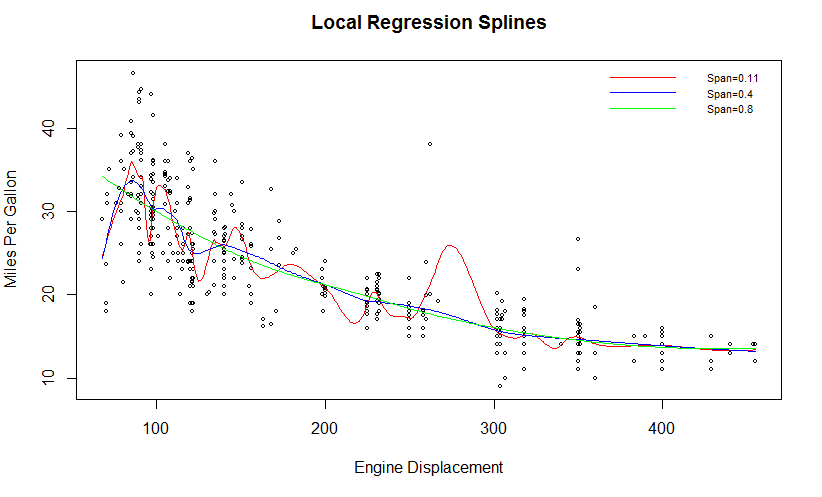
localreg3=loess(Auto$mpg~Auto$displacement,degree=2,span=0.8)

We see that at span = 0.11, curve fits the data at almost all the observations making it an overfit model. Confidence intervals to this curve is way wigglier than the curve itself.

At span = 0.4, curve is not fitted at every point and is smoother compared to the previous plot. Confidence intervals follow the fitted curve and are wider only at the boundaries.

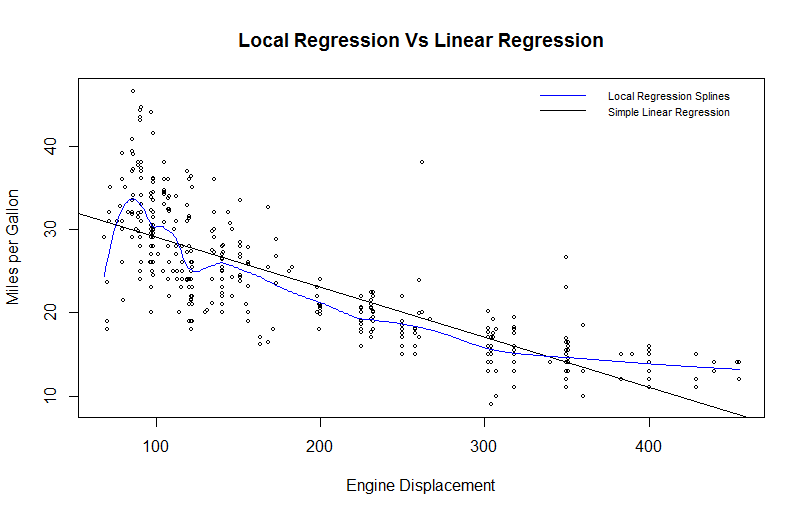
At span = 0.8, curve is way smoother and tends to be a simple linear curve for most of the part. Confidence intervals are closer to the fit and wide at boundaries.

Combining all the three graphs, we get a clear on how span smoothens the variance.



## 8.1 Comparing Local Regression Splines vs Linear Regression:

On comparing simple linear regression and local regression splines on Auto dataset, we see that local regression splines fit the data more effectively and curves according to the data intensity which is not captured by Linear regression.



# Generalized Additive Models

In this section, let us explore prediction of response Y, considering multiple variables using GAM. GAM provides flexible statistical methods to characterize nonlinear regression effects.

In general, multiple linear regression is modelled as

Additive regression replaces each linear term with more generalized functional form

where is non-parametric function which makes the model more flexible. We calculate for each ,and adding them together, makes the model additive.

In the Auto dataset, Miles per gallon(mpg) is predicted using other independent variables.

Build a linear relation between response and predictor variables using simple linear regression.

