**Smoothing Splines:**

We aim to minimize the Loss + Penalty function where represents the Loss and is a penalty. The parameter is used to control the smoothness of the function . Since every single data point is a knot in a smoothing spline, then when we will be able to minimize the Loss + Penalty function down to 0, by severely overfitting the data and hitting every single point. Alternatively, as , the penalty part of the function will overwhelm the loss and the smoothing spline will become a linear least squares line. The parameter is used to control the bias-variance tradeoff.

**R Code**

Package: 'splines'

smooth.spline(x, y, df, cv=T/F)

cv = False by default

False- 'generalized' cross-validation (GCV); True- ordinary leave-one-out

all.knots = True by default; could specify number of knots

**Choosing :**

As increases then the effective degrees of freedom , decreases to 2. Conversely, as decreases then can increase up to *n*. The effective degrees of freedom measure the amount of flexibility in the model rather than the number of free parameters. They are defined as where represents the diagonal of the matrix which is defined as .

LOOCV is a very efficient way to determine the best value of to use. The formula

computes the LOOCV residual sum of squares for any given .

**Local Regression:**

Local Regression is similar to the K-Nearest Neighbors method for classification. Except now instead of using the points nearby for classification, we are using them to build a regression model. We use the span, , where is the number of points that will be in the span and is the total number of points, to control the bias-variance trade-off in the model. A higher span will lead to less variance in the predictions because more data is being used.

**The Algorithm:**

1. Gather the span *s = k/n* of points closest to the chosen point
2. Weight the points using so that the points nearest to have the largest weights and the points further away have lower, almost 0, weights. All points not in *s* have a weight of 0.
3. Fit a weighted least squares regression by minimizing
4. The fitted value of the point is

**R Code:**

model <- loess(y ~ x, data, span = s) # where s is a value between 0 and 1

model <- lowess(y ~ x, data, span = s)

mse<- mean(model$fitted - data$y)^2