Correcting non-linearity: polynomial regression, categorical predictors

Stats 203, Winter 2024

Lecture 12 [Last update: February 25, 2024]

1 Introduction

- Refresher: last Wednesday, and the Wednesday before, we discussed violations of the linear model, in the form of heteroskedastic errors, non-Normal errors, nonlinear conditional mean, an outliers. At the start of class today we will finish up on outliers, and talk about unusual observations that do not violate the linear model, i.e. influential points and leverage points.
- For the second part of class, and on this Wednesday, we will discuss how to correct violations. We've already talked about correcting for heteroskedastic errors when we have a (linear) model for the error variance, using **weighted least squares**. Today we will talk about correcting for a nonlinear conditional mean.
- We assume the following regression model: $(X_1, Y_1), \ldots, (X_n, Y_n)$ are independently sampled, and given $\mathbf{X} \in \mathbb{R}^{p \times n}$,

  $$Y_i = m(X_i) + \epsilon_i, \quad \epsilon_i \sim N(0, \sigma^2),$$

  for an unknown function m. That is, all of the assumptions of the linear model are correct except m may not be linear in x.
- Last Wednesday we discussed diagnostics (residual plots, added variable plots) for assessing whether $m(x) := \mathbb{E}[Y_i | X_i = x]$ was truly linear in x.
- If $m(x)$ is not linear then we don't even know what parameters $\hat{\beta}$ is estimating. We will talk about this on Wednesday when we cover the bootstrap.
- The simplest correction for non-linearity is to include more predictors! For example, for $x \in \mathbb{R}$ the function $m(x) = \beta_1 x + \beta_2 x^2$ is not a linear function of x, but it **is** a linear function of $z = (x, x^2)$.
- Today we will talk about one approach for generating more predictors: polynomial regression. On Wednesday we will talk about two additional approaches, involving categorical variables and interaction effects. The good news is that for all of these cases, the underlying mechanics – that is, how we estimate the parameters, construct confidence intervals, and perform hypothesis tests – are more or less the same as OLS.

2 Polynomial regression

- In **polynomial regression** we treat $m(x)$ as being polynomial in x. Let's start with the simple case where the variable $X_i \in \mathbb{R}$ is just a number. We model the conditional mean function as

  $$m(x) = \beta_0 + \sum_{k=1}^{d} \beta_k x^k.$$

This is a **degree-d polynomial** in x. While it is not a linear function of x, it is a linear function of a transformation of the predictors $\varphi(x) = (x, x^2, x^3, \ldots, x^d)$, since

$$m(x) = \beta_0 + \sum_{k=1}^{d} \beta_j (\varphi(x))_k.$$

We know exactly how to estimate the parameters of this model. Let $\mathbf{\Phi} \in \mathbb{R}^{n \times (d+1)}$ be the matrix of transformed predictors, meaning $\mathbf{\Phi}_{i,k+1} = X_i^k$. (As usual, the first column is a vector of all-ones). Our OLS estimate of β is just $(\mathbf{\Phi}^\top \mathbf{\Phi})^{-1} \mathbf{\Phi}^\top Y$. We calculate standard error, Wald intervals, and T- and F-tests in exactly the same way as always, just with $\mathbf{\Phi}$ replacing $\mathbf{X}$.

- Polynomial regression also applies to the multivariate case. Suppose $x \in \mathbb{R}^p$ is a vector and we model

$$m(x) = \beta_0 + \sum_{j=1}^{p} \sum_{k=1}^{d} \beta_{jk} x_j^k.$$

Again, this is not linear in x but it is linear in $\varphi(x) = (x_1, x_1^2, \ldots, x_1^d, x_2, \ldots, x_2^d, \ldots, x_j, \ldots, x_j^d) \in \mathbb{R}^{n \times (dp+1)}$.

This model assumes the same degree polynomial for each variable. Of course, this need not be true.

- **Interpretation.** Interpreting the coefficients in a polynomial regression is tricky even when $p = 1$. Our usual interpretation would be based on the mathematical fact slopes represent differences in output when input is increased by one unit. In this context of polynomial regression, this means,

$$\beta_1 = \mathbb{E}[Y_i | X_i = x + 1, X_i^2 = x^2, \ldots, X_i^k = x^k] - \mathbb{E}[Y_i | X_i = x, X_i^2 = x^2, \ldots, X_i^k = x^k].$$

But obviously the first conditional expectation above makes no sense since, in general, how can $X_i = x+1$ but $X_i^2 = x^2$?

If we work hard, we can come up with ways of interpreting the parameters β_j that technically make sense. But they will be so complicated (meaning, they will depend on complicated polynomial functions of x) that they are basically impossible to interpret. Of course this is even more true when $x \in \mathbb{R}^p$.

Generally speaking, this means the only thing worth interpreting in polynomial regression is the conditional mean function $m(x)$, and the predictions $\hat{m}(x)$.

