Regression for Prediction

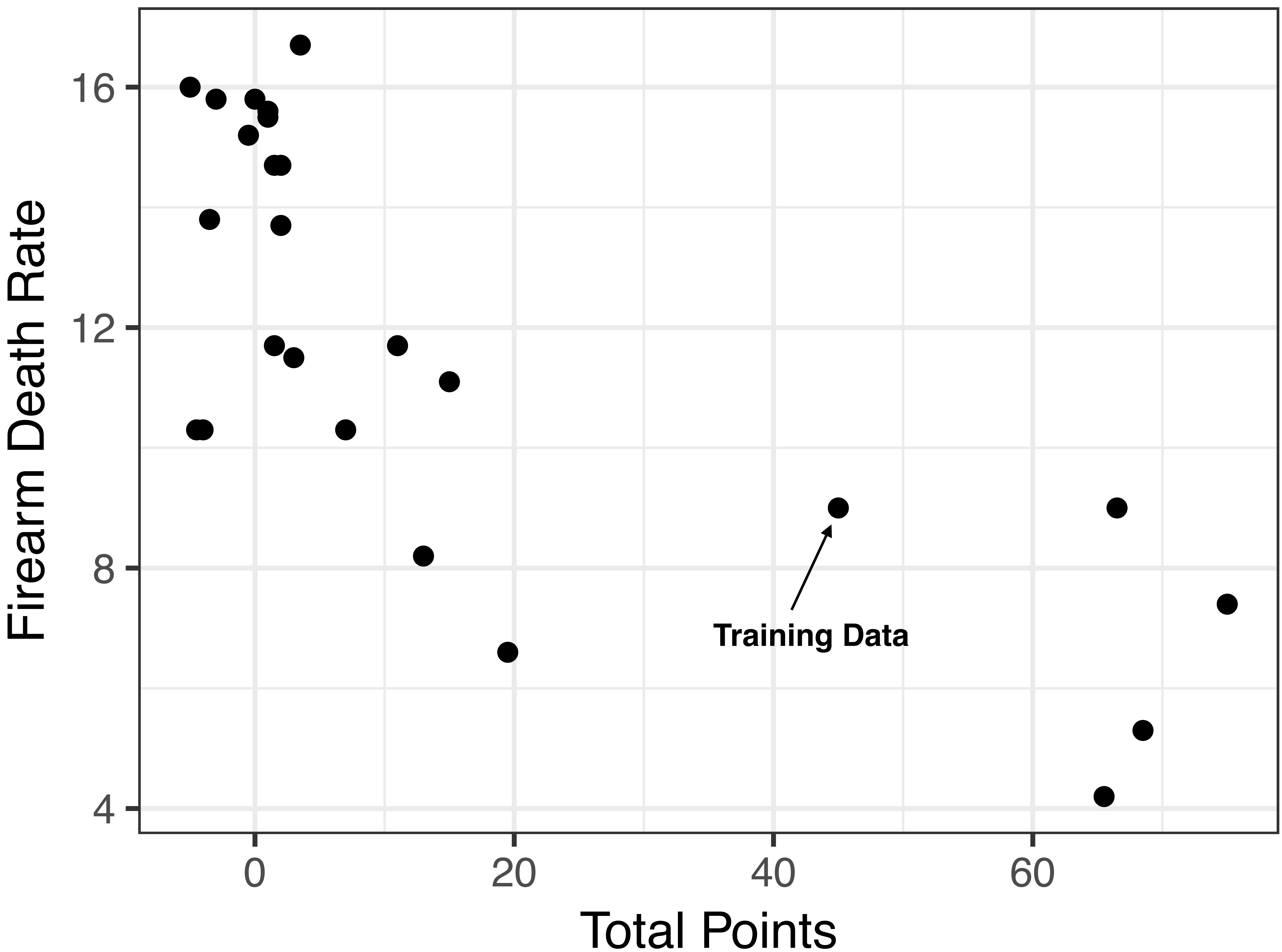
Balancing Complexity and Simplicity

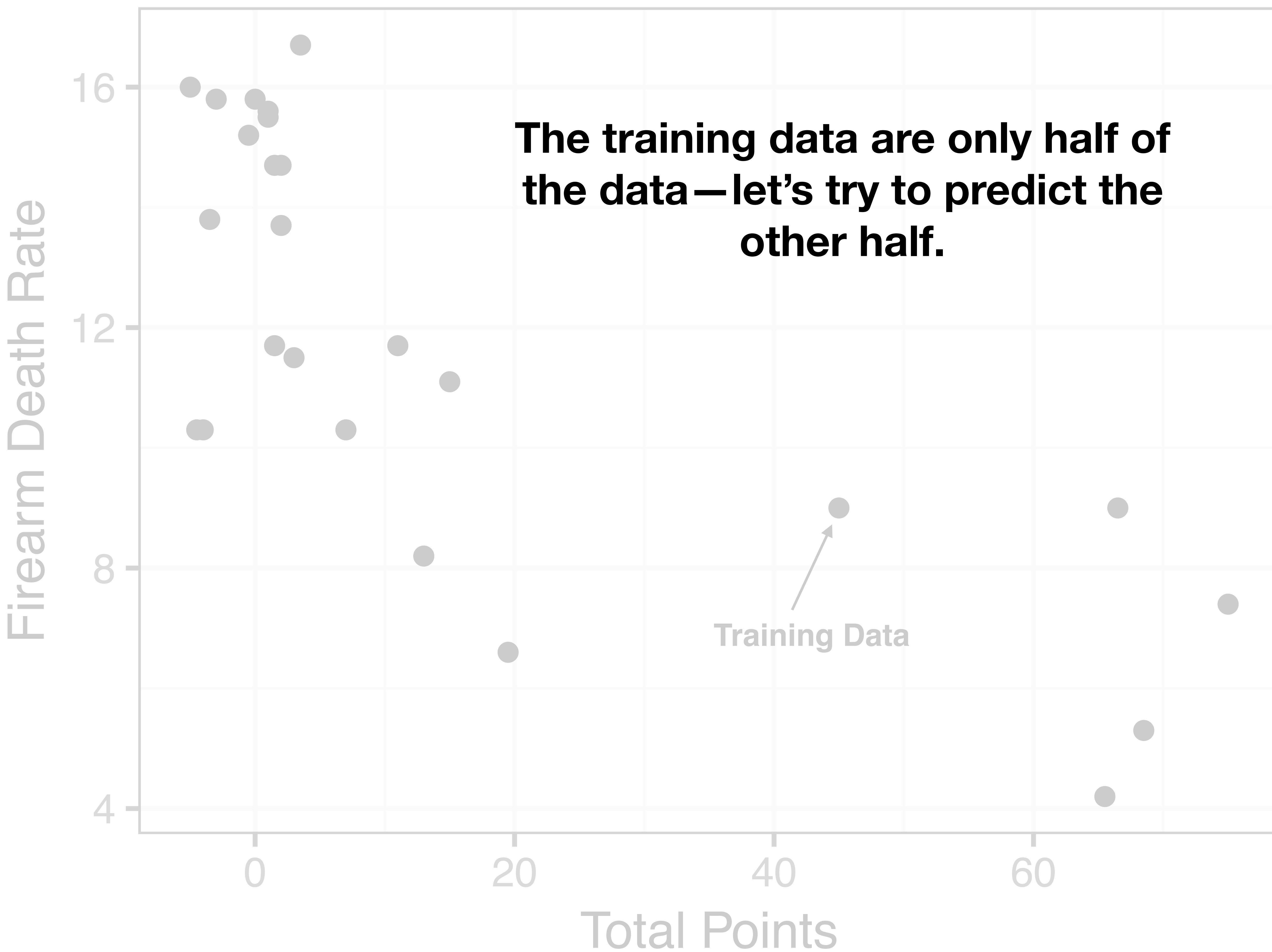
Goldilocks and the Three Bears

How do I balance simplicity and complexity?

