Polynomial Regression and Splines

Mirnes Salkic

12/21/2017

Polynomial Regression

Sometimes our data does not follow a linear trend and using models that assume linearity might not be useful and effective. After all, the scatterplot and the residual plot of the data will signal us that the data does not follow a linear trend. What could we do then? As one might expect, we might use a polynomial function of some degree to try to fit our data and that is the task of polynomial regression. In other words, we can say that polynomial regression is used when the relationship between the explanatory and response variable is nonlinear. Although polynomial regression allows for a nonlinear relationship between predictor and response variables, it is still considered a special case of multiple linear regression because it is linear in the parameters (coefficients). We can say that polynomial regression is a linear model with predictors $x_i, x_i^2, x_i^3 ... x_i^d$.

$$y_i = \beta_0 + \beta_1 x_i + \beta_2 x_i^2 + ... + \beta_d x_i^d$$

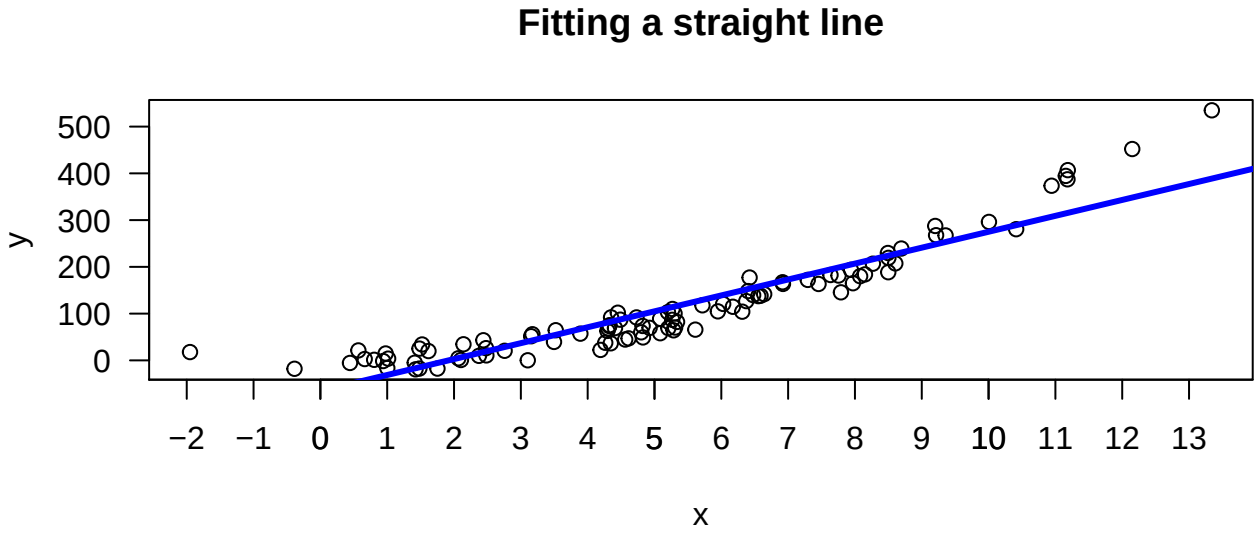
This means that polynomial regression inherits many of the properties of linear regression. For instance, the coefficients can be estimated using least squares linear regression (a line that minimizes the sum of squared residuals). To illustrate how the concept may work in practice we might consider an example. Let's consider an example where we have a y as the response variable and x as the predictor or explanatory variable. One might be tempted to use simple linear regression to model the data. This is what can be seen on the plot. (Data source: Udemy)