#Linear Regression model assuming linear relationship between response variable and predictors  
baseline=lm(mpg~.,data = Auto)

## 9.1 GAM using Natural splines

Let us fit GAM considering non-parametric function as Natural splines on Auto dataset. We use lm() function in the base package to build a multi linear regression model with appropriate choice of natural splines on predictors.

Writing mpg as a function of natural splines on variables horsepower, acceleration, displacement, year and origin with appropriate degrees of freedom(df).

gam1=lm(mpg~(ns(horsepower,df=4)+ns(acceleration,df=3)+ns(displacement,df=4)+ns(year,df=4)+ns(origin)),data=Auto)

Function ns() in splines library – Fits natural cubic spline for the predictor

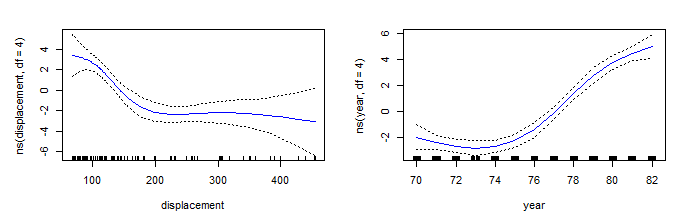
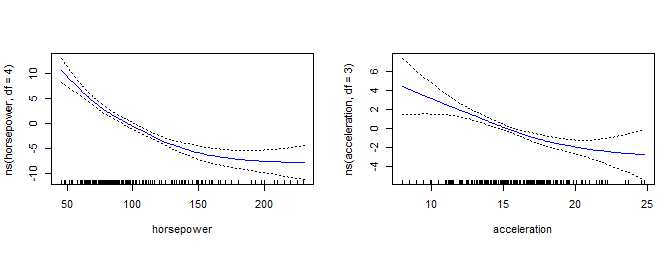
ns(x, df = NULL, knots = NULL)

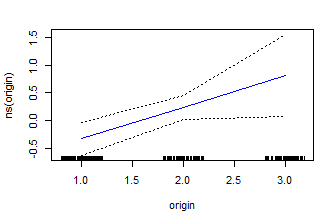
x is the predictor variable; df is degrees of freedom; knots are breakpoints that define the spline. The default is no knots; together with the natural boundary conditions this results in a basis for linear regression on x. Typical values are the mean or median for one knot, quantiles for more knots.

ns() choses knots (df - 1 – intercept) at suitable quantiles of x (ignoring missing values). By default, df = 1, corresponds to no knots. We can give either df or knots in the function ns().

*Results of generalized model fit on Auto dataset with mpg as predicting variable using least squares method*

gamoutput(gam1)   
## Coefficients:  
## Estimate Std. Error t value Pr(>|t|)   
## (Intercept) 39.5363 1.8717 21.123 < 2e-16 \*\*\*  
## ns(horsepower, df = 4)1 -10.8343 1.2974 -8.351 1.33e-15 \*\*\*  
## ns(horsepower, df = 4)2 -15.3808 1.6071 -9.571 < 2e-16 \*\*\*  
## ns(horsepower, df = 4)3 -24.2361 2.8629 -8.466 5.82e-16 \*\*\*  
## ns(horsepower, df = 4)4 -14.9039 2.0269 -7.353 1.23e-12 \*\*\*  
## ns(acceleration, df = 3)1 -5.1599 1.0811 -4.773 2.61e-06 \*\*\*  
## ns(acceleration, df = 3)2 -9.6339 3.1909 -3.019 0.002708 \*\*   
## ns(acceleration, df = 3)3 -5.6928 1.5442 -3.686 0.000261 \*\*\*  
## ns(displacement, df = 4)1 -6.4888 1.2934 -5.017 8.13e-07 \*\*\*  
## ns(displacement, df = 4)2 -4.8029 1.4254 -3.369 0.000831 \*\*\*  
## ns(displacement, df = 4)3 -6.4142 2.4546 -2.613 0.009332 \*\*   
## ns(displacement, df = 4)4 -6.1581 1.9240 -3.201 0.001488 \*\*   
## ns(year, df = 4)1 -0.1829 0.7202 -0.254 0.799616   
## ns(year, df = 4)2 5.7687 0.6864 8.404 9.08e-16 \*\*\*  
## ns(year, df = 4)3 5.4094 1.3002 4.161 3.94e-05 \*\*\*  
## ns(year, df = 4)4 7.6867 0.6019 12.771 < 2e-16 \*\*\*  
## ns(origin) 1.4179 0.6426 2.207 0.027945 \*   
## ---  
## Signif. codes: 0 '\*\*\*' 0.001 '\*\*' 0.01 '\*' 0.05 '.' 0.1 ' ' 1  
##   
## Residual standard error: 2.793 on 375 degrees of freedom  
## Multiple R-squared: 0.8772, Adjusted R-squared: 0.872   
## F-statistic: 167.4 on 16 and 375 DF, p-value: < 2.2e-16





