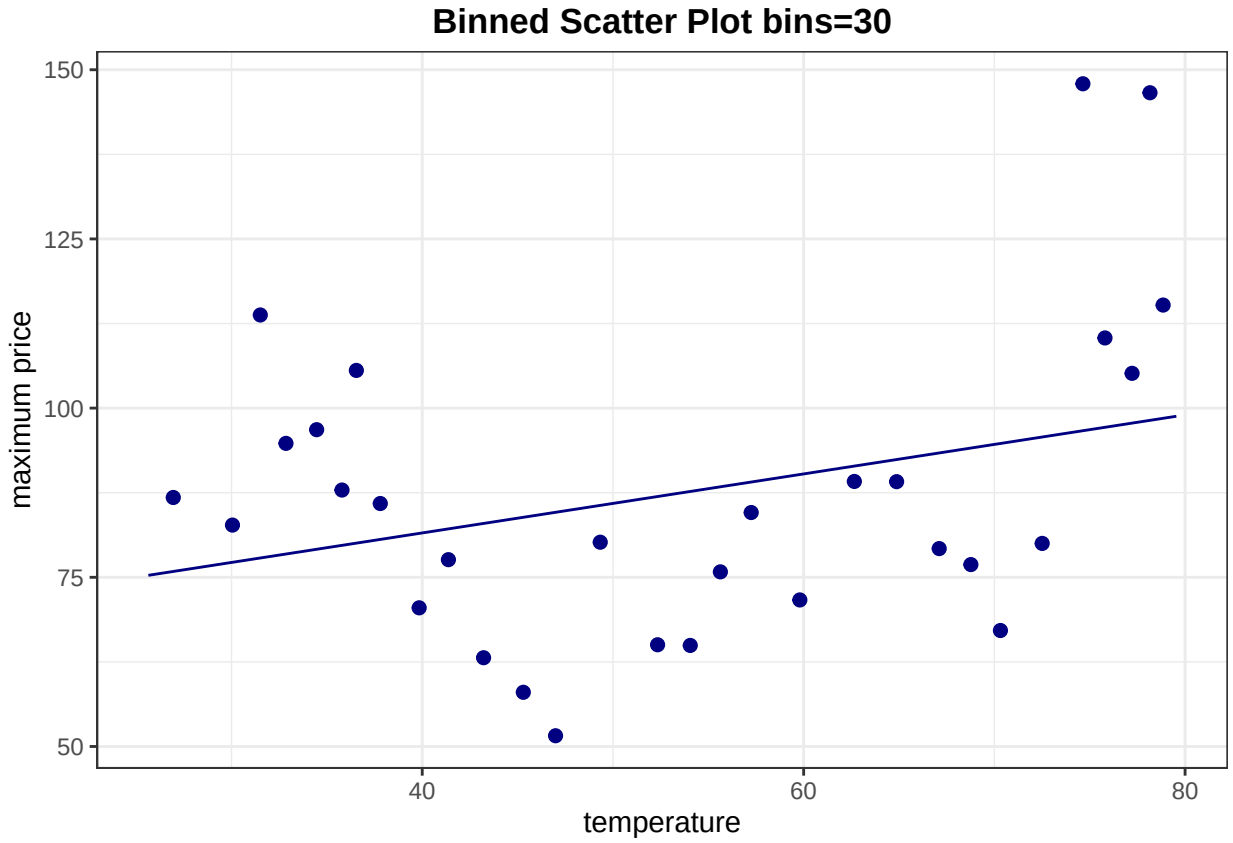
Empirical Methods for Applied Micro

Problem Set 2

Alberto Cappello

2/9/2021

Binned Scatterplot



The simple linear regression without higher order terms is not able to capture the evident U-shaped relationship between temperature and maximum price. In the next section we will test the performance of polynomial regression including terms up the the order 10 and select the optimal degree based on cross-validation.