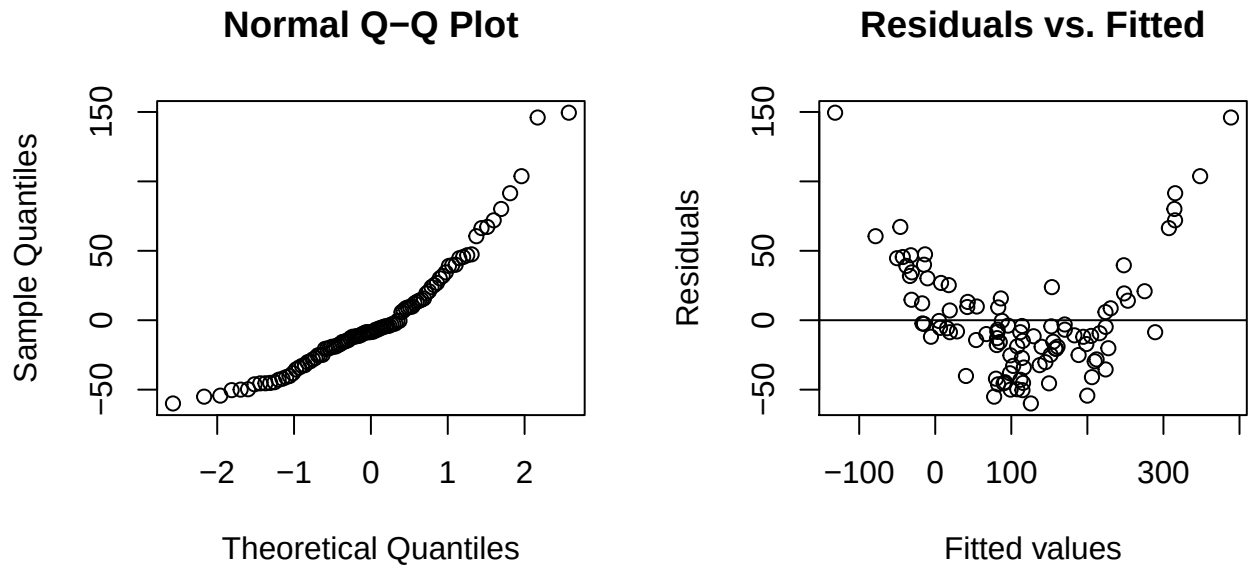
```
#fitting a linear model
lin_model <- lm(formula = dat$y~dat$x)
```

The line that best fits the data is given by the following equation:

$$\hat{y} = -65.273 * 34.035 * x$$

The adjusted R-squared is 0.8704. This means that about 87% of the variability in the response variable can be explained by the explanatory variable using a linear equation.



