Moving Beyond Linearity

ACTL3142 Statistical Machine Learning for Risk and Actuarial Applications

Slides: https://laub.au/ml

Patrick Laub

Disclaimer

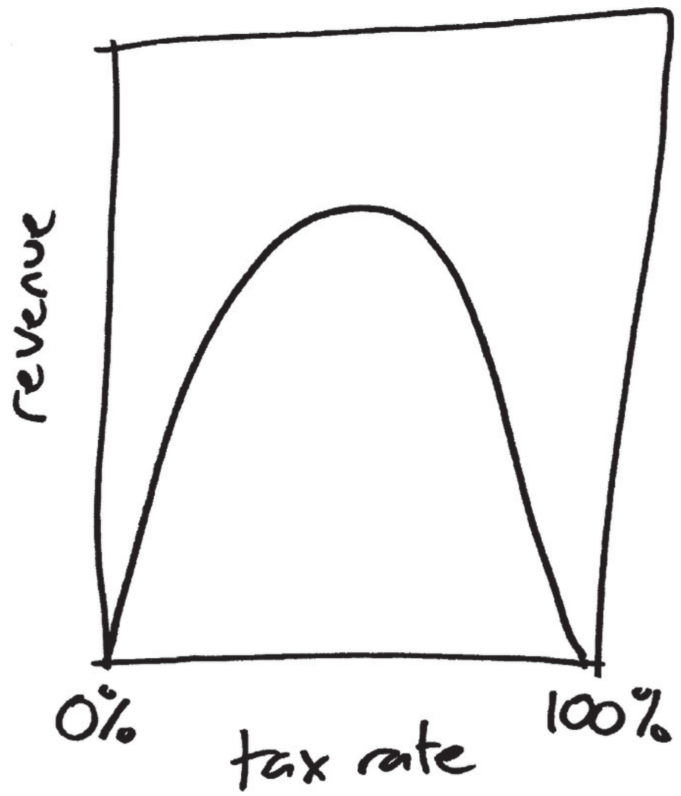
Some of the figures in this presentation are taken from “An Introduction to Statistical Learning, with applications in R” (Springer, 2021) with permission from the authors: G. James, D. Witten, T. Hastie and R. Tibshirani

Linearity & nonlinearity

Q: What's an example of a nonlinear relationship?

Nonlinear curves

> The legend of the Laffer curve goes like this: Arthur Laffer, then an economics professor at the University of Chicago, had dinner one night in 1974 with Dick Cheney, Donald Rumsfeld, and *Wall Street Journal* editor Jude Wanniski at an upscale hotel restaurant in Washington DC. They were tussling over President Ford's tax plan, and eventually, as intellectuals do when the tussling gets heavy, Laffer commandeered a napkin and drew a picture. The picture looked like this:

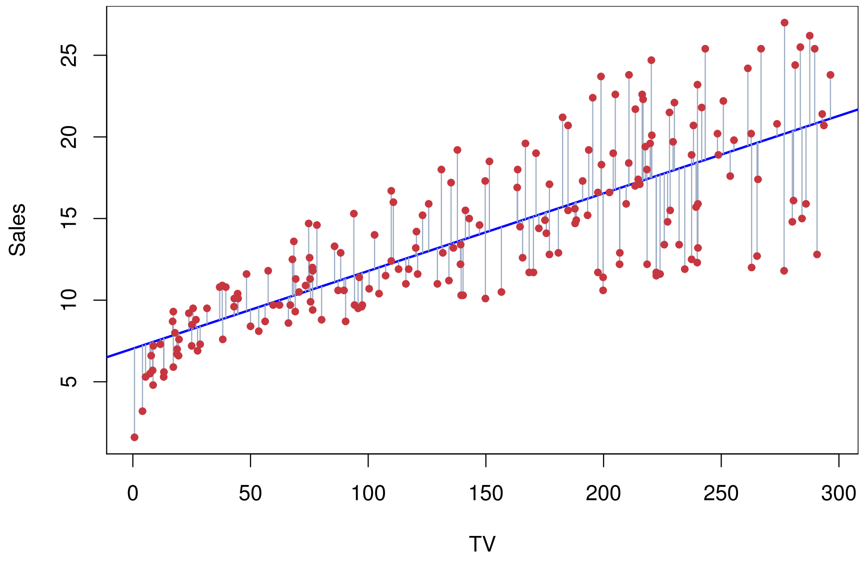


Laffer curve

Source: Jordan Ellenberg (2014), How Not to Be Wrong: The Power of Mathematical Thinking

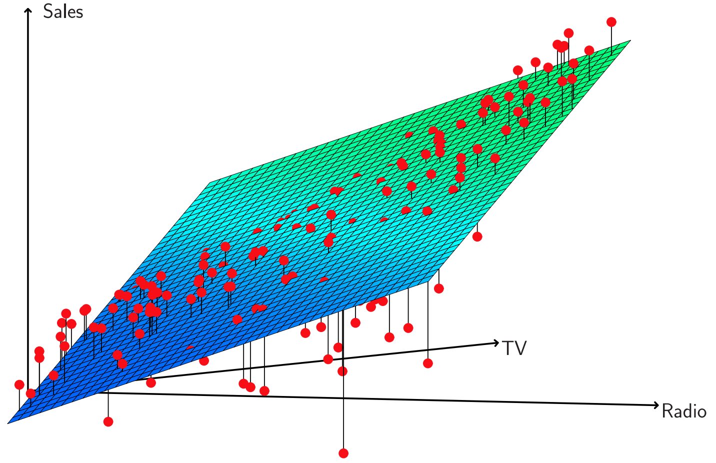
One predictor vs multiple predictors

$$\texttt{sales} \approx \beta_0 + \beta_1 \times \texttt{TV}$$



Linear regression

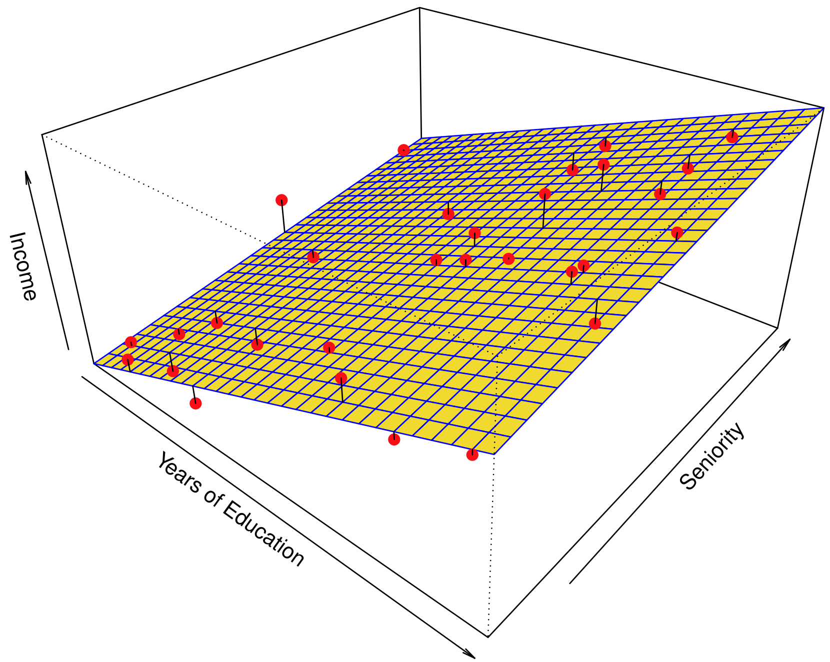
$$\texttt{sales} \approx \beta_0 + \beta_1 \times \texttt{TV} + \beta_2 \times \texttt{radio}$$

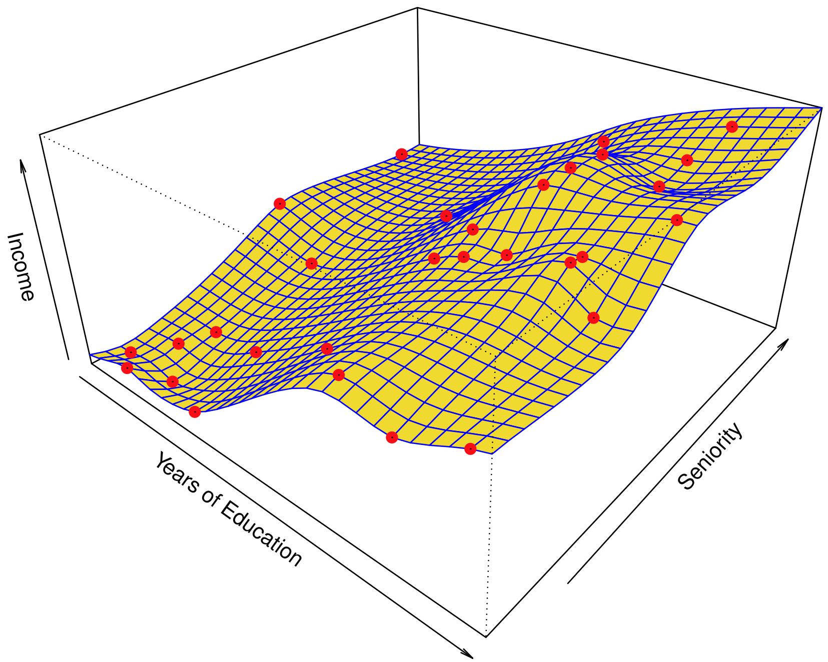


Multiple linear regression

Source: James et al. (2021), *An Introduction to Statistical Learning with Applications in R*, Figures 3.1 & 3.5.

By the end of today





Instead of just fitting lines (linear regression) or hyperplanes (multiple linear regression)…

You'll be able to fit nonlinear curves to multivariate data using *splines* and *Generalised Additive Models*.

Source: James et al. (2021), *An Introduction to Statistical Learning with Applications in R*, Figures 2.4 & 2.6.

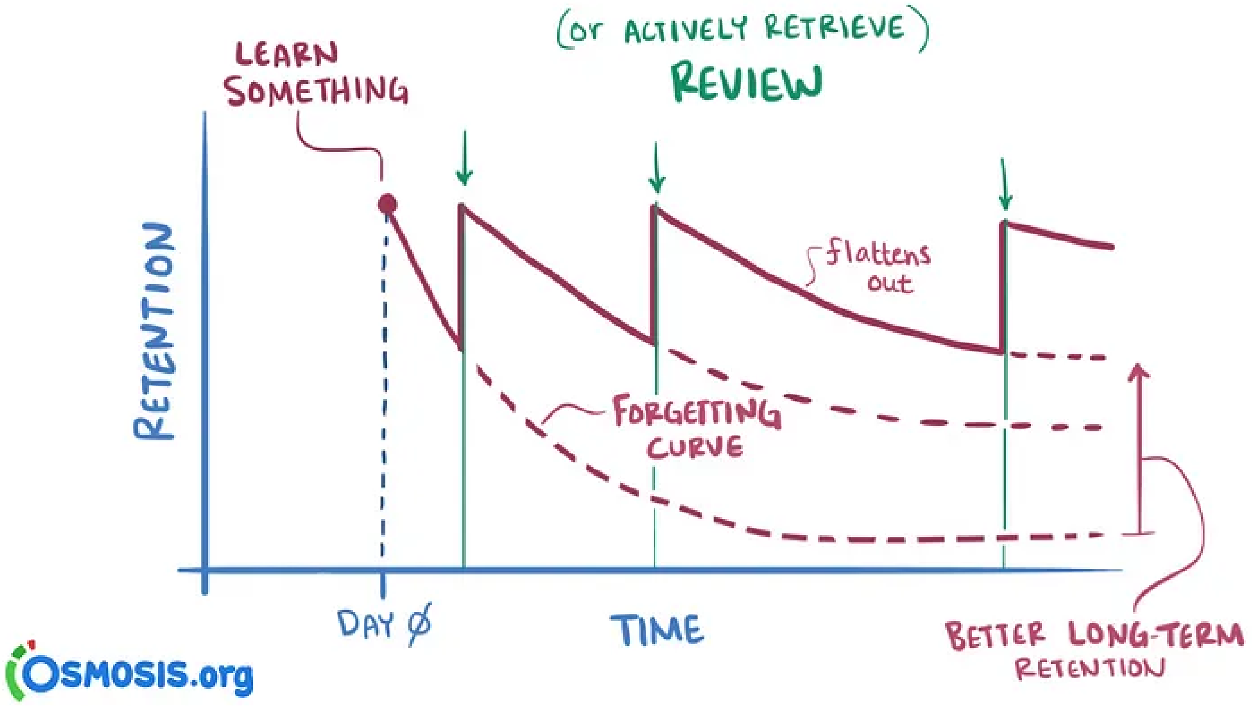
Moving beyond linearity

> Using a term like nonlinear science is like referring to the bulk of zoology as the study of non-elephant animals. (Stanisław Ulam)

- Linear models are highly interpretable
- Linear assumption can be *very* unrealistic
- Look for interpretable nonlinear models
- A machine learning view, not a statistical view

1. Polynomial regression
2. Step functions
3. Regression splines
4. Smoothing splines
5. Local regression
6. Generalised additive models

The methods from different perspectives



Today's topics will be presented in *concept*, in *code*, and in *math*.

> **Note**
>
> There's a fair bit of code today and in the rest of this course. This is to help you with understanding and with your *project*. Also, coding is a sizable part of an actuary's day-to-day work.

Source: Osmosis.org

In-class demonstration

I want you to ‘fit’ the data four different ways by drawing:

Top left: a straight line

- Draw a single straight line
- Don’t lift your pen from the page

Top right: a quadratic curve

- Draw a single smiley-face curve
- Don’t lift your pen from the page

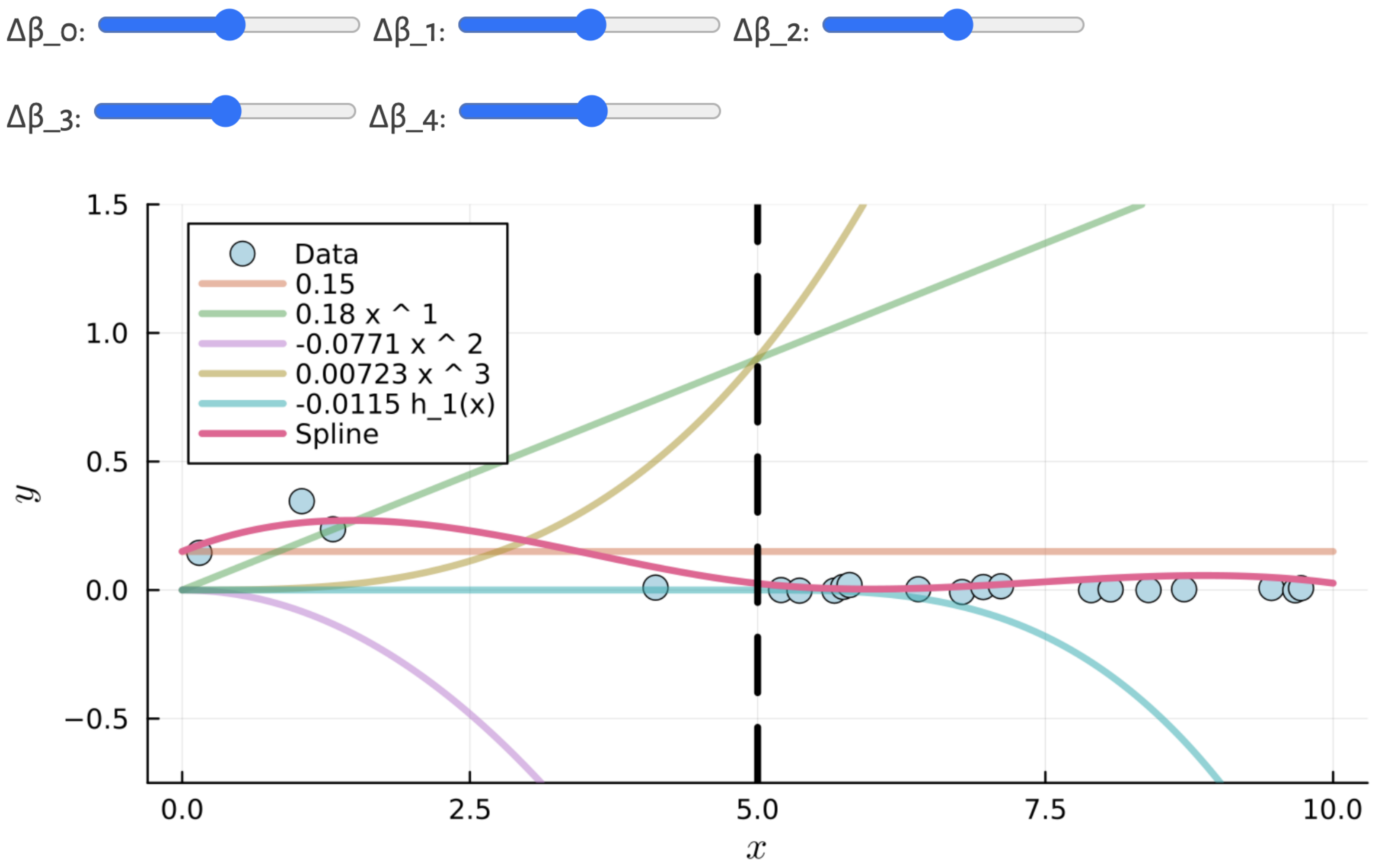
Bottom left: a step function

- Draw a sequence of flat lines
- Lift your pen between each line

Bottom right: a smooth curve

- Draw a single curve of any shape
- Avoid jagged changes of direction

Link to interactive notebook



See the spline demo notebook for a high-level view of these methods

Data science starts with data

Luxembourg Mortality Data

Download a file called `Mx_1x1.txt` from the Human Mortality Database.