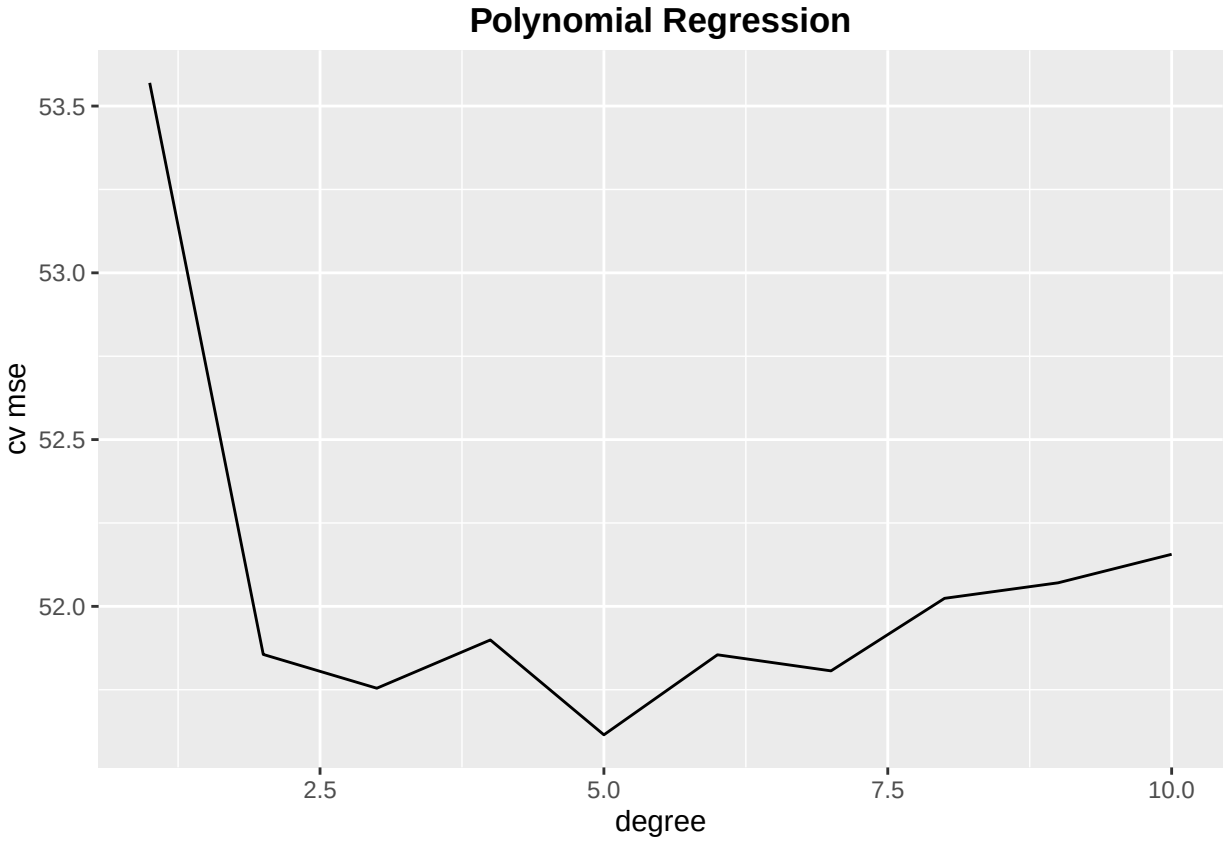
Polynomial Regression

As we can deduce from the table below we model that perform best in sample is the polynomial regression with degree 10 cause it is the one that minimizes the Residual Sum of Squares. This was expected since more complex models (higher degree) are able to fit the data better than less complex models (lower degree). However, more complex models allow to obtain low in sample bias at the expenses of large variance and

	Res.Df	RSS	Df	Sum of Sq	F	Pr(>F)
1	360	1151831.14				
2	359	1074482.49	1	77348.65	25.92	0.0000
3	358	1069812.47	1	4670.02	1.57	0.2118
4	357	1067825.95	1	1986.51	0.67	0.4151
5	356	1058421.54	1	9404.41	3.15	0.0767
6	355	1055415.63	1	3005.91	1.01	0.3162
7	354	1050341.67	1	5073.96	1.70	0.1931
8	353	1049634.46	1	707.21	0.24	0.6267
9	352	1048883.61	1	750.85	0.25	0.6162
10	351	1047367.22	1	1516.39	0.51	0.4764

