Day 5: Generalized Additive Models

Linear and Nonlinear Models

Linear Models (ARIMA and VAR)

- Make strong assumptions about the relationships between dependent and independent variables
- But they are easily interpretable

Non-linear Models

- Reduce (or eliminate) these assumptions
- But this is often done at the cost of interpretability

Non-linear Modeling

Non-linear models can be written generally as

$$y = g(x) + \epsilon$$

where $g(\cdot)$ can be **any** function.

- Tremendous flexibility
- Low likelihood of interpretability

Non-linear Modeling

If $g(\cdot)$ is a function of more than one parameter, interpretation may quickly become difficult.

$$y = x_1^2 x_2^2 + \epsilon$$

In this case, the marginal effect of x_1 on y is

$$\frac{\partial y}{\partial x_1} = 2x_1 x_2^2$$

and depends on the values of both x_1 and x_2.

Generalized Additive Models

GAMs allow us much of the flexibility of non-linear models, without the difficulty of interpretation.

- Each parameter's effect on the dependent variable is modeled as its own function
- Since the model is additive, interpretation is straightforward, and parameter effects can be isolated

Generalized Additive Models

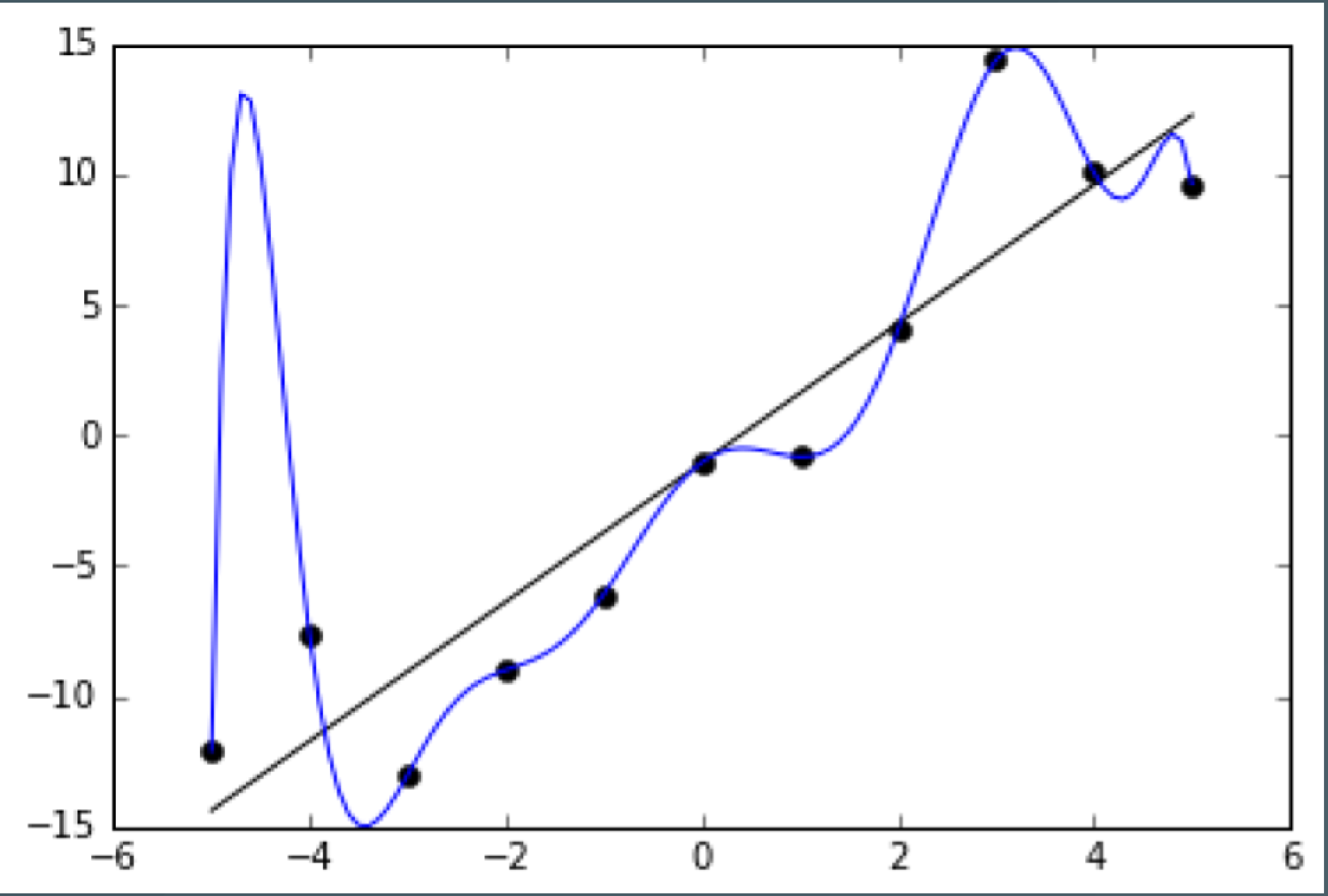
GAMs allow us much of the flexibility of non-linear models, without the difficulty of interpretation.