No-one is allowed to distribute the data, but you can download it for free. Here are the first few rows to get a sense of what it looks like.

```
Luxembourg, Death rates (period 1x1),   Last modified: 09 Aug 2023;  Methods Protocol: v6 (2017)

  Year          Age             Female          Male            Total
  1960           0             0.023863        0.039607        0.031891
  1960           1             0.001690        0.003528        0.002644
  1960           2             0.001706        0.002354        0.002044
  1960           3             0.001257        0.002029        0.001649
  1960           4             0.000844        0.001255        0.001051
  1960           5             0.000873        0.001701        0.001293
  1960           6             0.000443        0.000430        0.000437
```

Load packages

R setup:

```r
library(splines)
library(mgcv)
library(tidyverse)
```

Python setup:

```python
import seaborn as sns
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt

from patsy import dmatrix
import statsmodels.api as sm
import statsmodels.formula.api as smf
```

Setup & importing the data

R Python

```
lux <- read_table("Mx_1x1.txt", skip = 2, show_col_types = FALSE) %>%
  rename(age=Age, year=Year, mx=Female) %>%
  select(age, year, mx) %>%
  filter(age != '110+') %>%
  mutate(year = as.integer(year), age = as.integer(age), mx = as.numeric(mx))
```

```
lux
```

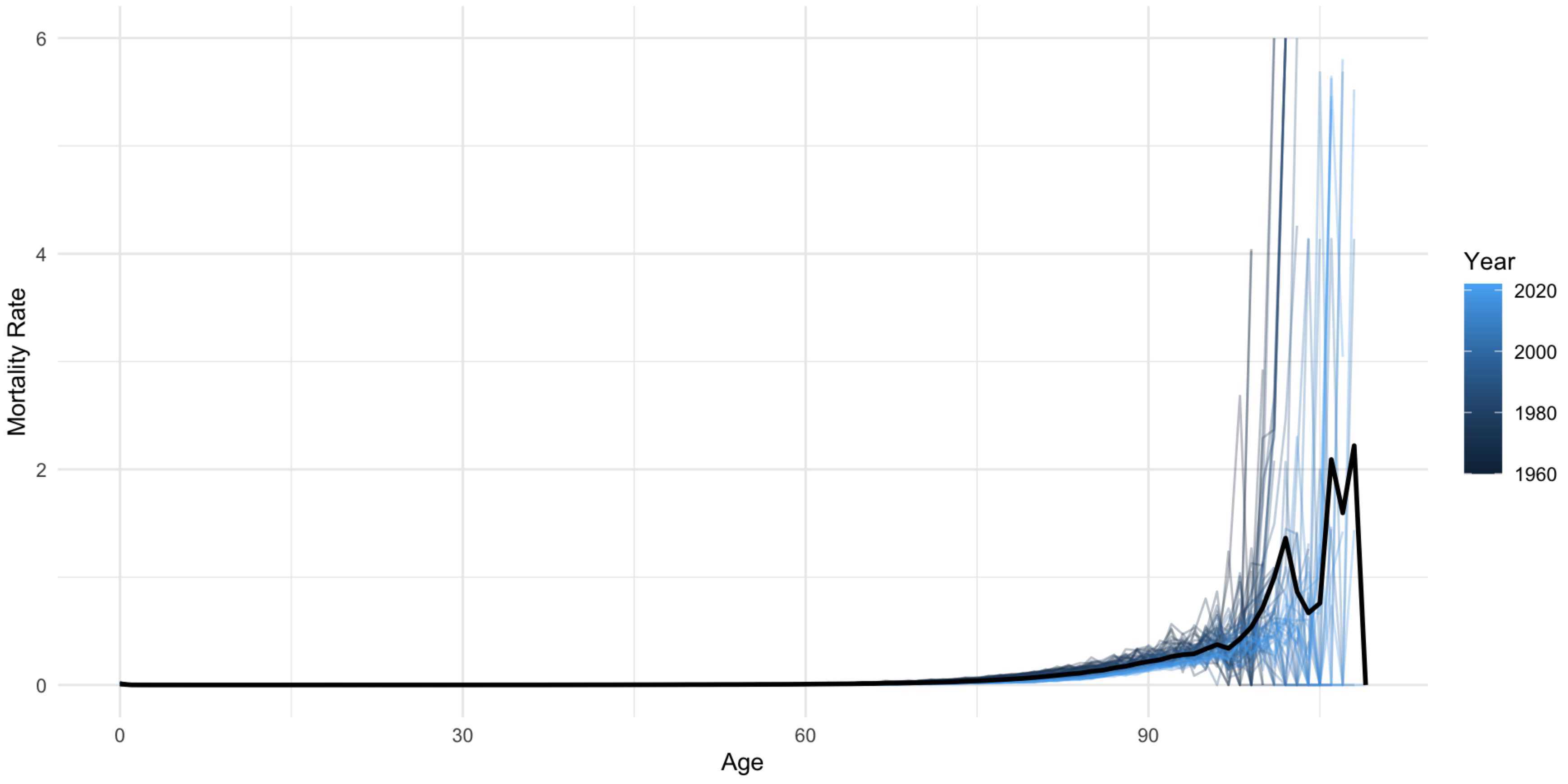
```
# A tibble: 6,930 × 3
     age  year       mx
   <int> <int>    <dbl>
 1     0  1960 0.0239
 2     1  1960 0.00169
 3     2  1960 0.00171
 4     3  1960 0.00126
 5     4  1960 0.000844
 6     5  1960 0.000873
 7     6  1960 0.000443
 8     7  1960 0
 9     8  1960 0.000951
10     9  1960 0
# ℹ 6,920 more rows
```

```
summary(lux)
```

```
      age             year            mx
 Min.   :  0.0   Min.   :1960   Min.   :0.0000
 1st Qu.: 27.0   1st Qu.:1975   1st Qu.:0.0004
 Median : 54.5   Median :1991   Median :0.0034
 Mean   : 54.5   Mean   :1991   Mean   :0.0920
 3rd Qu.: 82.0   3rd Qu.:2007   3rd Qu.:0.0418
 Max.   :109.0   Max.   :2022   Max.   :6.0000
                                NA's   :358
```

Mortality

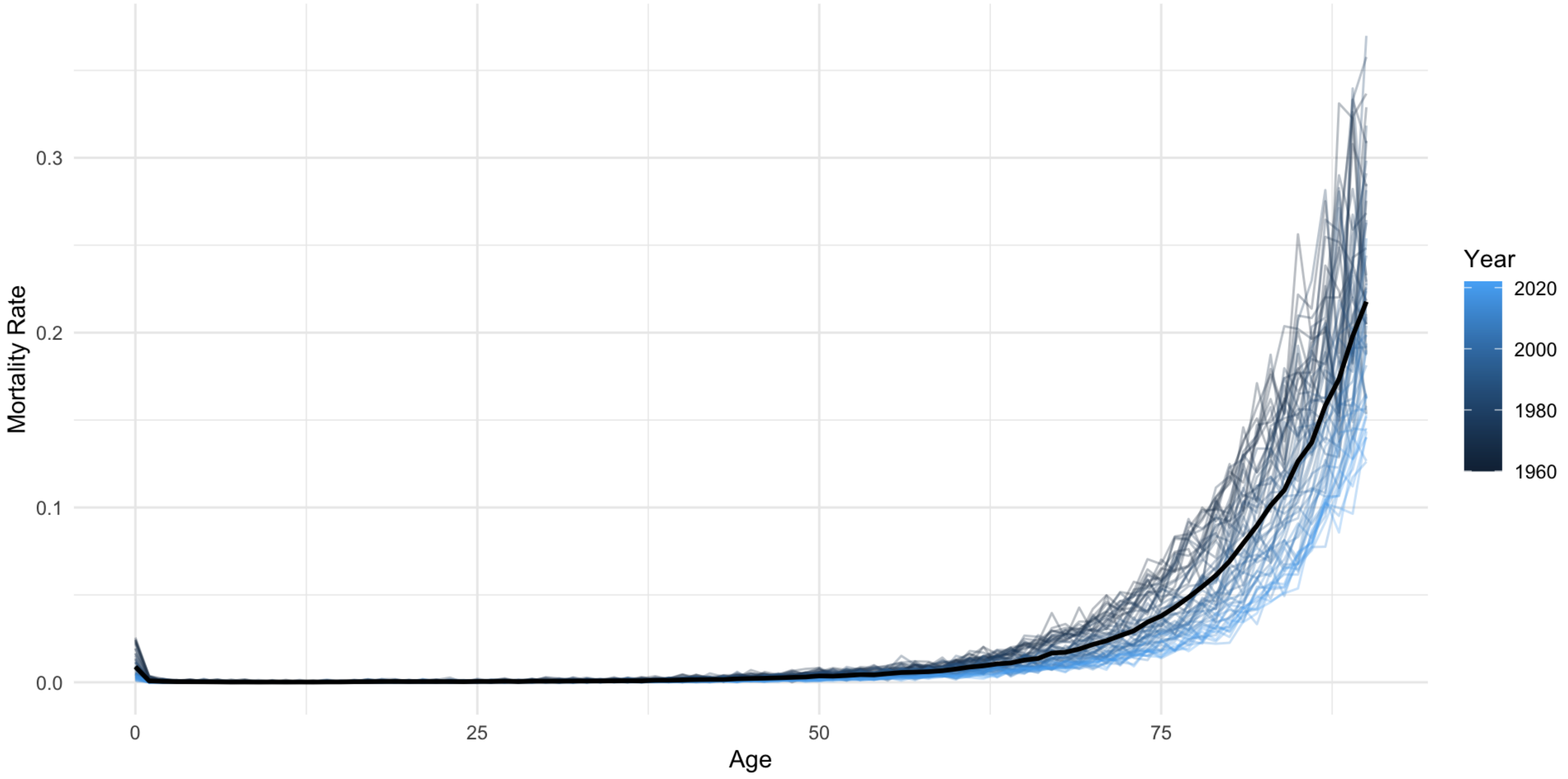
R | Python



Mortality (zoom in)

R | Python

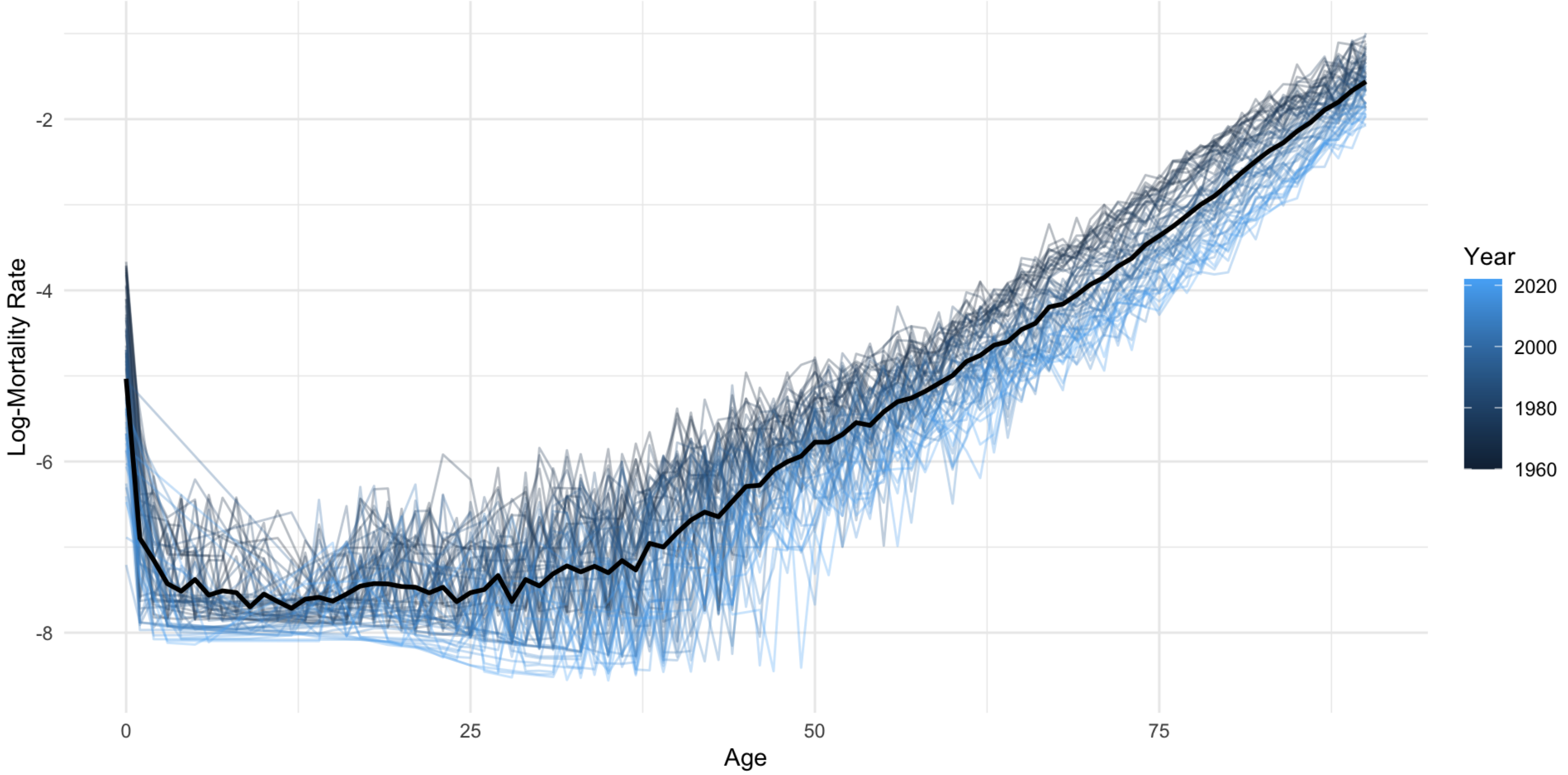
```r
lux <- lux %>% filter(age <= 90)
```



Log-mortality

R Python

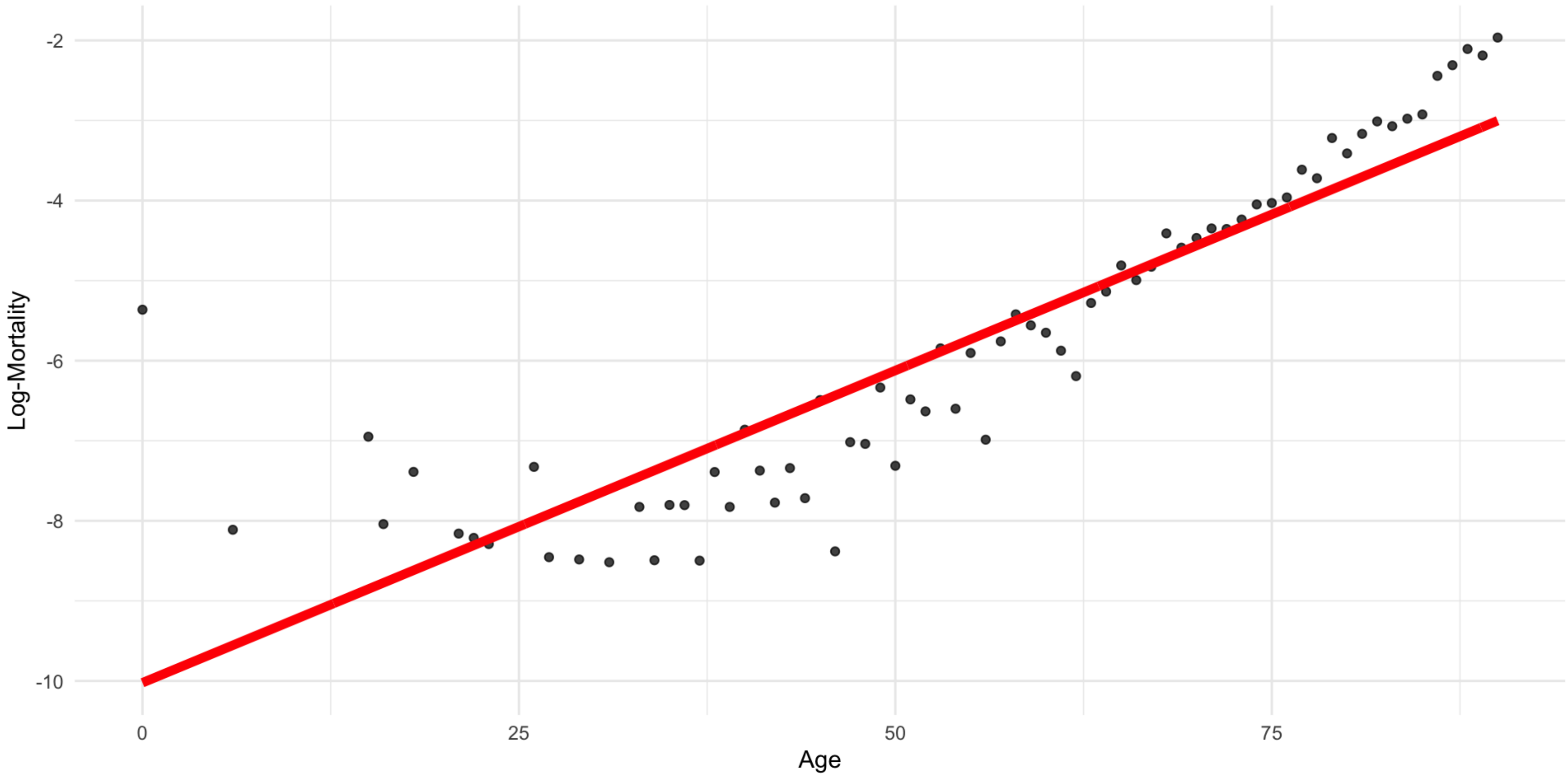
```r
lux$log_mx <- log(lux$mx)
lux <- lux[lux$log_mx != -Inf, ]
```



Linear regression

R Python

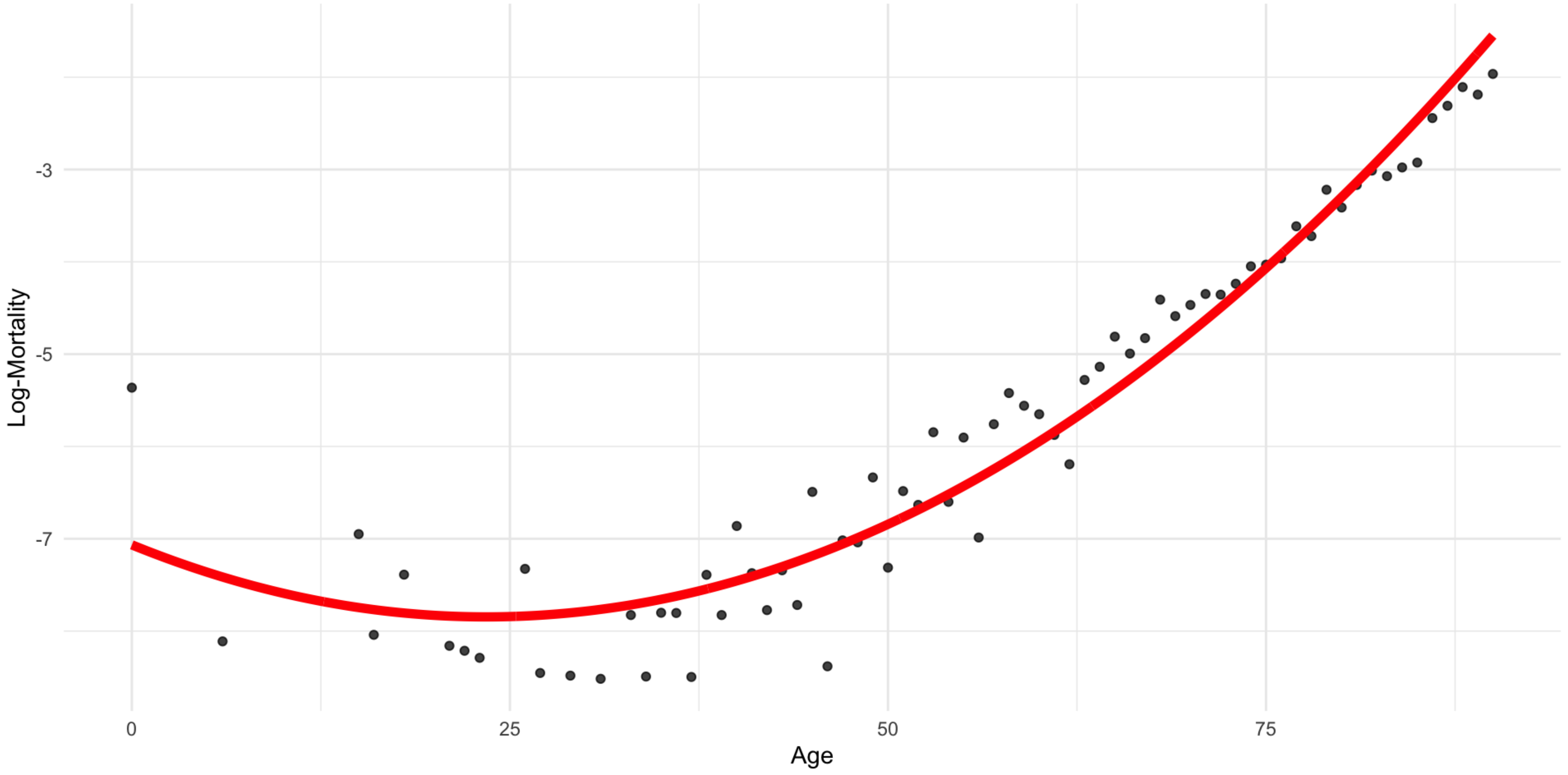
```r
lux_2020 <- lux %>% filter(year == 2020)
model_lr <- lm(log_mx ~ age, data = lux_2020)
```