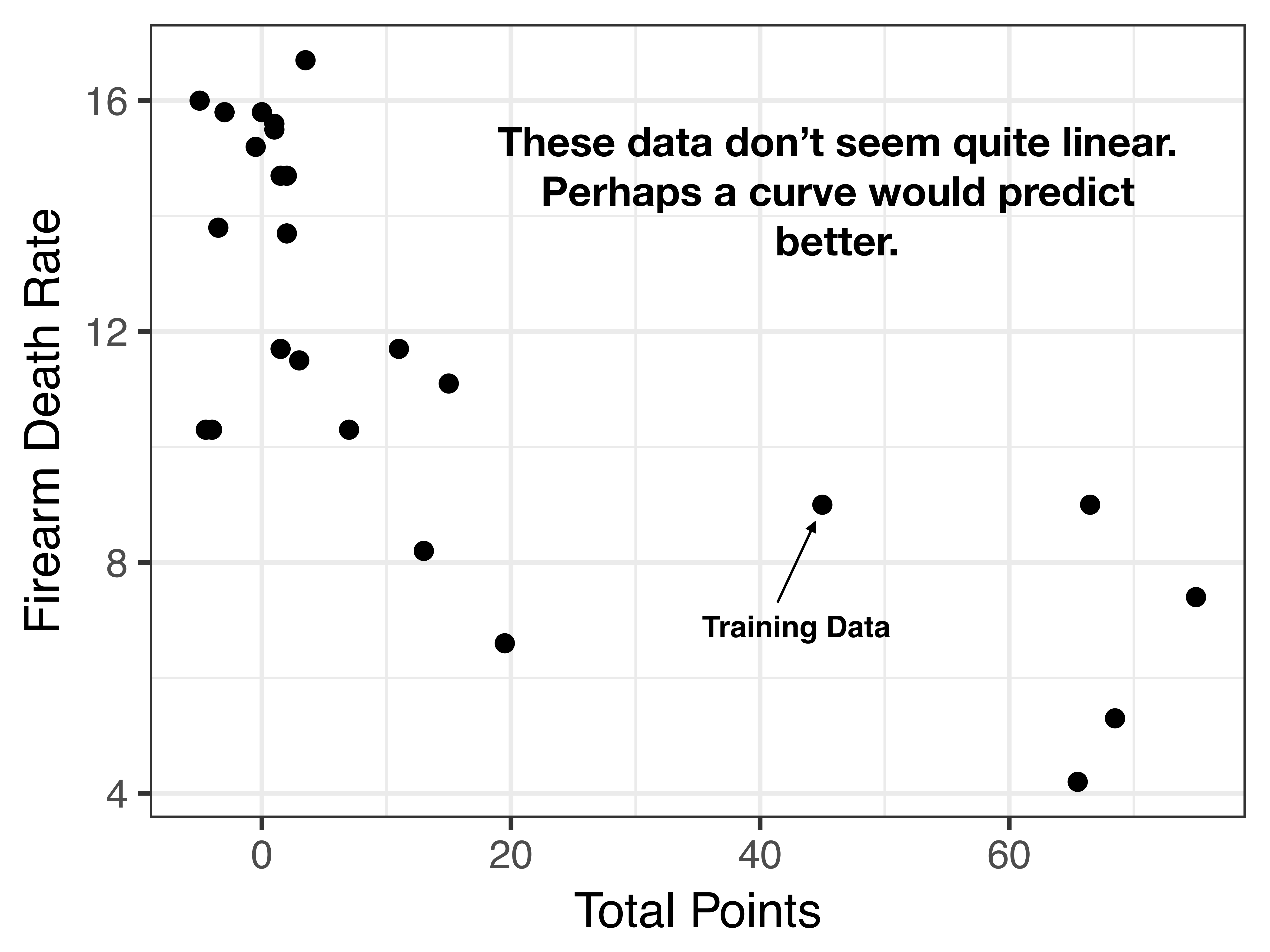
Three Types of Data Sets

- **training set**: cases and variables used to fit the models
- **prediction set**: cases to be predicted—includes same explanatory variables as training set, but missing the outcome of interest
- **test set**: has same cases as prediction set, but also includes the outcome of interest.


Firearm Death Rate
16
12
8
4
0
20
40
60
Total Points
Training Data



The training data are only half of the data—let's try to predict the other half.
Training Data
Firearm Death Rate
16
12
8
4
0
20
40
60
Total Points


These data don't seem quite linear. Perhaps a curve would predict better.
Training Data
Firearm Death Rate
16
12
8
4
0
20