- **Smoothness.** Obviously there are (infinitely) many ways to transform x besides taking a polynomial, and the preferred transformation will depend on the (true but unknown) functional form of $m(x)$. Polynomials are a sensible choice because they are **smooth** functions. Informally, this means small changes in x lead to small changes in $\varphi(x)$. Formally, it means they are infinitely many times differentiable, with all but their first-d derivatives being equal to 0.

For this reason, polynomial regression is most useful when we have a reason to believe that $m(x)$ is a smooth function of x: loosely, $m(x') \approx m(x)$ when $x' \approx x$. This is most often the case when x is a temporal (hour/day/week/year) or spatial (longitude, latitude, height) variable.

- **Picking the degree.** The most subtle part of polynomial regression is choosing d. There are a few ways to do this:
  - We might have side information telling us that the true conditional mean is (approximately or exactly) a degree d polynomial. For instance, Taylor's theorem tells us an upper bound on how much error we would incur if we replaced the true m (whatever it is) by a polynomial in x.
  - We can look at residual plots to determine whether m is truly linear in a given polynomial.
  - In a few lectures, we will discuss general methods for variable selection and model selection. These methods can be applied to choose the degree of the polynomial.

Finally, we can run a partial F-test to determine whether the model with a degree-d polynomial is sufficient, or whether a degree-$(d+1)$ polynomial need be included.

- **Overfitting.** Notice that as we pick a higher and higher degree polynomial, the assumption that m is linear in $\varphi(x)$ becomes more and more realistic. Of course, we can't take $d = \infty$ since then we would have more predictors than observations, our predictor matrix would be perfectly multicollinear, and our OLS estimates would not be well-defined. So why don't we take the degree to be as large as possible, i.e. $d = n - 1$?

  The answer is that the resulting estimator $\hat{m}(x)$ will be highly overfit to the data. For example, in the extreme case of $d = n - 1$, we would perfectly fit the data, and $\hat{m}(X_i) = Y_i$. But this is undesirable since Y_i is itself a bad estimate of $m(X_i)$ due to the errors.

  What we are seeing here is an example of what is called the **bias-variance tradeoff**. If we pick d too small, the model will be wrong and our estimate for $m(x)$ will be biased. If we pick d too large, the model will be right but our estimates for $m(x)$ will be highly variable. The methods for variable selection I mentioned above, applied in this context, are designed to optimally trade-off bias against variance.

- **Other transformations.** Transformations of the predictors besides polynomials can also be useful. One example is the log transform, where $\varphi(x) = \log_{10}(x)$. This is sometimes useful in financial applications where differences in daily returns are thought to be (roughly) linear on the log scale. Another example is the stepfunction $\varphi(x) = 1$ if $x > c$ and 0 otherwise. Stepfunctions are poor at modeling smooth functions m, but better than polynomials at picking up discontinuities in the conditional mean.

  We could also consider transforming the response. While this is sometimes done, it is a more delicate matter than transforming the predictors because transforming the response affects both the mean and error part of the regression. Indeed, the most classical reason for transforming the response is to address non-constant variance in the errors.[1]

3 Categorical predictors

We will get to this on Wednesday. I will update the lecture notes then.

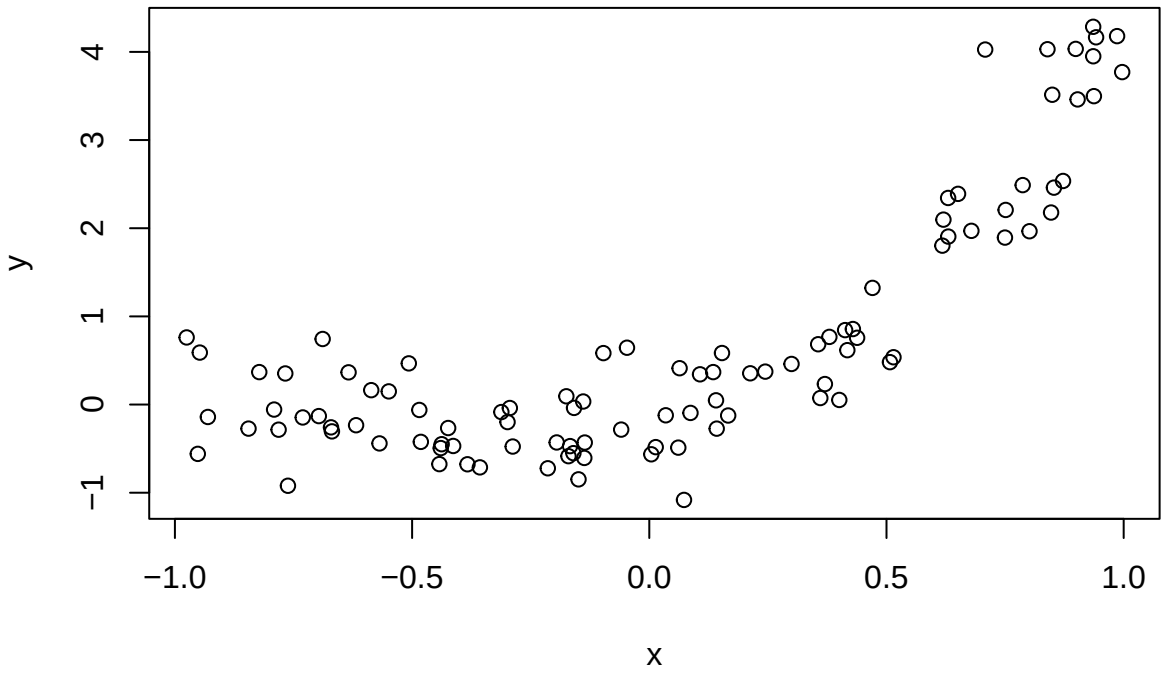
[1] We will not talk about this in class since we don't have the time and it's a bit of an old-fashioned idea. In you're interested, the keyword to search is **Box-Cox transformation**.