Quadratic regression

R | Python

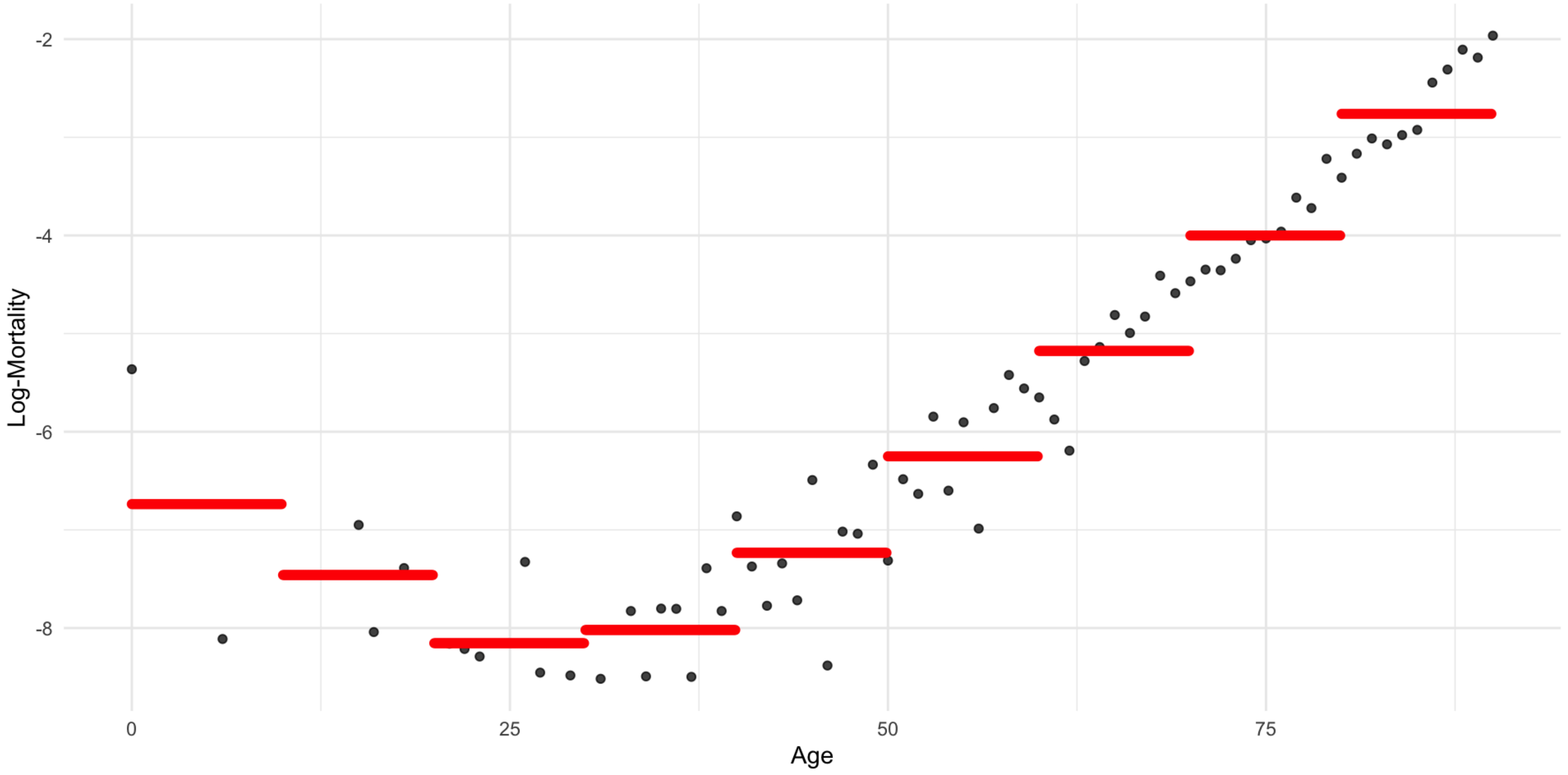
```r
model_quad <- lm(log_mx ~ poly(age, 2), data = lux_2020)
```



Step function regression

R Python

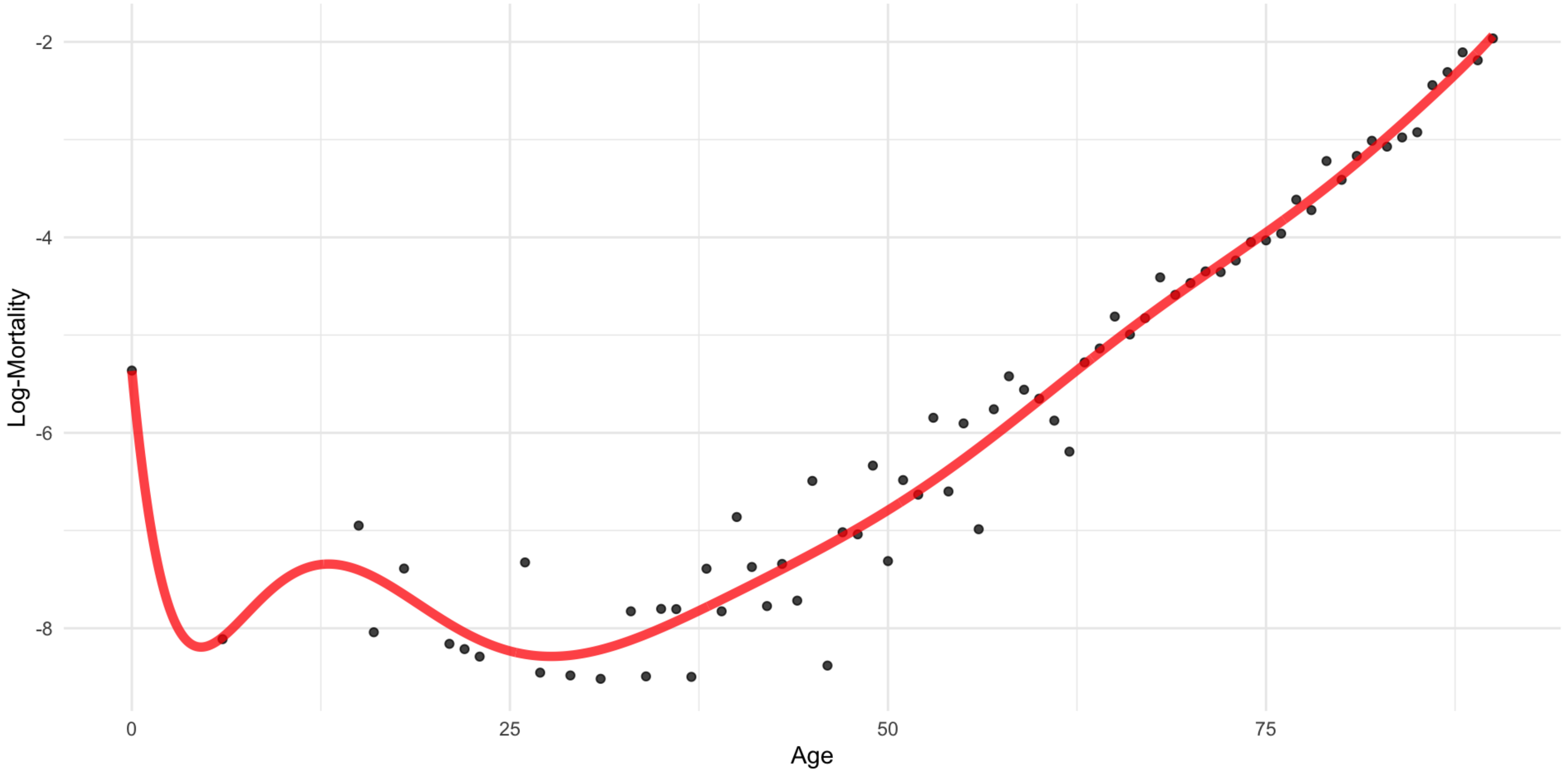
```
model_step <- lm(log_mx ~ cut(age, seq(0, 90, 10), right=F), data = lux_2020)
```



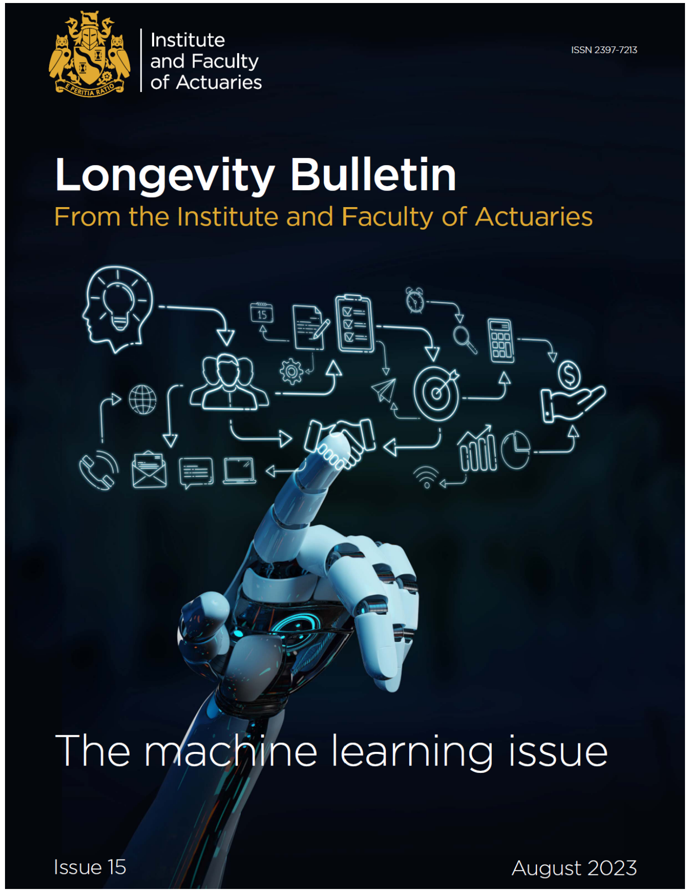
Regression spline

R | Python

```r
model_spline <- lm(log_mx ~ bs(age, degree=10), data=lux_2020) # Requires splines package
```



Industry approaches



IFoA bulletin on machine learning in mortality modelling

Methods from this class (p. 8–9):

- ridge regression
- lasso regression
- elastic net
- generalised linear models
- generalised additive models
- random forests
- dimension reduction
- (artificial neural networks)

Future courses

Take **ACTL3141** for proper mortality modelling

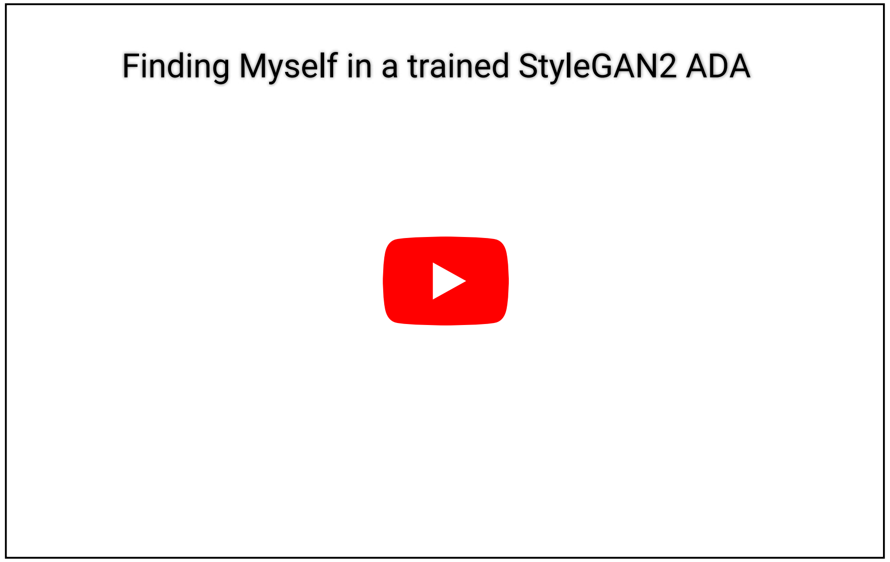
```
Methods from this class (p. 8--9):

- ridge regression
- lasso regression
- elastic net
- generalised linear models
- generalised additive models
- random forests
- dimension reduction
- (artificial neural networks)

Tonight, Andres Villegas will talk about _machine learning in mortality modelling_.
:::
:::
```

A real autocompletion from GitHub Copilot

Take **ACTL3143** (https://laub.au/ai) for AI in actuarial science



Linear Regression

The matrix approach

TV	radio	sales
230.1	37.8	22.1
44.5	39.3	10.4
17.2	45.9	9.3
151.5	41.3	18.5
180.8	10.8	12.9
8.7	48.9	7.2
57.5	32.8	11.8
120.2	19.6	13.2
8.6	2.1	4.8
199.8	2.6	10.6
66.1	5.8	8.6

$$\boldsymbol{y} = \mathbf{X}\boldsymbol{\beta} + \boldsymbol{\varepsilon}$$

$$\mathbf{X} = \begin{bmatrix} 1 & x_{1,1} & x_{1,2} & \dots & x_{1,p} \\ 1 & x_{2,1} & x_{2,2} & \dots & x_{2,p} \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ 1 & x_{n,1} & x_{n,2} & \dots & x_{n,p} \end{bmatrix} \boldsymbol{\beta} = \begin{bmatrix} \beta_0 \\ \vdots \\ \beta_p \end{bmatrix} \boldsymbol{y} = \begin{bmatrix} y_1 \\ y_2 \\ \vdots \\ y_n \end{bmatrix}$$

```r
site <- url("https://www.statlearning.com/s/Advertising.csv")
df_adv <- read_csv(site, show_col_types = FALSE)
X <- model.matrix(~ TV + radio, data = df_adv);
y <- df_adv[, "sales"]
```

```r
head(X)
```

```
  (Intercept)    TV radio
1           1 230.1  37.8
2           1  44.5  39.3
3           1  17.2  45.9
4           1 151.5  41.3
5           1 180.8  10.8
6           1   8.7  48.9
```

```r
head(y)
```

```
# A tibble: 6 × 1
  sales
  <dbl>
1  22.1
2  10.4
3   9.3
4  18.5
5  12.9
6   7.2
```

Design matrices

This is basically the 'Excel'-style covariates / predictors plus a column of ones.

If categorical variables are present, they are converted to *dummy variables*:

```r
fake <- tibble(
  speed = c(100, 80, 60, 60, 120, 40),
  risk = c("Low", "Medium", "High",
           "Medium", "Low", "Low")
)
fake
```

```
# A tibble: 6 × 2
  speed risk
  <dbl> <chr>
1   100 Low
2    80 Medium
3    60 High
4    60 Medium
5   120 Low
6    40 Low
```

```r
model.matrix(~ speed + risk, data = fake)
```

```
  (Intercept) speed riskLow riskMedium
1           1   100       1          0
2           1    80       0          1
3           1    60       0          0
4           1    60       0          1
5           1   120       1          0
6           1    40       1          0
attr(,"assign")
[1] 0 1 2 2
attr(,"contrasts")
attr(,"contrasts")$risk
[1] "contr.treatment"
```

Note

I'll assume it is the *training set* from last lecture's validation set approach. Otherwise, we have $\mathbf{X}_{\text{Train}}$, $\mathbf{X}_{\text{Val}}$, $\mathbf{X}_{\text{Test}}$, $\boldsymbol{y}_{\text{Train}}$, $\boldsymbol{y}_{\text{Val}}$, and $\boldsymbol{y}_{\text{Test}}$.

UNSW SYDNEY

Brief refresher

Fitting: Minimise the residuals sum of squares

$$\begin{aligned}\mathrm{RSS} &= \sum_{i=1}^{n}(y_i - \widehat{y}_i)^2 = \sum_{i=1}^{n}\left(y_i - \widehat{\beta}_0 - \widehat{\beta}_1 x_{i,1} - \ldots - \widehat{\beta}_p x_{i,p}\right)^2 \\ &= (\boldsymbol{y} - \mathbf{X}\boldsymbol{\beta})^\top(\boldsymbol{y} - \mathbf{X}\boldsymbol{\beta})\end{aligned}$$

If $\left(\mathbf{X}^\top\mathbf{X}\right)^{-1}$ exists, it can be shown that the solution is given by:

$$\widehat{\boldsymbol{\beta}} = \left(\mathbf{X}^\top\mathbf{X}\right)^{-1}\mathbf{X}^\top\boldsymbol{y}.$$

Predicting: The predicted values are given by

$$\widehat{\boldsymbol{y}} = \mathbf{X}\widehat{\boldsymbol{\beta}}.$$

R's `lm` and `predict`

$$\widehat{\beta} = (\mathbf{X}^\top \mathbf{X})^{-1} \mathbf{X}^\top \boldsymbol{y}$$

```r
model <- lm(log_mx ~ age, data = lux_2020)
coef(model)
```

```
 (Intercept)          age 
-10.02233982   0.07798979 
```

```r
X <- model.matrix(~ age, data = lux_2020)
y <- lux_2020$log_mx
beta <- solve(t(X) %*% X) %*% t(X) %*% y
beta
```

```
                    [,1]
(Intercept) -10.02233982
age           0.07798979
```

$$\widehat{\boldsymbol{y}} = \mathbf{X}\widehat{\beta}.$$

```r
ages <- data.frame(age = c(0, 20, 110))
predict(model, newdata = ages)
```

```
         1          2          3 
-10.022340  -8.462544  -1.443463 
```

```r
exp(predict(model, newdata = ages))
```

```
           1            2            3 
4.439695e-05 2.112340e-04 2.361086e-01 
```

```r
ages <- data.frame(age = c(0, 20, 110))
X_new <- model.matrix(~ age, data = ages)
X_new %*% beta
```

```
        [,1]
1 -10.022340
2  -8.462544
3  -1.443463
```

Dummy encoding & collinearity

Why do *dummy variables* drop the last level?

```
X_dummy = model.matrix(~ risk, data = fake)
as.data.frame(X_dummy)
```

```
  (Intercept) riskLow riskMedium
1           1       1          0
2           1       0          1
3           1       0          0
4           1       0          1
5           1       1          0
6           1       1          0
```

```
solve(t(X_dummy) %*% X_dummy)
```

```
            (Intercept)   riskLow riskMedium
(Intercept)           1 -1.000000       -1.0
riskLow              -1  1.333333        1.0
riskMedium           -1  1.000000        1.5
```

```
X_oh <- cbind(X_dummy, riskHigh = (fake$risk
as.data.frame(X_oh)
```

```
  (Intercept) riskLow riskMedium riskHigh
1           1       1          0        0
2           1       0          1        0
3           1       0          0        1
4           1       0          1        0
5           1       1          0        0
6           1       1          0        0
```