40
60
Total Points

How curvy should the line be?

$$
\begin{aligned}
y &= \beta_0 + \beta_1 x + u \\
y &= \beta_0 + \beta_1 x + \beta_2 x^2 + u \\
y &= \beta_0 + \beta_1 x + \beta_2 x^2 + \beta_3 x^3 + u \\
&\vdots \\
y &= \beta_0 + \beta_1 x + \beta_2 x^2 + \beta_3 x^3 + \cdots + \beta_k x^k + u
\end{aligned}
$$

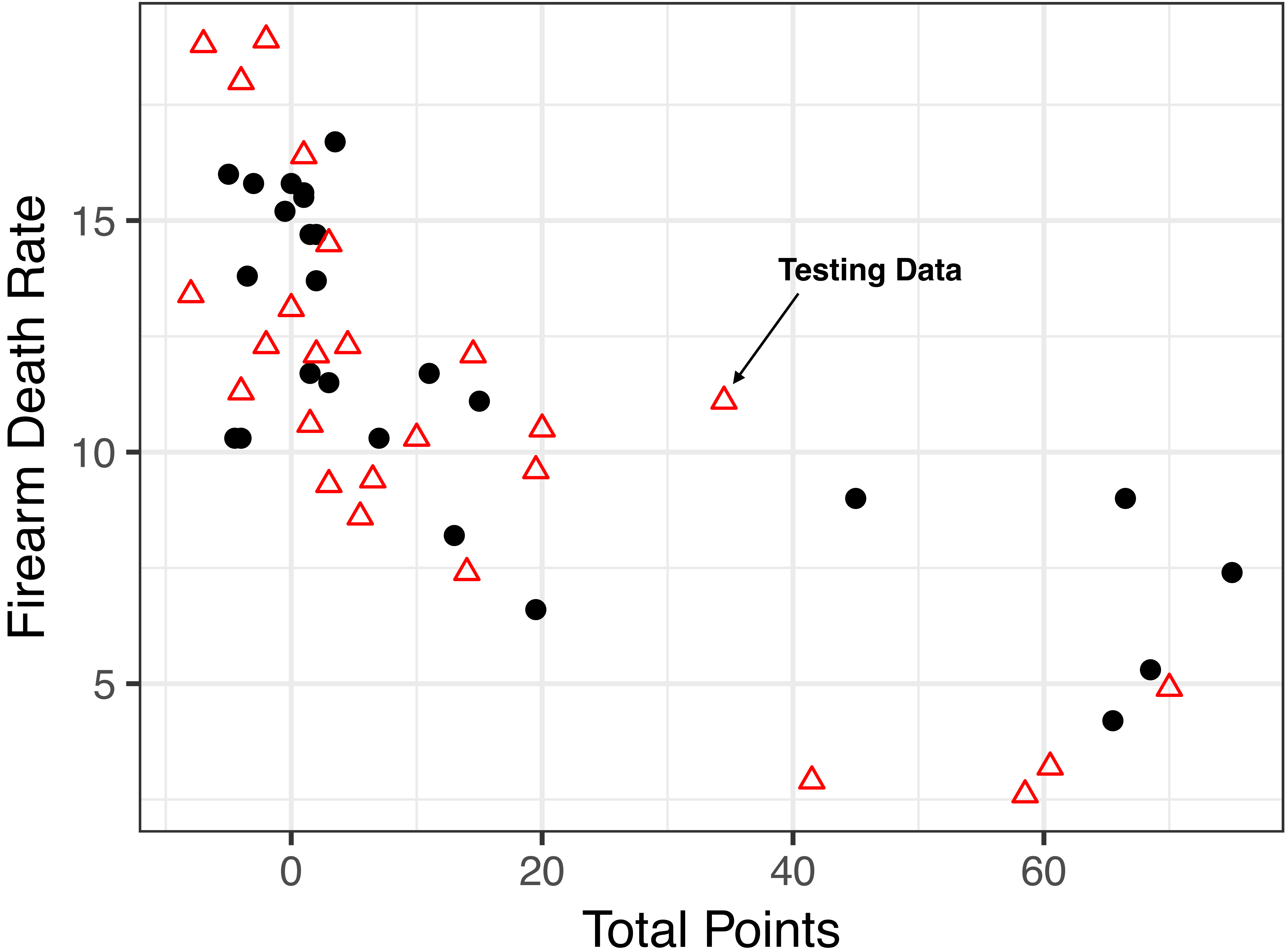
```
# 1st-order polynomial (i.e., linear)
> m <- lm(firearm_death_rate ~ total_points,
+         data = train)
> e <- residuals(m)
> sqrt(mean(e^2))
[1] 2.332054
```

```
# 2nd-order polynomial (i.e.,quadratic)
> m <- lm(firearm_death_rate ~ poly(total_points, 2),
+         data = train)
> e <- residuals(m)
> sqrt(mean(e^2))
[1] 2.198679
```