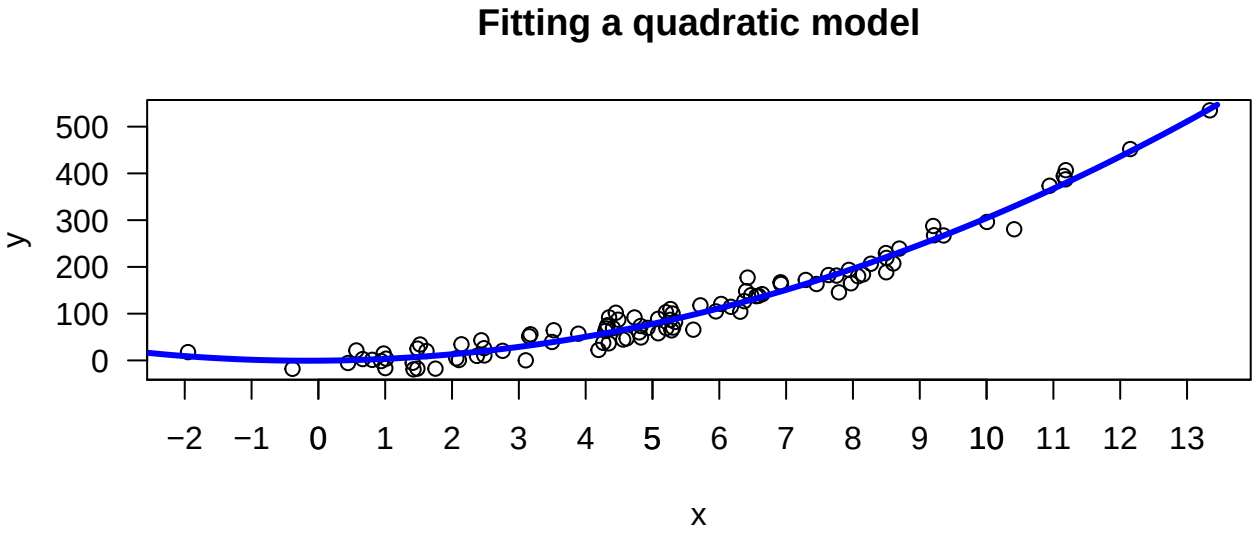


However, by observing the "Residuals vs. Fitted" graph, we can notice an obvious curvature instead of uniform randomness; this means that the data is nonlinear. The nonlinear trend is also evident from the normal QQ plot - the ends of the line are curved. We can try to fit a quadratic function and see if the fit or explained variability will improve relative to the linear function.

```
quad_model <- lm(dat$y~poly(dat$x, 2, raw=TRUE))
```

$$\hat{y} = 2.9522 * x^2 + 0.9719 * x - 0.5685$$

The adjusted R^2 is 0.9714. This is an incredible improvement over the linear fit.



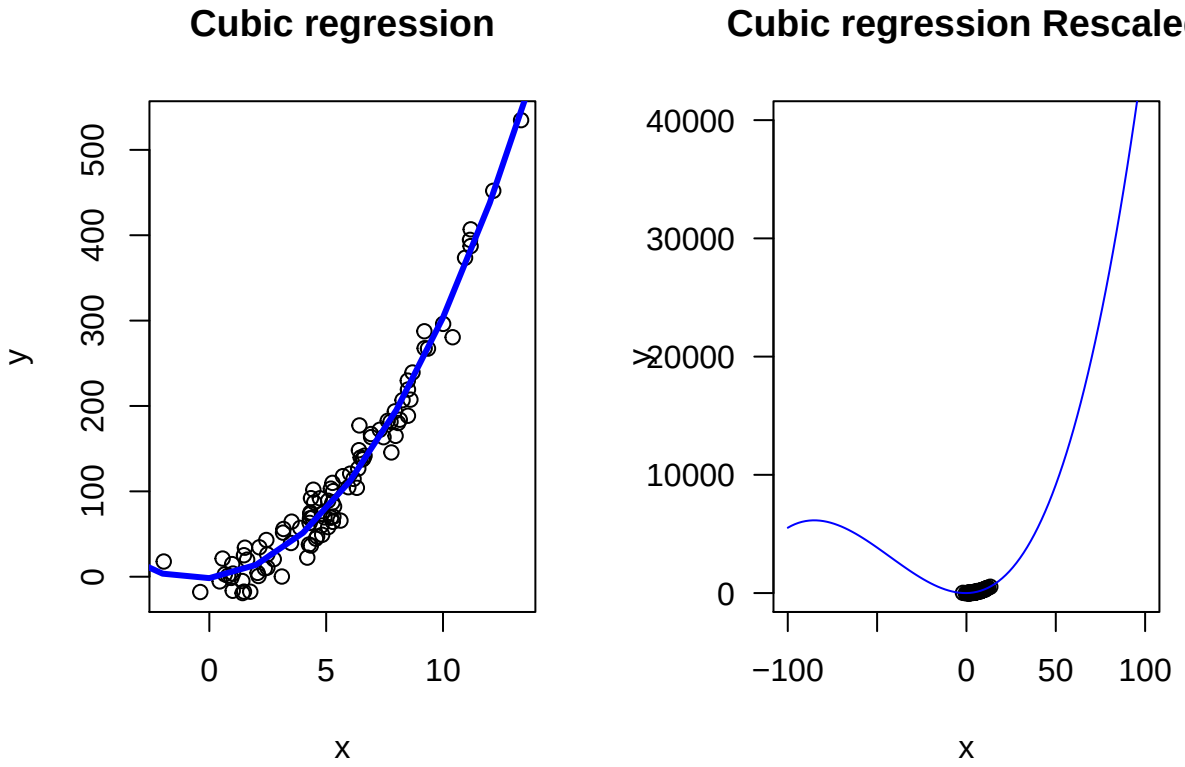
The quadratic polynomial seems to fit the data much better relative to the linear model. Let's see what happens if we fit a cubic model.

