Topic 12 – Non-linear regression

ENVX1002 Introduction to Statistical Methods

Liana Pozza
The University of Sydney

Dec 2024

THE UNIVERSITY OF SYDNEY

Recap

Last lecture…

Regressions

Simple linear regression

$$Y_i = \beta_0 + \beta_1 x_i + \epsilon_i$$

Ideal for predicting a continuous response variable from a single predictor variable: *"How does y change as x changes, when the relationship is linear?"*

Multiple linear regression

$$Y_i = \beta_0 + \beta_1 x_{1i} + \beta_2 x_{2i} + \ldots + \beta_k x_{ki} + \epsilon_i$$

"How does y change as x_1, x_2, ..., x_k change?"

Nonlinear regression

$$Y_i = f(x_i, \beta) + \epsilon_i$$

where $f(x_i, \beta)$ is a nonlinear function of the parameters β: "How do we model a change in y with x when the relationship is nonlinear?"

Nonlinear regression

Carl Friedrich Gauss (1777-1855) and Isaac Newton (1642-1726).

Fitting a nonlinear model

Linear relationships are simple to interpret since the rate of change is constant.

> "As one changes, the other changes at a constant rate."

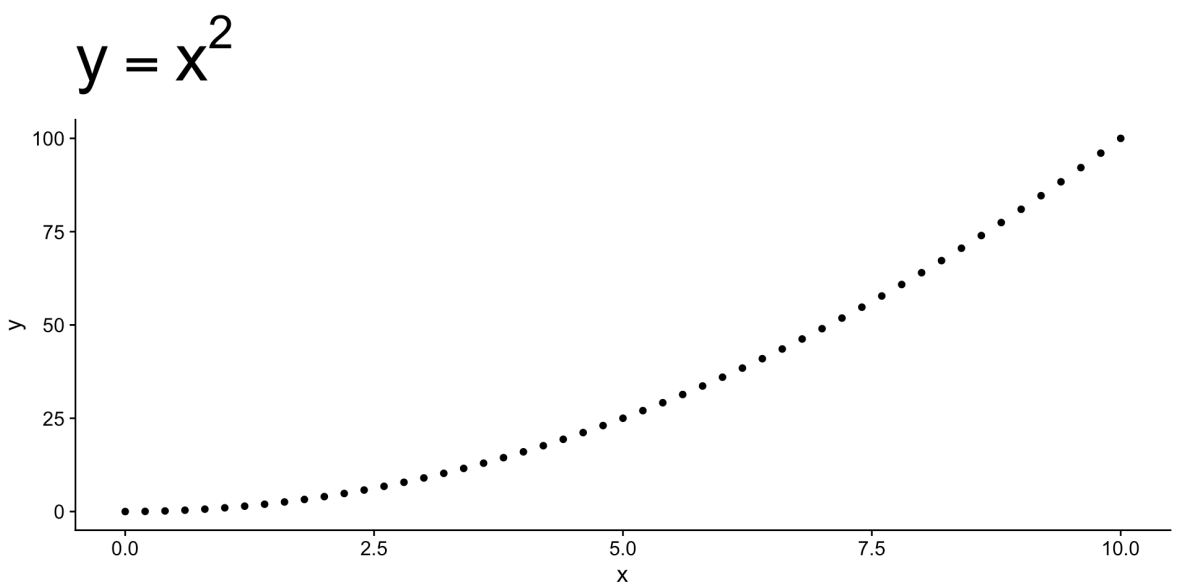
Nonlinear relationships often involve exponential, logarithmic, or power functions.

> "As one changes, the other changes at a rate that is *not proportional* to the change in the other."

Nonlinear relationships: exponents

- x^2 is the *square* of x.
- x^3 is the *cube* of x.
- x^a is x raised to the *power* of a.

> In a relationship where y is a function of x^a, as y increases, x increases at a rate that is equal to x to the power of a.



Nonlinear relationships: logarithms

- $log_e(x)$ is the *natural logarithm* of x.
- $log_{10}(x)$ is the *common logarithm* of x.
- $log_a(x)$ is the *logarithm* of x to the base a.

Interpretation:

- If $\log_a(y) = x$: as x increases, y increases at a rate of $y = a^x$.
- If $y = \log_a(x)$: as y increases, x also increases, at $x = a^y$.

Exponents and logarithms

	Exponents	Logarithms
Definition	If $a^n = b$, a is the base, n is the exponent, and b is the result.	If $\log_a b = n$, a is the base, b is the result, and n is the logarithm (or the exponent in the equivalent exponential form).
Example	$2^3 = 8$	$\log_2 8 = 3$
Interpretation	2 raised to the power of 3 equals 8.	The power to which you must raise 2 to get 8 is 3.
Inverse	The logarithm is the inverse operation of exponentiation.	The exponentiation is the inverse operation of logarithm.
Properties	$(a^n)^m = a^{n \cdot m}$, $a^n \cdot a^m = a^{n+m}$, $\frac{a^n}{a^m} = a^{n-m}$	$\log_a(b \cdot c) = \log_a b + \log_a c$, $\log_a\left(\frac{b}{c}\right) = \log_a b - \log_a c$, $\log_a(b^n) = n \cdot \log_a b$

Dealing with nonlinearity

Transformations

Often, a nonlinear relationship may be transformed into a linear relationship by applying a transformation to the response variable or the predictor variable(s).

- **Logarithmic**: $y = \log(x)$
- **Exponential**: $y = e^x$
- **Square-root**: $y = \sqrt{x}$
- **Inverse**: $y = \frac{1}{x}$

- All good when y changes monotically with x.
- What if relationship is not monotonic, or is more complex?

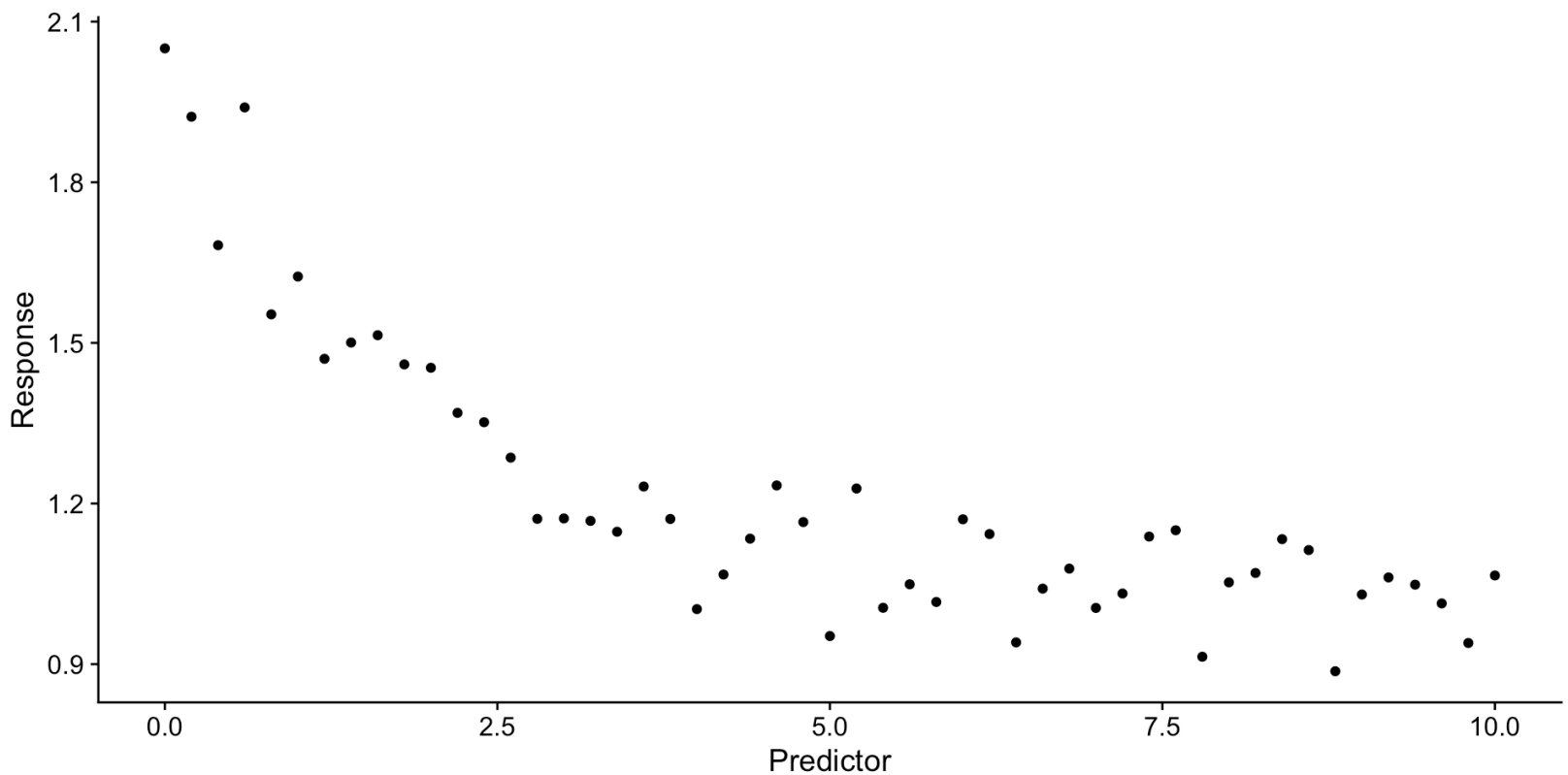
Common nonlinear functions

$f(x_i, \beta)$

Exponential decay relationship

Response variable *decreases* and approaches limit as predictor variable increases.

$$y = a \cdot e^{-b_x}$$

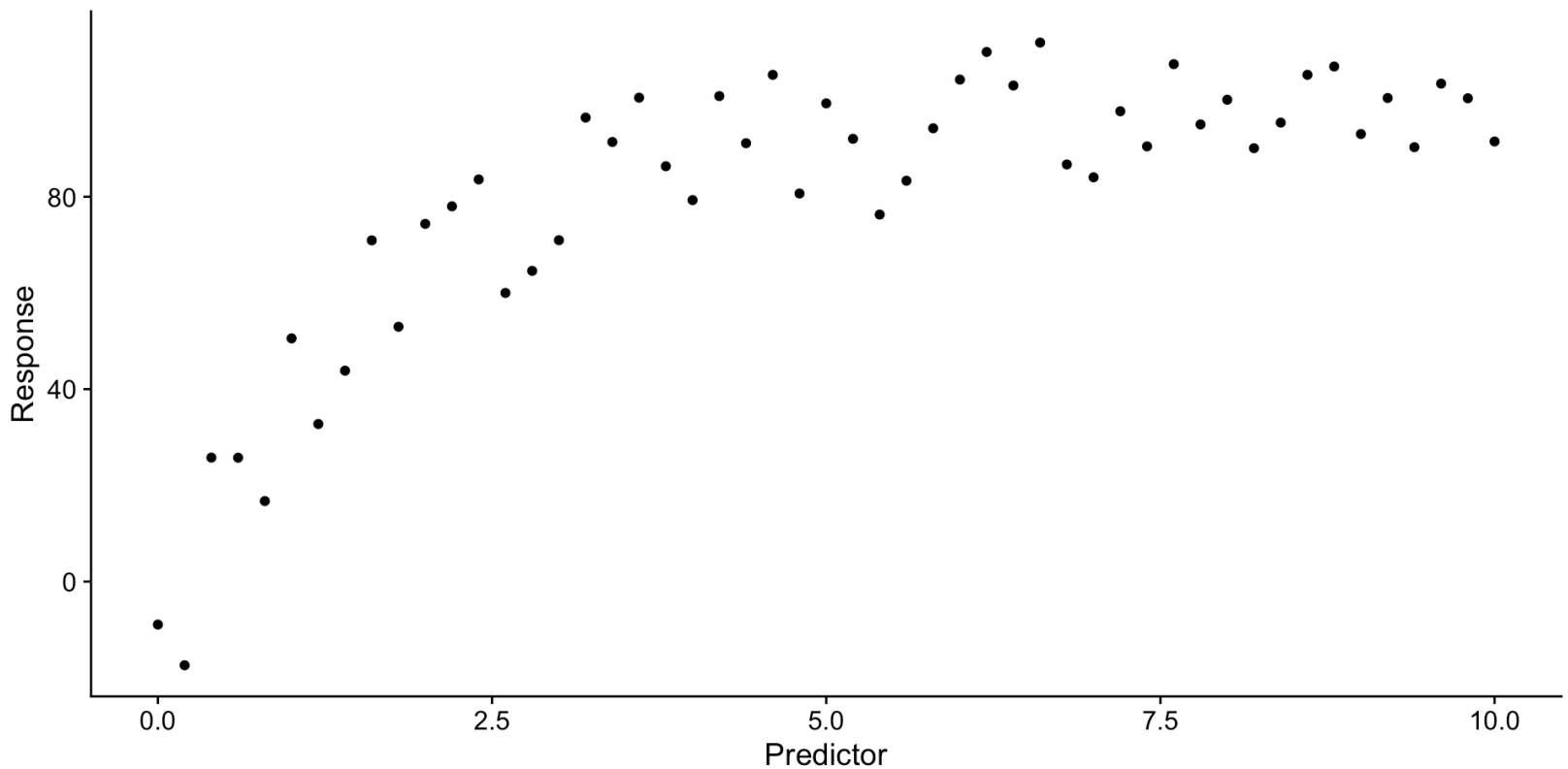


Examples: radioactive decay, population decline, chemical reactions.