*Figure 8: Each plot displays the fitted function and pointwise standard errors.*

**Interpretation of plots**: The first plot depicts that keeping acceleration, displacement year and origin constant, mpg tends to decrease with increase in horsepower. The second plot depicts that keeping other predictors fixed, mpg decreases with increase in acceleration. The third plot depicts that mpg decrease with increase in displacement and tends to becomes constant at certain point with increase in displacement. The fourth plot indicates that as year increases the mpg tends to increase; this may be due to advanced technology in the model manufactured in latest year. The fifth plot indicates that as origin increase the mpg tends to increase. Origin of car (1. American, 2. European, 3. Japanese); Japanese vehicle models give better mileage compared to other countries. The results are instinctual.

## 9.2 GAM using Smoothening splines

Let us fit GAM considering non-parametric function as Smoothening spline on Auto dataset.

#Load gam package to use smoothening spline functions available in this package  
library(gam)

Smoothening splines cannot be fit using least squares method. Install and load gam package in R. gam () function in this library can be used to fit smoothening splines using *backfitting* approach. gam uses backfitting algorithm to combine different smoothing or fitting methods.

gam.s1=gam(mpg~s(acceleration,3)+s(horsepower,df=3)+s(displacement,df=3)+s(year,df=3),data=Auto)

gam (formula, family, data ….)

formula - a formula expression as for other regression models, of the form response ~ predictors

family - a description of the error distribution and link function to be used in the model.

data - an optional data frame containing the variables in the model. If not found in data, the variables are taken from environment(formula), typically the environment from which gam is called.

s (x, df=4, spar=1)

is function to indicate a smooth term in a formula argument to gam

x - the univariate predictor, or expression, that evaluates to a numeric vector.

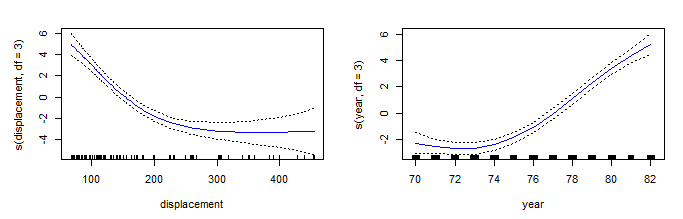
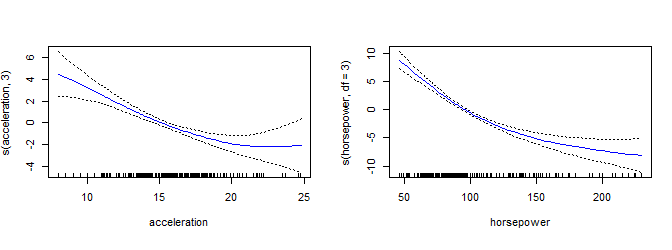
df - the target equivalent degrees of freedom, used as a smoothing parameter. The real smoothing parameter (spar below) is found such that df=tr(S)-1, where S is the implicit smoother matrix. Values for df should be greater than 1, with df=1 implying a linear fit. If both df and spar are supplied, the former takes precedence.

spar - can be used as smoothing parameter, with values typically in (0,1]

*Results of smoothening spline model fit on Auto dataset with mpg as predicting variable using backfitting algorithm*

gamoutput(gam.s1)