Polynomial regression with simulated data

In this simulation, $n = 100$ samples are drawn from a regression model where the true conditional mean function is a degree-3 polynomial in $x \in \mathbb{R}$.

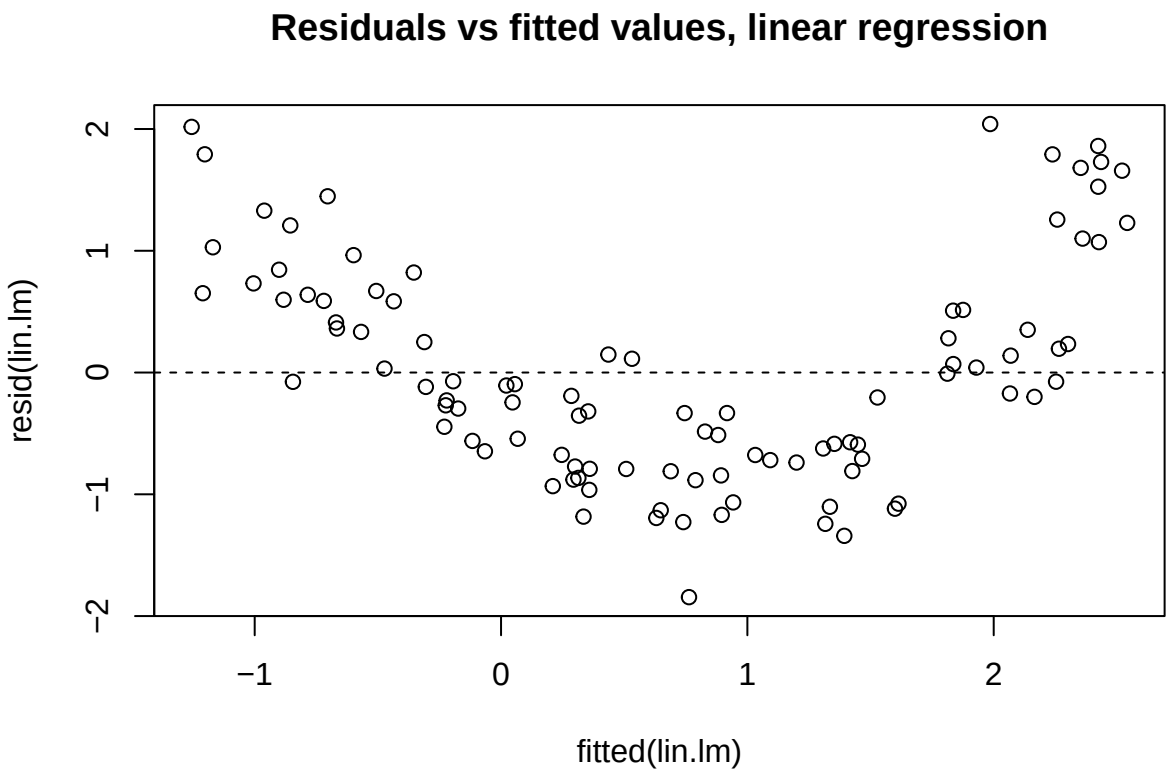
```
x = runif(100,-1,1)
sigma = .5
m = function(x) x + 2*x^2 + x^3
y = m(x) + rnorm(100,sd = sigma)
df.sim = data.frame(y = y, x = x)

plot(x,y)
```



I run a linear regression on the simulated data and plot the fitted values against the residuals.

```
lin.lm = lm(y ~ x, df.sim)
plot(fitted(lin.lm),resid(lin.lm),  main = "Residuals vs fitted values, linear regression")
abline(h = 0,lty = 2)
```



There appears to be a systematic trend in the residuals, which makes sense: $m(x)$ is not linear in x.