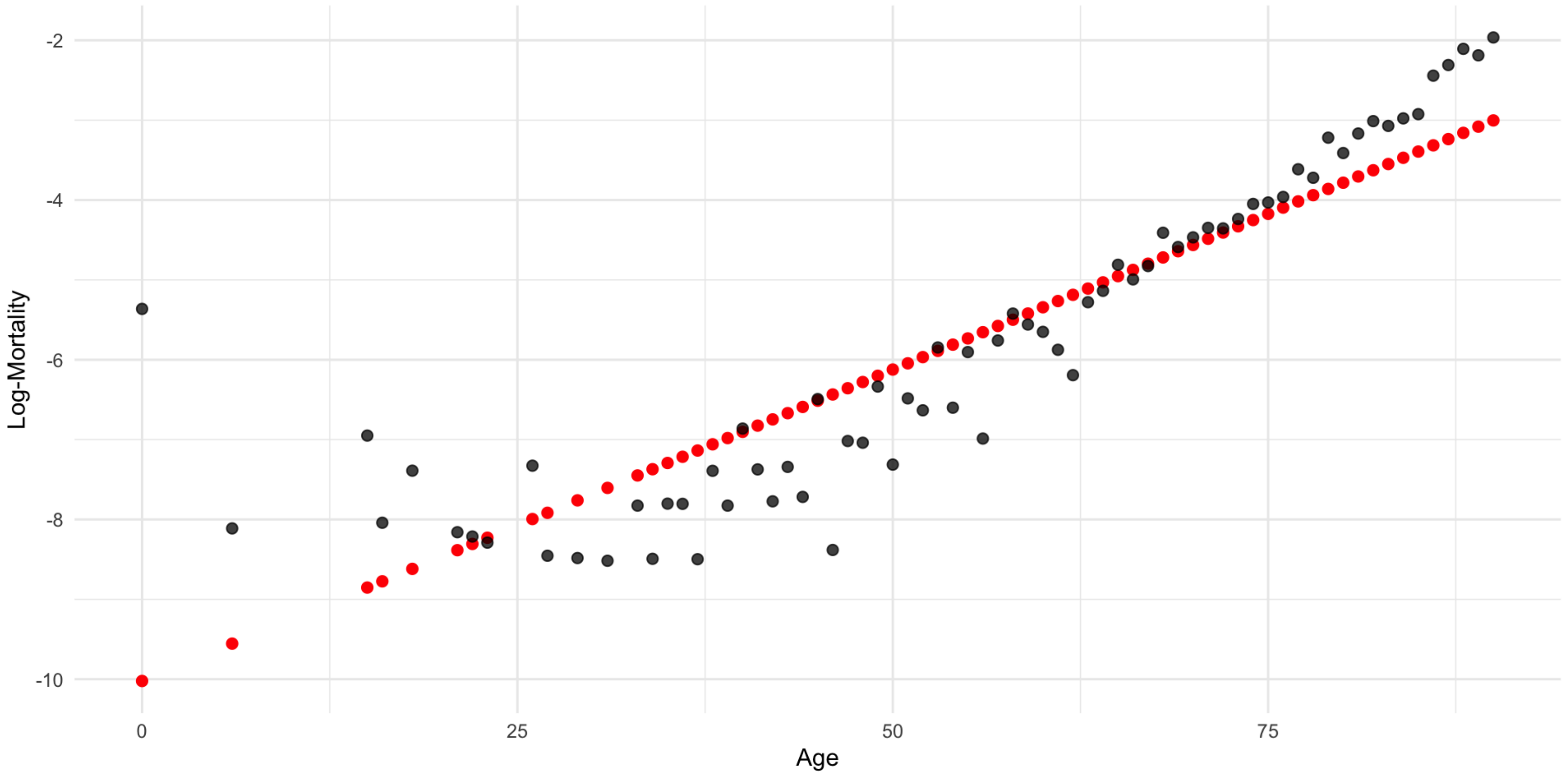
```
solve(t(X_oh) %*% X_oh)
```

```
Error in solve.default(t(X_oh) %*% X_oh): system is
computationally singular: reciprocal condition
number = 6.93889e-18
```

Plotting the fitted values

R Python

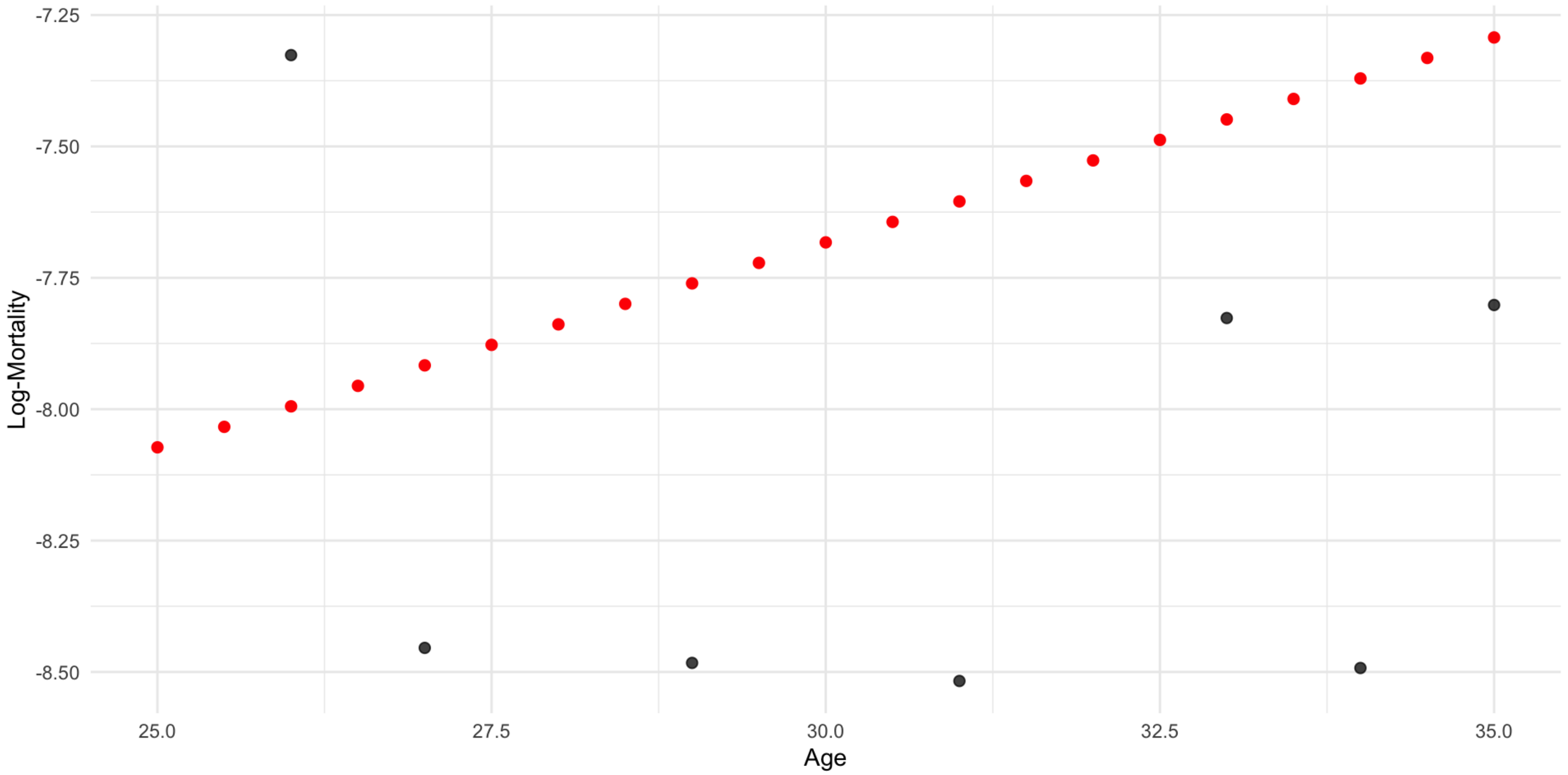
```r
ggplot(lux_2020, aes(x = age, y = log_mx)) + theme_minimal() +
  geom_point(aes(y = predict(model)), color = "red", size = 2) +
  geom_point(alpha = 0.75, size = 2) + labs(x = "Age", y = "Log-Mortality")
```



Interpolating

R Python

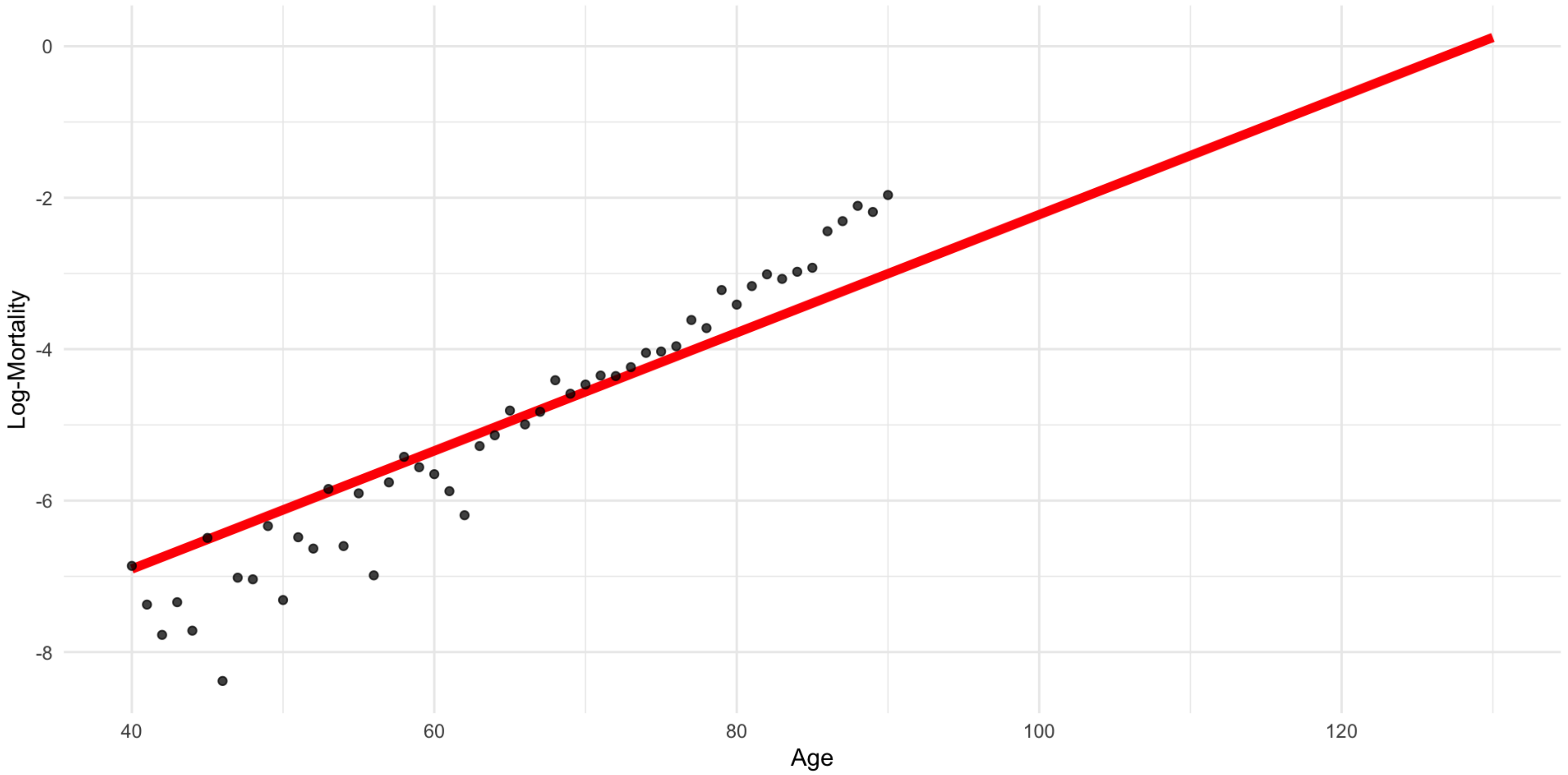
```r
df_grid <- data.frame(age = seq(25, 35, by = 0.5))
df_grid$log_mx <- predict(model, newdata = df_grid)
```



Extrapolating

R | Python

```r
df_grid <- data.frame(age = seq(40, 130))
df_grid$log_mx <- predict(model, newdata = df_grid)
```



Multiple linear regression

```r
df_mlr = lux[c("age", "year", "log_mx")]
head(df_mlr)
```

```
# A tibble: 6 × 3
    age  year log_mx
  <int> <int>  <dbl>
1     0  1960  -3.74
2     1  1960  -6.38
3     2  1960  -6.37
4     3  1960  -6.68
5     4  1960  -7.08
6     5  1960  -7.04
```

Fitting:

```r
linear_model <- lm(log_mx ~ age + year, data = df_mlr)
```

Predicting:

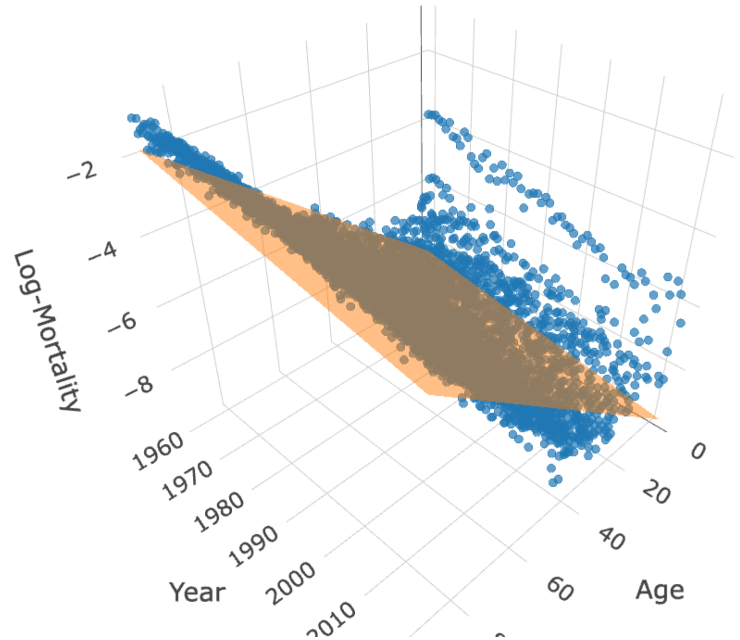
```r
new_point <- data.frame(year = 2040, age = 20)
predict(linear_model, newdata = new_point)
```

```
    1
-8.66
```

```r
coef(linear_model)
```

```
(Intercept)         age        year
34.58222358  0.07287039 -0.02191158
```

Fitted multiple linear regression



Polynomial Regression

Polynomial regression

Extend the standard linear model:

$$y_i = \beta_0 + \beta_1 x_i + \varepsilon_i$$

To:

$$y_i = \beta_0 + \beta_1 x_i + \beta_2 x_i^2 + \ldots + \beta_d x_i^d + \varepsilon_i$$

- Relaxes the assumption that predictor and response are linearly related
- Works almost identically to multiple linear regression, except the other "predictors" are just transformations of the initial predictor

Quadratic regression (by hand)

```r
df_pr <- tibble(age = lux_2020$age, age2 = lux_2020$age^2, log_mx = lux_2020$log_mx)
head(df_pr)
```

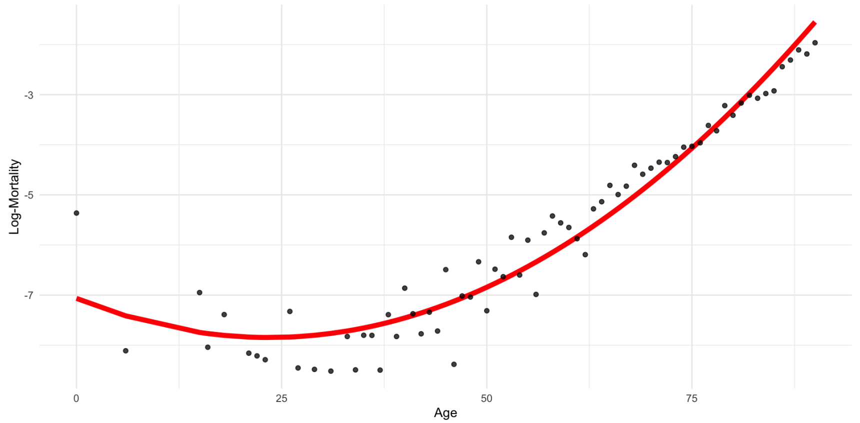
```
# A tibble: 6 × 3
    age  age2 log_mx
  <int> <dbl>  <dbl>
1     0     0  -5.36
2     6    36  -8.11
3    15   225  -6.95
4    16   256  -8.04
5    18   324  -7.39
6    21   441  -8.16
```

```r
poly_model <- lm(log_mx ~ age + age2,
    data = df_pr)
coef(poly_model)
```

```
 (Intercept)          age         age2 
-7.065977594 -0.066603952  0.001421058 
```

```r
bad_x <- data.frame(age = 20, age2 = 20)
predict(poly_model, newdata = bad_x)
```

```
        1 
-8.369635 
```



We just tricked R into thinking that `age2` is a separate variable!

This is a linear model of a nonlinearly transformed variable.

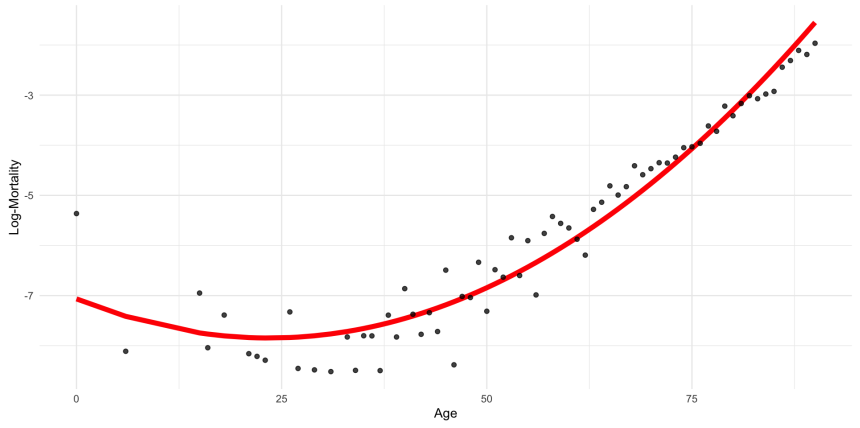
The poly function

```r
df_pr <- tibble(age = lux_2020$age, log_mx = lux_2020$log_mx)
head(df_pr)
```

```
# A tibble: 6 × 2
    age log_mx
  <int>  <dbl>
1     0  -5.36
2     6  -8.11
3    15  -6.95
4    16  -8.04
5    18  -7.39
6    21  -8.16
```

```r
poly_model <- lm(log_mx ~ poly(age, 2),
    data = df_pr)
coef(poly_model)
```

```
 (Intercept) poly(age, 2)1 poly(age, 2)2 
   -5.787494     14.534731      6.376355 
```



Now we *can't* put in `age^2` as a separate variable.

```r
new_input <- data.frame(age = 20)
predict(poly_model, newdata = new_input)
```

```
        1 
-7.829633 
```

Note

The coefficients are different, but the predictions are the same!