Asymptotic relationship

Response variable *increases* and approaches a limit as the predictor variable increases.

$$y = a + b(1 - e^{-cx})$$

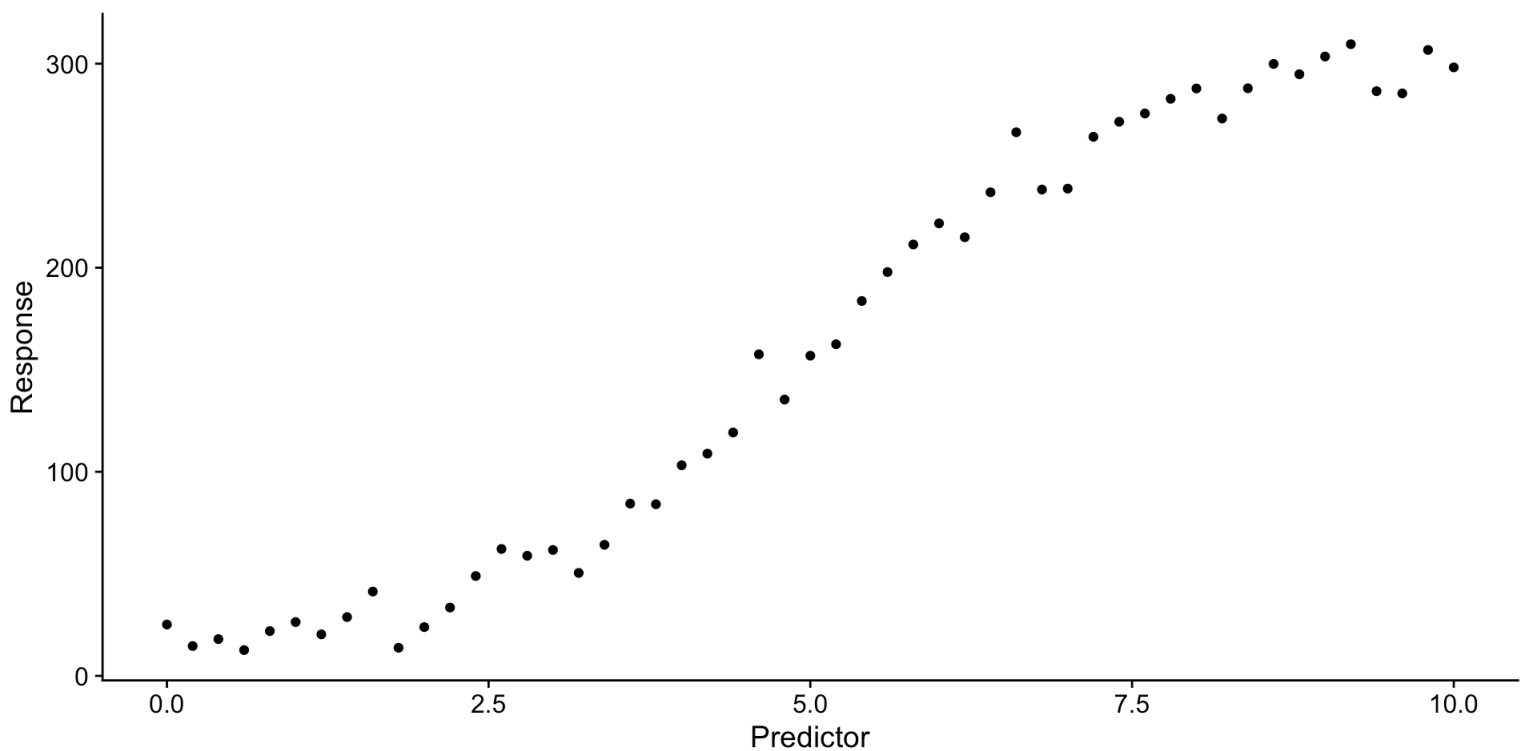


Examples: population growth, enzyme kinetics.

Logistic relationship

An S-shaped relationship, where the response variable is at first exponential, then asymptotic.

$$y = c + \frac{d - c}{1 + e^{-b(x-a)}}$$

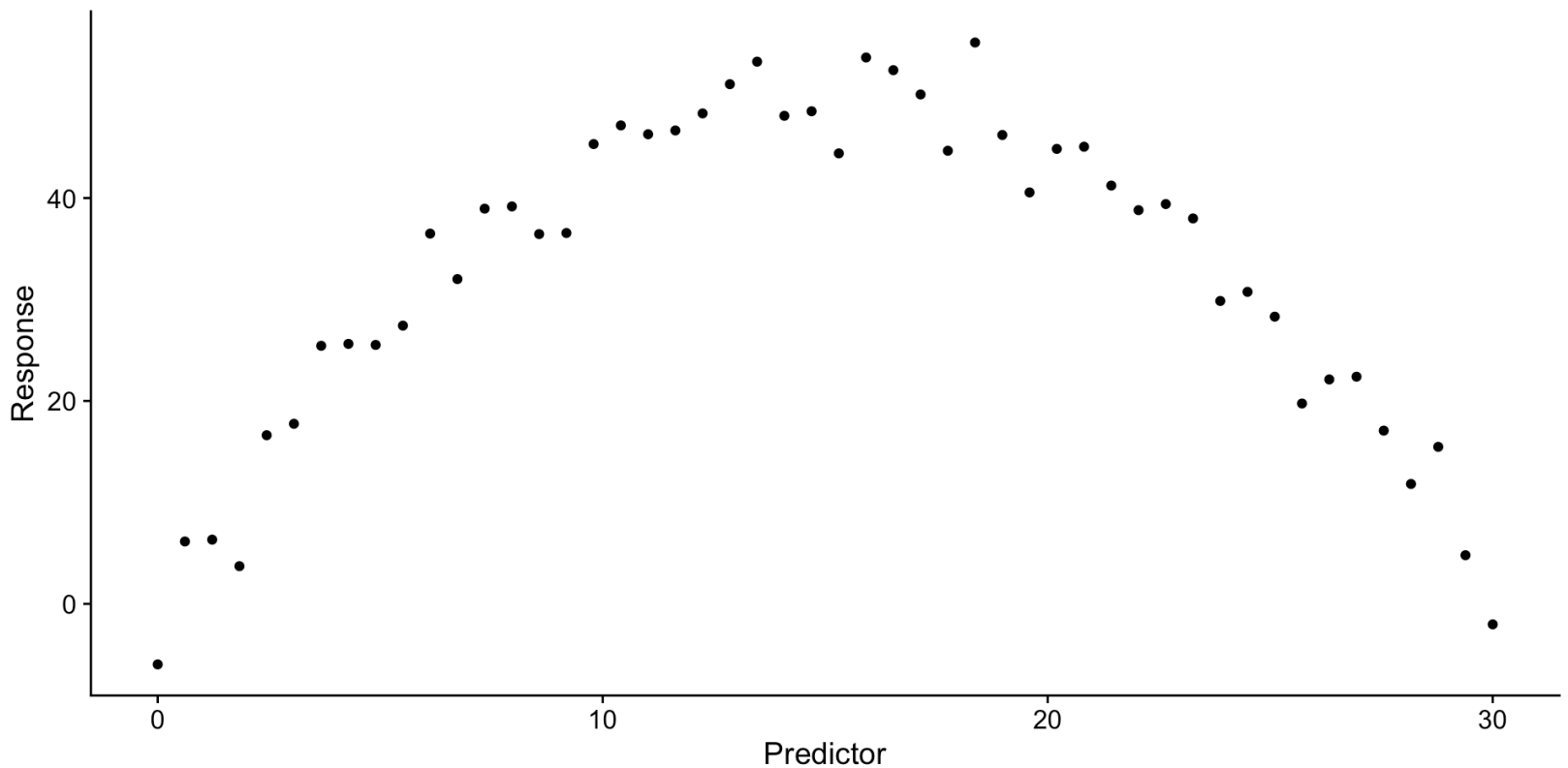


Examples: growth of bacteria, disease spread, species growth.

Curvilinear relationship

Response variable changes in a variety of ways as the predictor variable changes.

$$y = a + bx + cx^2 + dx^3 + ...$$



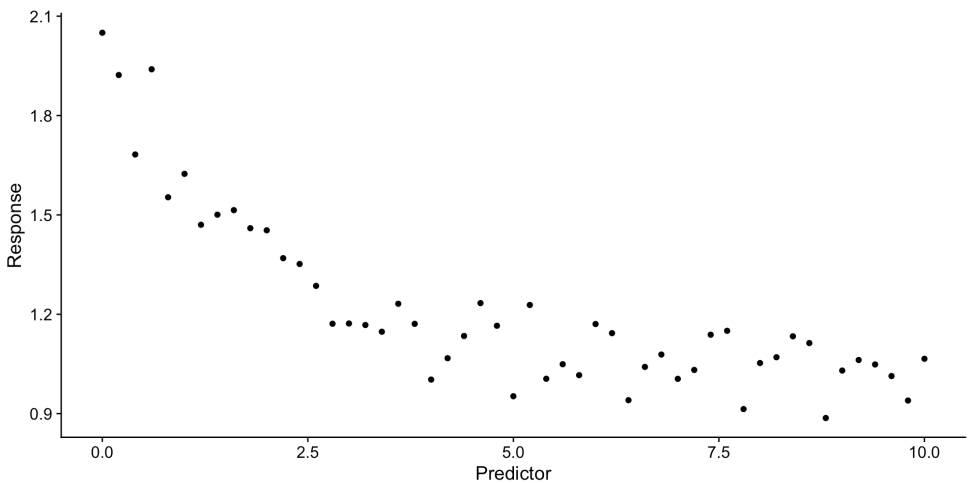
Examples: food intake, drug dosage, exercise.

Transformations

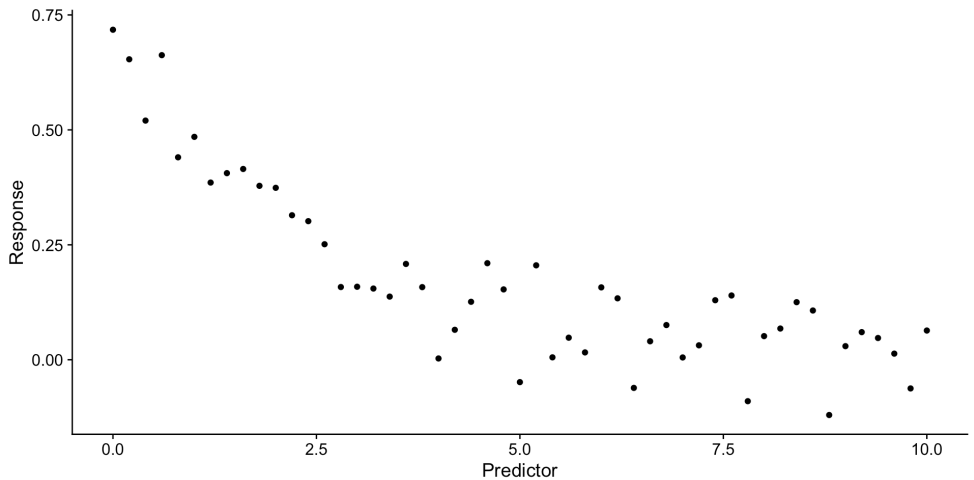
> How far can we go?