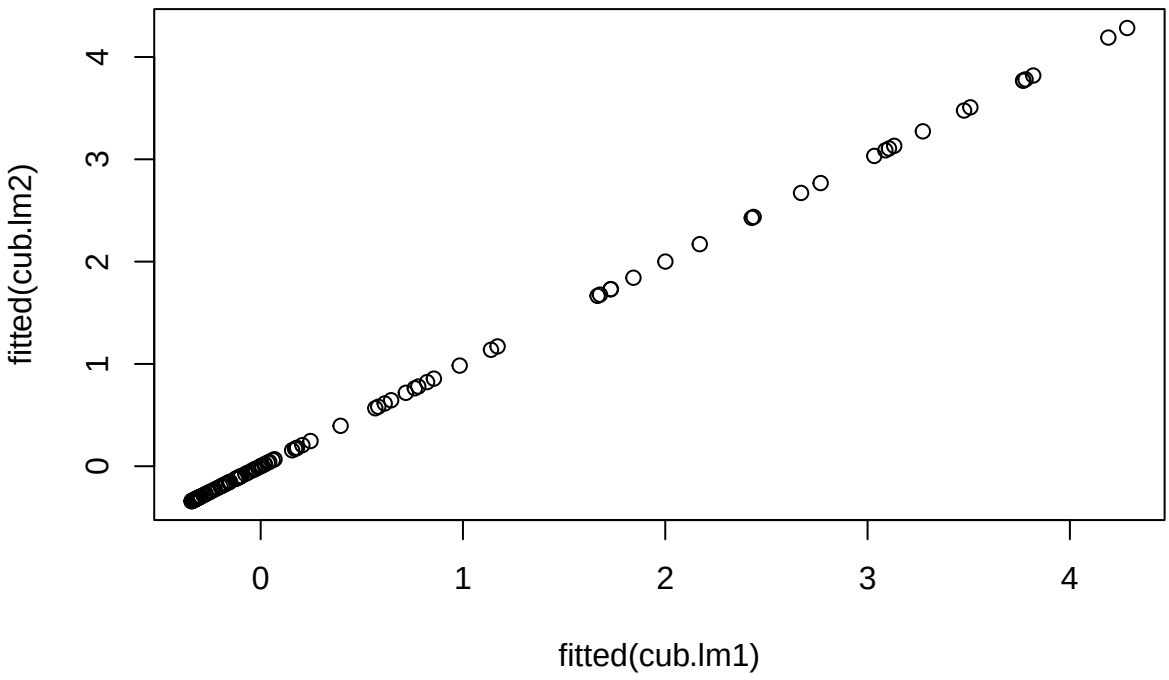
Cubic fit. Since we know the true degree of the polynomial is 3, let's try cubic regression, i.e. fitting a degree-3 polynomial. `R` makes this very simple if you know how to correctly write the formula.

One way to specify this formula is by writing `y ~ x + I(x^2) + I(x^3)`. The operators `I` stand for identity. They distinguish this from the formula `y ~ x + x^2 + x^3` which is **incorrect**: `R`s interpretation of this is completely different, and has to do with interaction effects (we will get to interactions on Wednesday) not polynomials.

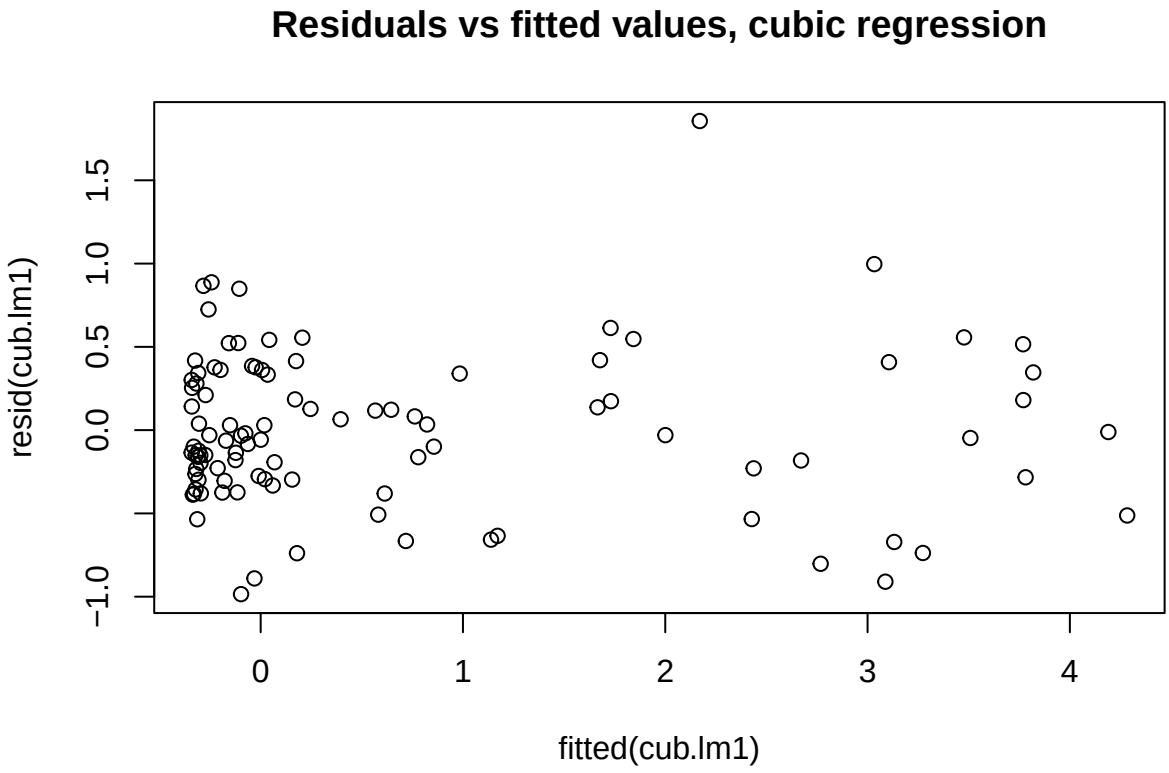
The slightly better and more idiomatic way to write the formula is `y ~ poly(x,d)`. Here `d` stands for the degree of the polynomial you want, so for a cubic `d = 3`. This leads to a much simpler formula (and numerically stabler fit) if the degree of the polynomial is large.