Polynomial regression: notes and problems

Pros:

- Can model more complex relationships
- Can also use this in logistic regression, or any linear-like regression for that matter

Cons:

- Normally stick to polynomials of degree 2-4; shape can get very erratic with higher degrees
- Can be computationally unstable with high degrees
- Can be difficult to interpret
- Non-local effects in the errors

Polynomial expansion

```r
head(lux$age)
```

```
[1] 0 1 2 3 4 5
```

```r
age_poly <- model.matrix(~ poly(age, 2), data = lux)
head(age_poly)
```

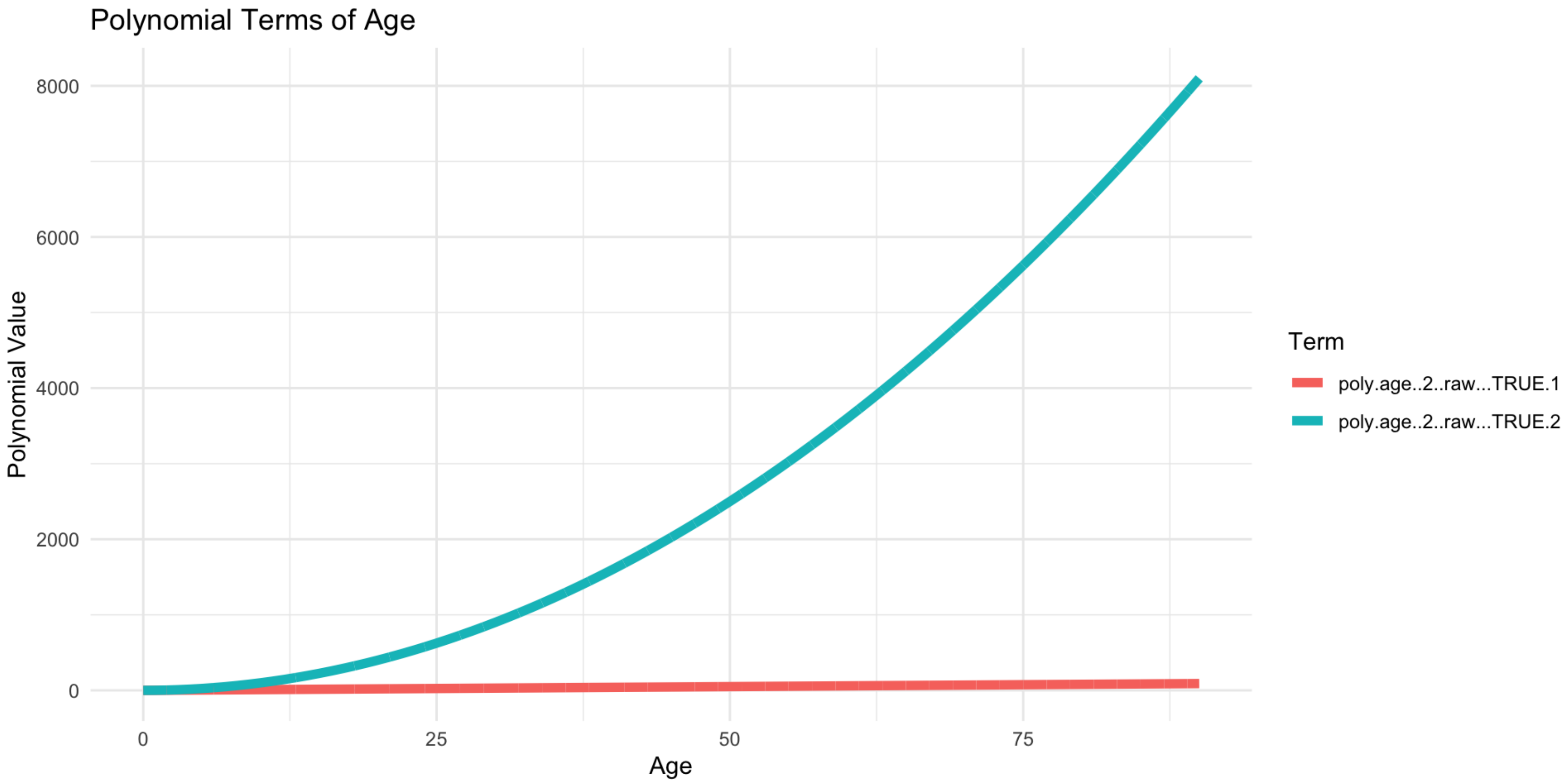
```
  (Intercept) poly(age, 2)1 poly(age, 2)2
1           1   -0.03020513    0.03969719
2           1   -0.02961226    0.03744658
3           1   -0.02901939    0.03524373
4           1   -0.02842652    0.03308866
5           1   -0.02783365    0.03098136
6           1   -0.02724077    0.02892183
```

```r
age_poly <- model.matrix(~ poly(age, 2, raw=TRUE), data = lux)
head(age_poly)
```

```
  (Intercept) poly(age, 2, raw = TRUE)1 poly(age, 2, raw = TRUE)2
1           1                         0                         0
2           1                         1                         1
3           1                         2                         4
4           1                         3                         9
5           1                         4                        16
6           1                         5                        25
```

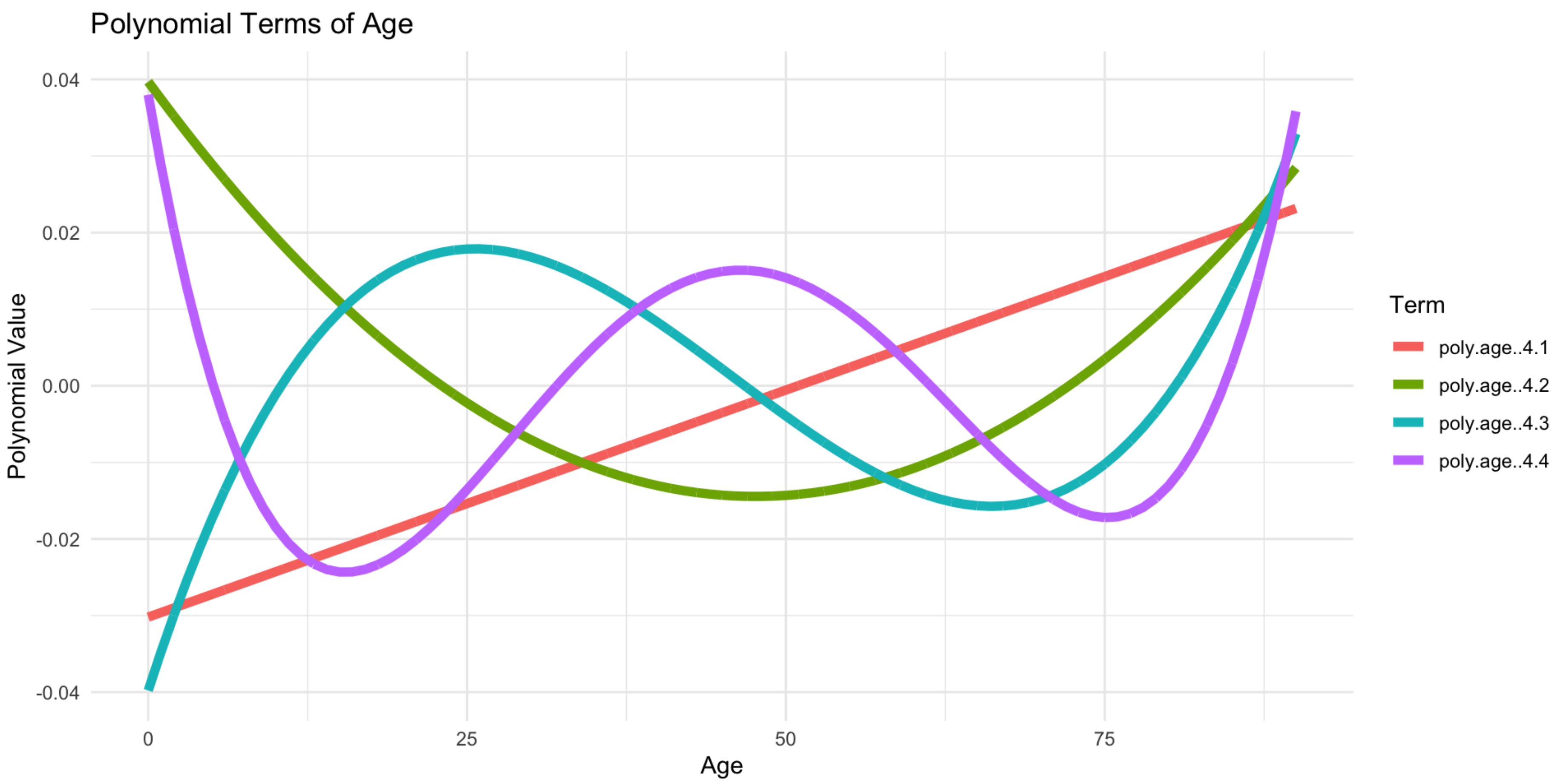
Monomials plotted (`raw=TRUE`)

```
age_poly <- model.matrix(~ poly(age, 2, raw=TRUE), data = lux)
```



Orthogonal polynomials plotted (default)

```r
age_poly <- model.matrix(~ poly(age, 4), data = lux)
```



Why? Collinearity

```r
X <- model.matrix(~ poly(age, 2), data = lux)
kappa(t(X) %*% X)
```

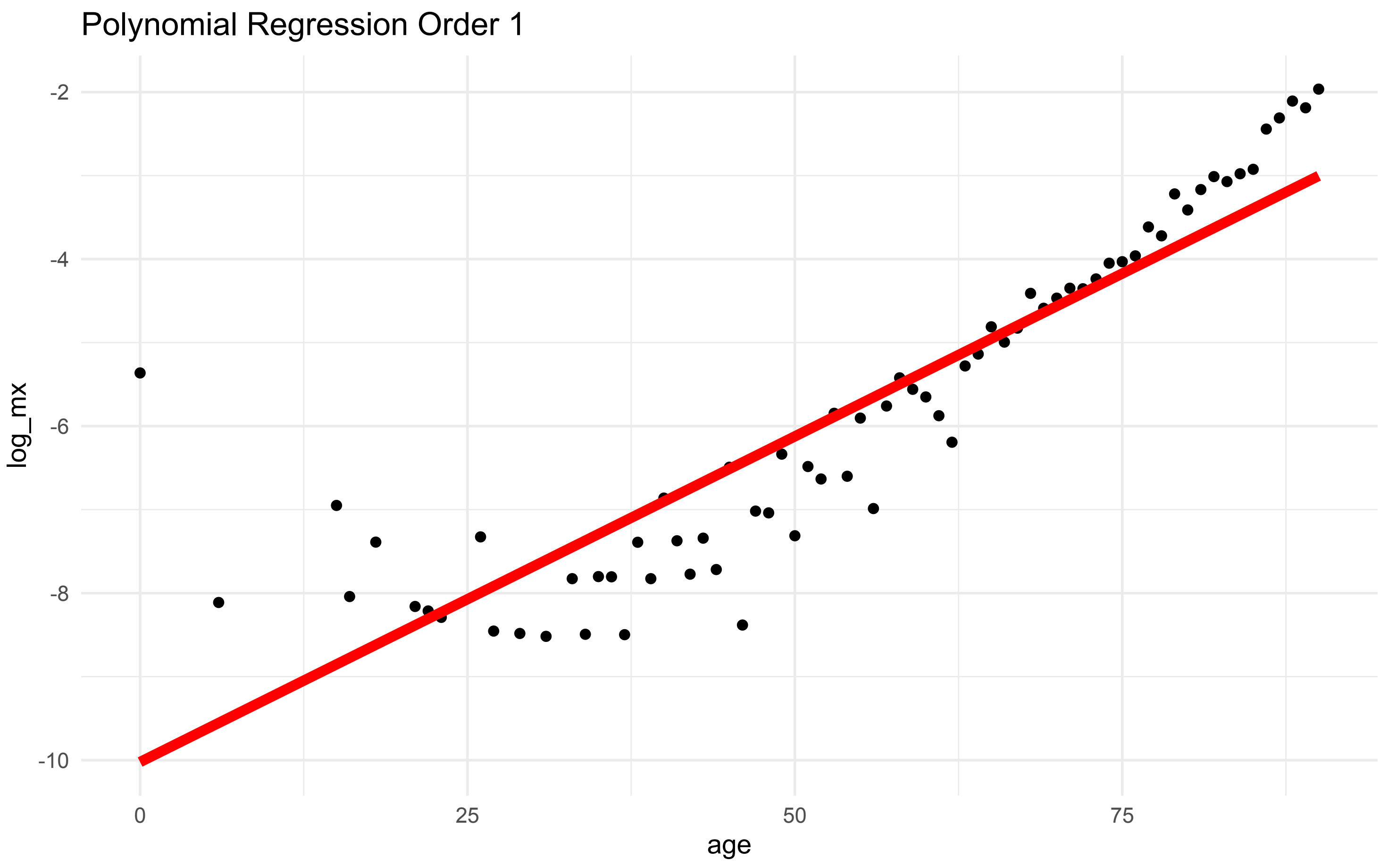
```
[1] 4789.5
```

```r
X <- model.matrix(~ poly(age, 2, raw=TRUE), data = lux)
kappa(t(X) %*% X)
```

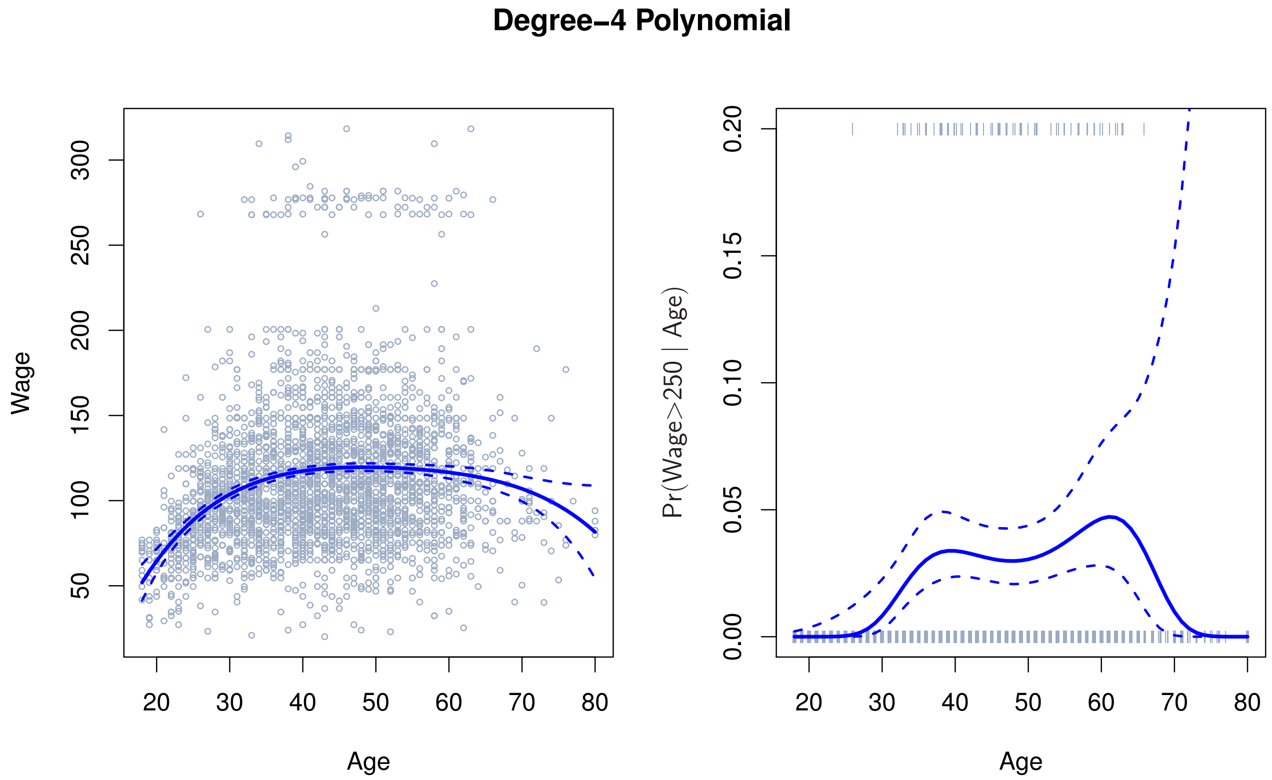
```
[1] 211226485
```

UNSW SYDNEY

Example: Polynomial regression



Can easily use polynomials in classification



(Right Side:) Model of binary event `Wage` > 250 via logistic regression

$$\mathbb{P}(y_i > 250|x_i) = \frac{\exp(\beta_0 + \beta_1 x_i + \beta_2 x_i^2 + \beta_3 x_i^3 + \beta_4 x_i^4)}{1 + \exp(\beta_0 + \beta_1 x_i + \beta_2 x_i^2 + \beta_3 x_i^3 + \beta_4 x_i^4)}$$

Source: James et al. (2021), *An Introduction to Statistical Learning with Applications in R*, Figure 7.1.

Step Functions

Step functions

Polynomial regression imposes a *global structure* on the nonlinear function; an alternative is to use step functions.

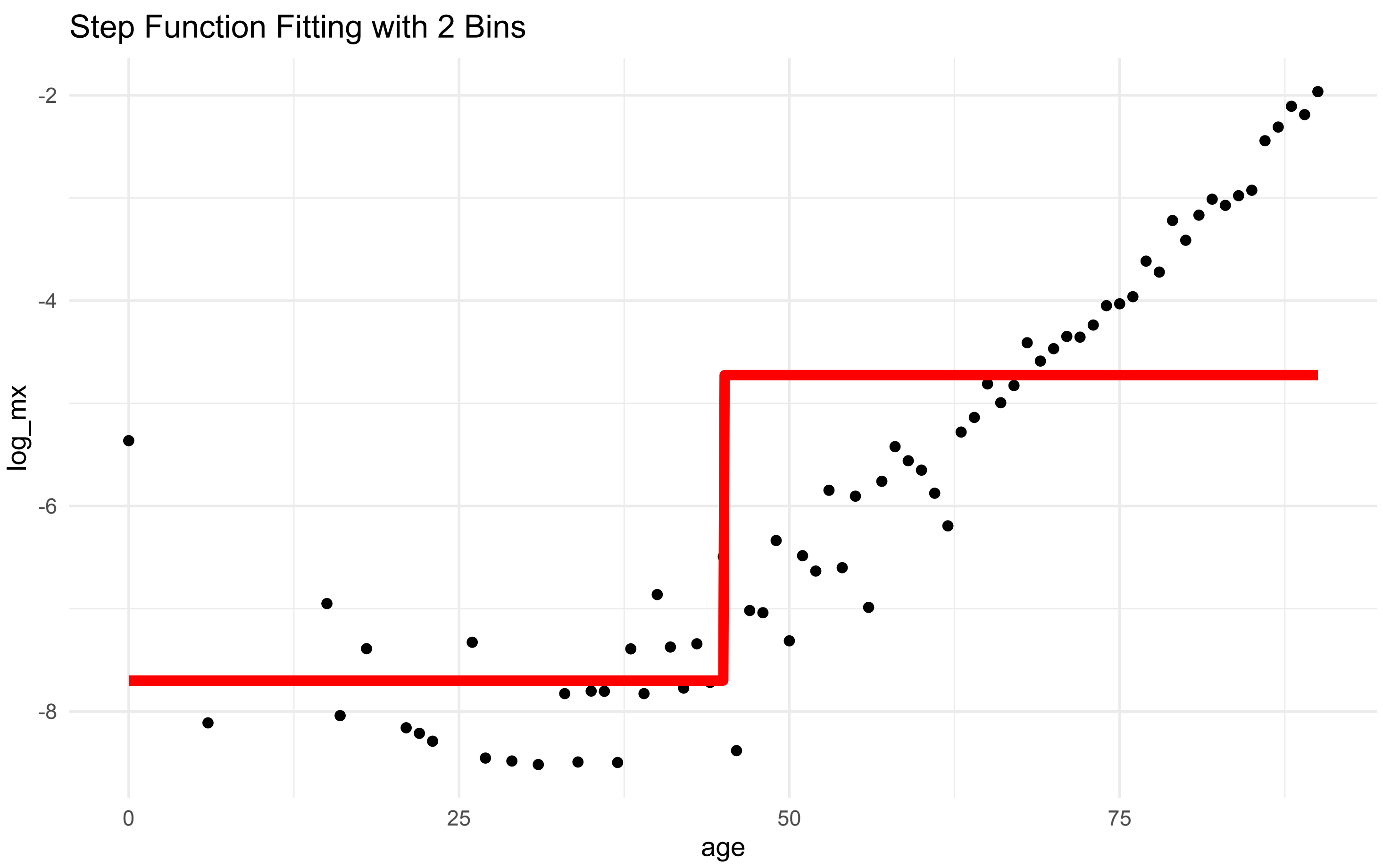
Break up range of x into k distinct regions