Fitting a cubic equation

```
cubic_model <- lm(dat$y~poly(dat$x, 3, raw=TRUE))
```

$$\hat{y} = 0.02024 * x^3 + 2.60142 * x^2 + 2.49528 * x - 1.67268$$

The adjusted R^2 is 0.9712 is similar to the quadratic regression model. However, the sum of squared residuals decreased from 34,582 to 34,484. Overall, there isn't a statistically significant difference between the quality of the two models given the p-value of 0.6.



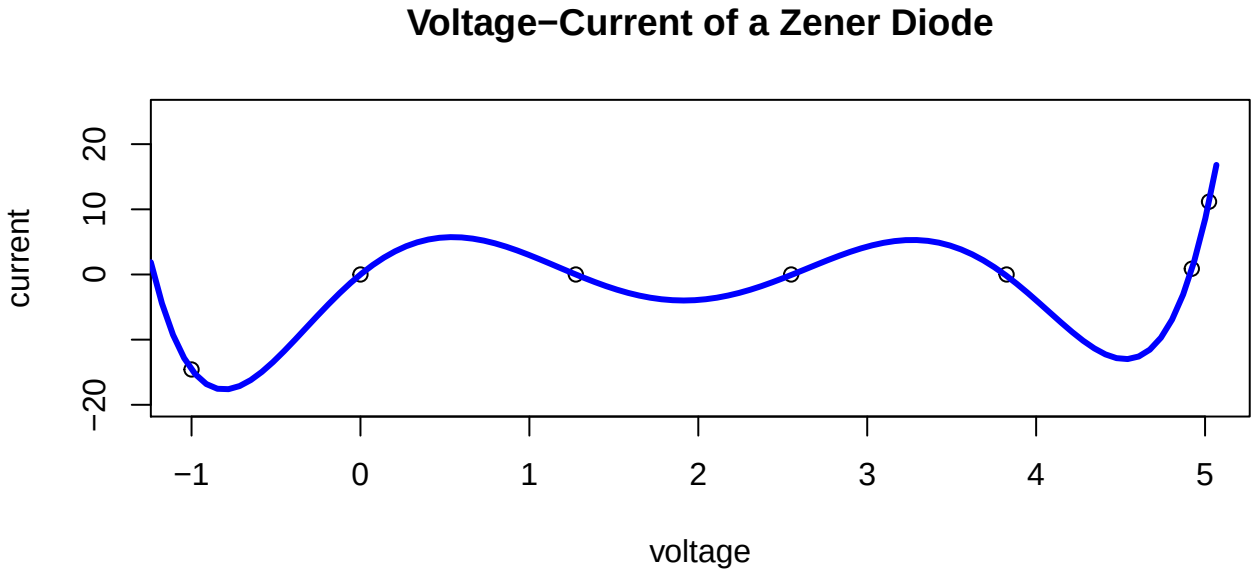
We have seen that both the quadratic and cubic regression fitted the data very well. We could surely interpolate and get fairly accurate predictions on unseen data. The problem that arises if we want to extrapolate. As we can see in the "Cubic Regression Rescaled" graph the curve shoots upwards and downwards at the ends meaning it is heavily influenced by the end data points. Therefore, doing extrapolation would not be reasonable.

Interpolation seems to work fairly well for this data set where we used lower degree polynomials. What if we had a data set that cannot be fitted well using quadratic or cubic regression? One could increase the order of the polynomial until a function that reasonably fits the data is found. But, there are downsides for that as will be discussed in the following example. A better solutions exists...