$$y = \sum_{i=1}^{N} f_i(x_i) + \epsilon$$

For two parameters, this could be expressed as

$$y = f_1(x_1) + f_2(x_2) + \epsilon$$

Non-linearity and Smoothness

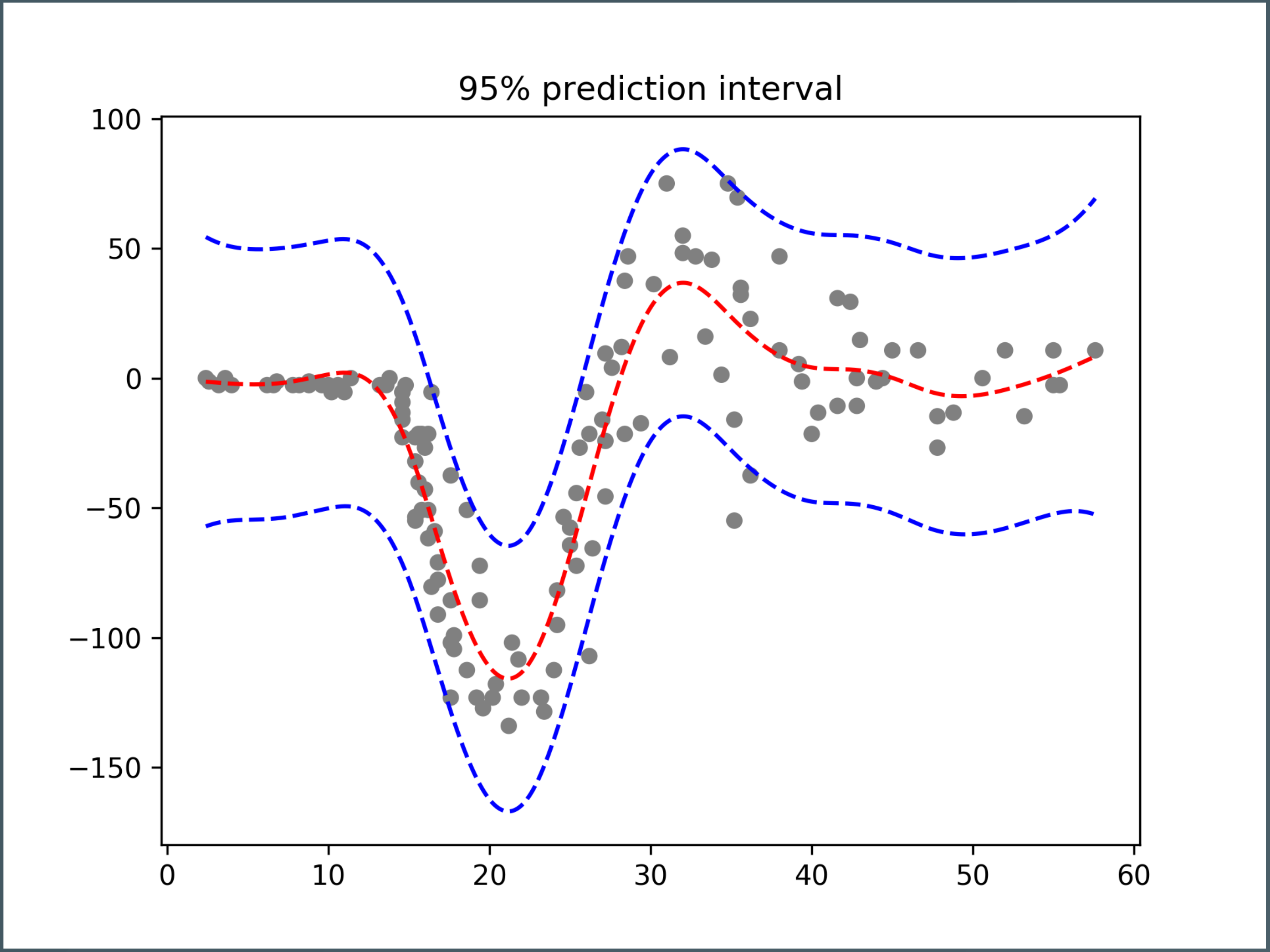


Non-linearity and Smoothness

On the previous slide, a high-order polynomial was fitted to a parameter.

- Was the fit perfect? **Yes**
- Was it likely to fit the true data-generating process? **No**

Non-linearity and Smoothness



Non-linearity and Smoothness

This time, our high-order polynomial actualy appears to represent the true relationship between the input and the output.

- Take care not to overfit your model
- Our true test will be when we fit a model, and use it to make predictions out-of-sample
- In sample, we can never do worse by applying a more complex functional form
- Out of sample, excess complexity can ruin our predictions

GAM Fitting Procedure

If we want to fit an additive model, we need to create a loss function that we can optimize. For one parameter, we need to optimize

$$y = a + f(x) + \epsilon$$

Sum of squared errors for this function is

$$SSE = \sum_{i=1}^{n} (y_i - a - f(x_i))^2$$

Choosing GAM Smoothness

In addition to minimizing the SSE term, we need to include a term that will regulate how smooth our function is, penalizing our model for "less smooth" functional forms.

Our *Penalized* Sum of Squared Errors (PSSE) is

$$\sum_{i=1}^{n}(y_i - a - f(x_i))^2 + \lambda \int_0^1 (f''(x))^2 dx$$

Choosing GAM Smoothness

λ is the parameter that we can adjust in order to choose how much we want to penalize our function for increased complexity.