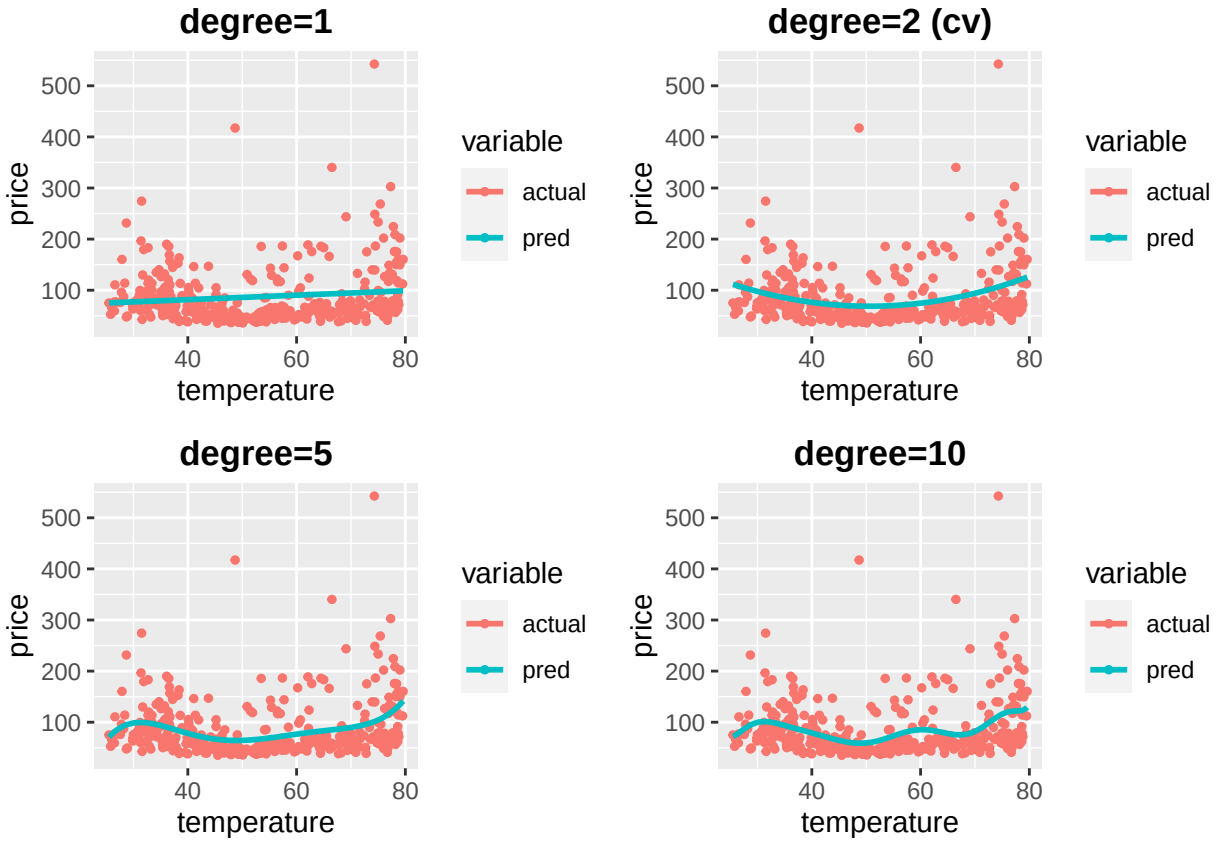
therefore are likely to perform poorly on out of sample data. Hence, in the next section we will select the degree of the polynomial regression that best balance the bias-variance trade off via cross validation.