Problems with higher order polynomials

Let's consider a fairly small data set of only seven points of a voltage-current characteristic of a zener diode (Data source: Pelinovsky). The best polynomial that fits the data is of the sixth degree. The 6^{th} degree polynomial fully fits the data and thus has a sum of squared residuals of 0.

```
#fitting a polynomial degree 6
volt_model <- lm(current~poly(voltage, 6, raw=TRUE), data=dset)
```



While the 6^{th} degree polynomial fully captures the data, there are problems with it. The downside of the model is that it is fairly complicated and difficult to interpret compared to linear, quadratic or even cubic models. For instance, a linear model could be interpreted based on the slope which represents a constant rate of change while in a quadratic model the rate of change increases or decreases at a constant rate (Schwartz). We could ask the question what does the 6^th degree polynomial say about the change in the current if the voltage is between -1 and 0 and how does it compare with the the change in voltage between 0 and 1?

The equation is given below:

$$\hat{current} = 1.128 * 10^{-12} + 21.21 * v - 17.47 * v^2 - 9.212 * v^3 + 11.38 * v^4 - 3.219 * v^5 + 0.284 * v^6$$

Note that v stands for voltage.

As one might expect it is impossible to come up with an answer that is meaningful. The second problem associated with higher order polynomials is known as the **polynomial wiggle**. In simple terms, the phenomenon of the polynomial wiggle refers to the large number of oscillations or swings the function makes to fit the data. We know that polynomial functions (of degree 2 and greater) have turning points (read as extreme points: maxima, minima) depending on the order of the polynomial. A quadratic function (polynomial of order 2) has one extreme point, a cubic function (order 3) has 2. So the number of extreme points a function has is dependent on its order and can be calculated by subtracting 1 from the order of the polynomial. The polynomial from our example, as can be seen from the graph, has to make 5 turns to fit the data. Why is the polynomial wiggle a problem? If the data is not polynomial in its very nature, the polynomial wiggle would lead to large errors when interpolating. If we look at the graph carefully, we will notice 4 points that have the current equal to 0. Is there a reason to believe that the current would be positive when the voltage is between 0 and 1, and negative when the voltage 1 and about 2.5? How about when the voltage is between 4 and 5? Probably not. The pronounced problem with interpolation makes higher order polynomials unattractive to use to fit models. As discussed earlier extrapolation should also be avoided while using polynomials, or at least one should be extremely cautious when doing it (e.g. when one is ensured that the data is polynomial in nature) (Shwartz; London&Wright).

Splines

A solution to using higher order polynomials to fit the data is given by splines. A spline, as opposed to polynomial regression, does not try to construct a curve that goes over the whole interval of the data, it rather divides the data into smaller pieces and fits simpler models to each of the sub-intervals. This allows us to use simpler functions to fit smaller ranges of data using linear, quadratic or cubic polynomials (Schwartz). Splines

make it possible to interpolate in a more reasonable fashion. Additionally, we can compare segments of data and interpret the rate of change at different intervals of the data. If we recall from the earlier discussion this was not possible using higher order polynomials. I will focus on the previous two examples to illustrate this concept. Before that I will explain how to make splines in R.

Splines and R

There are a few different functions that could be used to construct splines in R. I will use the *bs* (basic spline) function from the *splines* package. The *bs* function takes several inputs: *the predictor variable*, *degrees of freedom*, *degree of the piecewise polynomial* (default=3), *knots*, and a few others not so important for our discussion.