Transformations: Exponential decay

Before transformation

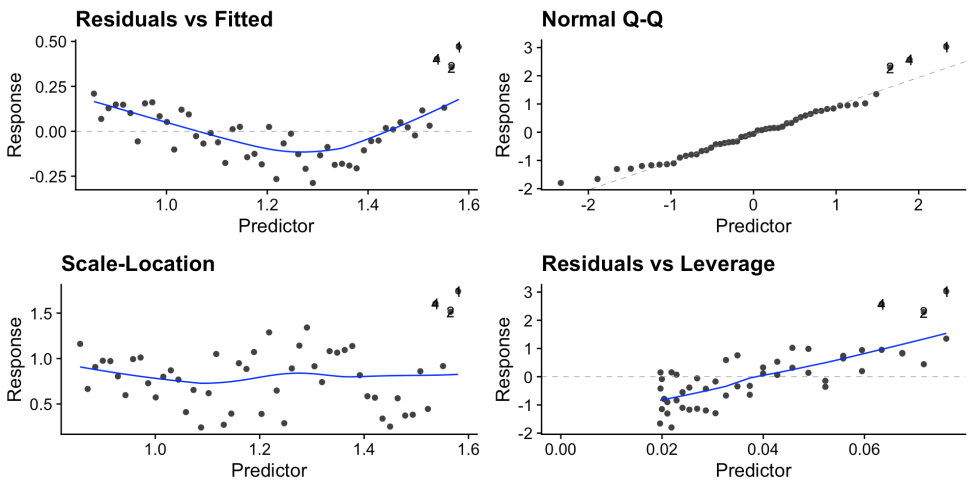


After $\log_e$ transform

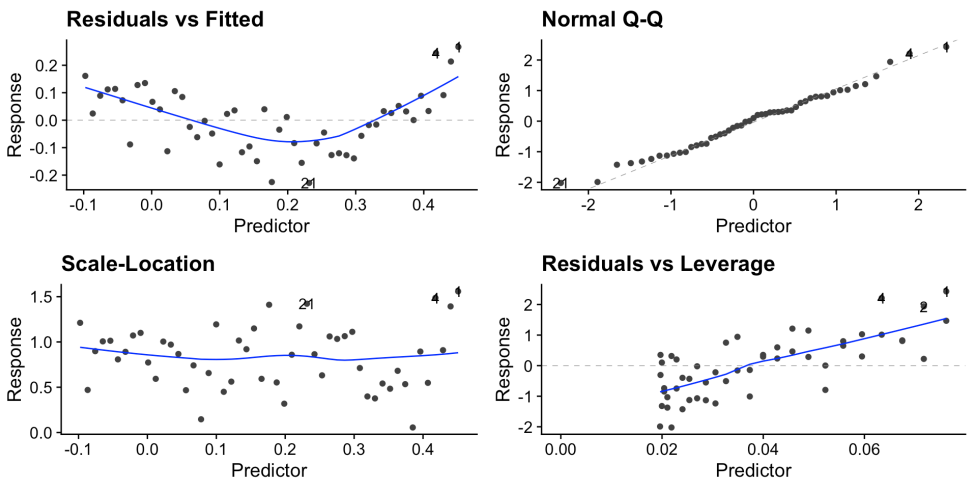


Transformations: Exponential decay

Before transformation

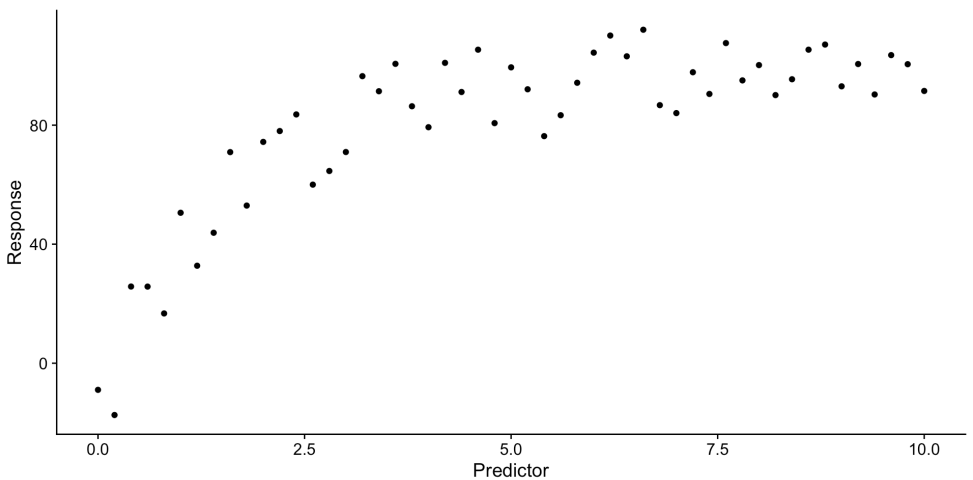


After $\log_e$ transform

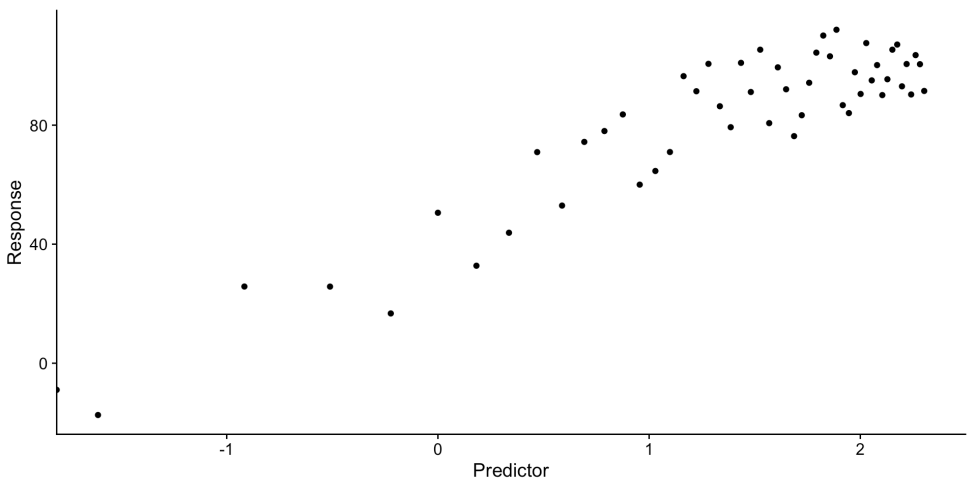


Transformations: Asymptotic relationship

Before transformation

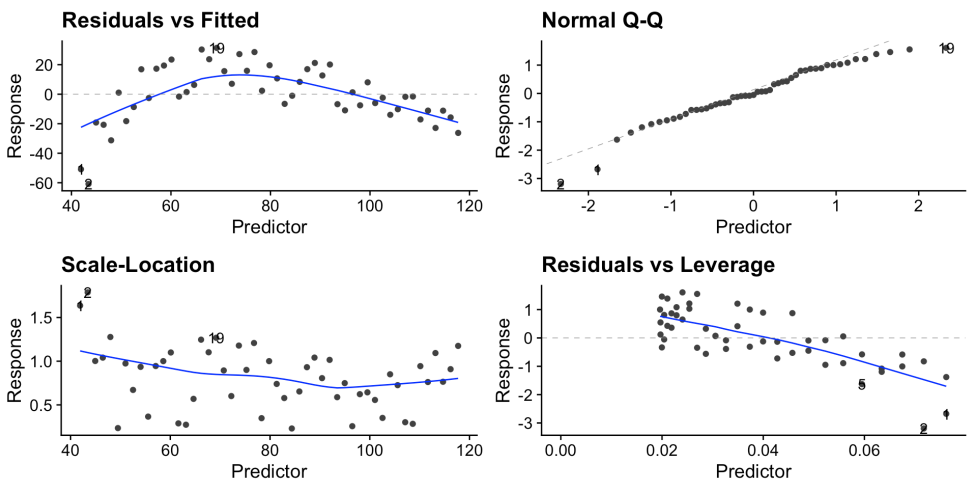


After $\log_e$ transform

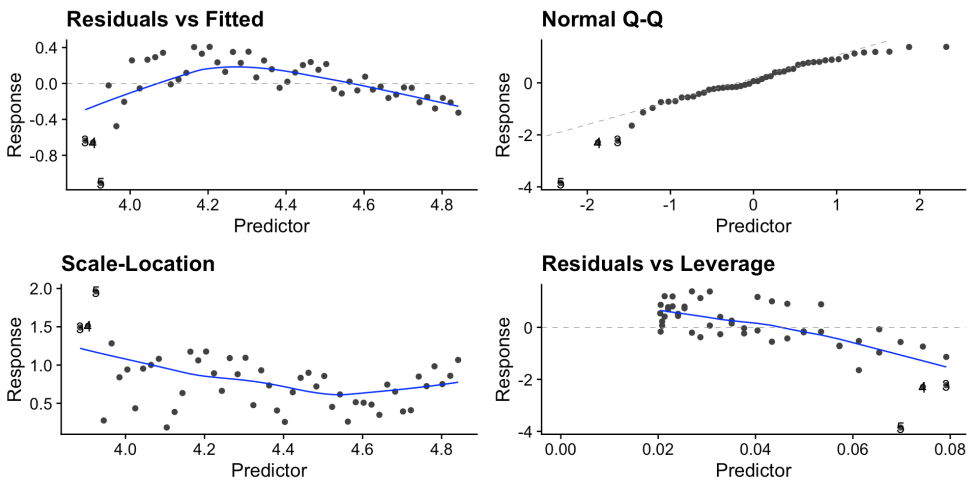


Transformations: Asymptotic relationship

Before transformation

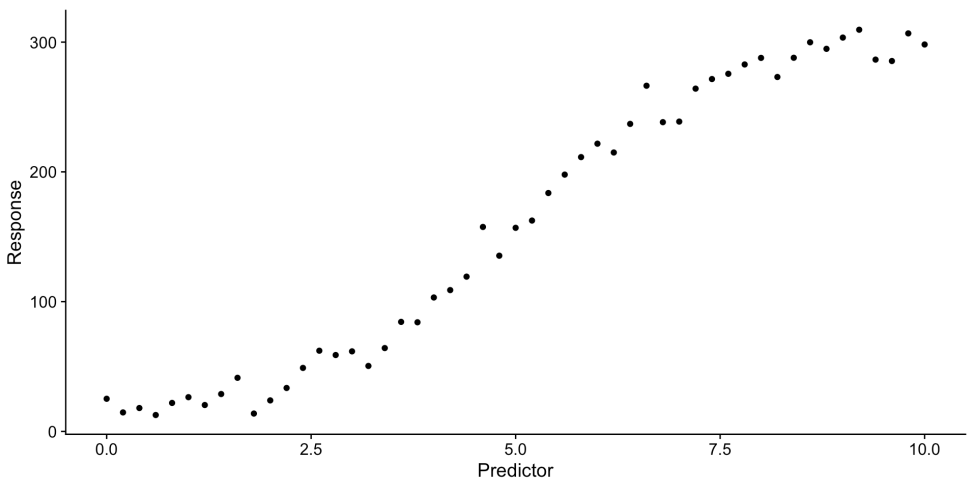


After $\log_e$ transform

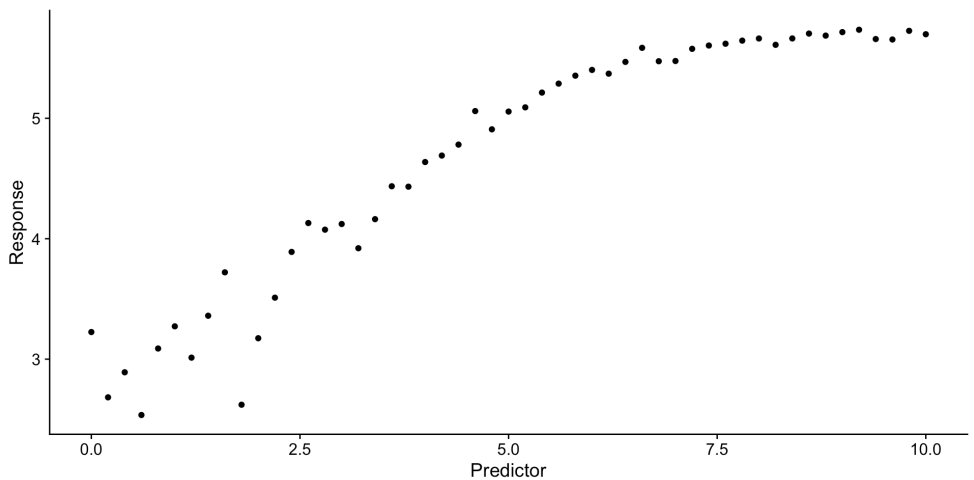


Transformations: Logistic relationship

Before transformation

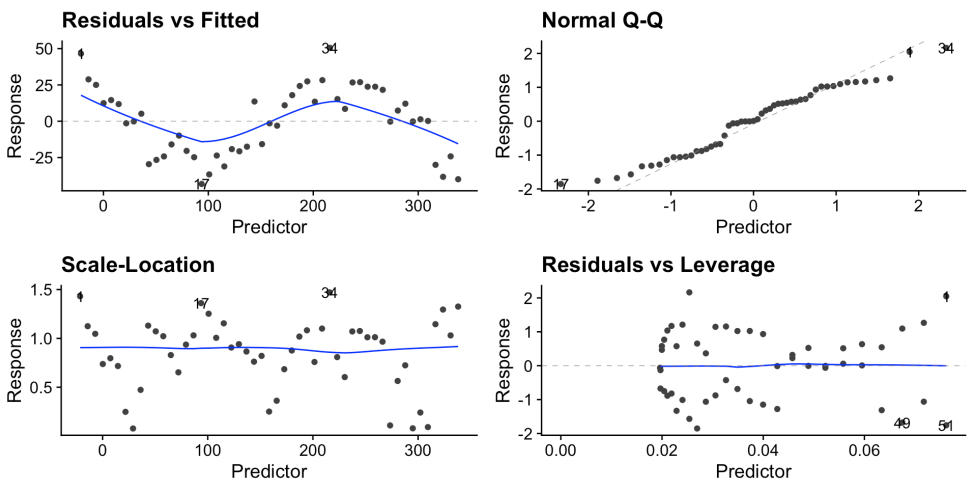


After $\log_e$ transform

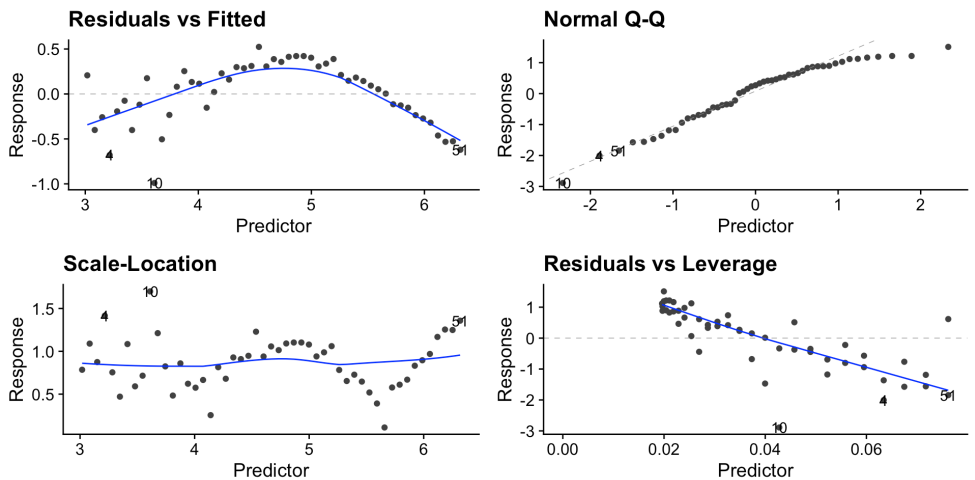


Transformations: Logistic relationship

Before transformation

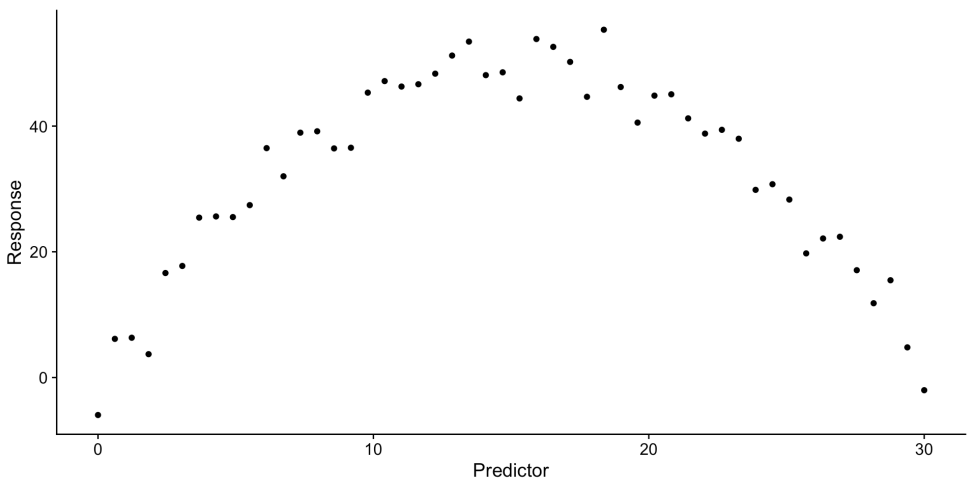


After $\log_e$ transform

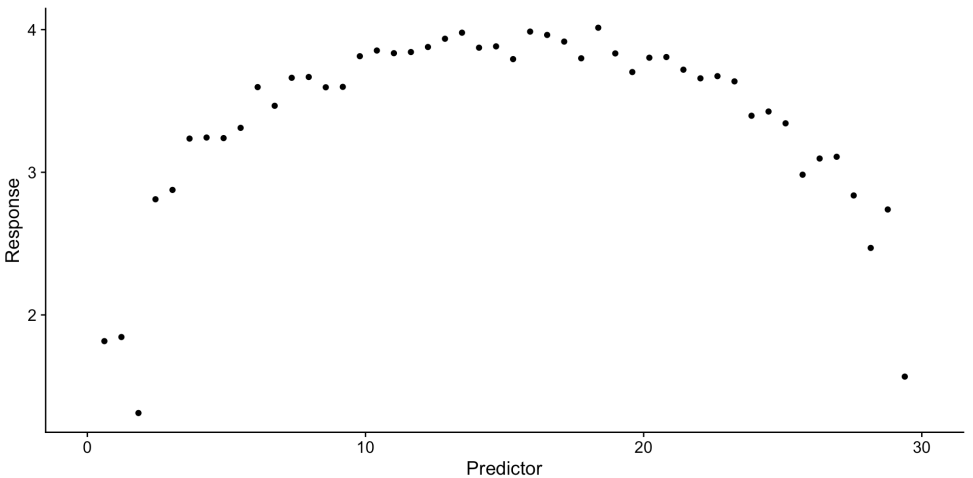


Transformations: Curvilinear relationship

Before transformation

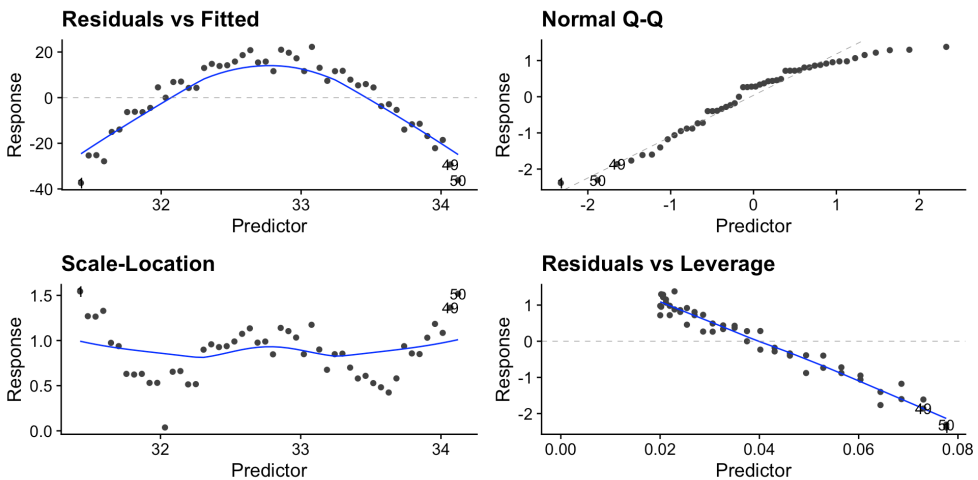


After $\log_e$ transform

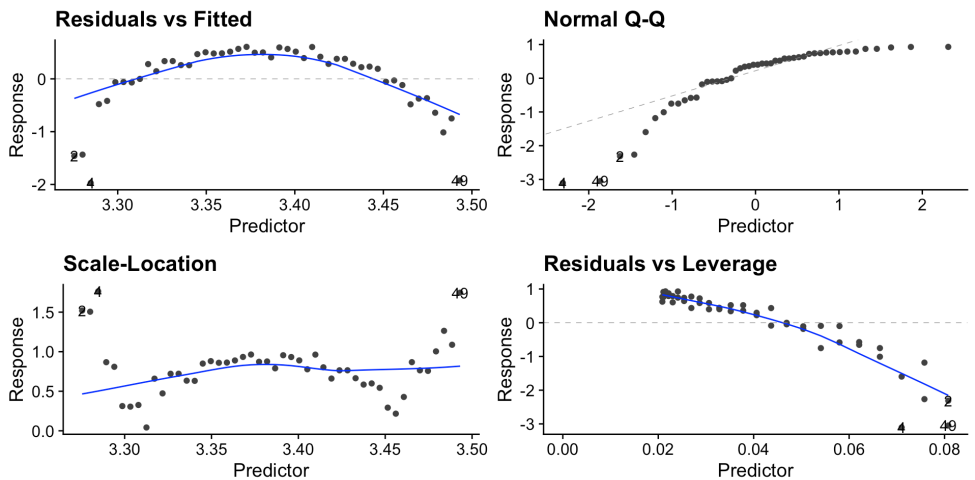


Transformations: Curvilinear relationship

Before transformation



After $\log_e$ transform



Did the transformations work?