Let's confirm that our two ways of writing the formulas give the same values.

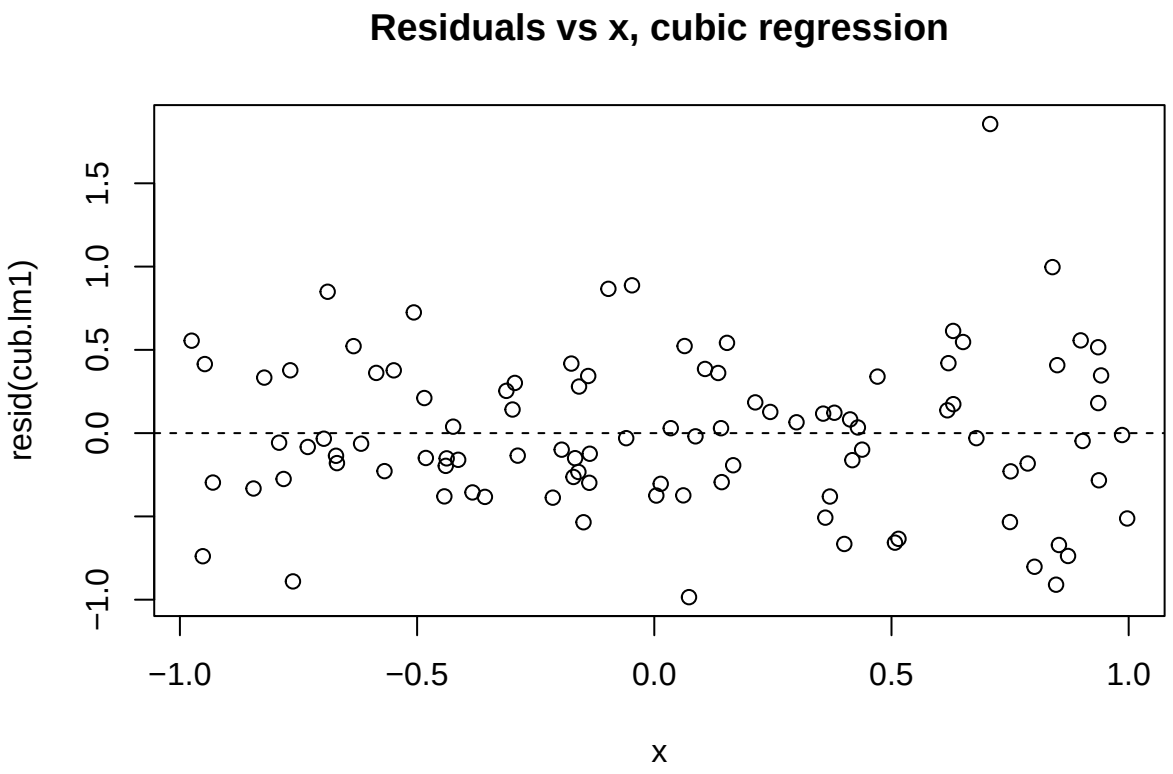
```
cub.lm1 = lm(y ~ x + I(x^2) + I(x^3), df.sim)
cub.lm2 = lm(y ~ poly(x,3), df.sim)
plot(fitted(cub.lm1),fitted(cub.lm2))
```




```
plot(fitted(cub.lm1),resid(cub.lm1), main = "Residuals vs fitted values, cubic regression")
```