$$c_0 < c_1 < \cdots < c_k$$

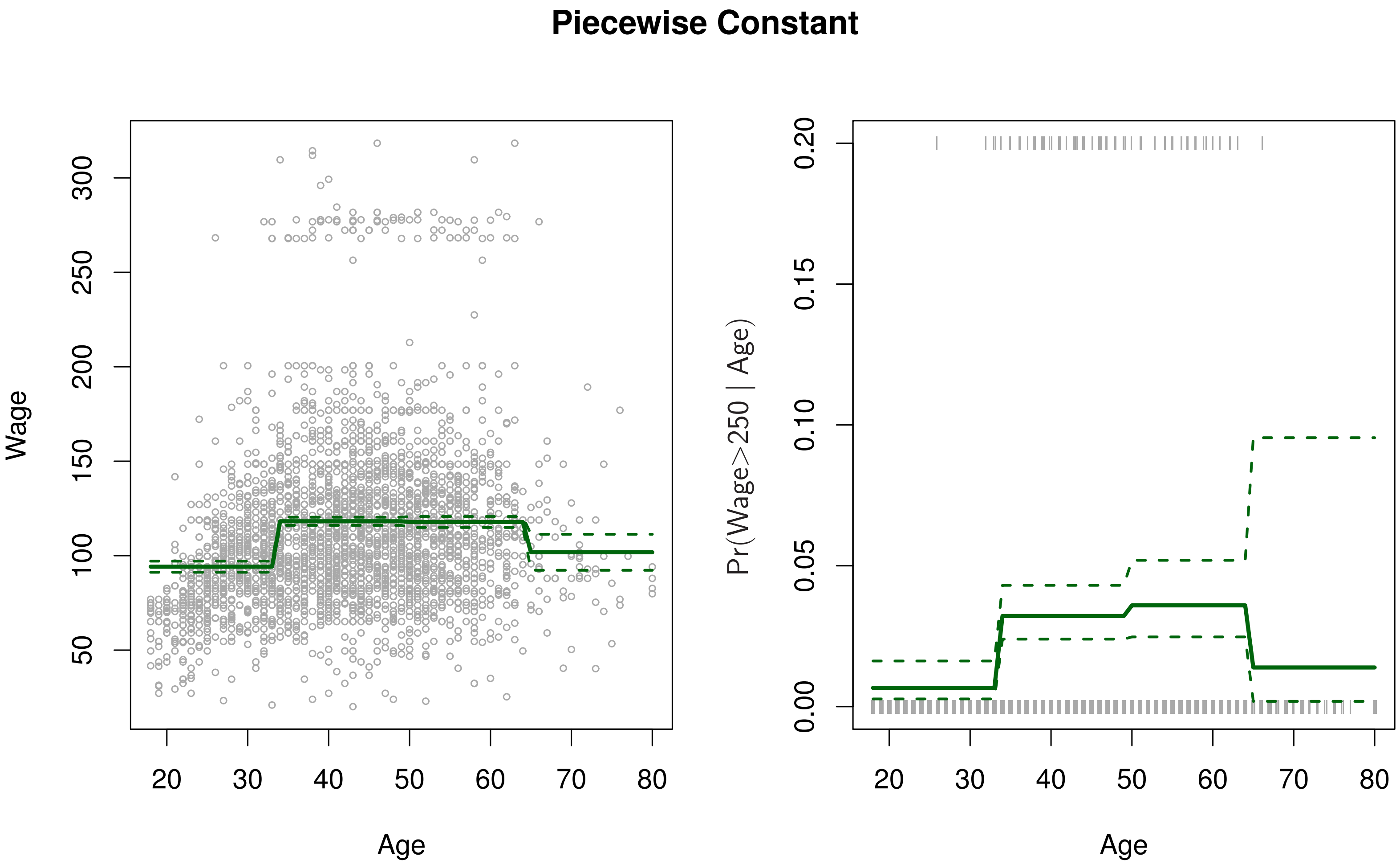
Do a least squares fit on

$$y_i = \beta_0 + \beta_1 I(c_1 \leq x_i \leq c_2) + \beta_2 I(c_2 \leq x_i < c_3) + \cdots + \beta_{k-1} I(c_{k-1} \leq x_i \leq c_k)$$

Example: Step functions



Step function regression on Wage data



Same Wage example as before but with step functions.

Source: James et al. (2021), *An Introduction to Statistical Learning with Applications in R*, Figure 7.2.

Using `I` and `cut`

```r
head(model.matrix(~ age + I(age >= 3), data =
```

```
  (Intercept) age I(age >= 3)TRUE
1           1   0               0
2           1   1               0
3           1   2               0
4           1   3               1
5           1   4               1
6           1   5               1
```

```r
head(cut(lux$age, c(0, 5, 100), right=FALSE))
```

```
[1] [0,5)   [0,5)   [0,5)   [0,5)   [0,5)   [5,100)
Levels: [0,5) [5,100)
```

```r
head(model.matrix(~ age + cut(age, c(0, 5, 100), right=F), data = lux))
```

```
  (Intercept) age cut(age, c(0, 5, 100), right = F)[5,100)
1           1   0                                        0
2           1   1                                        0
3           1   2                                        0
4           1   3                                        0
5           1   4                                        0
6           1   5                                        1
```

```r
model_step <- lm(log_mx ~ cut(age, c(0, 5, 100), right=F), data = lux)
coef(model_step)
```

```
                             (Intercept)
                               -6.555113
cut(age, c(0, 5, 100), right = F)[5,100)
                                1.300758
```

More general viewpoint: Basis functions

Fit the model:

$$y_i = \beta_0 + \beta_1 b_1(x_i) + \beta_2 b_2(x_i) + \cdots + \beta_k b_k(x_i)$$

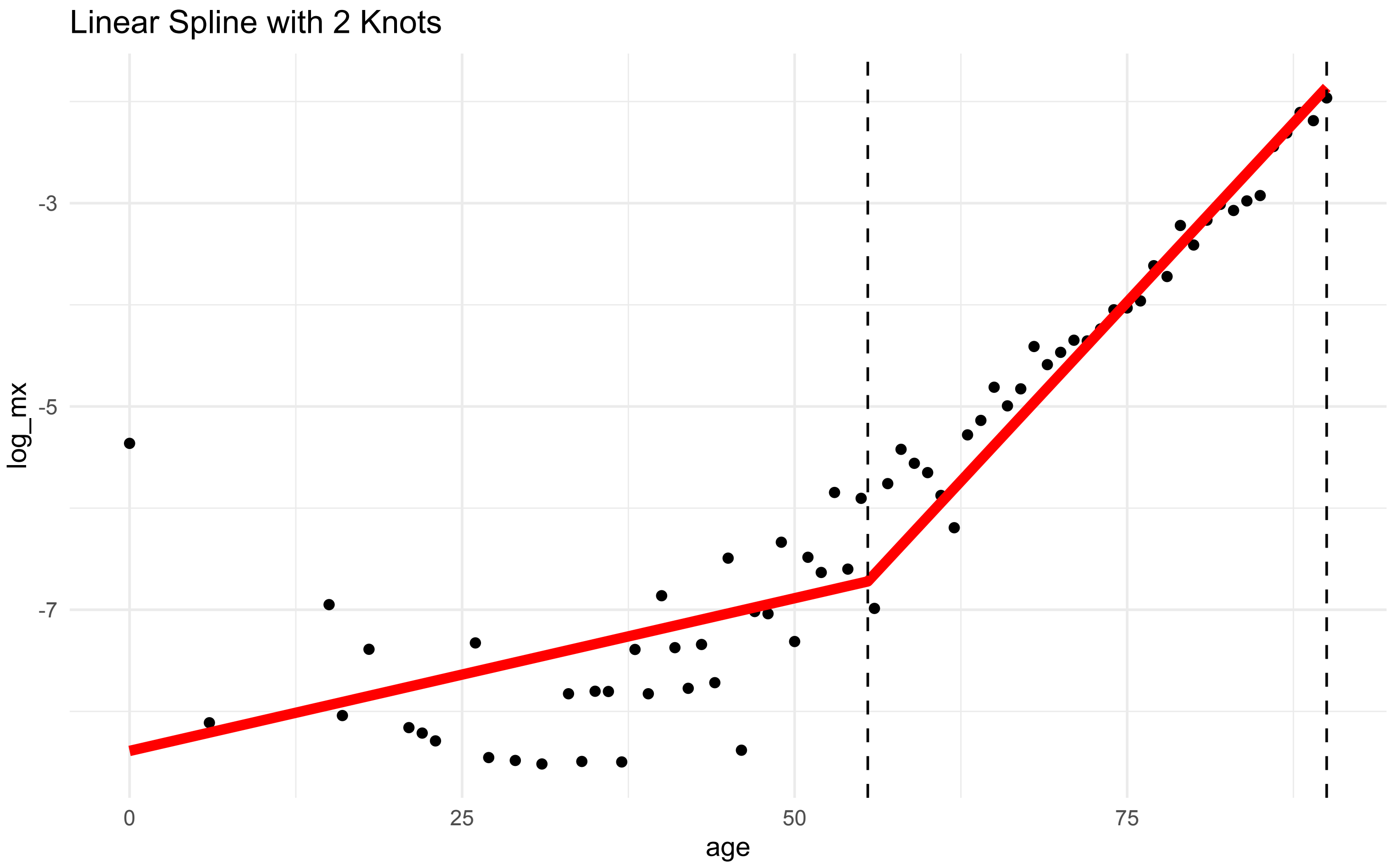
- $b_1(x_i), b_2(x_i), \ldots, b_k(x_i)$ are the basis functions
- Transform the predictor before fitting it, and split it into multiple derived "predictors"
- For polynomial regression, $b_j(x_i) = x_i^j$
- For step function regression, $b_j(x_i) = I(c_j \leq x_i < c_{j+1})$ if $j = 1, \ldots, k-1$

Regression Splines

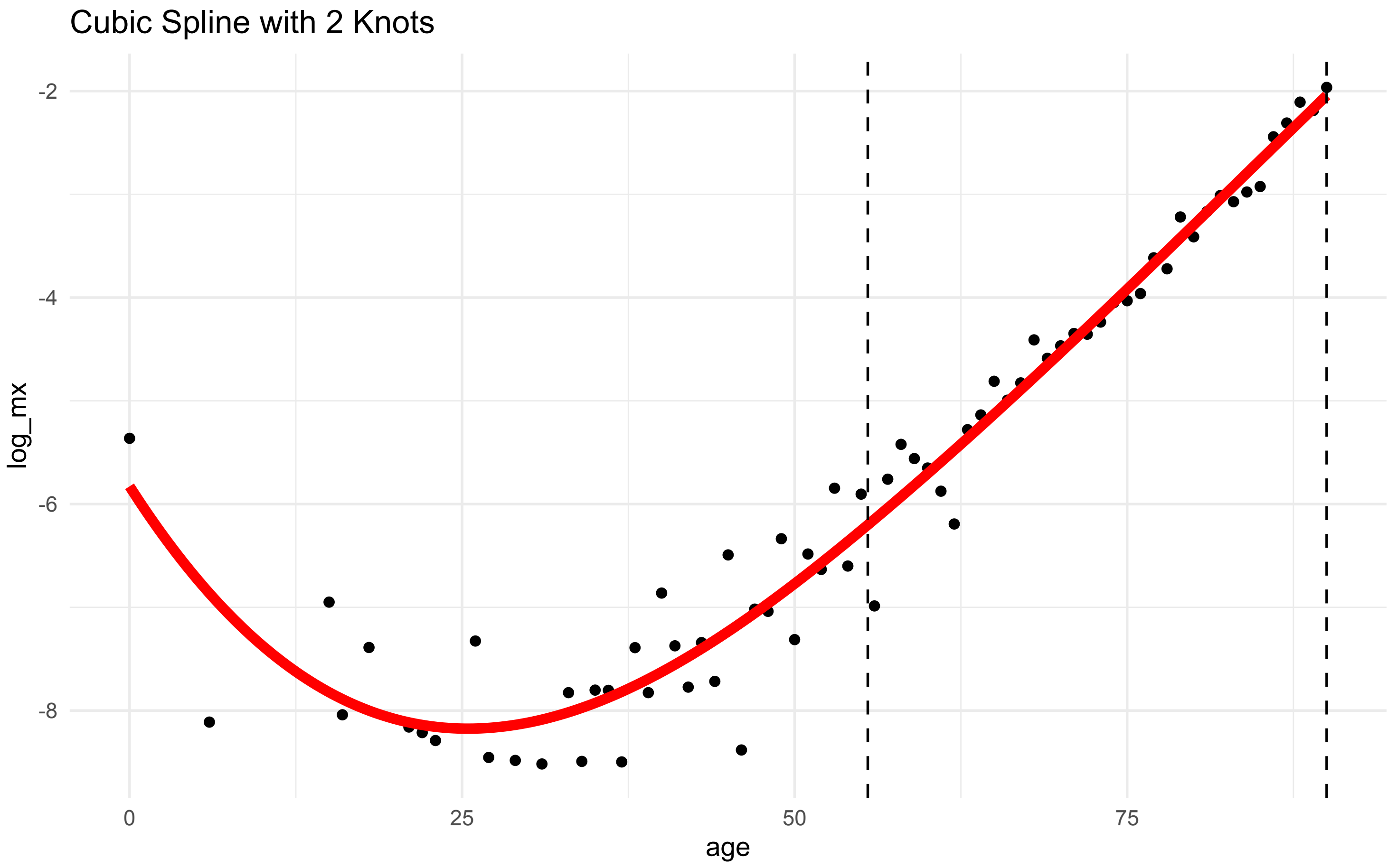
Recommended viewing

The Continuity of Splines

Example: Piecewise linear



Example: Piecewise cubic



Example: Piecewise cubic regression

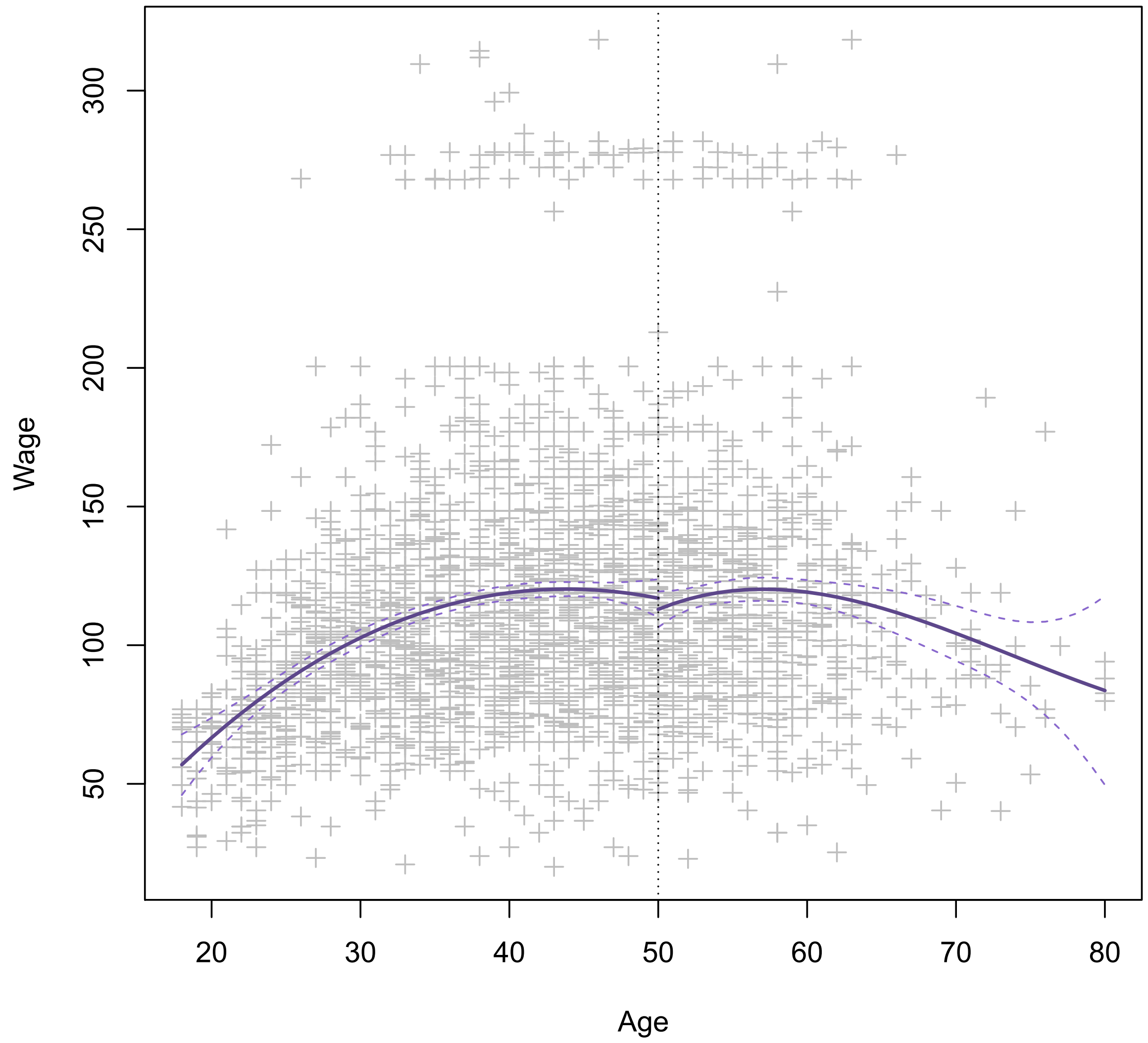
Example: Fitting a piecewise cubic polynomial with one "knot"

$$y_i = \begin{cases} \beta_{0,1} + \beta_{1,1}x_i + \beta_{2,1}x_i^2 + \beta_{3,1}x_i^3 & \text{if } x_i < c \\ \beta_{0,2} + \beta_{1,2}x_i + \beta_{2,2}x_i^2 + \beta_{3,2}x_i^3 & \text{if } x_i \geq c \end{cases}$$

- Each cubic equation is a spline
- c is a knot: a point of our choosing where the model changes from one to another