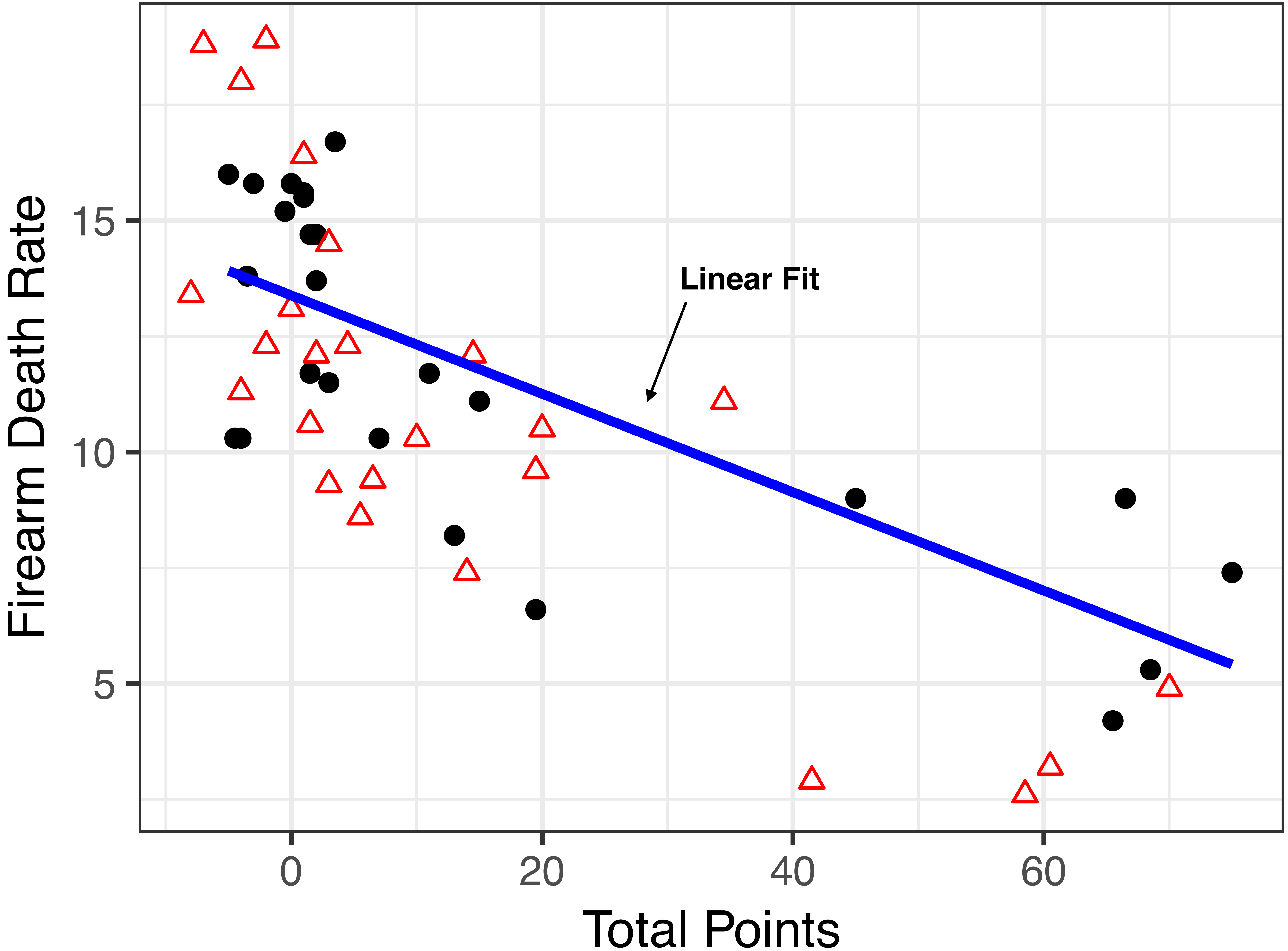
```
# 3rd-order polynomial (i.e., cubic)
> m <- lm(firearm_death_rate ~ poly(total_points, 3),
+         data = train)
> e <- residuals(m)
> sqrt(mean(e^2))
[1] 2.197419
```