##   
## Anova for Parametric Effects  
## Df Sum Sq Mean Sq F value Pr(>F)   
## s(acceleration, 3) 1 4887.6 4887.6 612.542 < 2.2e-16 \*\*\*  
## s(horsepower, df = 3) 1 11568.1 11568.1 1449.793 < 2.2e-16 \*\*\*  
## s(displacement, df = 3) 1 468.2 468.2 58.679 1.566e-13 \*\*\*  
## s(year, df = 3) 1 2185.3 2185.3 273.877 < 2.2e-16 \*\*\*  
## Residuals 379 3024.1 8.0   
## ---  
## Signif. codes: 0 '\*\*\*' 0.001 '\*\*' 0.01 '\*' 0.05 '.' 0.1 ' ' 1  
##   
## Anova for Nonparametric Effects  
## Npar Df Npar F Pr(F)   
## (Intercept)   
## s(acceleration, 3) 2 3.409 0.03408 \*   
## s(horsepower, df = 3) 2 39.374 2.220e-16 \*\*\*  
## s(displacement, df = 3) 2 33.982 2.665e-14 \*\*\*  
## s(year, df = 3) 2 28.685 2.515e-12 \*\*\*  
## ---  
## Signif. codes: 0 '\*\*\*' 0.001 '\*\*' 0.01 '\*' 0.05 '.' 0.1 ' ' 1



*Figure 9: Each plot displays the fitted function and pointwise standard errors for GAM using smoothening spline model.*

The plots in Figure 8 and Figure 9 are similar except the functions used in figure 9 is smoothening splines.

## 9.3 GAM using Local Regression

We can create general additive models using any combination of spline functions, local regression and polynomial regression for the predictor variables.

Let us predict mpg using combination of local regression and smoothening splines to fit the model.

gam.lo=gam(mpg~s(acceleration,3)+s(horsepower,3)+lo(displacement,span=0.7)+lo(year,span=0.7),data=Auto)

lo() - function to indicate a local regression term in a formula argument to gam

lo(x, span=0.5, degree=1)

x - the univariate predictor

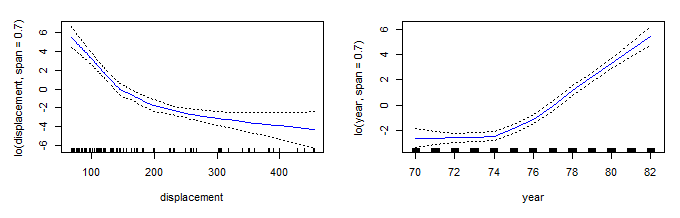
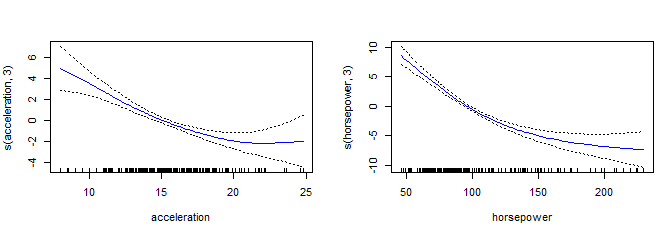
span - the number of observations in a neighbourhood. This is the smoothing parameter for a loess fit. If specified, the full argument name span must be written.

degree - the degree of local polynomial to be fit; currently restricted to be 1 or 2. If specified, the full argument name degree must be written

Smoothening spline is used to fit acceleration and horsepower, displacement and year is fitted using local regression with span = 0.7.

*Results of smoothening spline and local regression model to fit on Auto dataset with mpg as predicting variable.*

gamoutput(gam.lo)  
## Anova for Parametric Effects  
## Df Sum Sq Mean Sq F value Pr(>F)   
## s(acceleration, 3) 1.00 4738.3 4738.3 588.514 < 2.2e-16 \*\*\*  
## s(horsepower, 3) 1.00 11653.9 11653.9 1447.466 < 2.2e-16 \*\*\*  
## lo(displacement, span = 0.7) 1.00 525.9 525.9 65.314 8.615e-15 \*\*\*  
## lo(year, span = 0.7) 1.00 2271.0 2271.0 282.074 < 2.2e-16 \*\*\*  
## Residuals 379.72 3057.2 8.1   
## Anova for Nonparametric Effects  
## Npar Df Npar F Pr(F)   
## (Intercept)   
## s(acceleration, 3) 2.0 4.493 0.01179 \*   
## s(horsepower, 3) 2.0 40.094 2.220e-16 \*\*\*  
## lo(displacement, span = 0.7) 1.8 36.256 3.553e-14 \*\*\*  
## lo(year, span = 0.7) 1.4 35.888 8.565e-12 \*\*\*



*Figure 10: Each plot displays the fitted function and pointwise standard errors for GAM using combination of smoothening spline and local regression.*

The plots in Figure 8 and Figure 10 are similar except the functions used in figure 9 is local regression and smoothening splines.