Cross Validation



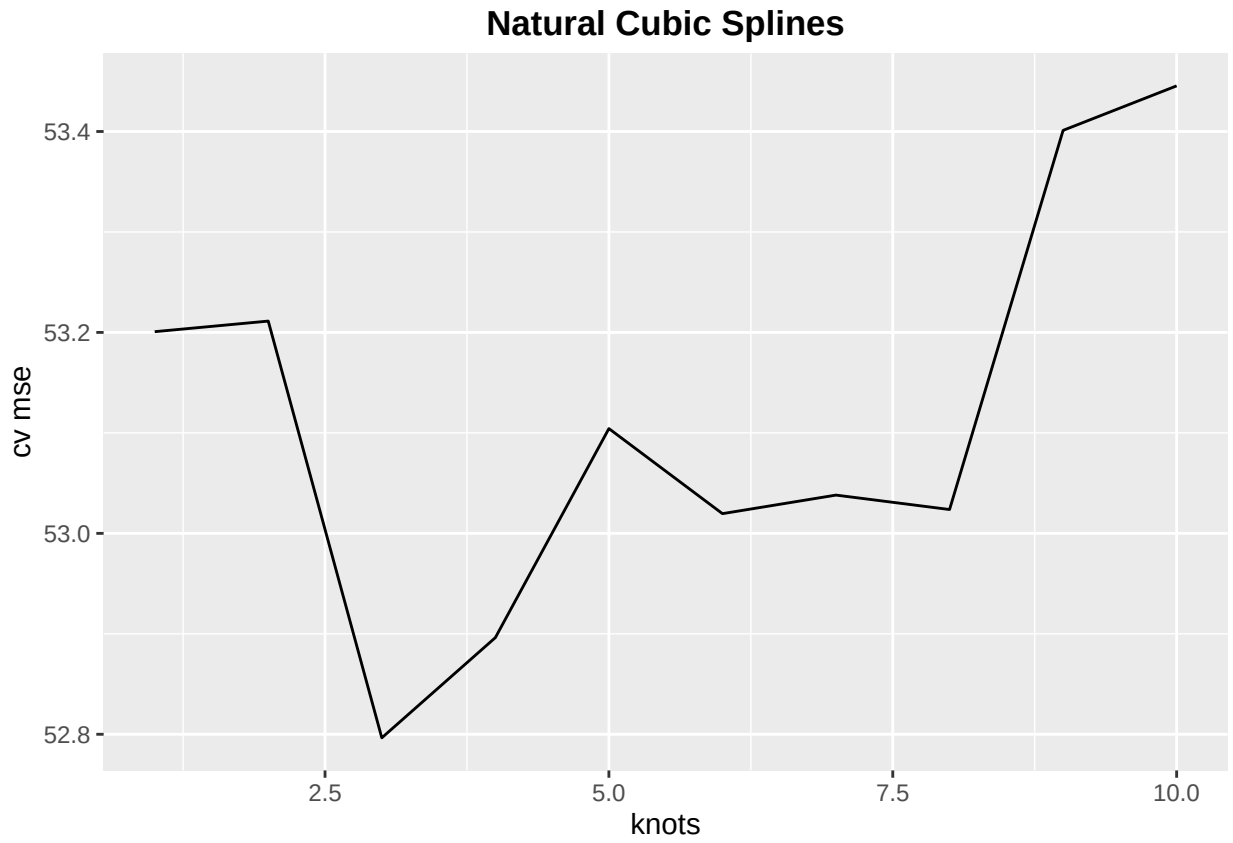
The cross validated mean squared error (CVMSE) is minimized at degree 5. However, the difference between the estimated CVMSE between degree 2 and degree 5 is considerably small. Therefore we have reasons to choose the simpler model (degree 2) over the more complex model (degree 5).