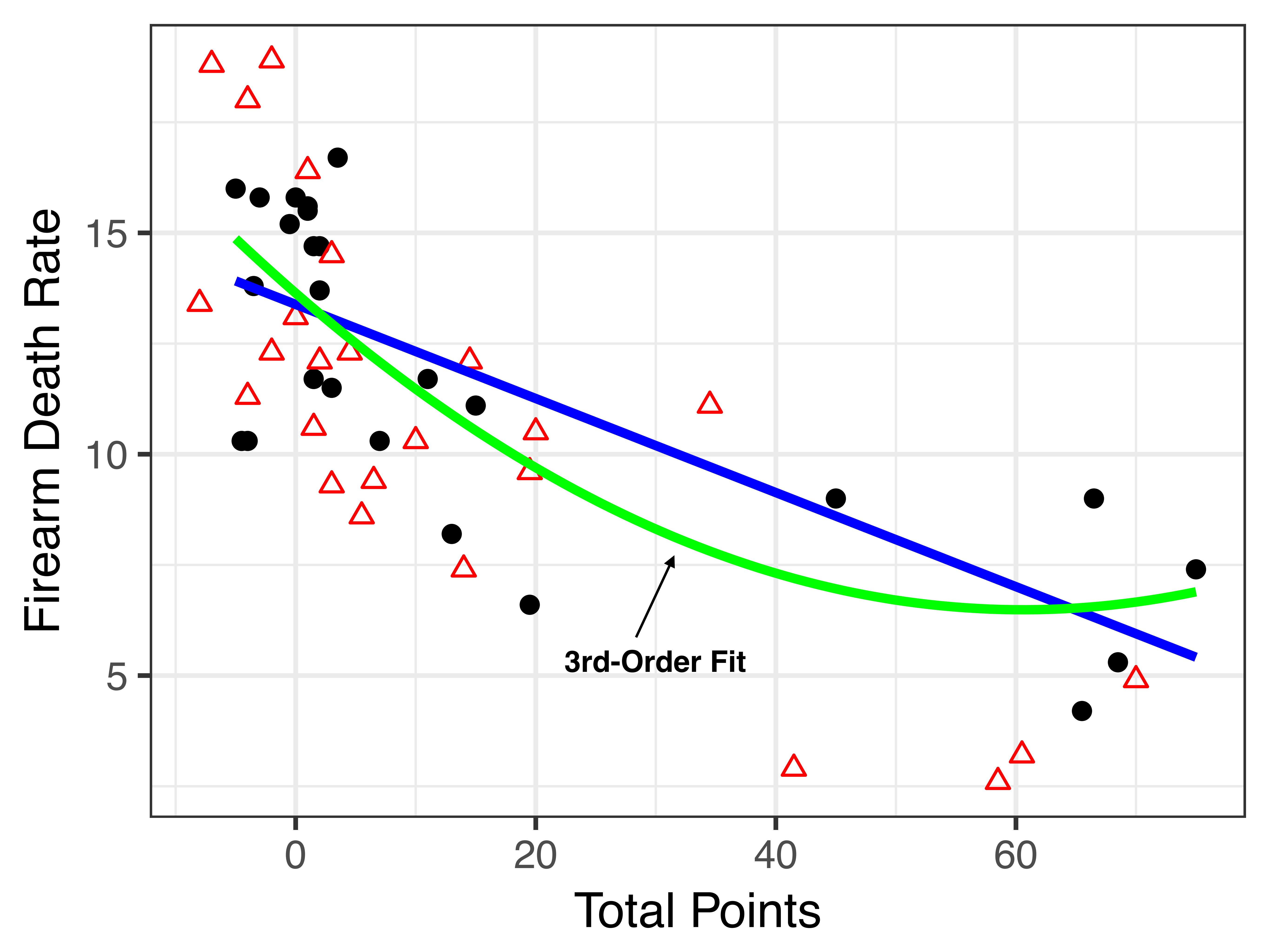
```
# 10th-order polynomial
> m <- lm(firearm_death_rate ~ poly(total_points, 10),
+         data = train)
> e <- residuals(m)
> sqrt(mean(e^2))
[1] 1.569003
```

Will the regression with the lowest r.m.s. error have the most predictive power?