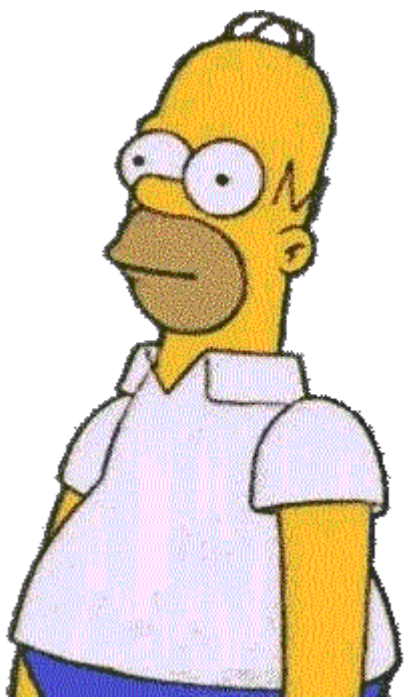
- To a *certain* extent…
- **Problems**:
  - Relationships typically do not meet the linear assumption, but seem “ok” for other assumptions.
  - Poor fit to the data (over or underfitting in some areas).
  - Difficult to interpret the results.

Nonlinear regression

- A way to model complex (nonlinear) relationships.
  - ➠ i.e. phenomena that arise in the natural and physical sciences e.g. biology, chemistry, physics, engineering.
- At least *one* predictor is not linearly related to the response variable.

Performing nonlinear regression

> "You need to know a bit of *calculus*."



Wait!

- It's easier than you think in R.
- **Polynomial regression**: still linear in the parameters and a good place to start.
- **Nonlinear regression**: use the `nls()` function to fit the following nonlinear models:
  - Exponential growth
  - Exponential decay
  - Logistic

Polynomial regression

> A special case of multiple linear regression used to model nonlinear relationships.

Model

$$Y_i = \beta_0 + \beta_1 x_i + \beta_2 x_i^2 + ... + \beta_k x_i^k + \epsilon_i$$

where k is the degree of the polynomial.

- The model is still linear in the parameters β and can be fitted using least squares.
- Instead of multiple predictors, we have multiple *terms* of the same predictor.
- Can still be fit using `lm()`.

Adding polynomial terms

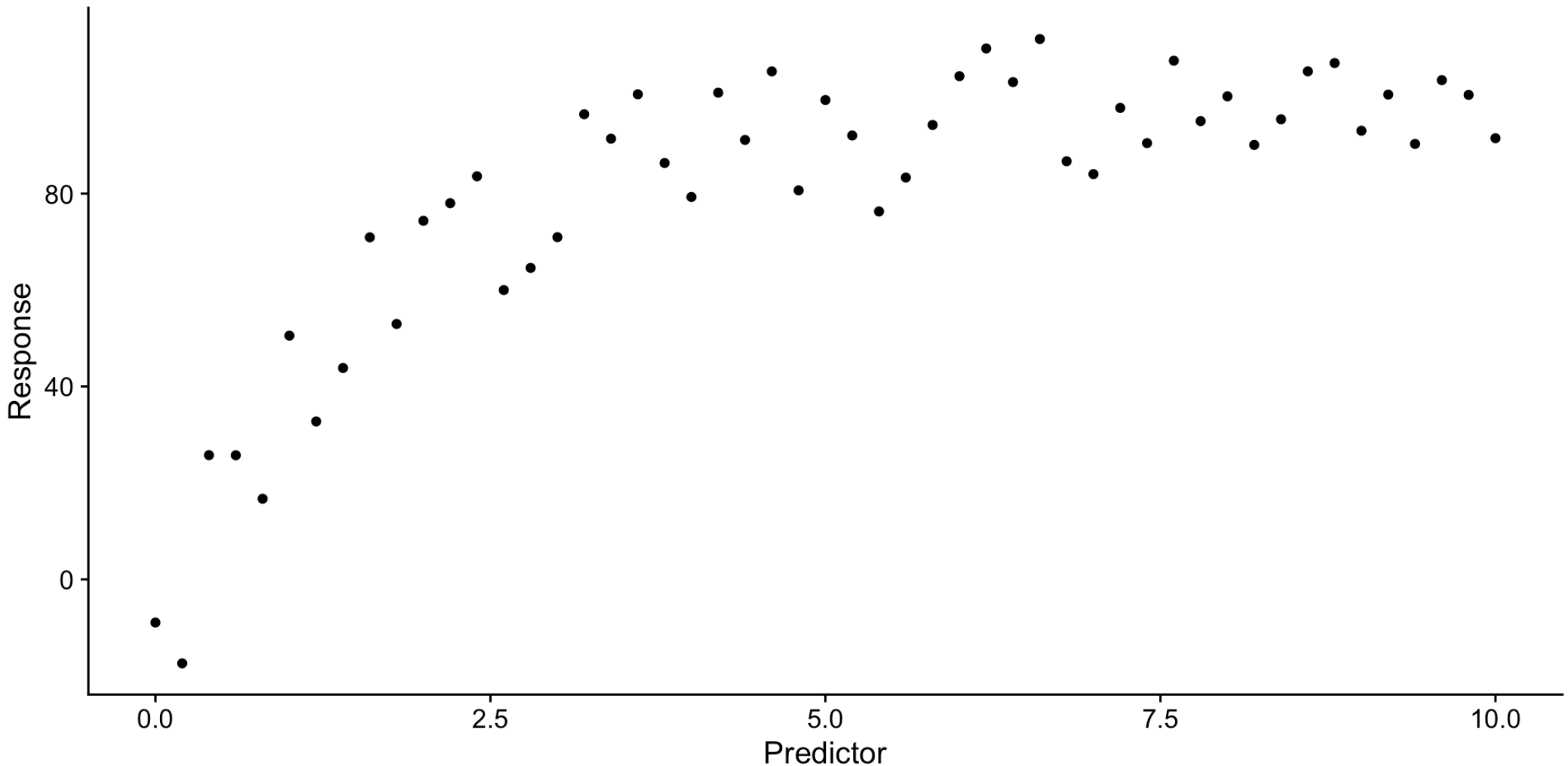
- Linear: $y = \beta_0 + \beta_1 x$
- Quadratic: $y = \beta_0 + \beta_1 x + \beta_2 x^2$
- Cubic: $y = \beta_0 + \beta_1 x + \beta_2 x^2 + \beta_3 x^3$
- Each level increases the power of the predictor by 1.

Polynomial fitting

Using the `asymptotic` data

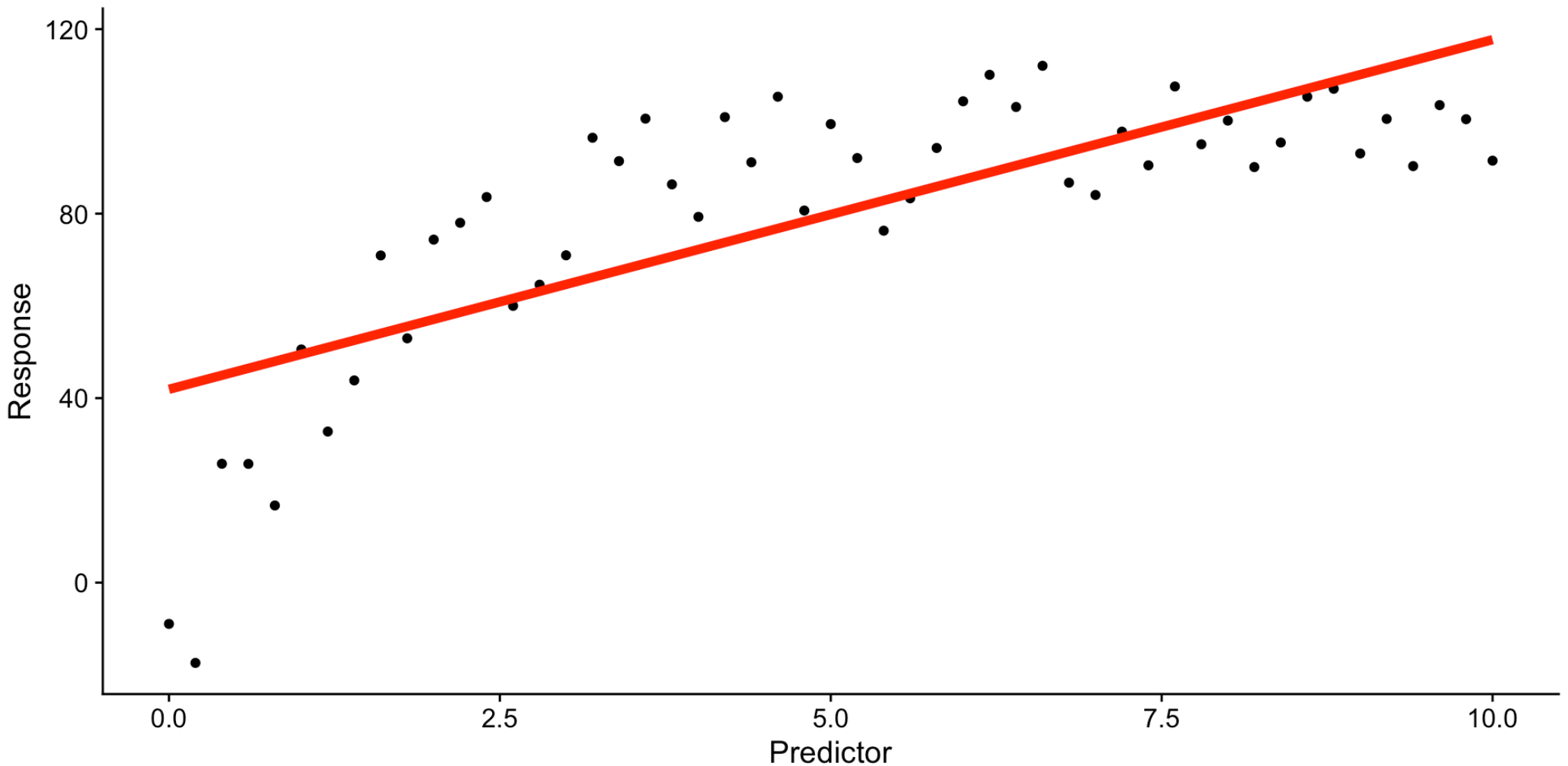
The data

See Slide 11 for the relationship and mathematical expression.



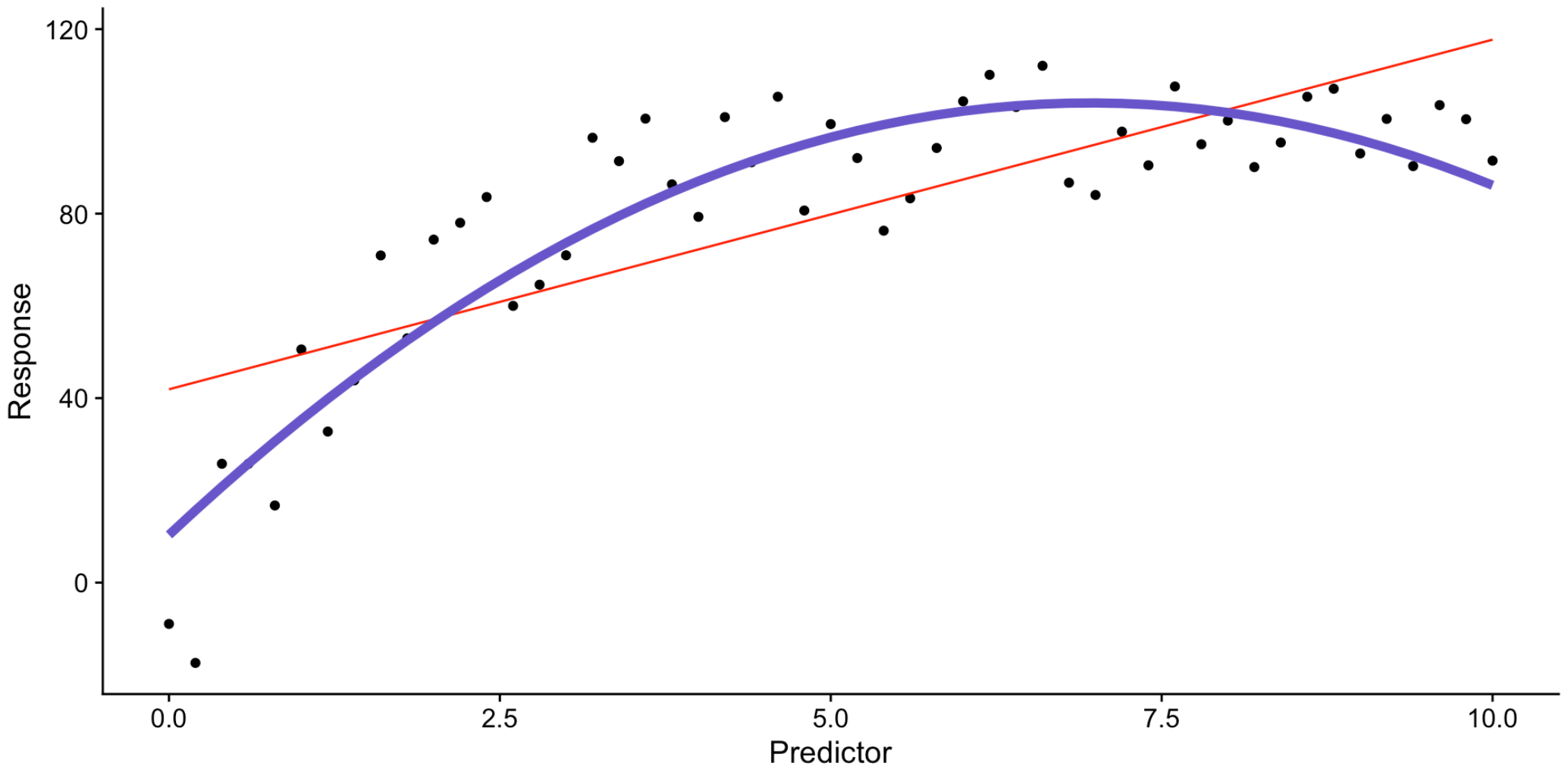
Fitting the model (linear)