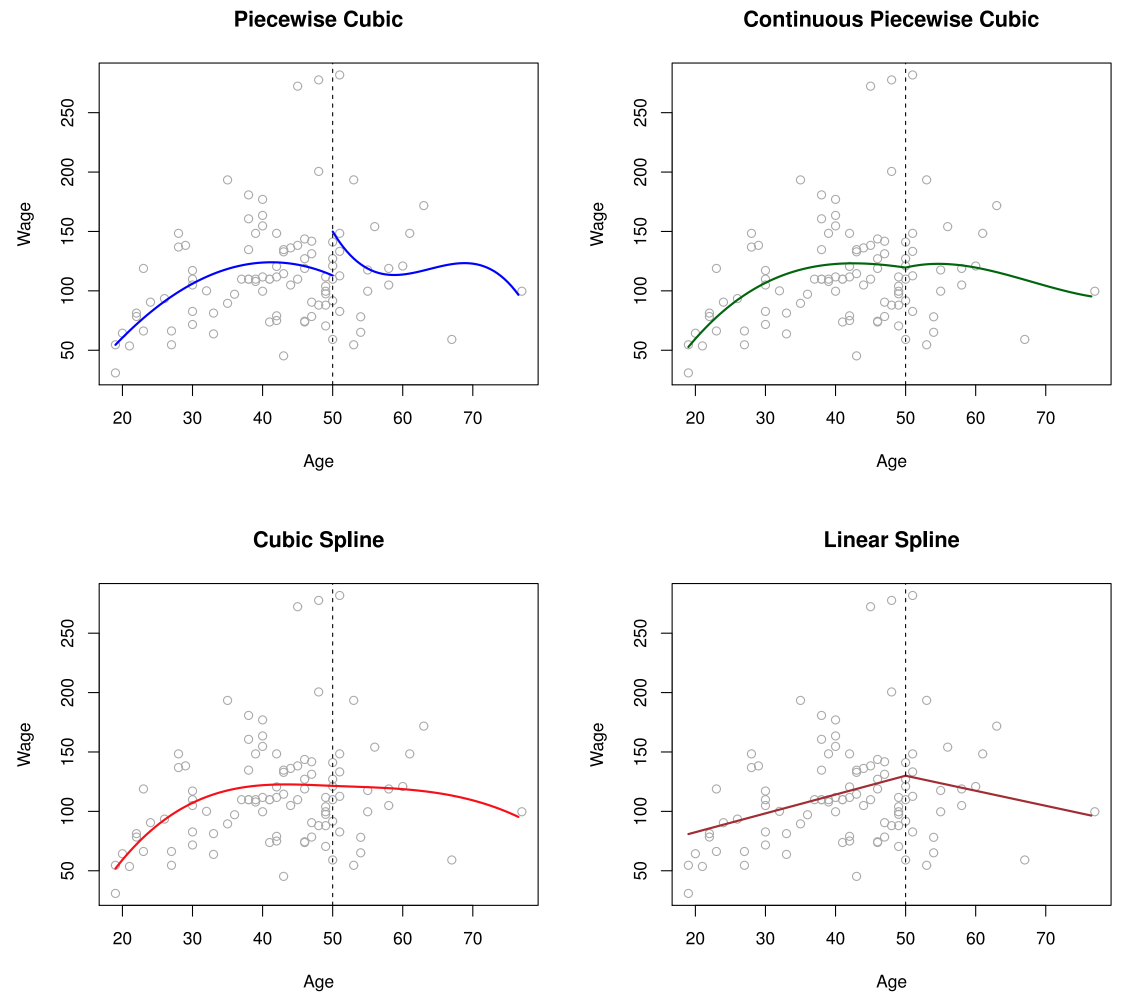
UNSW SYDNEY

Unconstrained cubic regression



Unconstrained cubic regression on `Wage` data

Examples: Different types of splines



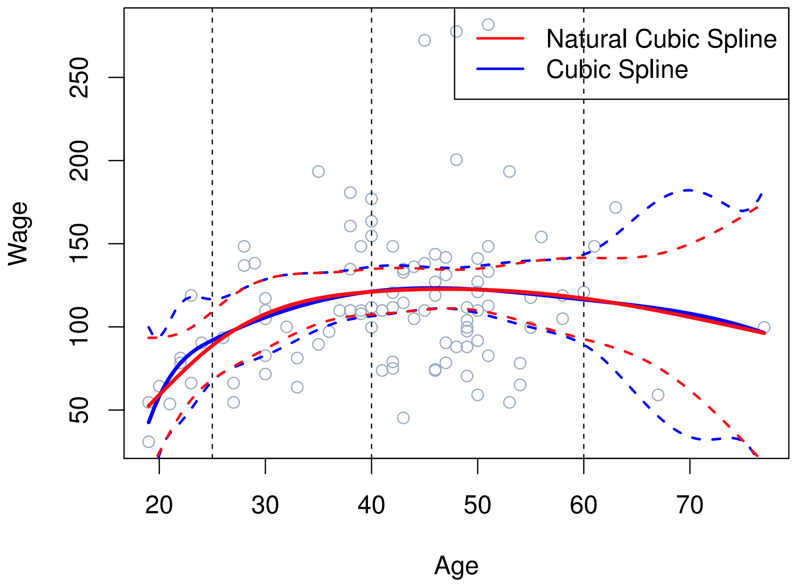
Four varieties of splines fit on a subset of the `Wage` data

Source: James et al. (2021), *An Introduction to Statistical Learning with Applications in R*, Figure 7.3.

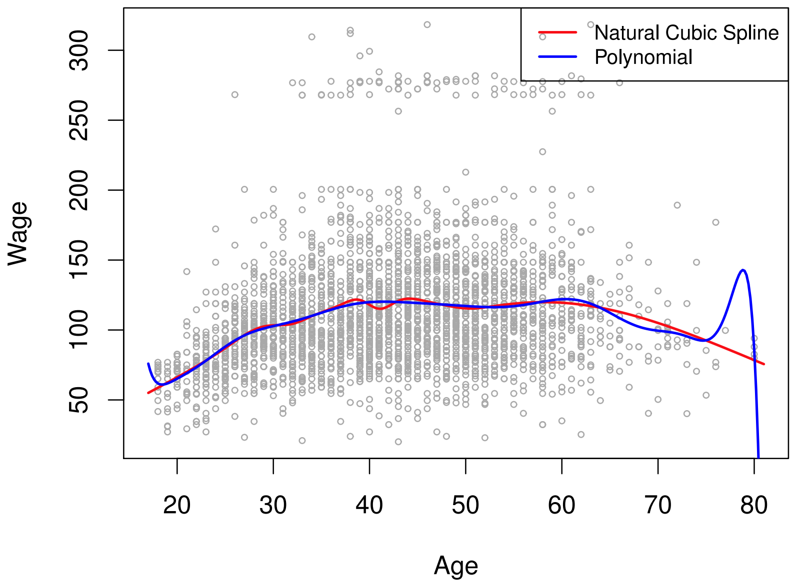
Cubic Splines: constraints and knots

- In order to have smoothness, one can impose further constraints on a cubic spline
  - Continuity
  - Continuity in 1st and 2nd derivatives
  - Linearity at the boundaries
- Procedure is similar for splines of different degrees, but cubic is preferred since knots aren't visible without very close inspection
- Discussion: How can one determine the location and number of the knots?

Natural cubic splines on `Wage` data



Cubic spline & natural cubic spline fit to `Wage` subset



Degree-15 spline & natural cubic spline fit to `Wage` data

Source: James et al. (2021), *An Introduction to Statistical Learning with Applications in R*, Figures 7.4 and 7.7.

Smoothing Splines

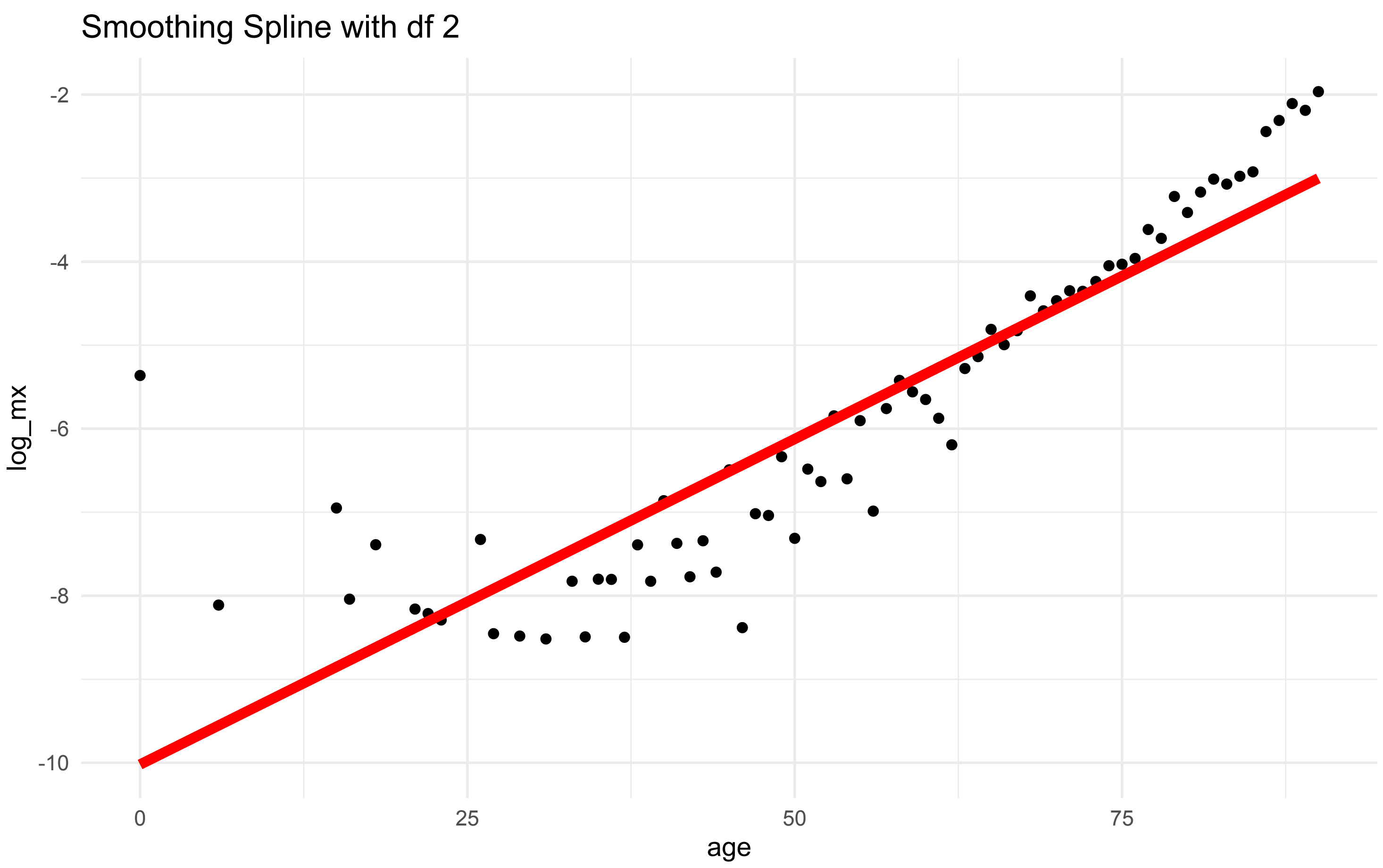
Smoothing splines

Find a function g which minimises:

$$\sum_{i=1}^{n}(y_i - g(x_i))^2 + \lambda \int g''(t)^2 dt$$

- Goal: fit a function which minimises the RSS whilst still being 'smooth'
- λ is the tuning parameter which penalises a rougher fit
- $\lambda = 0$: g will be very lumpy and will just interpolate all training data points (more flexible: less bias for more variance)
- $\lambda \to \infty$: g will be a straight line fit (less flexible: more bias for less variance)
- g turns out to be a (shrunken) natural cubic spline, with knots at every training data point.

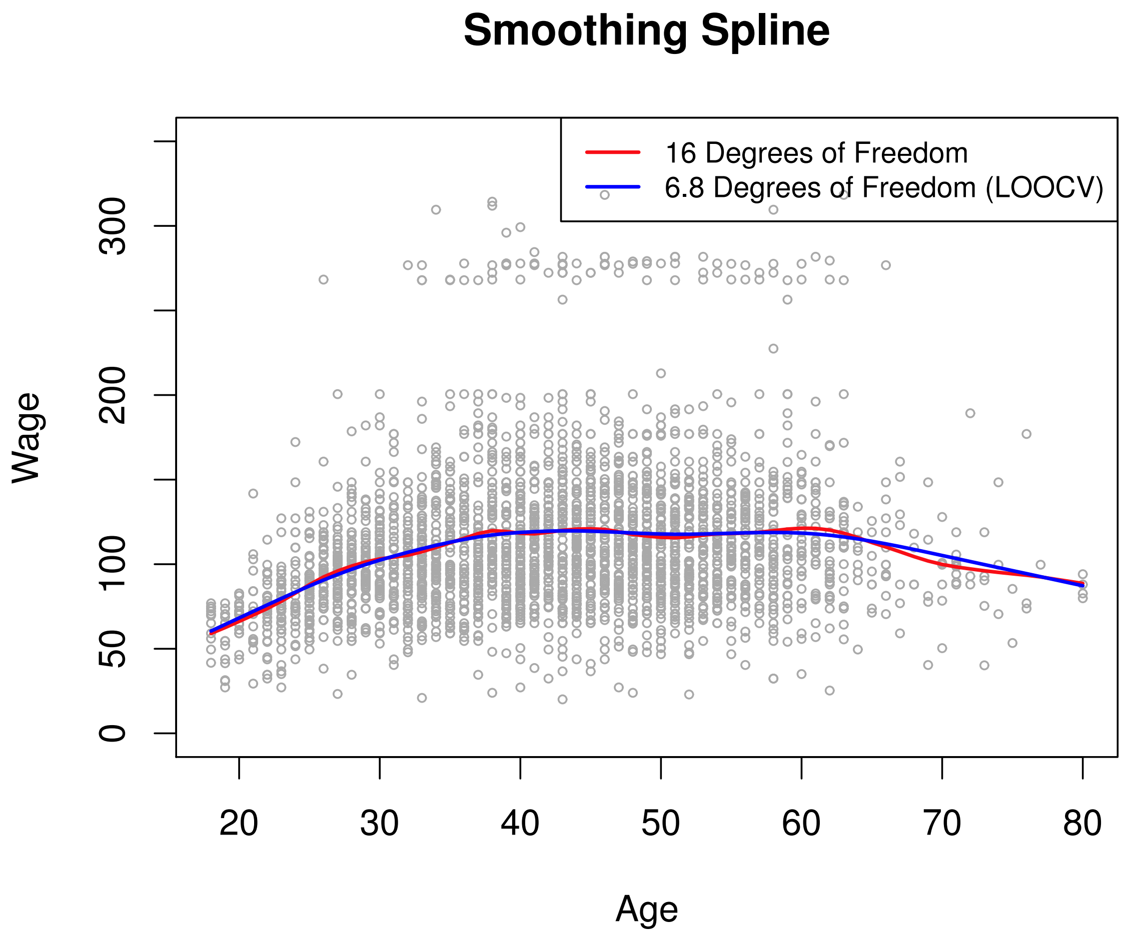
Example: Smoothing splines



Choosing λ

- df_λ: Effective degrees of freedom: measures the flexibility of the smoothing spline.
  - Can be non-integer since some variables are constrained, so they are not free to vary
  - Note that the location and degree of the knots is all determined.
- Choice of λ via Cross validation.
- In particular - LOOCV error can be computed using only ONE computation for each λ - extremely computationally efficient.

Smoothing splines on `Wage` data



Comparing smoothing splines

Source: James et al. (2021), *An Introduction to Statistical Learning with Applications in R*, Figure 7.8.

Local regression

Recommended viewing

What is LOESS and When Should I Use It?

Local regression

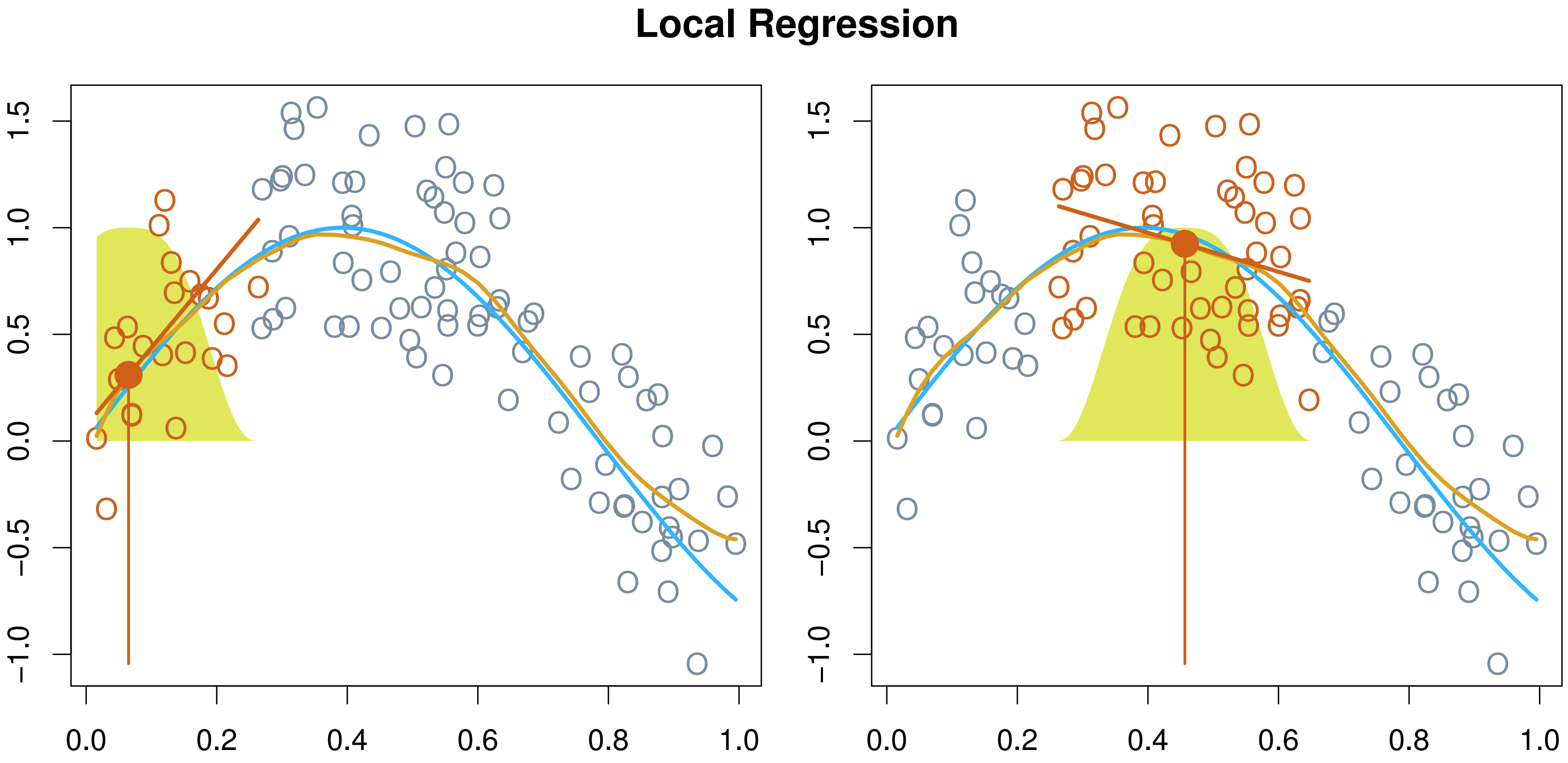
(Example) Algorithm

1. Get the fraction $s = k/n$ nearest neighbours to the point x_0
2. Assign each a weight $K_{i0} = K(x_i, x_0)$ based on how close they are to x_0. Closer: higher weight. Furthest point in the k should get weight zero. Points outside the k selected should have a zero weight as well
3. Minimise