$$\int_0^1 (f''(x))^2 dx$$

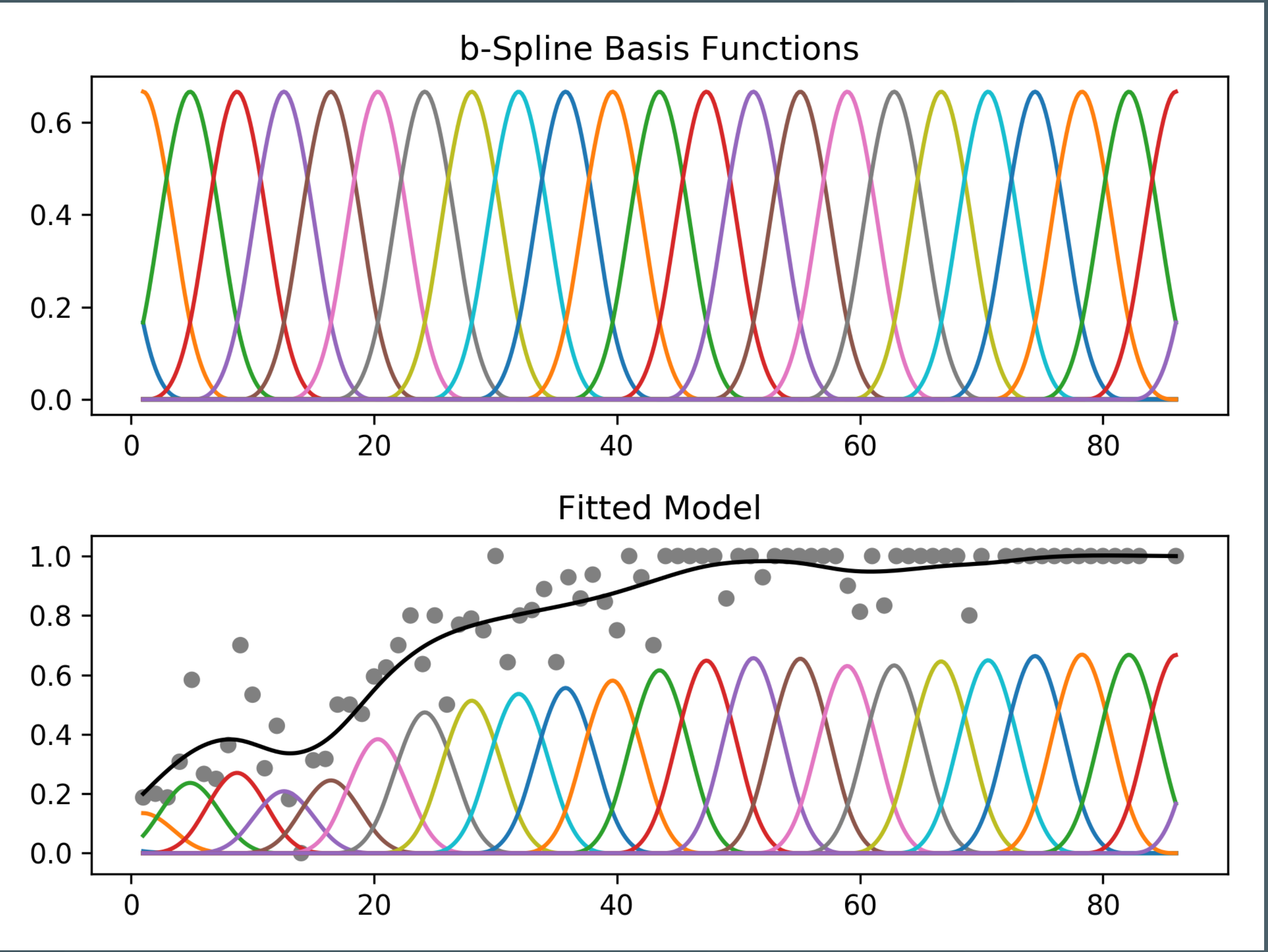
The integral term takes into account how quickly the slope of our function is changing over the interval [0,1], and penalizes our SSE when this value is high.

Fitting Functional Forms

In order to fit a GAM to the data, we need to be able to choose an arbitrary function from among nearly infinite options.

Splines are a way for us to generate these functions without having to use computationally expensive searches through the function space (the group of possible function matches to the true function)