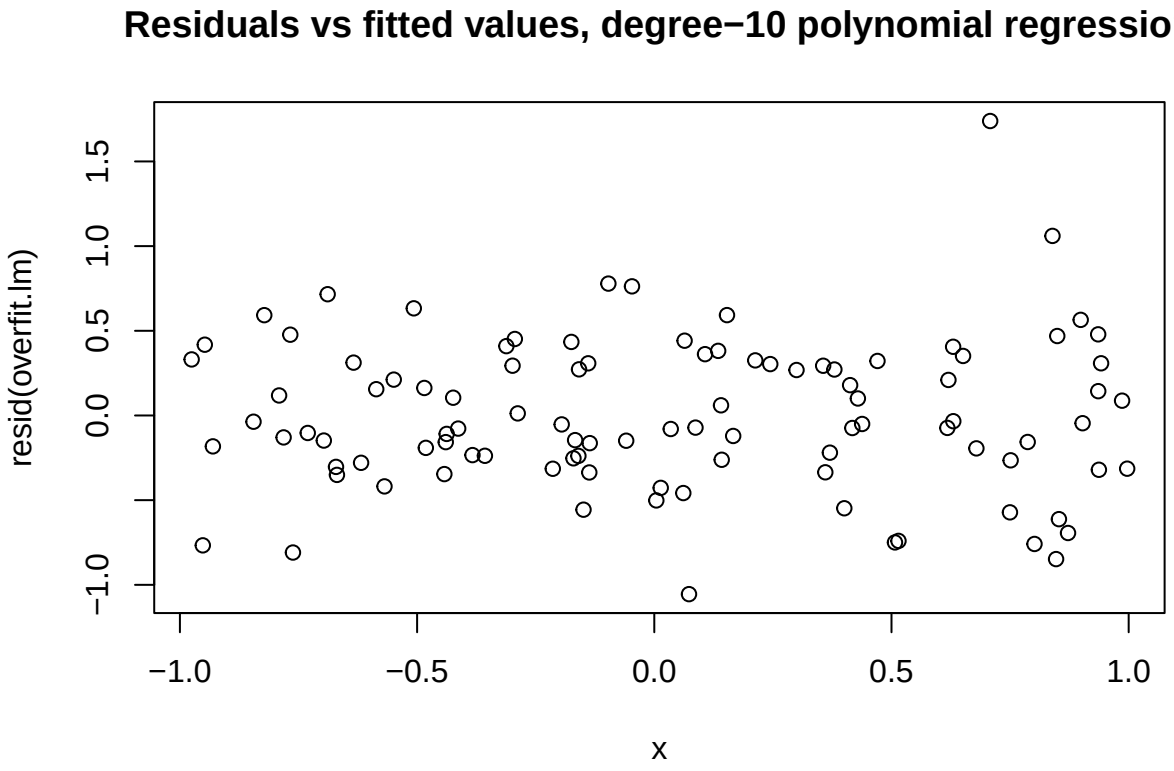

```
plot(x,resid(cub.lm1), main = "Residuals vs x, cubic regression")
abline(h = 0,lty = 2)
```



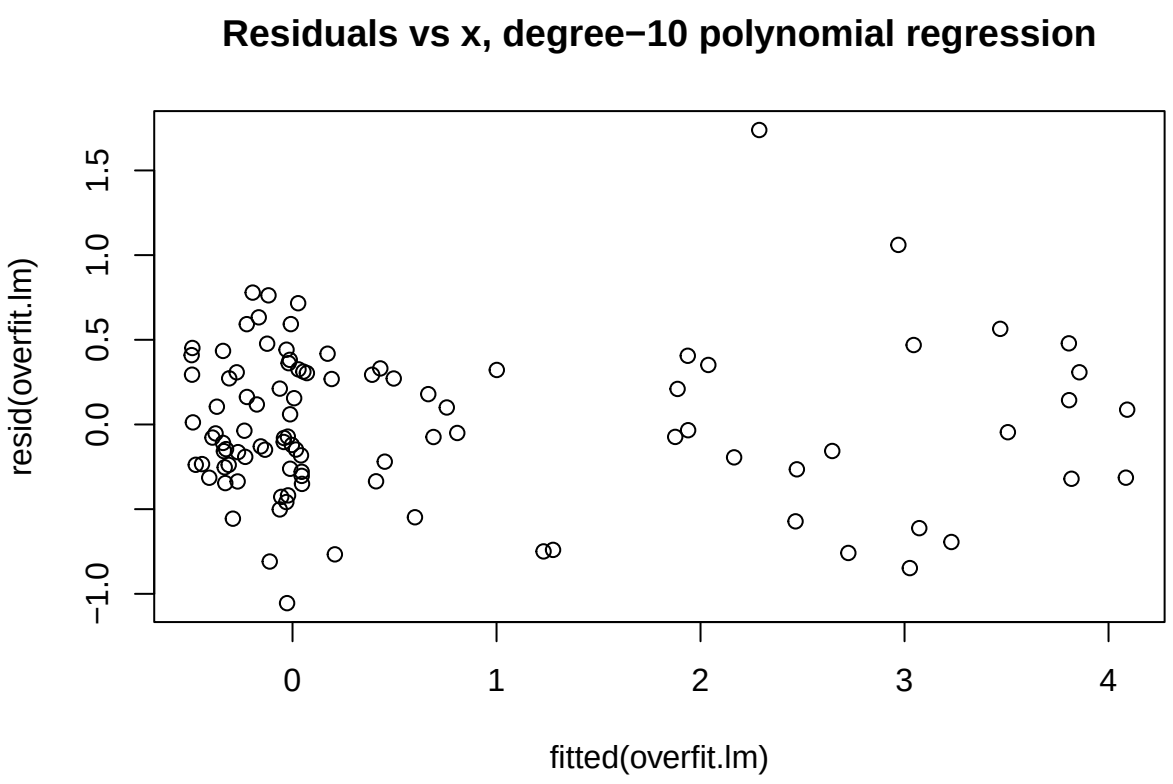
That's better: now the residuals have no obvious trend in the fitted values. **Diagnostics note**: for residual plots with polynomial regression, there is no need to plot the residuals against $x, x^2, \ldots$. You can just plot against x (and against $\hat{y}$ as usual.)