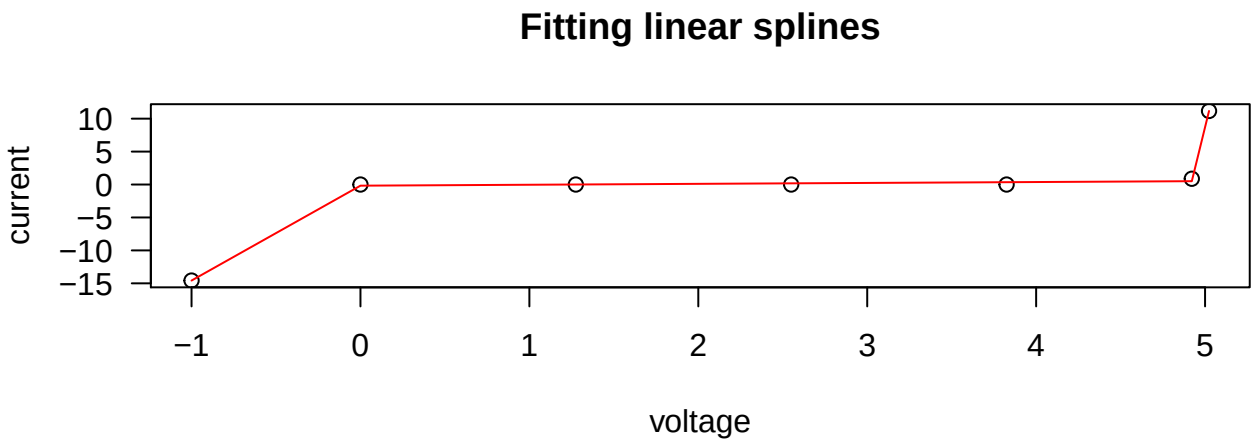
- The degrees of freedom determine how wiggly the curve will be - the higher the degrees of freedom the more wiggly the curve will be as it tries to fit more and more data. We will see this in the example.
- The degree simply refers to the degree of the piece-wise polynomial we are trying to fit.
- Knots refer to the number of connected points of the piece-wise polynomials. For instance, if we are trying to fit two linear lines they will meet at one point which represents one knot. If we know that some change is occurring at some point, we could also explicitly specify the location of the knot. In general, we do not have to use this parameter because changing the degrees of freedom will automatically change the number of knots.

Besides the *bs* spline function, there are *ns* (natural spline), *ps* (p-splines). One could also use the smooth.spline function. They all have different characteristics. For example, the p-spline stands for "penalized B-spline" and it penalizes for over-fitting and ensures smoothness (Wikipedia).

Splines and the voltage data

For this example fitting a few straight lines makes sense; this is why I selected degree=1. Since the current significantly changes when the voltage is 0 and 5, I decided to specify that the knots at those points.

```
library(splines)
lmfit2 <- lm(dset$current~ bs(dset$voltage, degree=1, knots = c(0,5)))
```