Using Splines



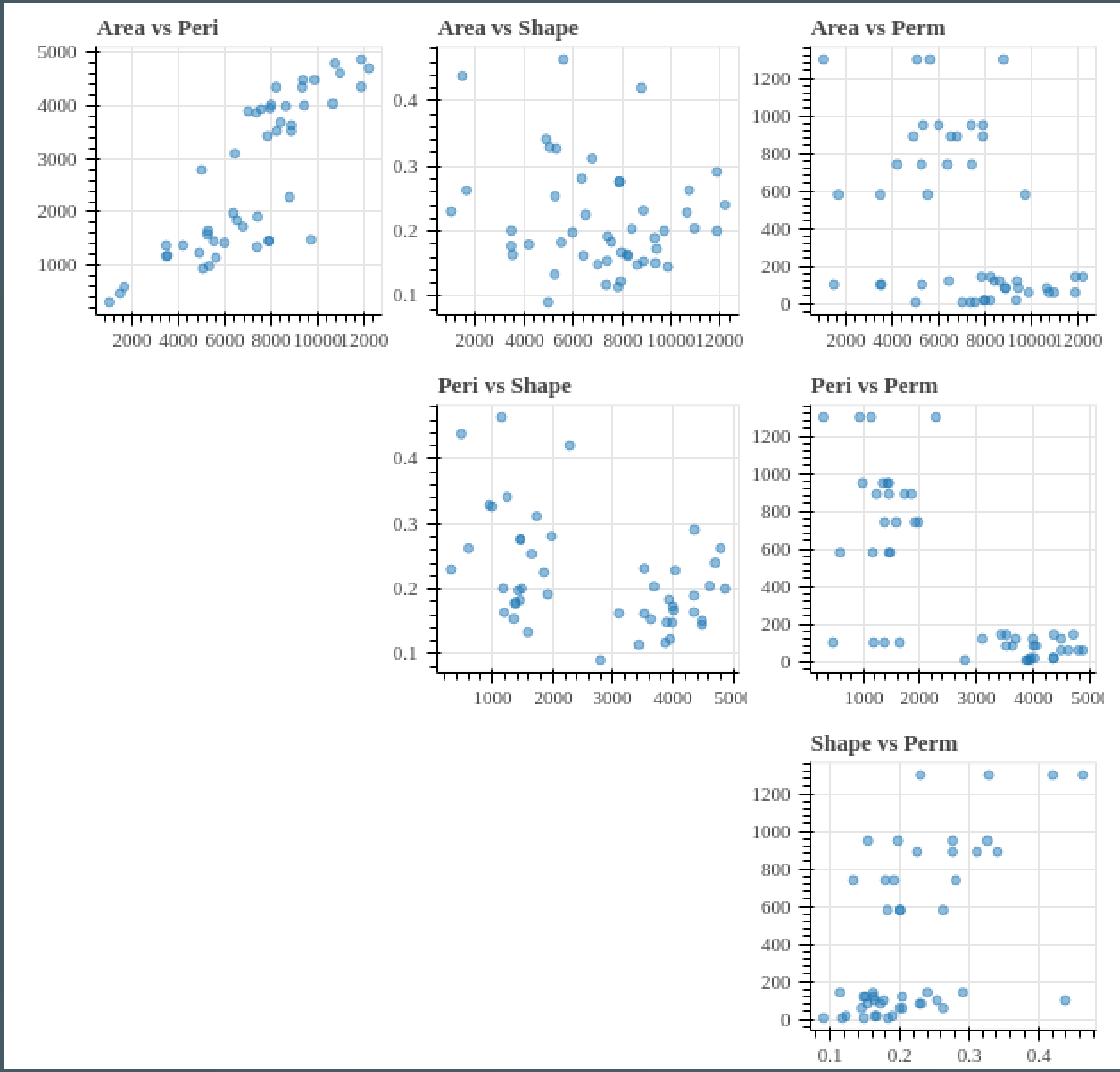
Implementing a GAM

```
# Our import statements
import pandas as pd
import numpy as np
import datetime
import patsy as pt
from pygam import LinearGAM
from bokeh.plotting import figure, show
from bokeh.layouts import gridplot

# Importing data from the web
path = 'http://www.stat.cmu.edu/~larry/' \
        'all-of-nonpar/=data/rock.dat'

data = pd.read_csv(path, sep=' *', engine='python')
```

Implementing a GAM



Implementing a GAM

```
X = data[['peri','shape','perm']]
y = data['area']

adjy = y - np.mean(y) # For plotting purposes

gam = LinearGAM(n_splines=10).gridsearch(X, y)
```

In order to be able to estimate our function, we need to choose a number of splines. This helps us to dictate how smooth our functional form will be. This combines with the λ term to determine overall smoothness of the function.

Implementing a GAM

```
# First, we need to create a mesh on the x-axis, so that
#   we are able to plot equidistant points on our marginal
#   effect diagrams

XX = generate_X_grid(gam)

titles = ['peri', 'shape', 'perm']

# Calculate the marginal effects of each variable at
#   all points on the x-axis mesh XX, as well as
#   calculating the confidence intervals (at 95% level)

pdep, confi = gam.partial_dependence(XX, width=.95)
```