$$Y_i = \beta_0 + \beta_1 x_i + \epsilon_i$$



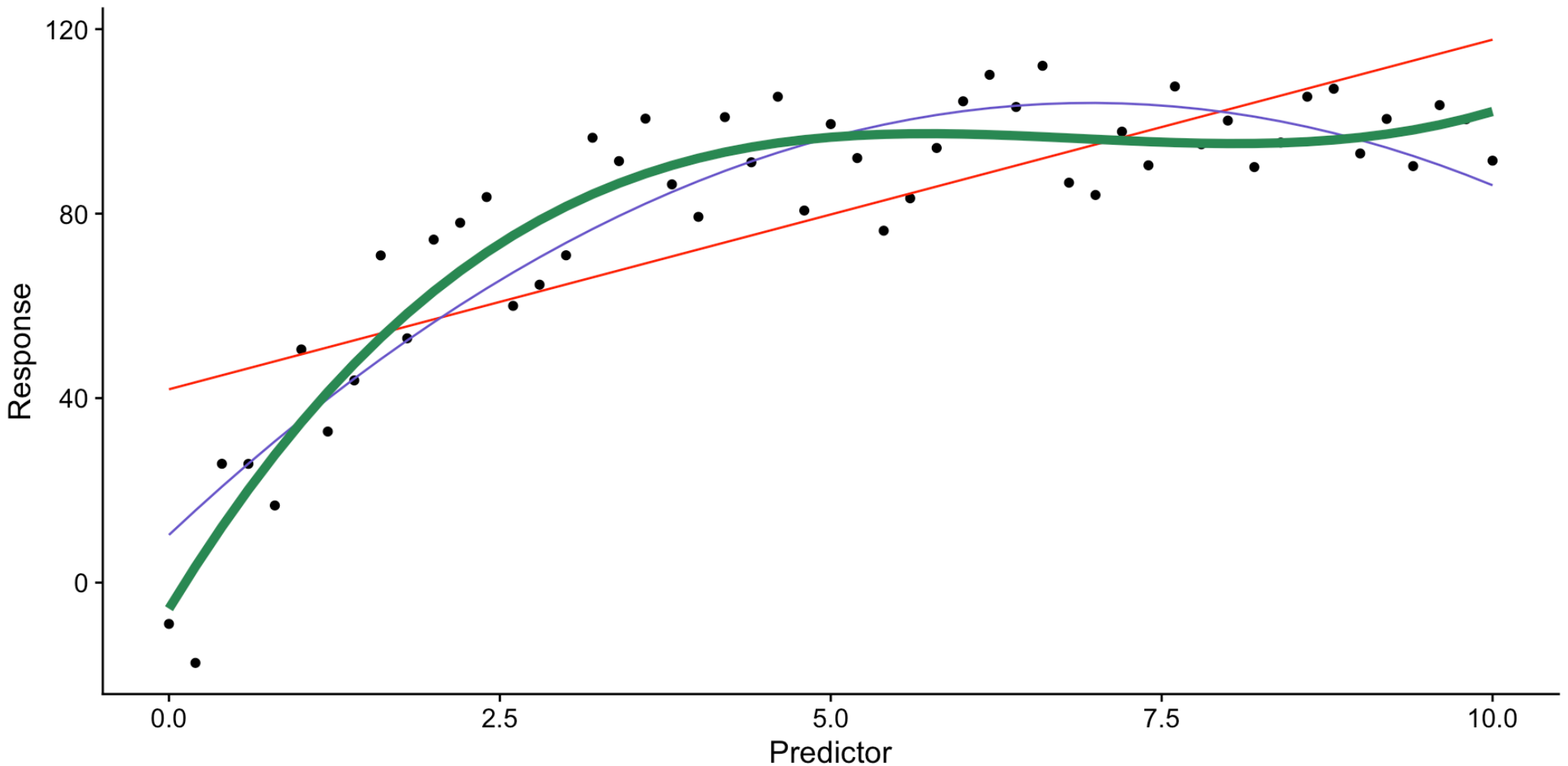
Fitting the model (poly order 2)

$$Y_i = \beta_0 + \beta_1 x_i + \beta_2 x_i^2 + \epsilon_i$$



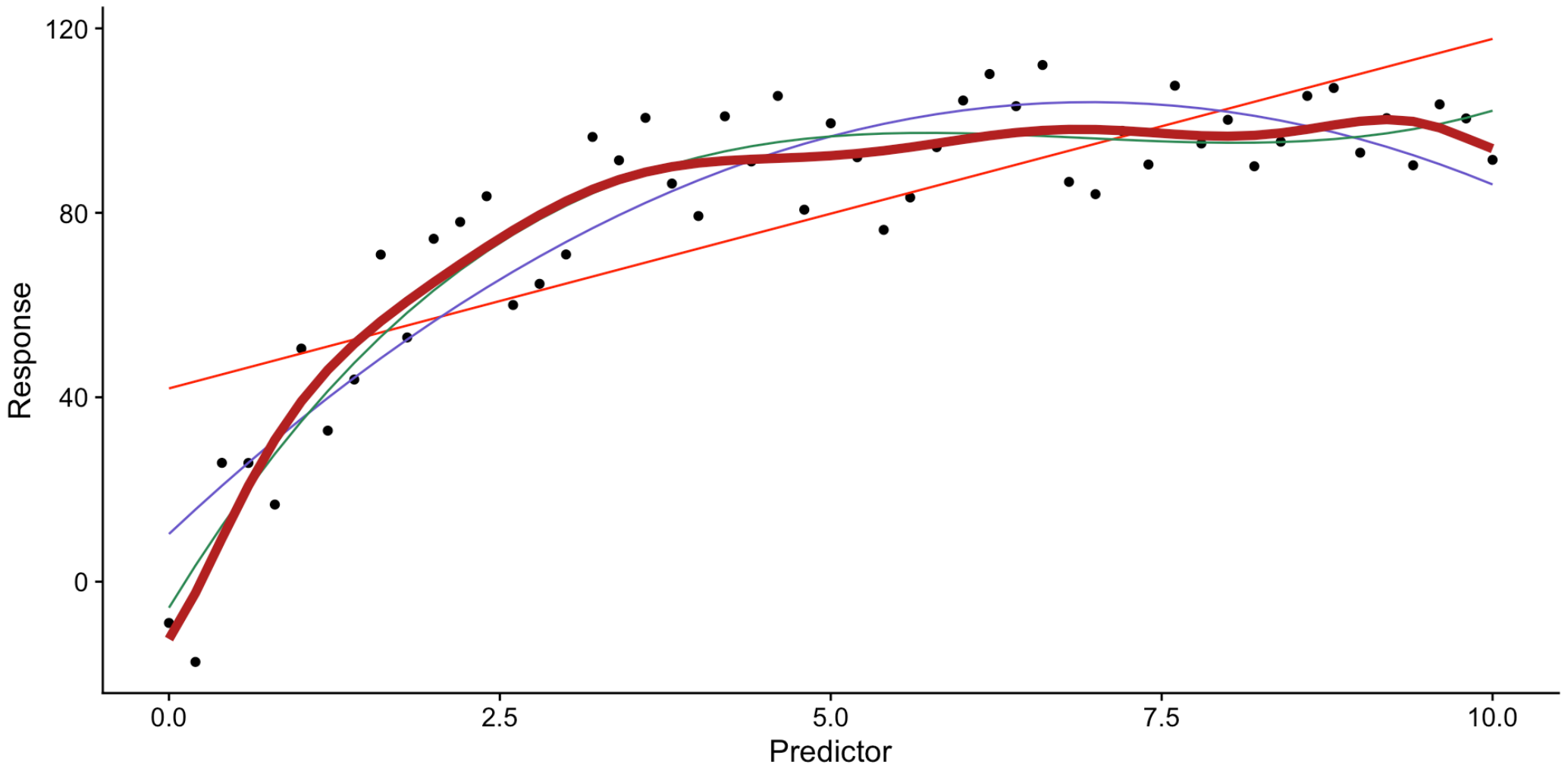
Fitting the model (poly order 3)

$$Y_i = \beta_0 + \beta_1 x_i + \beta_2 x_i^2 + \beta_3 x_i^3 + \epsilon_i$$



Fitting the model (poly order 10)

$$Y_i = \beta_0 + \beta_1 x_i + \beta_2 x_i^2 + ... + \beta_1 0 x_i^{10} + \epsilon_i$$



Limitations

- Meaning of the coefficients is not always clear.
- Extrapolation can be *dangerous*.
- Extra terms can lead to overfitting and are difficult to interpret:

```
Call:
lm(formula = response ~ poly(predictor, 10), data = asymptotic)

Residuals:
     Min       1Q   Median       3Q      Max 
-17.1659  -8.6908  -0.0494   8.8003  16.4012 

Coefficients:
                     Estimate Std. Error t value Pr(>|t|)    
(Intercept)            79.818      1.552  51.426  < 2e-16 ***
poly(predictor, 10)1  159.368     11.084  14.378  < 2e-16 ***
poly(predictor, 10)2 -106.939     11.084  -9.648 5.37e-12 ***
poly(predictor, 10)3   48.570     11.084   4.382 8.28e-05 ***
poly(predictor, 10)4  -19.411     11.084  -1.751   0.0876 .  
poly(predictor, 10)5    1.193     11.084   0.108   0.9148    
poly(predictor, 10)6   -2.769     11.084  -0.250   0.8040    
poly(predictor, 10)7   -1.343     11.084  -0.121   0.9042    
poly(predictor, 10)8   -4.009     11.084  -0.362   0.7195    
poly(predictor, 10)9   -2.851     11.084  -0.257   0.7984    
poly(predictor, 10)10   5.769     11.084   0.520   0.6056    
---
```

But:

- Easy to fit: just add polynomial terms to the model.
- Simple to perform: use `lm()`.

Nonlinear fitting

Fitting a nonlinear model

If you have some understanding of the underlying relationship (e.g. mechanistic process) between the variables, you can fit a nonlinear model.

Mathematical expression

$$Y_i = f(x_i, \beta) + \epsilon_i$$

where $f(x_i, \beta)$ is a nonlinear function of the parameters β.

- Y_i is the continuous response variable.
- x_i is the vector of predictor variables.
- β is the vector of unknown parameters.
- ϵ_i is the random error term (residual error).

Assumptions

Like the linear model, the nonlinear model assumes:

- Error terms are normally distributed (**Normality**).
- Error terms are independent (**Independence**).
- Error terms have constant variance (**Homoscedasticity**).

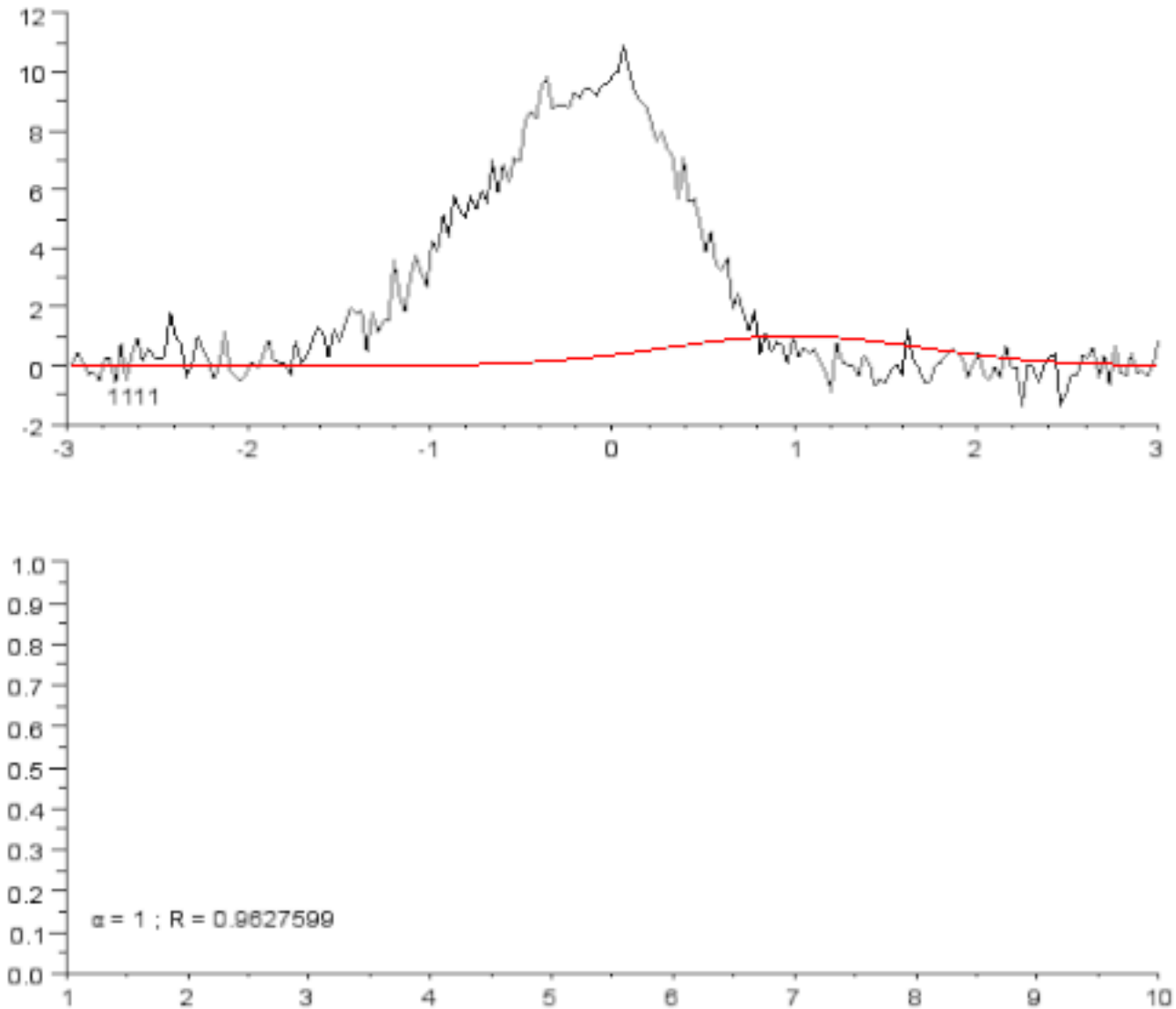
Basically:

$$\epsilon_i \sim N(0, \sigma^2)$$

Like all other models we have seen, we focus on the residuals to assess the model fit, since the residuals are the only part of the model that is random.