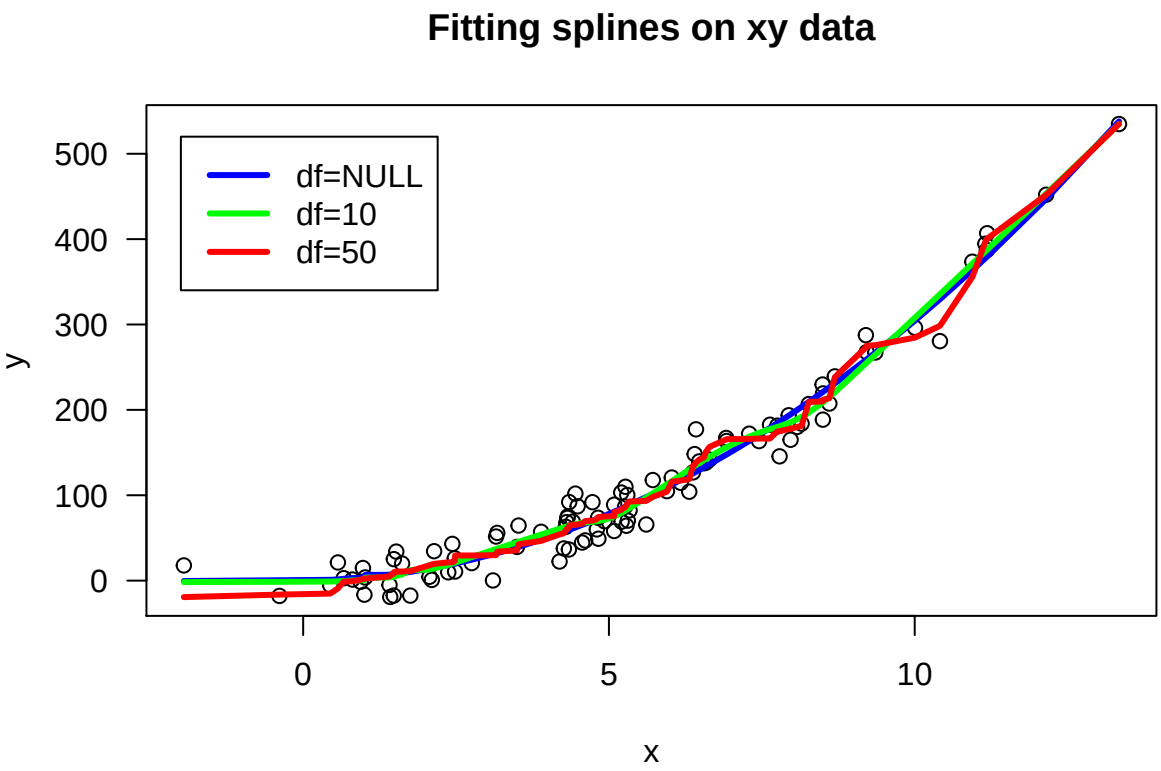


Splines and x-y data

I have chosen to build three different spline models. **The first model** (*spline_xy*) is a spline of degree 2 without specified df; this makes it identical to the quadratic polynomial from earlier. **The second model**, *spline_xy_*10 is set to have 10 degrees of freedom which increases the number of knots to 9 - it allows the

piece-wise polynomials to wiggle and better fit the data. Based on the ANOVA test, the sum of the squared residuals decreased from 34582 - quadratic regression/spline_xy to 31272. **The third model** with df=50, tries to captures even more data, but as we can see it leads to over-fitting. It learns the specific patters, the noise, of the data set and it would not generalize well to unseen data. Therefore, it is important to avoid over-fitting.

```
spline_xy <- lm(dat$y~ bs(dat$x, degree=2))
spline_xy_10 <- lm(dat$y~ bs(dat$x, degree=2, df=10))
spline_xy_50 <- lm(dat$y~ bs(dat$x, degree=2, df=50))
```



As we have seen, using polynomial regression up to a certain degree is reasonable when modeling data and doing interpolation. However using higher degrees polynomials may lead to the phenomenon polynomial wiggle and thus may not be useful for interpolation; these functions are also difficult to interpret. The problem is solved using splines of lower degree piece-wise polynomials. We always have to make sure that our model does not overfit the data, it should rather capture the general trend.

References

[1] James, Gareth. Witten, Daniela. Hastie, Trevor. Tibshirani, Robert. *An Introduction to Statistical Learning with Applications in R*. 2013.
[2] Schwarz, J. Carl. Simon Fraser University. *Intro to spline notes. Intro to spline slides.* http://people.stat.sfu.ca/~cschwarz/Consulting/Trinity/Phase2/TrinityWorkshop/Workshop-material-Simon/Intro_to_splines/
[3] Pelinovsky, Dmitry. McMaster University. http://dmpeli.math.mcmaster.ca/Matlab/Math4Q3/Lecture2-1/Example2-1.html
[4] Wikipedia. https://en.wikipedia.org/wiki/B-spline
[5] Udemy. Polynomial Regression, R, and ggplot. https://www.udemy.com/machlearn2/learn/v4/t/lecture/2973414?start=0
[6] London, Kamala. Wright, Daniel. *Modern Regression Techniques Using R*. 2009.