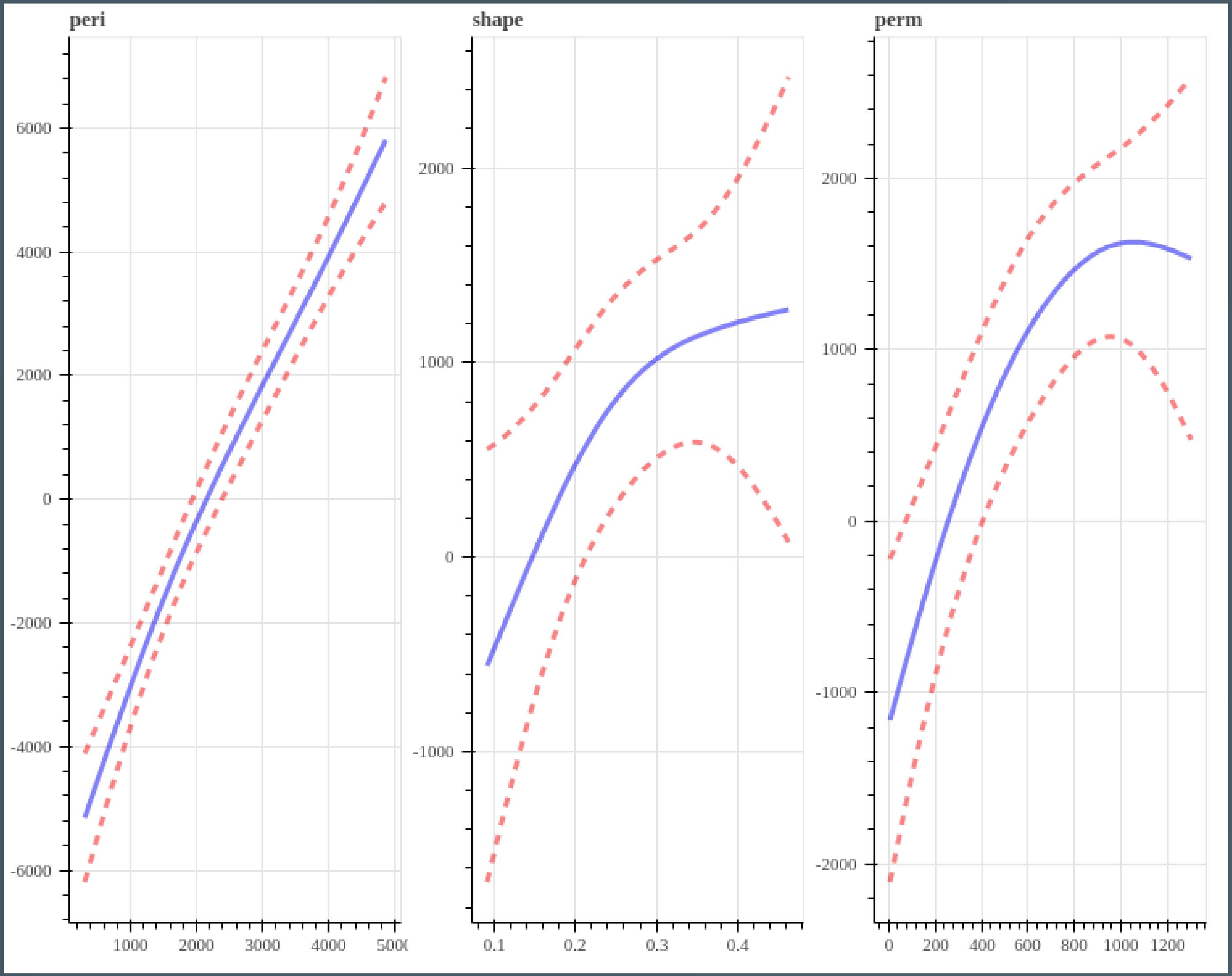
Implementing a GAM

```
# Create a list so that we can embed plots in it
p = list()

# Plot the effects for each variable
for i in range(3):
    p.append(figure(title=titles[i], plot_width=250,
        toolbar_location=None))
    p[i].line(XX[:, i], pdep[:,i], color='blue',
        line_width=3, alpha=0.5)
    p[i].line(XX[:, i], confi[i][:, 0], color='red',
        line_width=3, alpha=0.5, line_dash='dashed')
    p[i].line(XX[:, i], confi[i][:, 1], color='red',
        line_width=3, alpha=0.5, line_dash='dashed')

# Generate a grid of the plots for each effect
show(gridplot([p]))
```

Implementing a GAM



For Lab Today

In your teams, work to model future temperature based on the Omaha NOAA weather data in the file `omahaTemp.csv`. You should attempt to generate a model with high R^2, since this suggests that you have appropriately forecast the variation in temperature given the data provided.

Does changing the smoothness parameter λ or the number of splines allow you to improve your model accuracy?