Predicted vs. Actual Prices



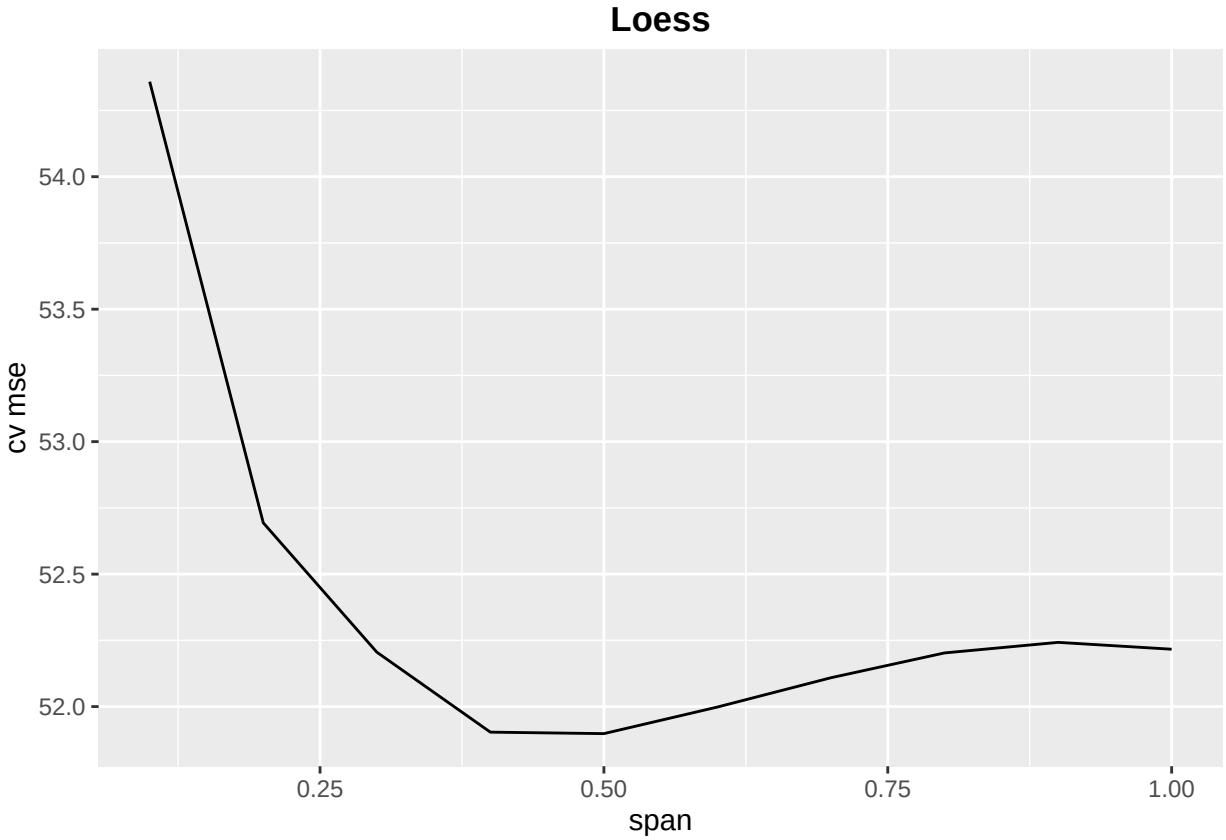
The plot above shows the predicted versus actual maximum prices for the polynomial regression models with degree 1,2,5 and 10. As we can notice the model selected after cross validation (degree 2) is capturing the U-shaped relationship mentioned above, whereas the model that minimizes the cross validated MSE is not adding anything apart from being more flexible at fitting the data at the boundaries of temperature levels.