Overfitting: Suppose we try fitting a degree-10 polynomial.

```
overfit.lm =  lm(y ~ poly(x,10), df.sim)
plot(x,resid(overfit.lm), main = "Residuals vs fitted values, degree-10 polynomial regression")
```




```
plot(fitted(overfit.lm), resid(overfit.lm), main = "Residuals vs x, degree-10 polynomial regression")
```



The residual plots look fine, which makes sense – certainly $m(x)$ is a degree-10 polynomial, its just that all of the slopes for the monomials $x^3, x^4, \ldots$ are equal to 0.

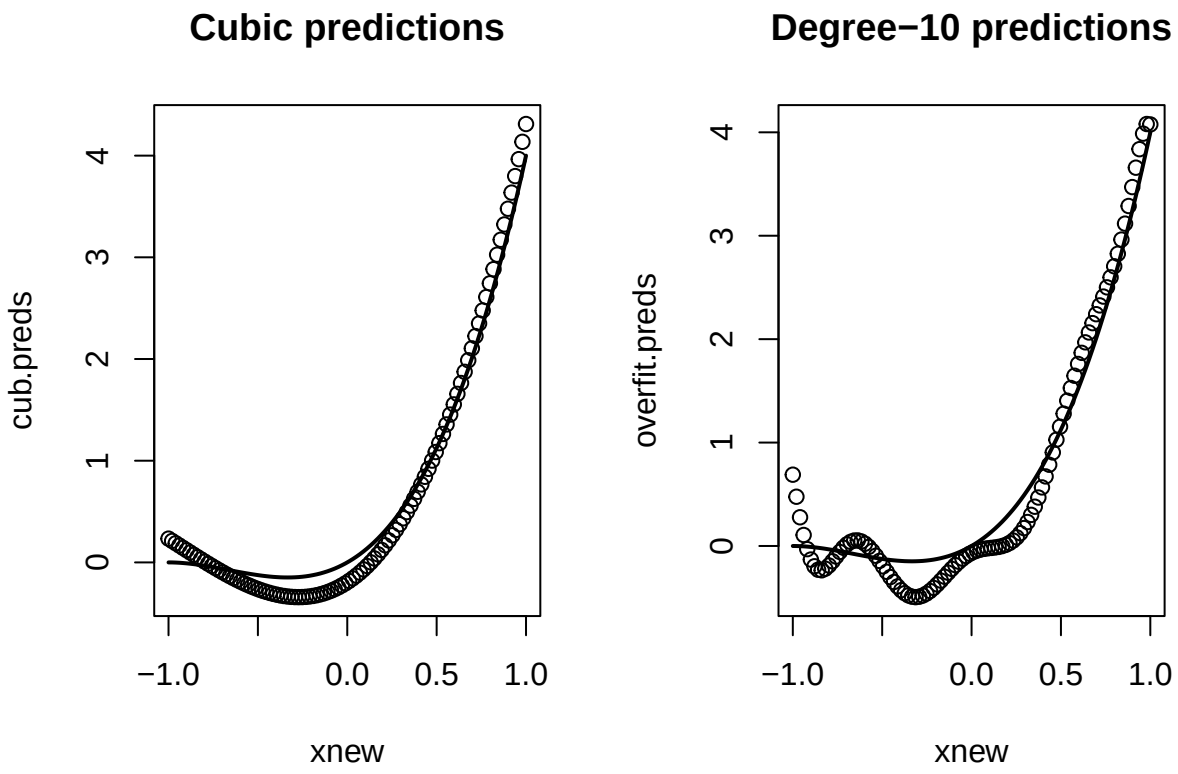
Now let's compare the predictions made by our cubic fit with our degree-10 polynomial fit. Notice that the `predict` function *automatically* knows that a cubic fit should be used.

```
par(mfrow = c(1,2))
# New predictors
xnew = seq(-1,1,length.out = 100)
newdata = data.frame(x = xnew) # New predictors

# Predictions of each model
cub.preds = predict(cub.lm1,newdata)
overfit.preds = predict(overfit.lm,newdata)

# Plot predictions of cubic model
plot(x = xnew,y = cub.preds, main = "Cubic predictions")
lines(x = xnew,y = m(xnew),lwd = 2) # True conditional mean

# Plot predictions of the degree-10 model
plot(x = xnew,y = overfit.preds, main = "Degree-10 predictions")
lines(x = xnew,y = m(xnew),lwd = 2) # True conditional mean
```



We can see that the fit by runnning OLS with a degree-10 polynomial is noticeably (albeit only slightly) wigglier and less accurate. This is empirical evidence of overfitting.

Extrapolation

One thing that we will talk about more seriously when we get to prediction and model selection next week are the dangers of extrapolation. But in a nutshell – if x' is a new value of x, very different than anything we have seen in our samples, then our predictions $\hat{m}(x')$ are untrustworthy.