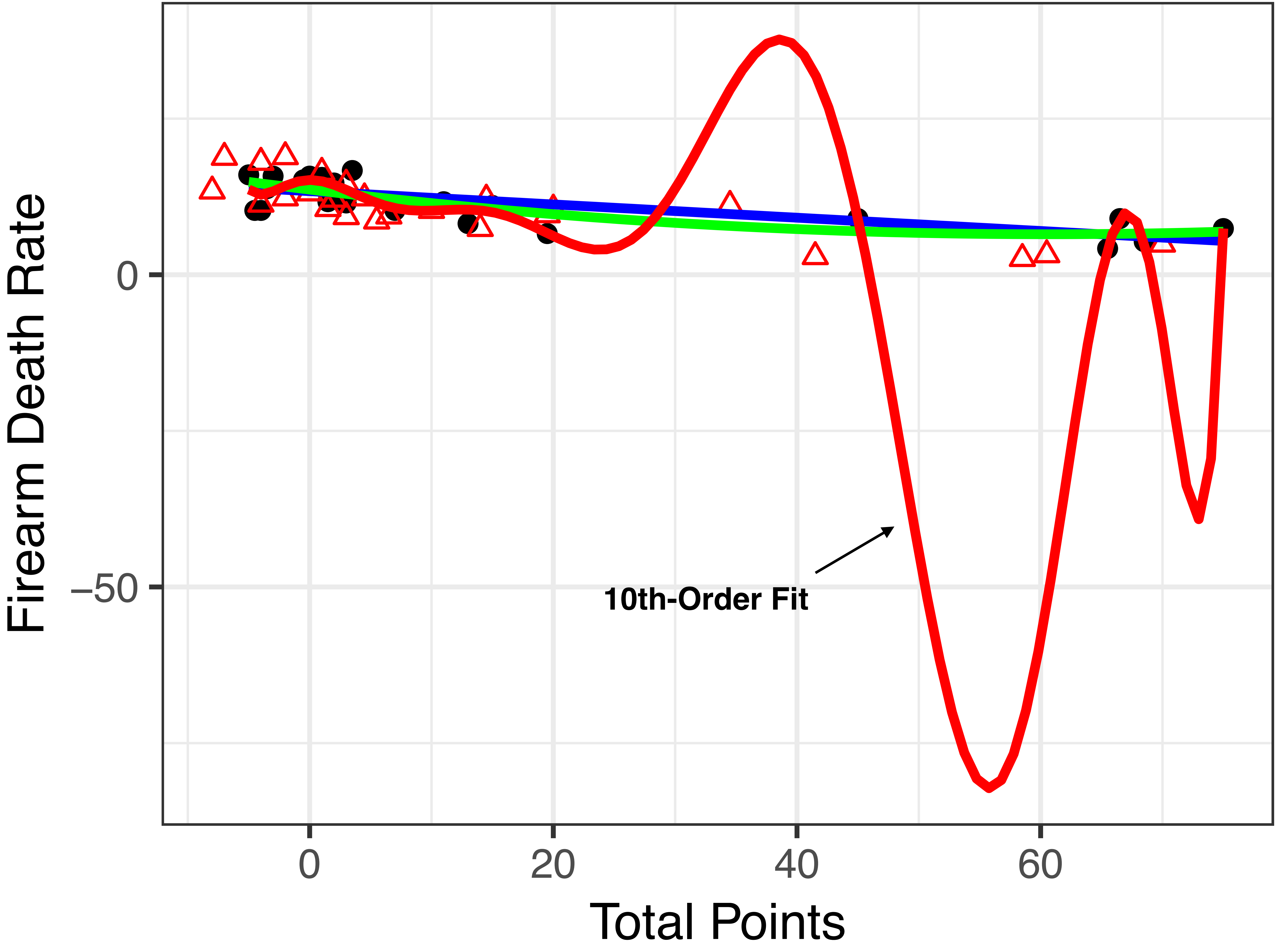


Testing Data
Firearm Death Rate
15
10
5
0
20
40
60
Total Points

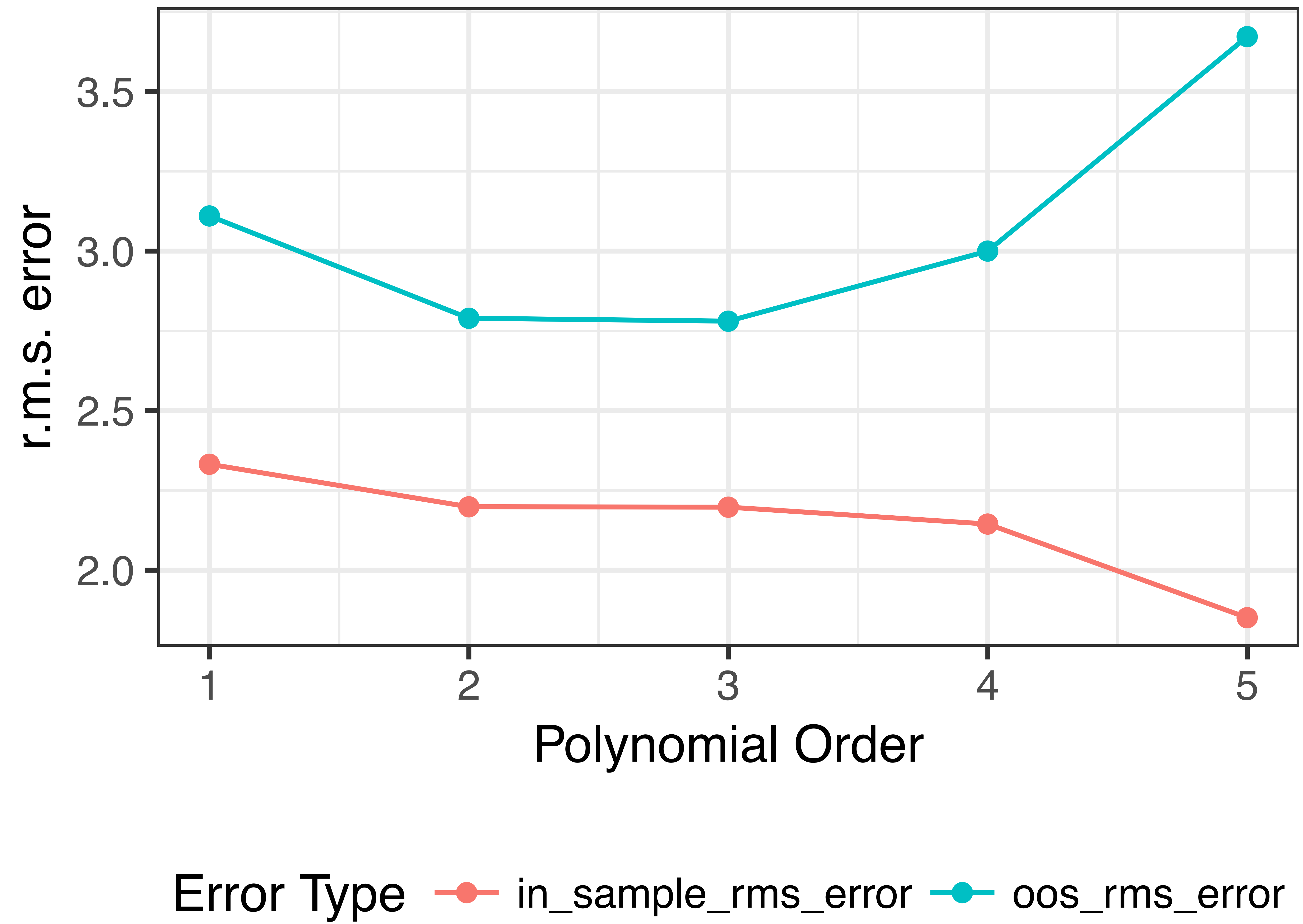

Linear Fit