Let us perform ANOVA Test to determine which of the four multiple regression models is best in predicting Miles per gallon(mpg)

anova(baseline,gam1,gam.s1,gam.lo,test="F")

## Analysis of Variance Table  
## Model 1: mpg ~ cylinders + displacement + horsepower + weight + acceleration +   
## year + origin + name  
## Model 2: mpg ~ (ns(horsepower, df = 4) + ns(acceleration, df = 3) + ns(displacement,   
## df = 4) + ns(year, df = 4) + ns(origin))  
## Model 3: mpg ~ s(acceleration, 3) + s(horsepower, df = 3) + s(displacement,   
## df = 3) + s(year, df = 3)  
## Model 4: mpg ~ s(acceleration, 3) + s(horsepower, 3) + lo(displacement,   
## span = 0.7) + lo(year, span = 0.7)  
## Res.Df RSS Df Sum of Sq F Pr(>F)   
## 1 85.00 438.73   
## 2 375.00 2924.87 -290.00000 -2486.14 1.6609 0.003119 \*\*  
## 3 379.00 3024.10 -4.00017 -99.23 4.8058 0.001525 \*\*  
## 4 379.72 3057.19 -0.71628 -33.09 8.9503 0.007801 \*\*  
## ---  
## Signif. codes: 0 '\*\*\*' 0.001 '\*\*' 0.01 '\*' 0.05 '.' 0.1 ' ' 1

The baseline linear regression model is not the best model as we could observe non-linear relationship between response and predictor variables in Auto dataset.

GAM built using smoothening splines outperforms other two GAM built using natural splines, and combination of smoothening splines and local regression.

But all the three GAMs produce close result as their p-values are significant (α = 0.05).

Further model can be tuned to get better results by using different combination of splines, local and polynomial regression. Use cross-validation techniques to choose span value in local regression and degrees of freedom (or knots) in spline regression

# References

Text: An Introduction to Statistical Learning with Applications in R, [Gareth James](http://www-bcf.usc.edu/~gareth), [Daniela Witten](http://www.biostat.washington.edu/~dwitten/), [Trevor Hastie](http://www.stanford.edu/~hastie/) and [Robert Tibshirani](http://www-stat.stanford.edu/~tibs/)

<https://cran.r-project.org/web/packages/ISLR/index.html>

<https://cran.r-project.org/web/packages/gam/index.html>

<http://www.stat.ucla.edu/~cocteau/stat204/readings/cleveland.pdf>

# 11. Appendix

Below is a function that we created to predict the outcomes of various nonlinear models, calculate 95% confidence intervals, and then plot the fit and confidence intervals

allparam<-function(model)  
{  
 Disp =range(Auto$displacement)  
 Dispgrid=seq(from=Disp [1],to=Disp [2])  
 predictions<-predict(model, newdata=list(displacement=Dispgrid), se=T)  
 standarderror=cbind(predictions$fit +2\* predictions$se.fit ,predictions$fit -2\* predictions$se.fit)  
 plot(Auto$displacement ,Auto$mpg ,xlim=Disp ,cex =.5,col=" black ")  
 lines(Dispgrid ,predictions$fit,lwd=2,col="blue")  
 matlines(Dispgrid ,standarderror ,lwd=1, col=" blue",lty=3)  
}

Below is the function to display the plot model function fitting, standard errors and model summary for the GAMs built

gamoutput<-function(gammodel)  
{  
 library(gam)  
 par(mfrow=c(2,3))  
 plot.gam(gammodel,se=TRUE,col="blue",cex=2)#plot method for GAM objects  
 return(summary(gammodel))  
}