Estimating the model parameters

- The parameters are estimated using the **method of least squares**.
- For nonlinear models, a nonlinear optimization algorithm is used to find the best fit, rather than ordinary least squares, e.g.:
  - ➟ Gauss-Newton algorithm
  - ➟ Levenberg-Marquardt algorithm
- This can only be performed iteratively and depends on a “best guess” of the parameters *as a start*.
  - ➟ **i.e. we need to provide a starting point for a nonlinear least squares algorithm to begin**.



Source: Wikipedia

Implementation

use `nls()` function in R.

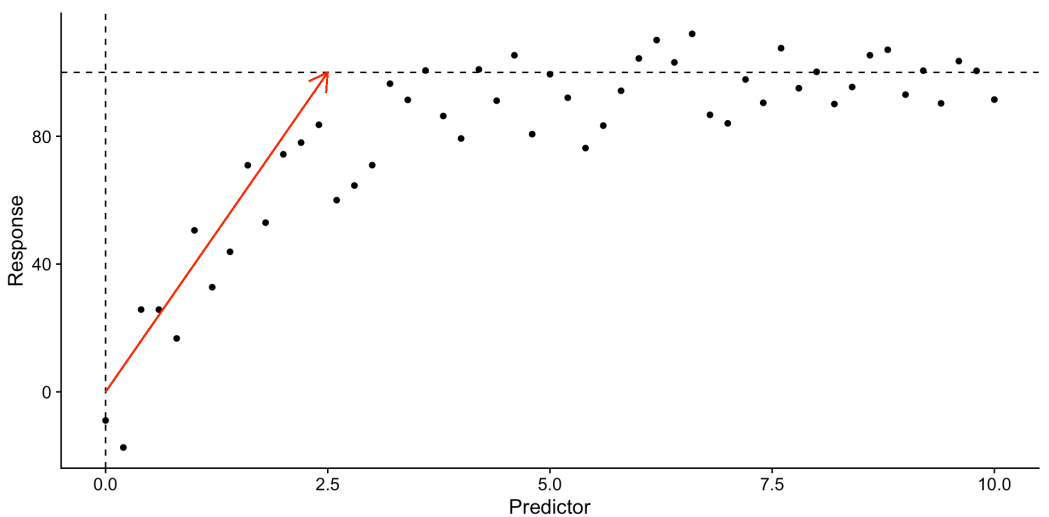
- `formula`: a formula object, with the response variable on the left of a ~ operator, and the predictor variable(s) on the right.
- `data`: a data frame containing the variables in the model.
- `start`: a named list of starting values for the parameters in the model.

Example: Fitting an asymptotic model

Finding starting values

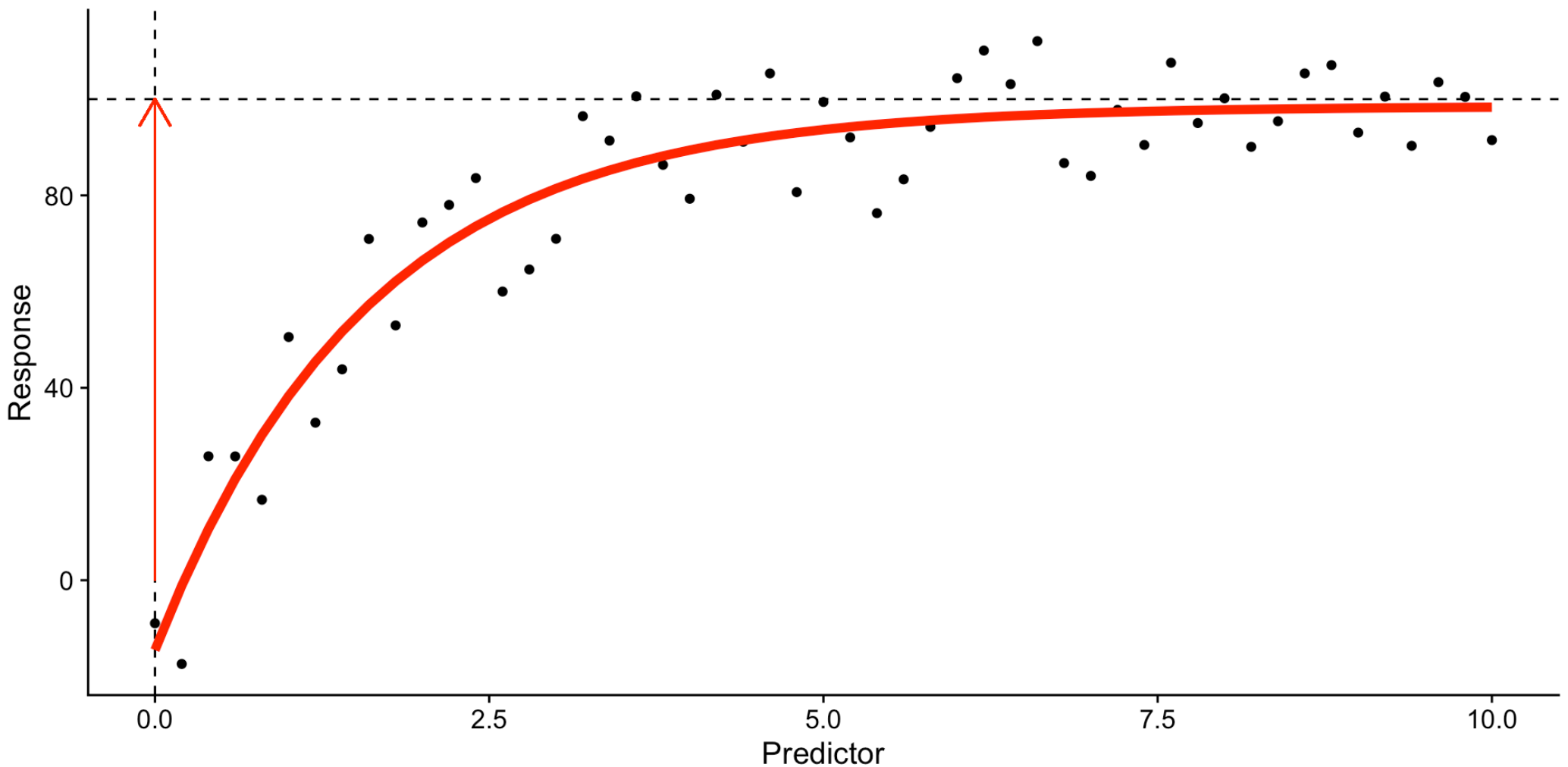
$$y = a + b(1 - e^{-cx})$$

- a is value of y when $x = 0$.
- b is the upper limit: the maximum value of y.
- c is the rate of change: the rate at which y approaches the upper limit.

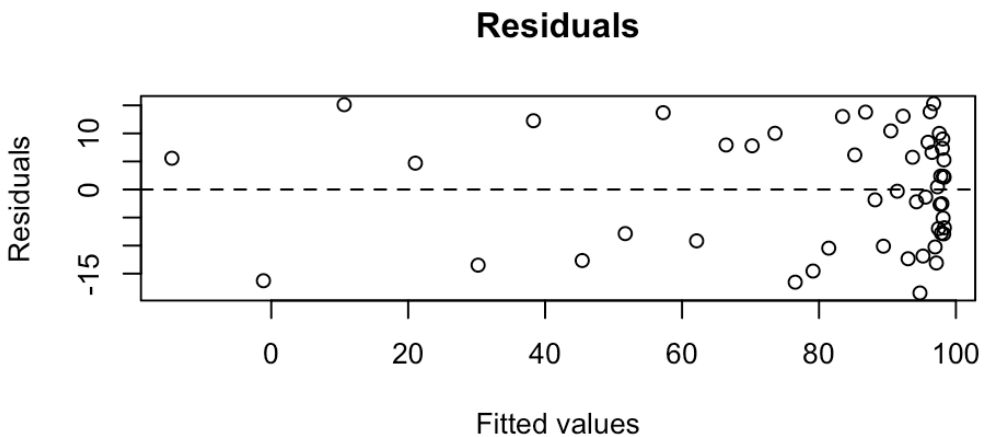


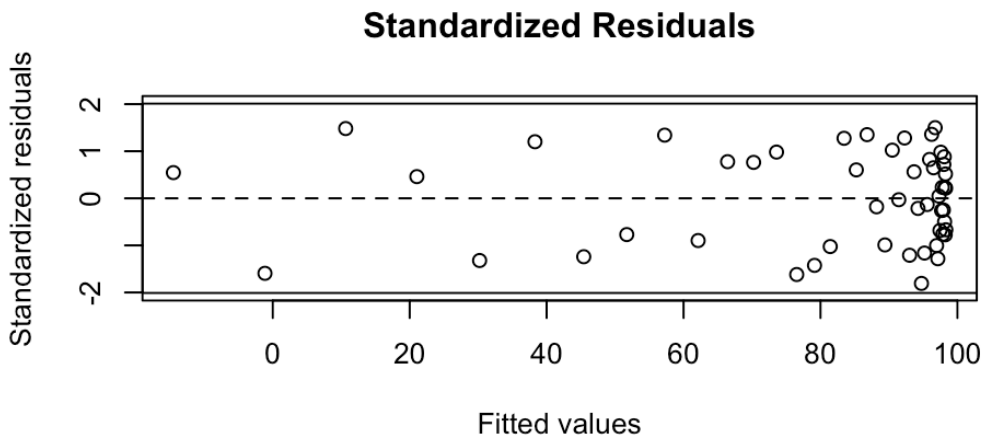
First guess

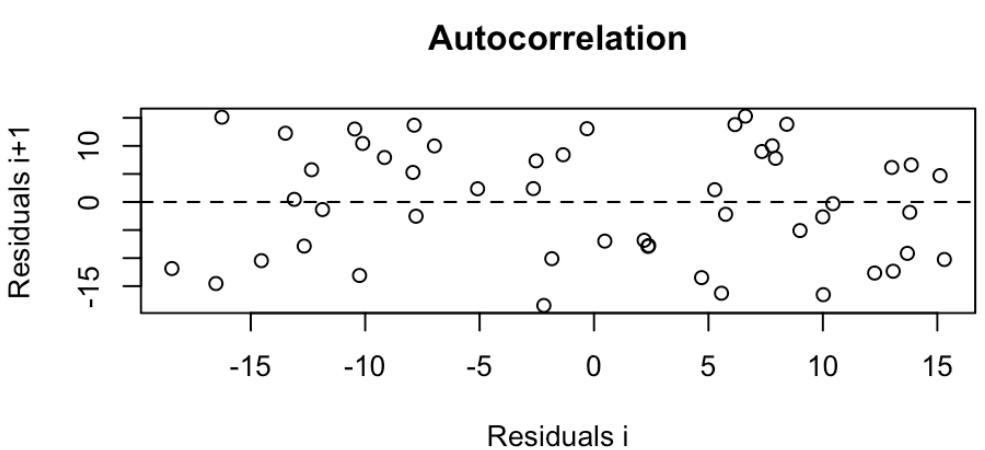
$$y = a + b(1 - e^{-cx})$$

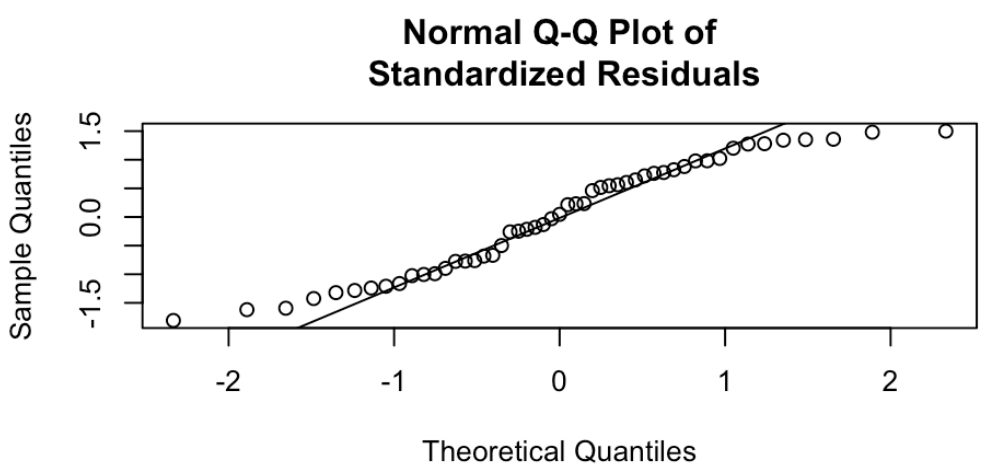


Check the fit









Interpretation

```
Formula: response ~ a + b * (1 - exp(-c * predictor))

Parameters:
   Estimate Std. Error t value Pr(>|t|)
a -14.51755    6.64162  -2.186   0.0337 *
b 113.03797    6.44716  17.533  < 2e-16 ***
c   0.62962    0.07142   8.816 1.32e-11 ***
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

Residual standard error: 10.21 on 48 degrees of freedom

Number of iterations to convergence: 4
Achieved convergence tolerance: 1.178e-06
```

- The model is significant since the p-value is less than 0.05 for all parameters.
- The parameterised model is:

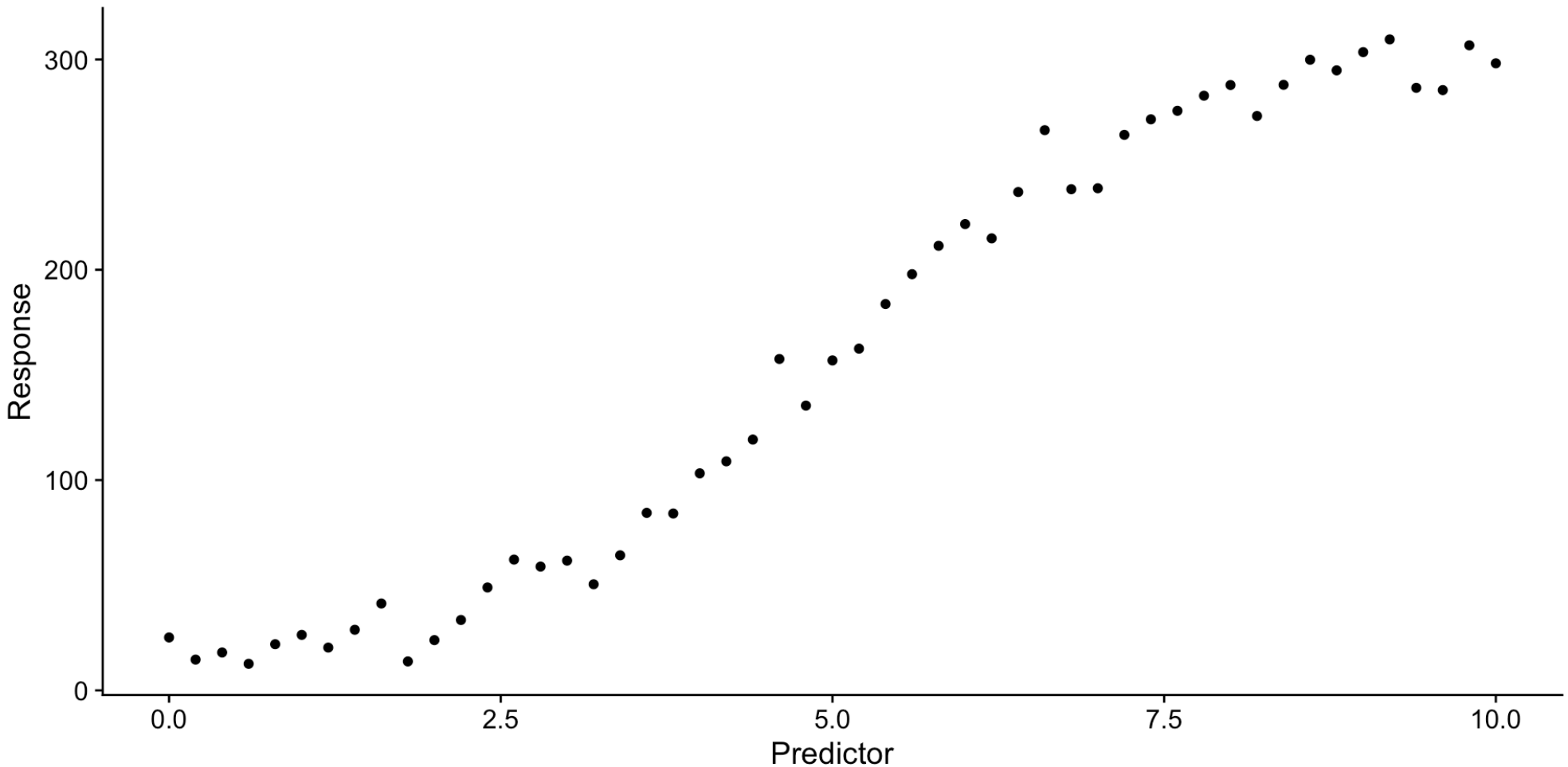
$$y = -14.5 + 113.04(1 - e^{-0.63x})$$

The R-square value is not reported for nonlinear models as the sum of squares is not partitioned into explained and unexplained components.

Another example: Fitting a logistic model

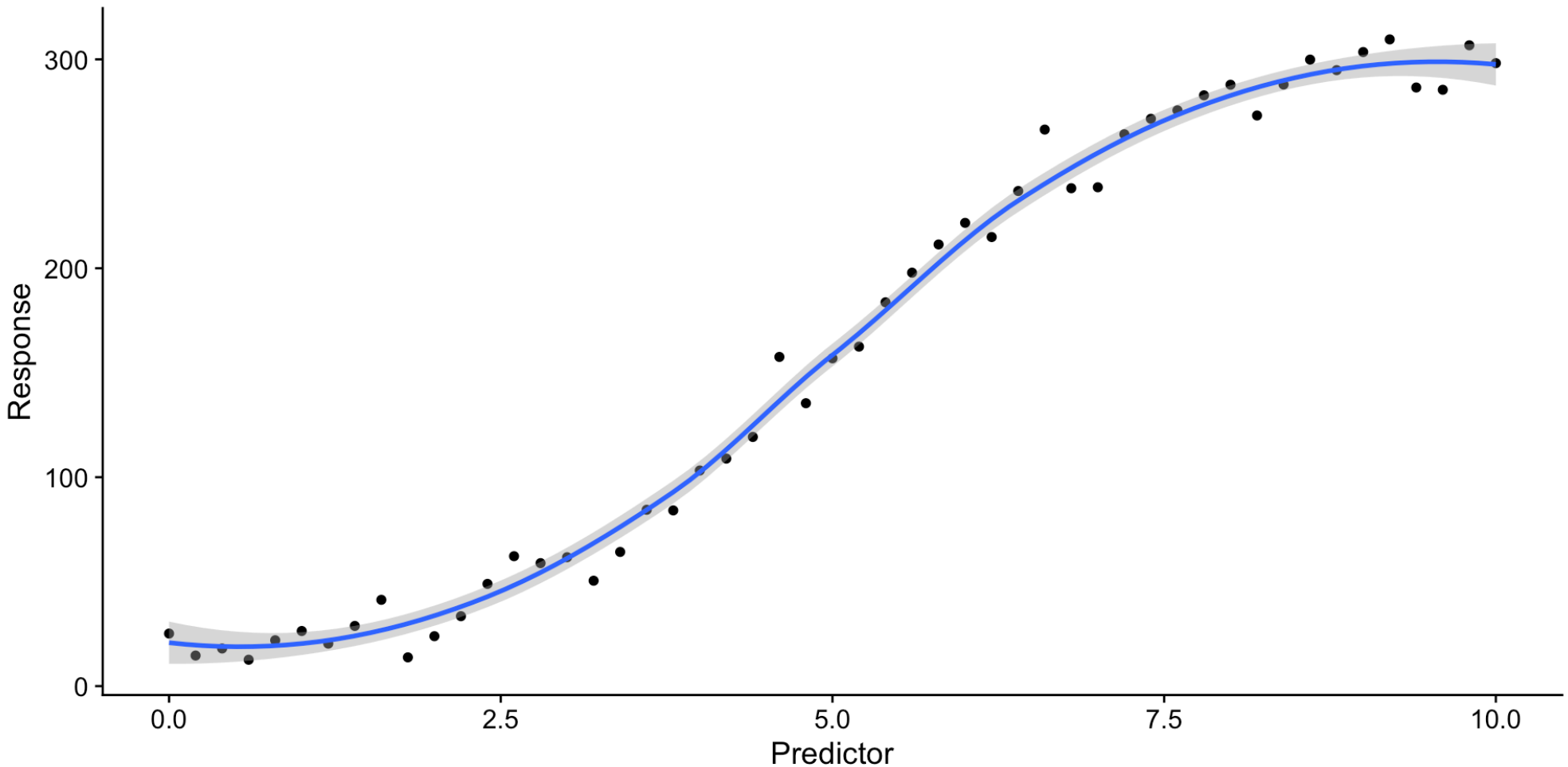
Recap on logistic relationship

$$y = c + \frac{d - c}{1 + e^{-b(x-a)}}$$



Recap on logistic relationship

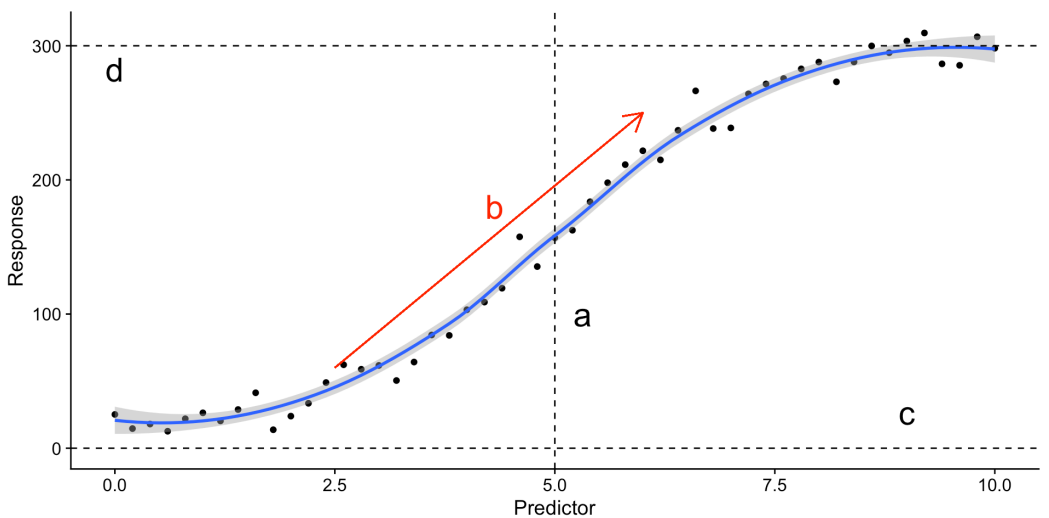
$$y = c + \frac{d - c}{1 + e^{-b(x-a)}}$$



Finding the starting values

$$y = c + \frac{d - c}{1 + e^{-b(x-a)}}$$

- c is the lower limit: the minimum value of y.
- d is the upper limit: the maximum value of y.
- a is the value of x when y is halfway between the lower and upper limits.
- b is the rate of change: the rate at which y approaches the upper limit.



Finding the starting values

$$y = c + \frac{d - c}{1 + e^{-b(x-a)}}$$

- c is the lower limit: the minimum value of y.
- d is the upper limit: the maximum value of y.
- a is the value of x when y is halfway between the lower and upper limits.
- b is the rate of change: the rate at which y approaches the upper limit.

NOPE

Automating the process (sort of)

Self-starting functions

- The `nls()` function requires a formula and starting point(s) for the parameters.
    - ➦ *How about starting to nope out…*

Wait!

- Several self-starting functions are available in R that can be used to estimate the starting values.
- These functions are named after the model they fit, e.g. `SSasymp()`, `SSlogis()`, `SSmicmen()`, `SSweibull()`, etc.

> **Important**
>
> We still need to have some understanding of the underlying relationship between the variables to pick the right function.

Revisiting the logistic model

$$y = c + \frac{d - c}{1 + e^{-b(x-a)}}$$

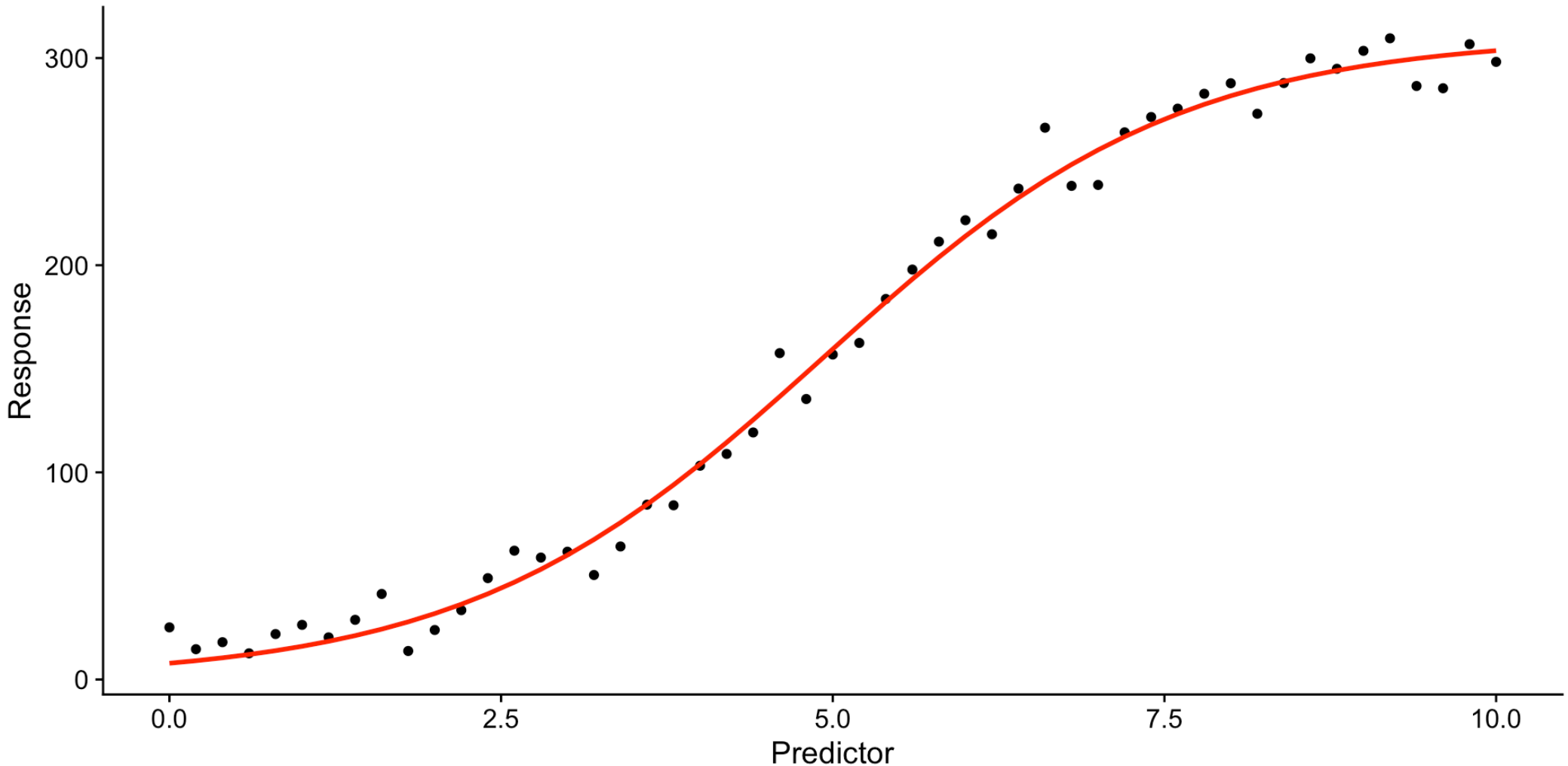
- `input`: the predictor variable.
- `Asym`: the upper limit.
- `xmid`: the value of x when y is halfway between the lower and upper limits.
- `scal`: the rate of change.

The equation ia *different*: see `?SSlogis`:

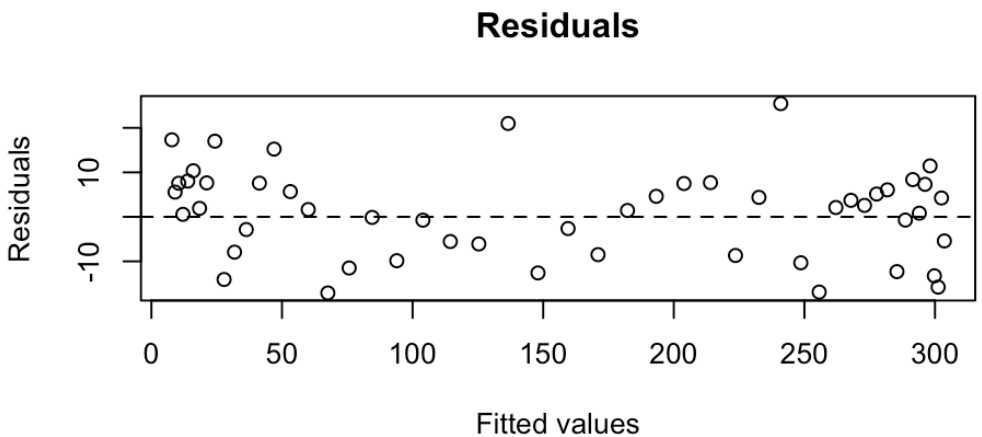
$$y = \frac{Asym}{1 + exp\frac{xmid - input}{scal}}$$

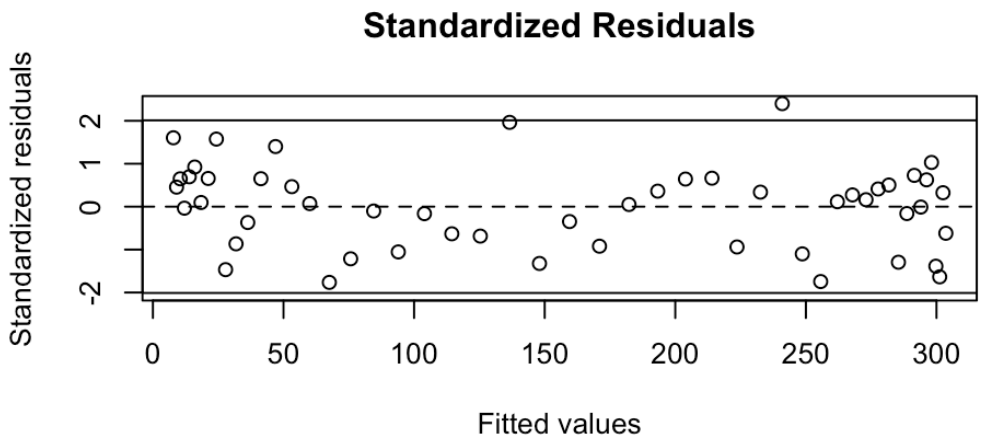
Other than `input`, the other parameters can be left to the function to estimate.