Firearm Death Rate
15
10
5
0
20
40
60
Total Points



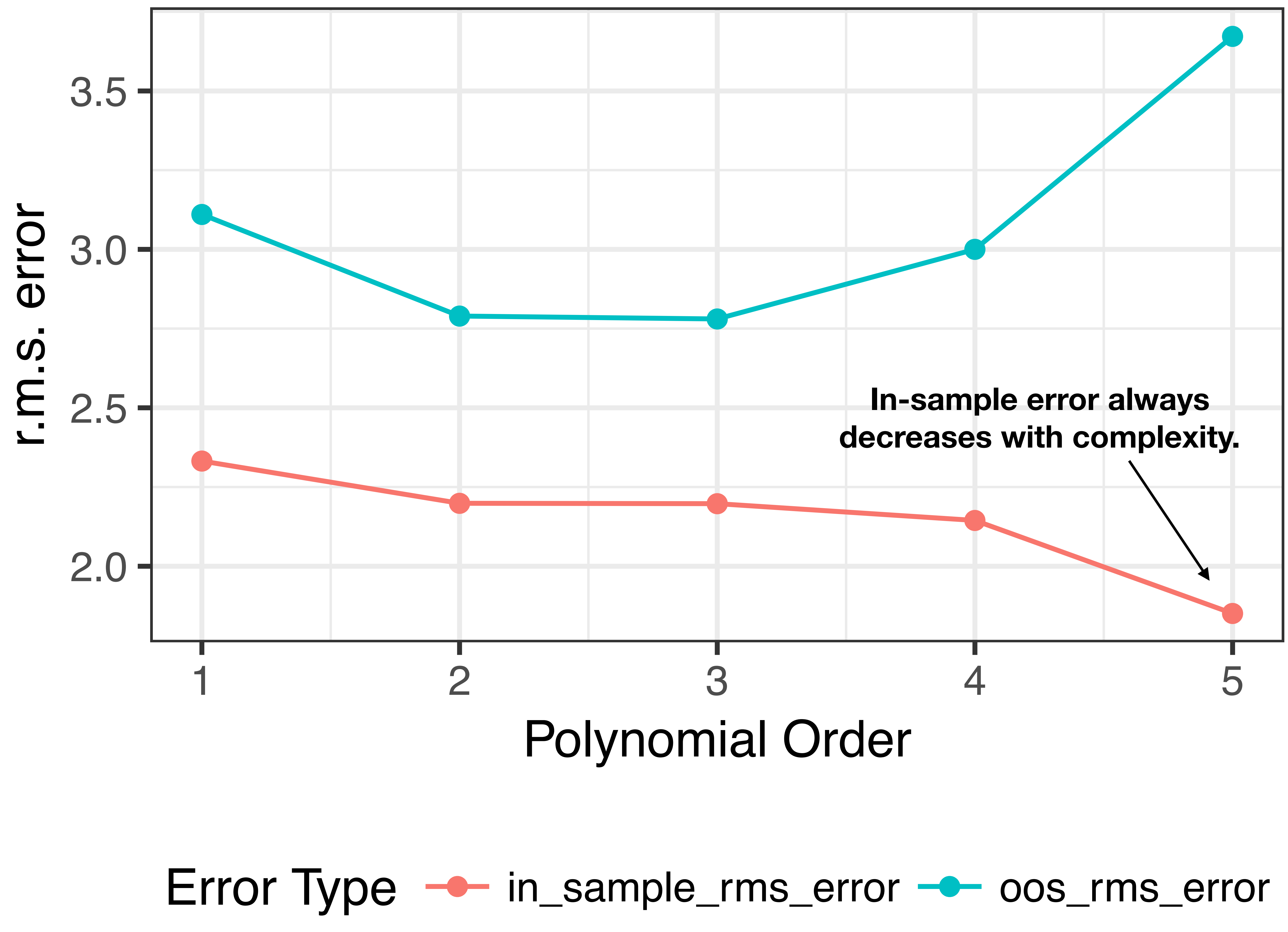
Firearm Death Rate
15
10
5
0
20
40
60
Total Points
3rd-Order Fit