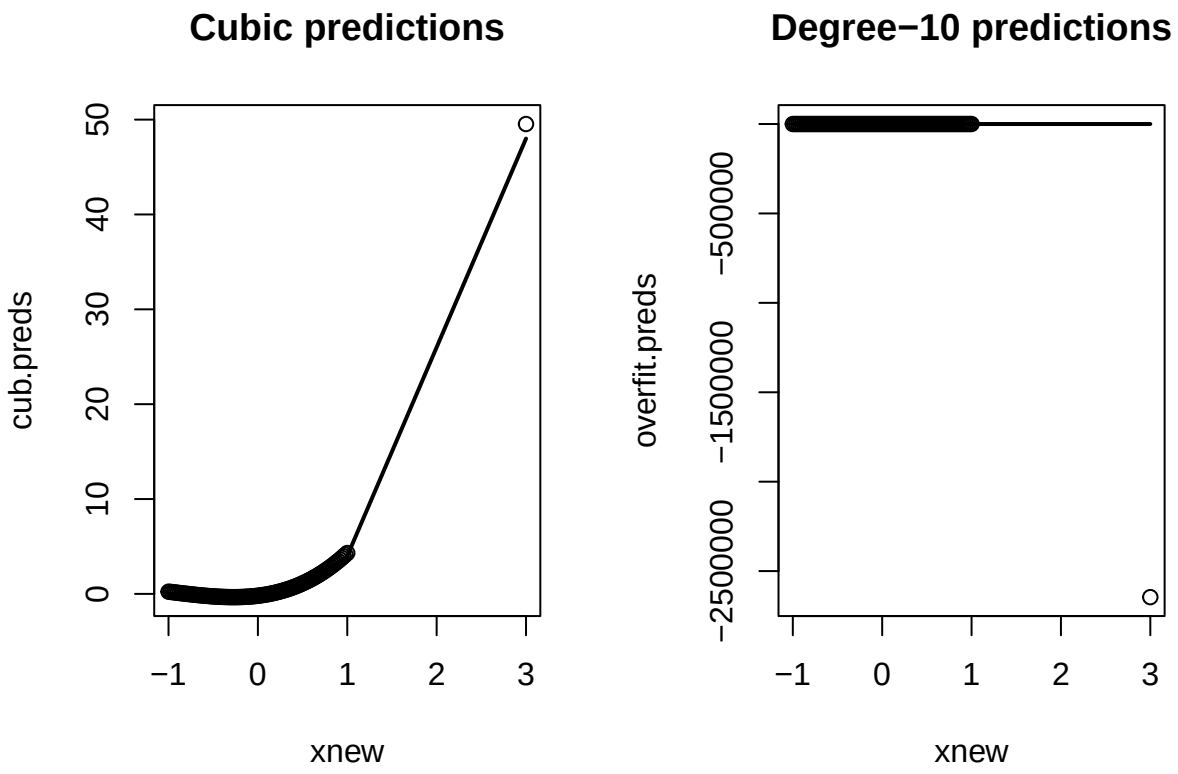
```
par(mfrow = c(1,2))

# New predictors
xnew = c(xnew,3) # one new x point is unlike the others...
newdata = data.frame(x = xnew)

# Predictions of each model
cub.preds = predict(cub.lm1,newdata)
overfit.preds = predict(overfit.lm,newdata)

# Plot predictions of cubic model
plot(x = xnew,y = cub.preds, main = "Cubic predictions")
lines(x = xnew,y = m(xnew),lwd = 2) # True conditional mean

# Plot predictions of the degree-10 model
plot(x = xnew,y = overfit.preds, main = "Degree-10 predictions")
lines(x = xnew,y = m(xnew),lwd = 2) # True conditional mean
```



Both cubic and degree-10 polynomial fits are much worse when extrapolating. The problem is particular severe with the more overfit model.

Orthogonal polynomials There is one thing that I swept under the rug when introducing the `poly` function, which can be seen by comparing the *coefficients* (rather than the fitted values) of our two (equivalent) cubic fits.

```
cub.lm1
```

```
##
## Call:
## lm(formula = y ~ x + I(x^2) + I(x^3), data = df.sim)
##
## Coefficients:
## (Intercept)            x       I(x^2)       I(x^3)
##     -0.1943       1.1462       2.4664       0.8923
```

```
cub.lm2
```

```
##
## Call:
## lm(formula = y ~ poly(x, 3), data = df.sim)
##
## Coefficients:
## (Intercept)  poly(x, 3)1  poly(x, 3)2  poly(x, 3)3
##      0.6965      10.9926       7.5729       1.3185
```

We see that despite giving the same predictions, the coefficients of the two regression are different. What gives? The answer is the `poly(x,3)` does not use the monomials x, x^2, x^3 as predictors. Instead it orthogonalizes the

polynomials, by Gram-Schmidt. This leads to a more stable numerical procedure. So the two regressions use predictors matrices $\mathbf{\Phi}, \mathbf{\Phi}'$ that have different columns, but whose column space is the same: $\text{col}(\mathbf{\Phi}) = \text{col}(\mathbf{\Phi}')$.