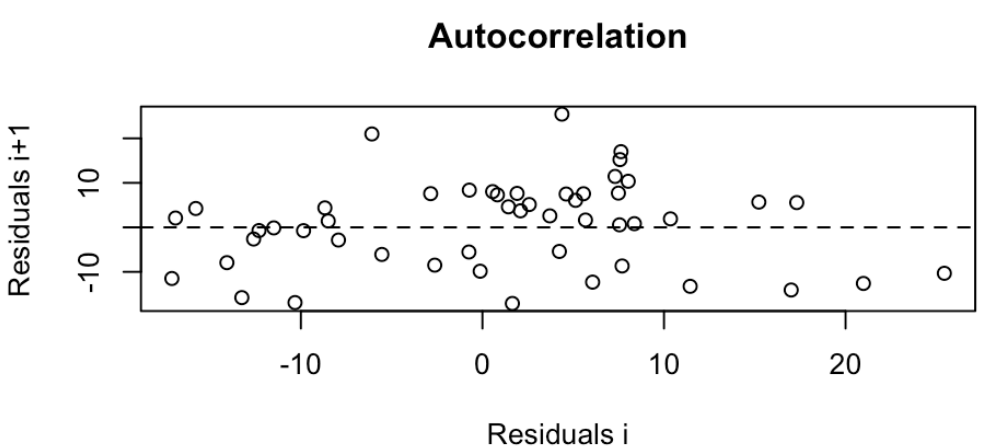
What does the fit look like?

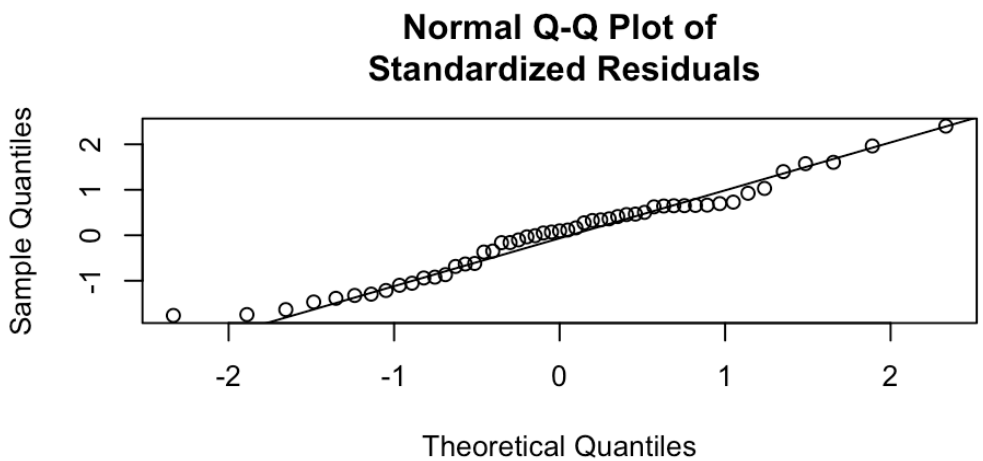


Check the fit









Interpretation

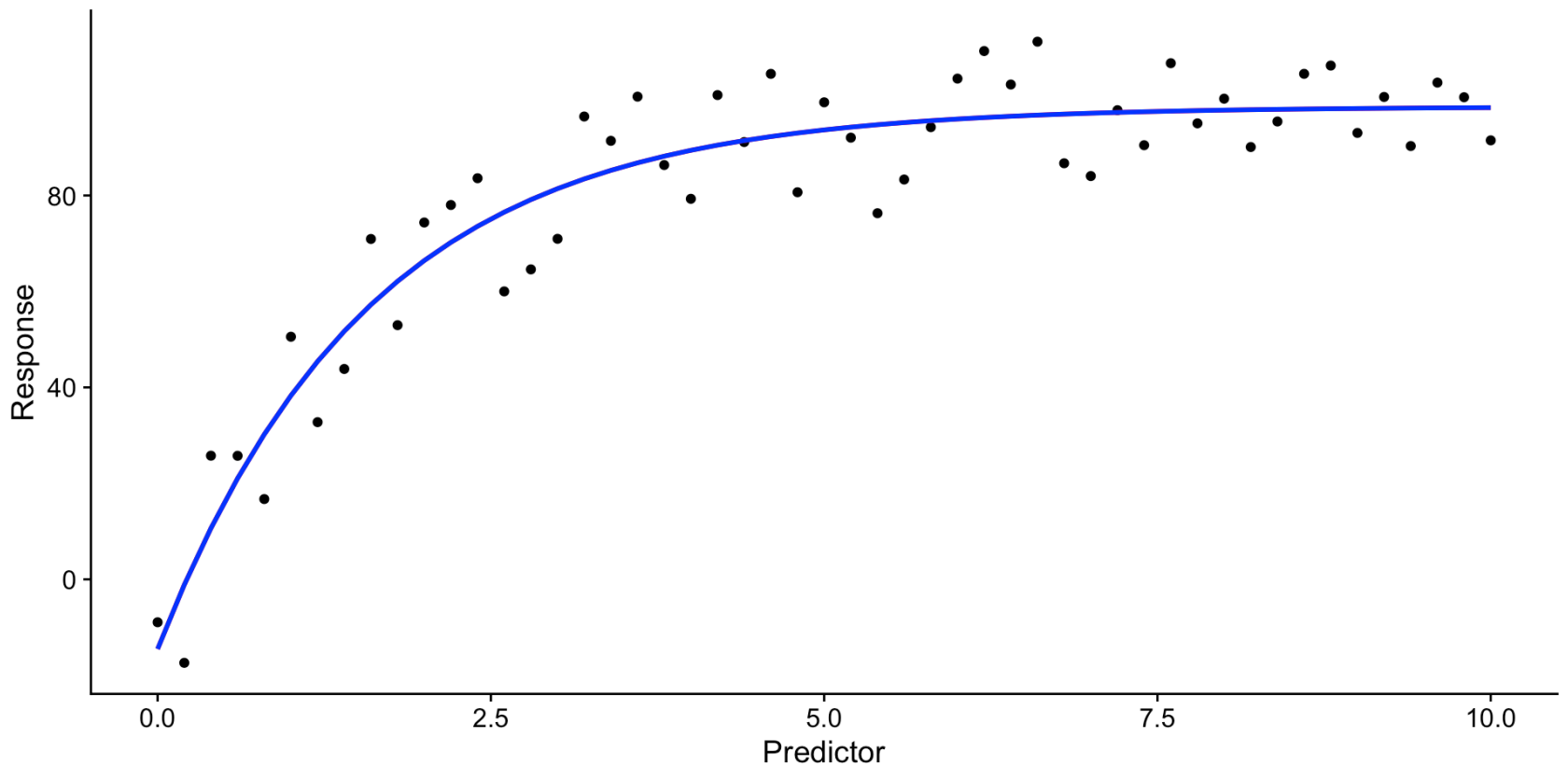
```
Formula: response ~ SSlogis(predictor, Asym, xmid, scal)

Parameters:
      Estimate Std. Error t value Pr(>|t|)
Asym 310.64727    4.62579   67.16   <2e-16 ***
xmid   4.92715    0.07142   68.99   <2e-16 ***
scal   1.34877    0.05418   24.90   <2e-16 ***
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

Residual standard error: 10.22 on 48 degrees of freedom

Number of iterations to convergence: 1
Achieved convergence tolerance: 6.626e-07
```

Back to asymptotic model

Comparing manual fit to self-starting function

The self-starting function for the asymptotic model is `SSasymp()`.

Comparing outputs:



In some cases, the fits are identical, but in others, they are not.

Summary

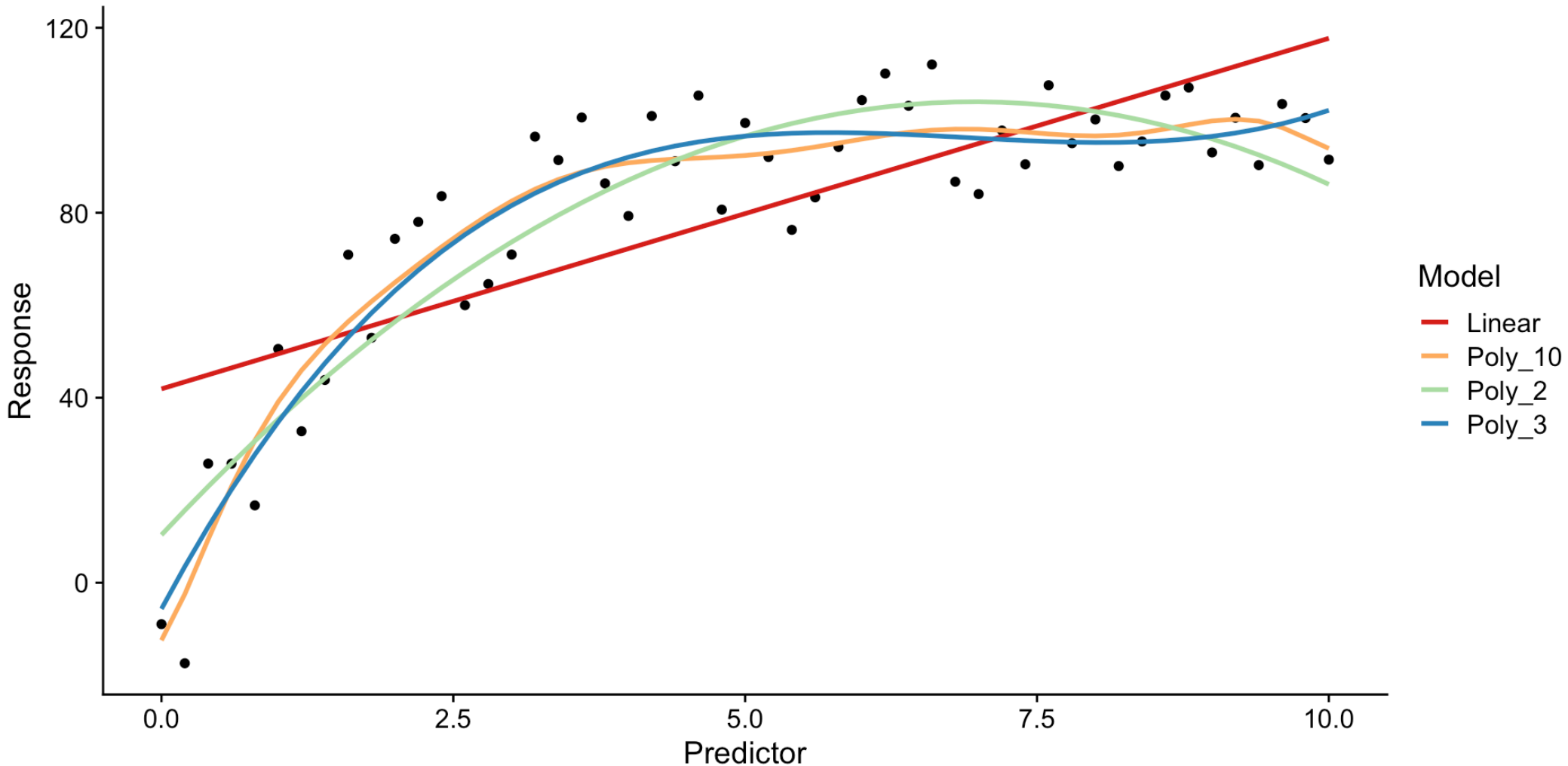
- When fitting a nonlinear model, there are three possible approaches:
  1. **Linearize** the model by transforming the response variable or predictor variable(s):
     - ➟ Fit: easy/difficult
     - ➟ Interpret: difficult
  2. Approximate the model by adding **polynomial** terms:
     - ➟ Fit: easy
     - ➟ Interpret: difficult
  3. Fit the model using a **nonlinear** least squares algorithm:
     - ➟ Fit: difficult
     - ➟ Interpret: easy

- Nonlinear models:
  - ➟ Useful for modelling complex relationships.
  - ➟ Require some understanding of the underlying relationship between the variables, especially asympotic and logistic models.
  - ➟ Most useful when prediction is the goal, since we do not necessarily need to interpret the parameters to assess the model fit.

Bonus: How do we know which model is better?

Note: this is non-examinable content but might be useful for your project.

Example: polynomial regression



Prediction quality

We can use prediction quality metrics to compare the fits.

- Akaike information criterion (AIC) and Bayesian information criterion (BIC).
  - ➟ Useful for comparing model fits.
- root mean squared error (RMSE) and mean absolute error (MAE).
  - ➟ Useful for *assessing the quality* of the fit.

AIC and BIC

Use the `broom` package to extract the AIC and BIC values from the model fits.

```
# A tibble: 4 × 3
  model    AIC   BIC
  <chr>  <dbl> <dbl>
1 linear  453.  459.
2 poly2   409.  416.
3 poly3   392.  402.
4 poly10  402.  425.
```

- The smaller the AIC or BIC, the better the fit compared to other models.
- However, for better performance, **cross-validation is recommended** as it explains how well the model will perform on **new** data, rather than just assessing the fit to the data.

Cross-validation

1. Split the data into training and testing sets.
2. Fit the model to the training set.
3. Predict the response variable using the testing set.
4. Calculate the RMSE or MAE between the predicted and observed values.
5. Repeat for each fold of the data.
6. Average the RMSE or MAE across all folds.
7. The model with the lowest RMSE or MAE is the best model.

Looks like a lot of work, but it's easy in R using the `caret` package.

Performing cross-validation on the polynomial fits

	MAE	RMSE
linear	15.751679	19.76680
quadratic	9.881451	11.93538
cubic	9.741090	11.27664
poly10	11.774449	13.70916

From the results, the cubic model has the lowest RMSE and MAE, so is the best model.

Thanks!