Polynomial Regression

## Loading data

To load the wildfires data set, run the following code chunk:

wildfires <- read.csv("https://aloy.github.io/stat230-materials/data/wildfires.csv")

## Fitting a polynomial regression model

To fit a polynomial regression model we still use the lm() command, but we expand our formula to include polynomial terms. To include polynomial terms in a regression model, we need to use the I() function to indicate that we want to calculate a polynomial term. For example, to fit the quadratic model we have already discussed in class, we use the following code:

quadratic\_lm <- lm(Acres ~ Year + I(Year^2), data = wildfires)

Once you have your fitted model, we can explore it like we have with simple linear regression models.

## Exploring a cubic model

Let’s fit a cubic model to the wildfires data set. The cubic model has the form

Use the lm() command to fit the cubic model where Year is used to predict Acres.

To plot the fitted cubic model, you can use the gf\_point() and gf\_lm() functions from the ggformula package. The following code will create a scatter plot of the data and add the fitted cubic regression line:

gf\_point(Acres ~ Year, data = wildfires, xlab = "Year", ylab = "Acres burned") |>  
 gf\_lm(formula = y ~ poly(x, 3), size = .5)

Warning: Using the `size` aesthetic with geom\_line was deprecated in ggplot2 3.4.0.  
ℹ Please use the `linewidth` aesthetic instead.



|  |
| --- |
| Note |
| In the gf\_lm() layer we use the poly(x, 3) function to specify that we want to fit a cubic polynomial. You can use poly() to fit polynomials of any degree by changing the second argument, and you can also use this function within the lm() function to fit polynomial regression models if you’d like. |

Does the cubic model appear to be necessary? Use the summary() function to explore the fitted model and run a hypothesis test for the cubic term. What do you conclude?

What degrees of freedom did R for the t-distribution used to calculate the p-value for the test of the cubic term?

Do you notice anything curious about the inferential results for the linear and quadratic terms?

The issue here is that the polynomial terms for year are highly correlated with each other (i.e., year, year, and year are correlated). This can lead to numerical instability and make it difficult to interpret the coefficients. This is a situation called **multicollinearity**. We’ll talk more about this later. One way to remedy this issue in polynomial regression is to use orthogonal polynomials, which are uncorrelated with each other.

## An alternative way to fit polynomials

To fit polynomial model with uncorrelated polynomial terms use the poly() function. For example, to fit a cubic model using orthogonal polynomials, we can run the following code:

cubic\_lm\_ortho <- lm(Acres ~ poly(Year, 3), data = wildfires)

Use the summary() function to explore the fitted model. What do you notice about the inferential results for the linear, quadratic, and cubic terms? How does this compare to the previous cubic model we fit? How does it compare to the quadratic model?

The method of constructing the polynomial terms in our regression model does not change our predictions, but it can change the inferential results for the polynomial terms.

## Function quick reference

The following table summarizes the functions we learned today:

| Function | Purpose |
| --- | --- |
| lm(formula, data) | Fit a linear model. For polynomial regression the formula should include polynomial terms or use poly(). |
| I() | Used to create polynomial terms in a regression model |
| poly(x, degree) | Create orthogonal polynomial terms |