Natural Cubic Splines



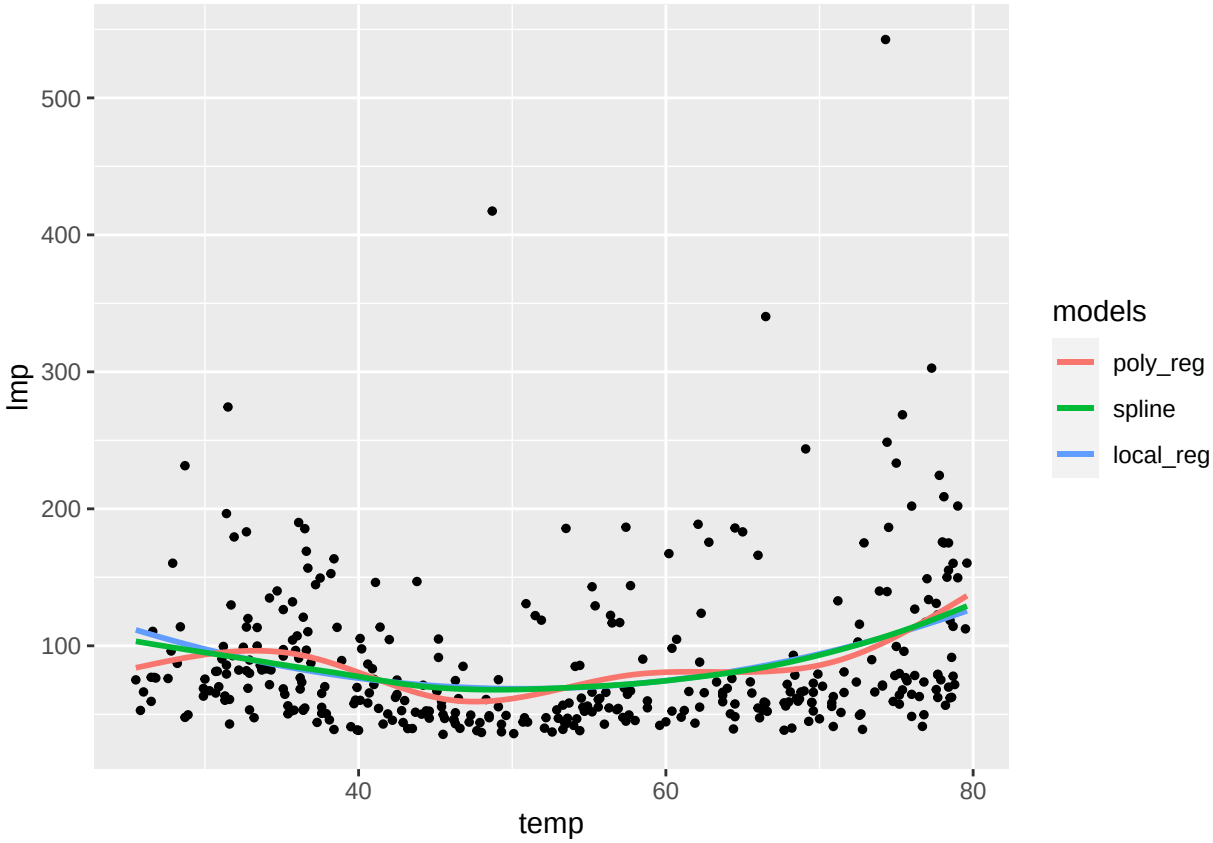
The cross validated MSE is unambiguously minimized at 3 knots. Therefore, the 4 regions where a polynomial of degree 3 is fitted corresponds to the for percentiles of the sample.