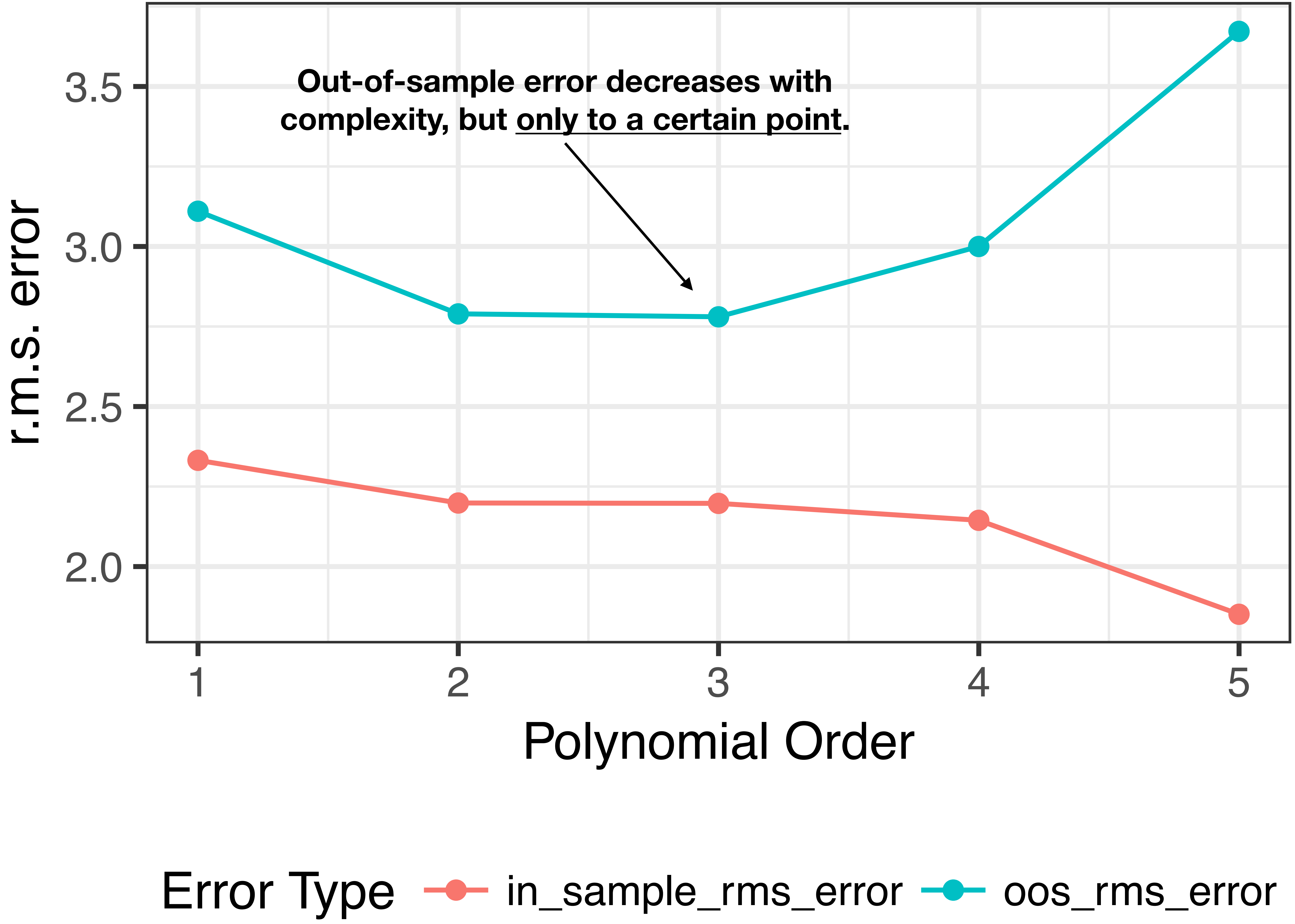
Firearm Death Rate
0
−50
0
20
40
60
Total Points
10th-Order Fit



r.m.s. error
3.5
3.0
2.5
2.0
1
2
3
4
5
Polynomial Order
Error Type
in_sample_rms_error
oos_rms_error



r.m.s. error
3.5
3.0
2.5
2.0
1
2
3
4
5
Polynomial Order
In-sample error always
decreases with complexity.
Error Type
in_sample_rms_error
oos_rms_error



Out-of-sample error decreases with
complexity, but only to a certain point.
r.m.s. error
3.5
3.0
2.5
2.0
1
2
3
4
5
Polynomial Order
Error Type
in_sample_rms_error
oos_rms_error

How do I balance simplicity and complexity?

How do [illegible] balance simplicity and complexity?

How do I balance simplicity and complexity?

INTUITION