LOESS



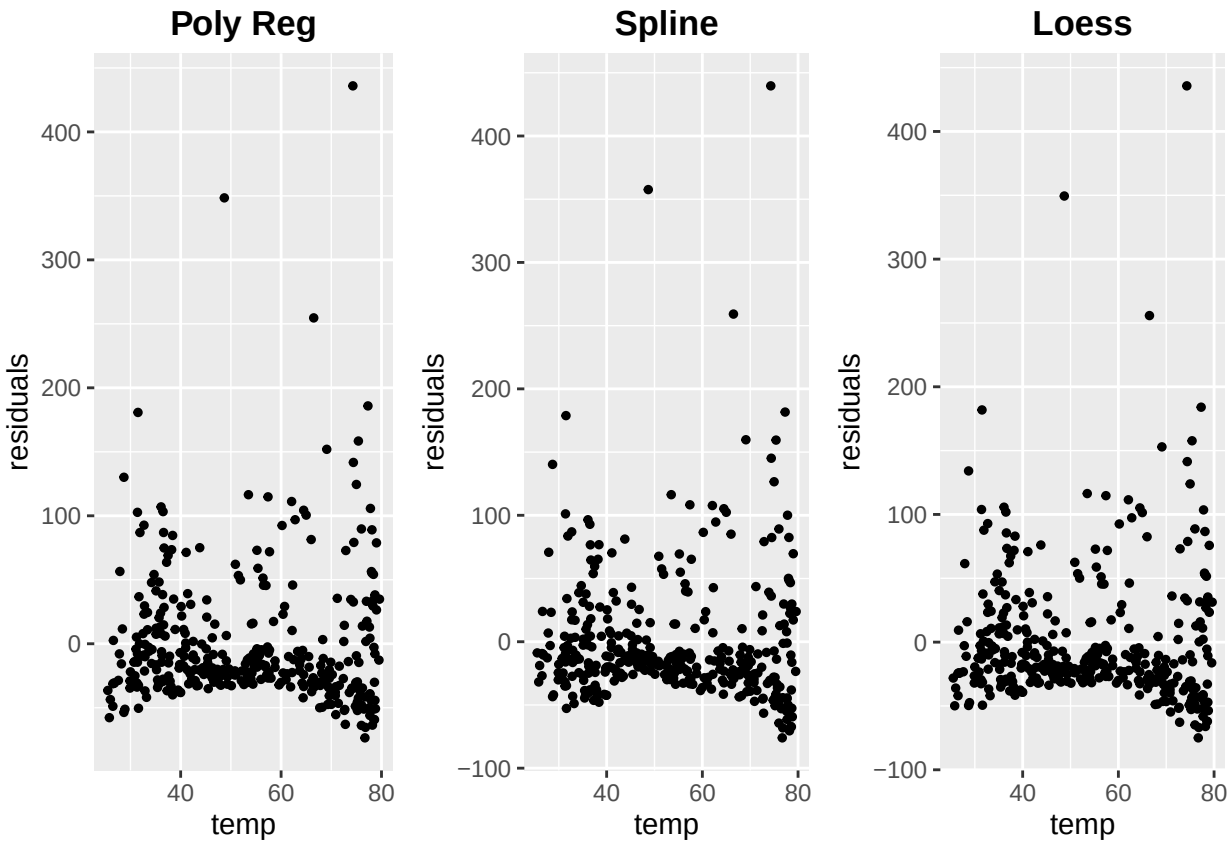
The cross validated MSE is unambiguously minimized at span 0.4, but since the function is flat round that point we will choose 0.75 as the optimal value of the span.

Compare Predictions



The plot above represents the predicted relationship between temperature and maximum local marginal price. The simpler model is the polynomial regression and is able to capture the U-shaped relationship that we see in the data, while the other two more complex model (natural cubic spline with 3 knots and local regression with span=0.75) are also capturing some local non-linear relationship the we can observe in each of the 4 percentiles of the sample.

Compare Residuals



The residual plots look very similar across the three models and they fit all the parts of the distribution equally well. Hence, we would not be able to tell which model perform best based on this plot. Also the cross validated MSE values for the three models are quite similar.

Appendix: Cross validation function

```
cross_val = function(data,model,df,K){

  #df = degrees of freedom (eg degrees of the polynomial or knots)
  #set.seed(1)
  data.shuffled <- data[sample(nrow(data)),]
  folds <- cut(seq(1,nrow(data.shuffled)),breaks=K,labels=FALSE)
  #Creating empty object to hold fit information
  rmse = matrix(NA,nrow=K,ncol=length(df))
  for(i in 1:K){
    #Segement data by fold using the which() function
    testIndexes <- which(folds==i,arr.ind=TRUE)
    testData <- data.shuffled[testIndexes, ]
    trainData <- data.shuffled[-testIndexes, ]
    #Use the test and train data partitions
    #Model fitting and evaluation
    for (j in 1:length(df)){
      d = df[[j]]
```

```
      #training regression model on training folds
      if(model == "polyreg"){
        fit.train = lm(lmp ~ poly(temp,d), data = trainData)
      }
      if(model == "spline"){
        fit.train = lm(lmp ~ ns(temp, df = d+2), data = trainData)
      }
      if(model == "loess"){
        fit.train = loess(lmp ~ temp, span = d, data = trainData)
      }
      #evaluating fit on the test fold
      fit.test = predict(fit.train, newdata=testData)
      rmse[i,j] = sqrt(mean(na.omit(fit.test - testData$lmp)^2))
    }
  }
  return(rmse)
}
```