$$\sum_{i=1}^{n} K_{i0}(y_i - \beta_0 - \beta_1 x_i)^2$$

4. $\hat{f}(x_0) = \hat{\beta}_0 + \hat{\beta}_1 x_0$

Local regression example



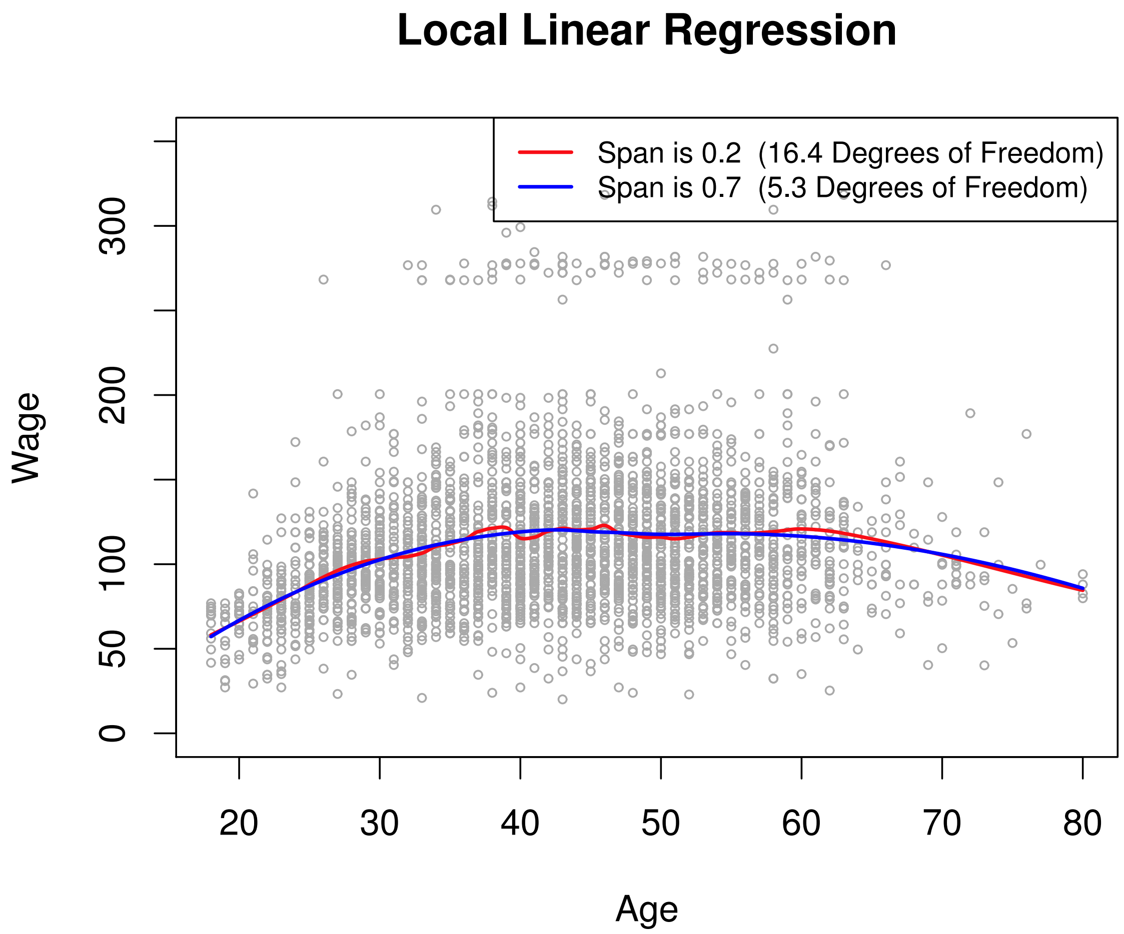
Example of making predictions with local regression at $x \approx 0.05$ and $x \approx 0.45$

Source: James et al. (2021), *An Introduction to Statistical Learning with Applications in R*, Figure 7.9.

Local regression cont.

- Local regression does a weighted regression of the points about some predictor value x_0. Can obtain an estimate for the response value at x_0 from this
- Needs to be re-run each time an estimate at a different point is desired
- Useful for adapting model to recent data
- Possible to extend to 2 or 3 predictors: just have the weights based on distance in 2D or 3D space.
- Things start to get problematic if $p > 4$ as there will be very few training observations

Local regression on Wage data



You can adjust the smoothness by changing the span

Source: James et al. (2021), *An Introduction to Statistical Learning with Applications in R*, Figure 7.10.

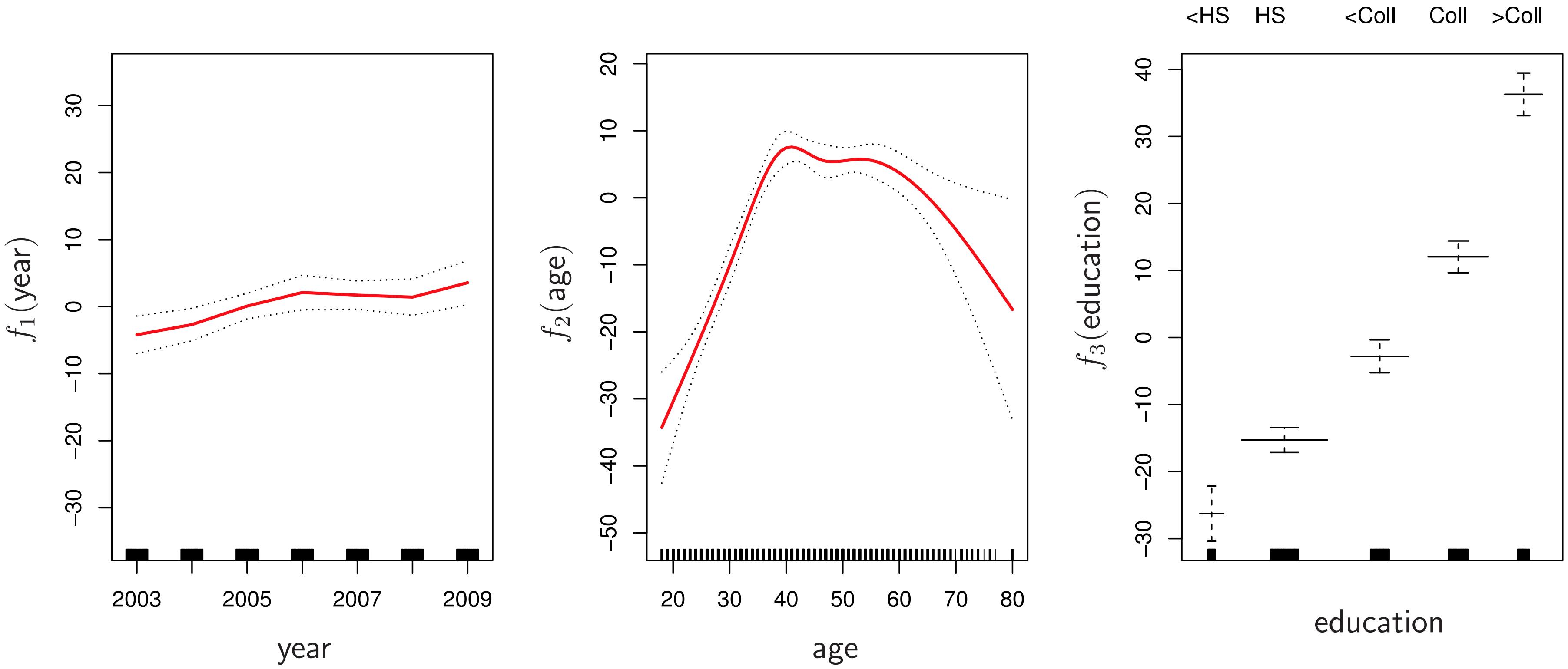
Generalised additive models (GAMs)

GAMs

$$y_i = \beta_0 + f_1(x_{i,1}) + f_2(x_{i,2}) + \ldots + f_p(x_{i,p}) + \varepsilon_i$$

- Non-linearly fit multiple predictors on a response, whilst keeping the additive quality
- f_i can be virtually *any* function of the parameter, including the ones discussed earlier
- Find a separate f_i for each predictor, and add them together
- Can also be used on categorical responses in a logistic regression setting

Example: GAM on `Wage` data

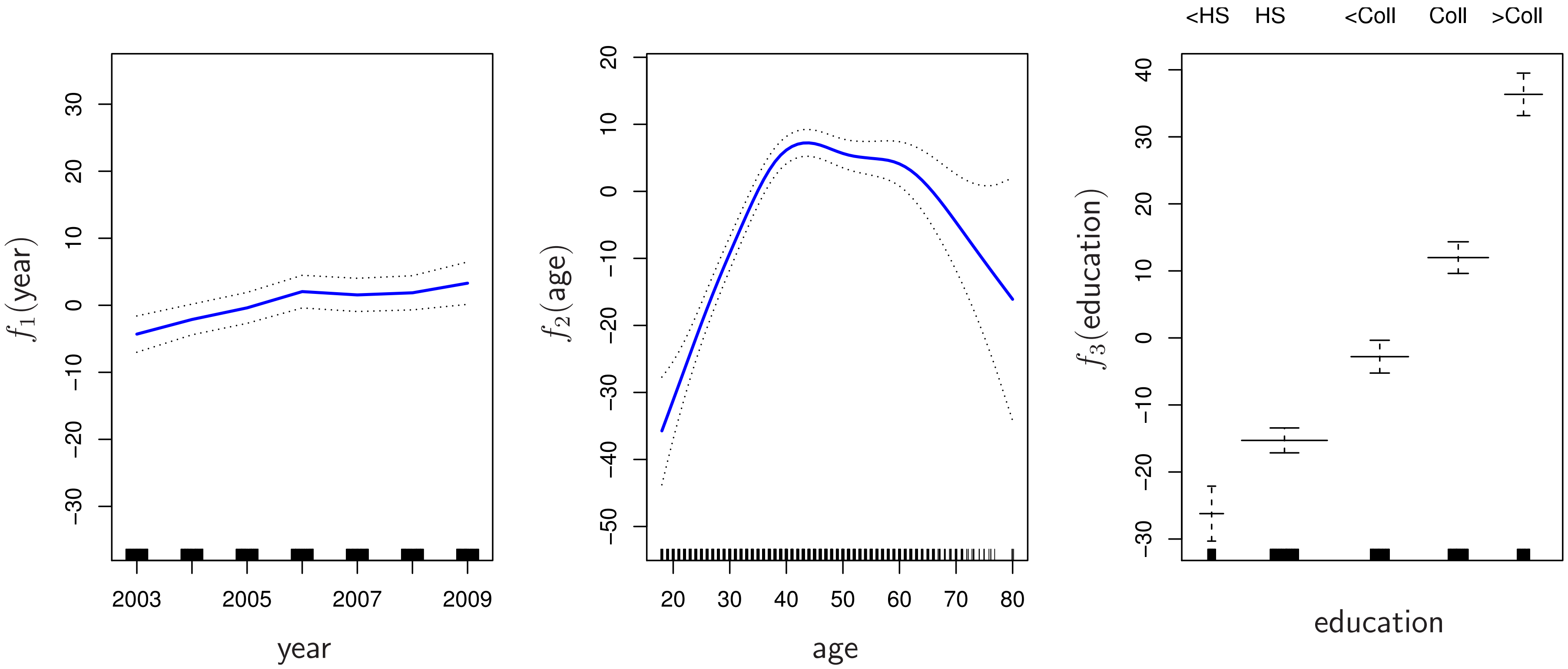


GAM fit using regression splines

- Each plot shows the contribution of each predictor to `wage`
- `education` is qualitative. The others are fit with natural cubic splines

Source: James et al. (2021), *An Introduction to Statistical Learning with Applications in R*, Figure 7.11.

Example: GAM on Wage data



GAM fit using smoothing splines

- Each plot shows the contribution of each predictor to `wage`
- `education` is qualitative. The others are fit with smoothing splines

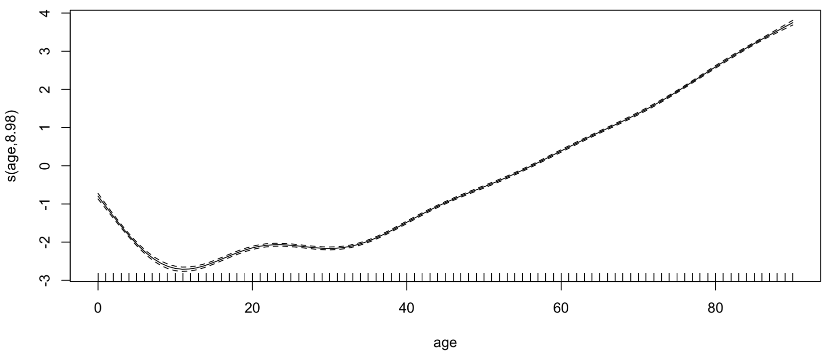
GAMs: pros and cons

- GAMs allow us to consider nonlinear relationships between the predictors and response, which can give a better fit
- Model is additive: can still interpret the effect of a single given predictor on the response
- However, model additivity ignores interaction effects between predictors. Could always add two-dimensional function parameters, e.g. $f_{j,k}(x_j, x_k)$

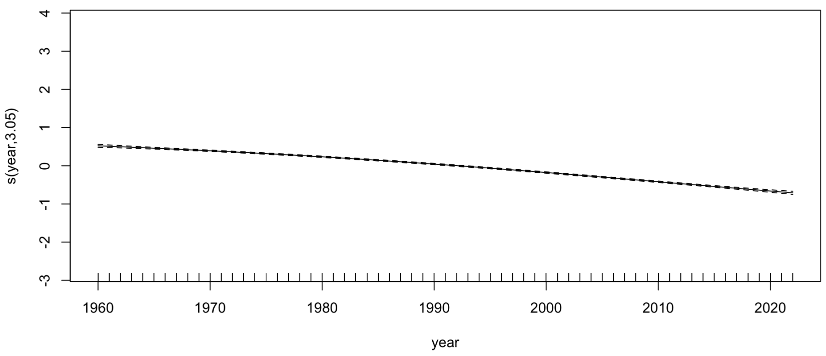
GAMs on Luxembourg data (mgcv)

```r
model_gam <- gam(log_mx ~ s(age) + s(year), data=lux)
```

```r
plot(model_gam, select=1)
```

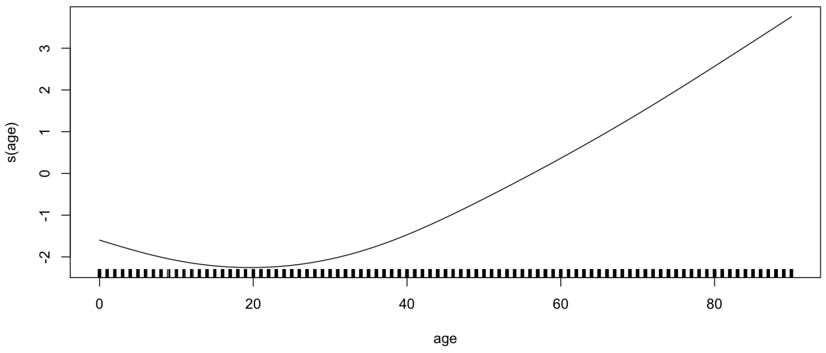


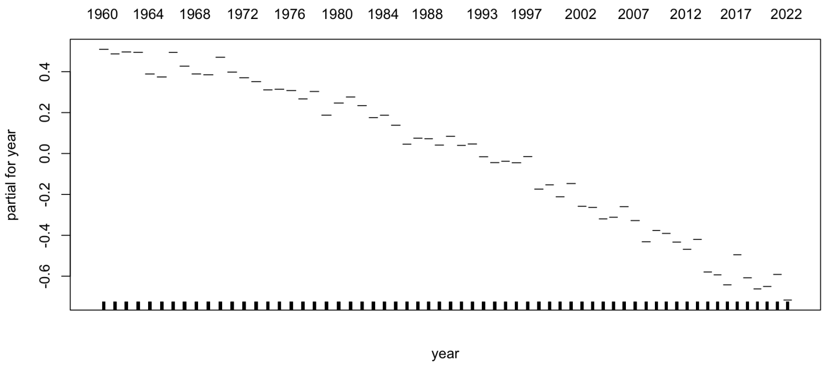

```r
plot(model_gam, select=2)
```

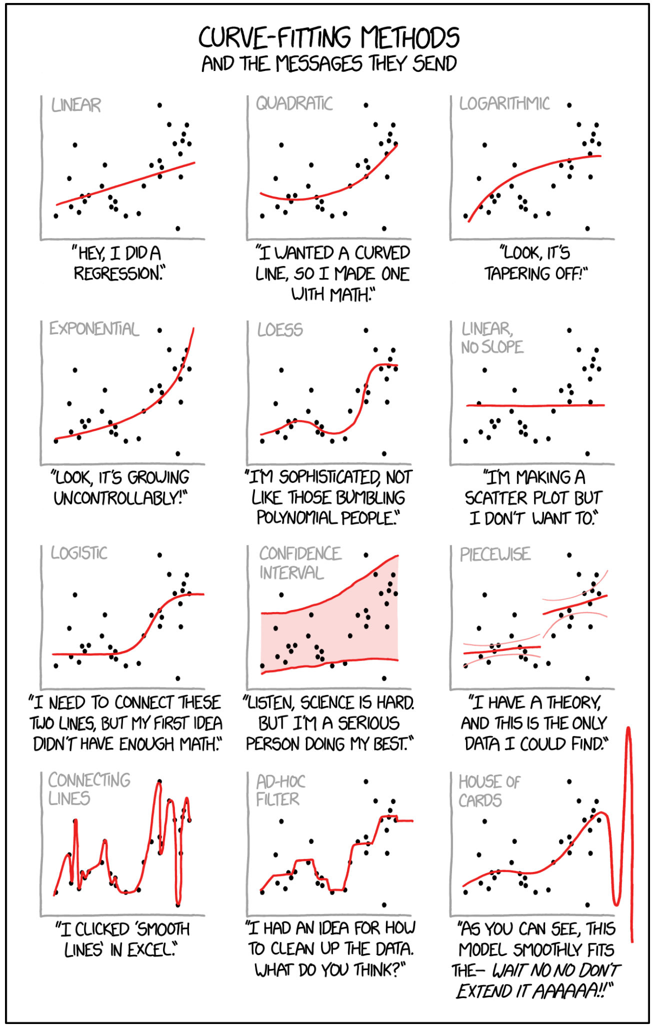


GAMs on Luxembourg data (gam)

```
library(gam)
lux_factor <- lux %>% mutate(year = factor(ye
model_gam <- gam(log_mx ~ s(age) + year, data
plot(model_gam)
```







Source: xkcd

Appendix

Glossary

- interpolation & extrapolation
- polynomial regression
  - monomials
  - orthogonal polynomials
- step functions
  - basis function expansion
  - piecewise polynomial functions
- regression splines
  - knots
  - natural splines
  - cubic splines
- smoothing splines
- local regression
- generalised additive models (GAMs)

OLS solution

$$\boldsymbol{y} = \mathbf{X}\boldsymbol{\beta} + \boldsymbol{\varepsilon}$$

Sum of squared residuals:

$$L(\boldsymbol{\beta}) = (\boldsymbol{y} - \mathbf{X}\boldsymbol{\beta})^\top(\boldsymbol{y} - \mathbf{X}\boldsymbol{\beta})$$

Find the minimum by taking the derivative and setting to zero:

$$\frac{\partial L}{\partial \boldsymbol{\beta}} = -2\mathbf{X}^\top(\boldsymbol{y} - \mathbf{X}\boldsymbol{\beta}) = 0$$

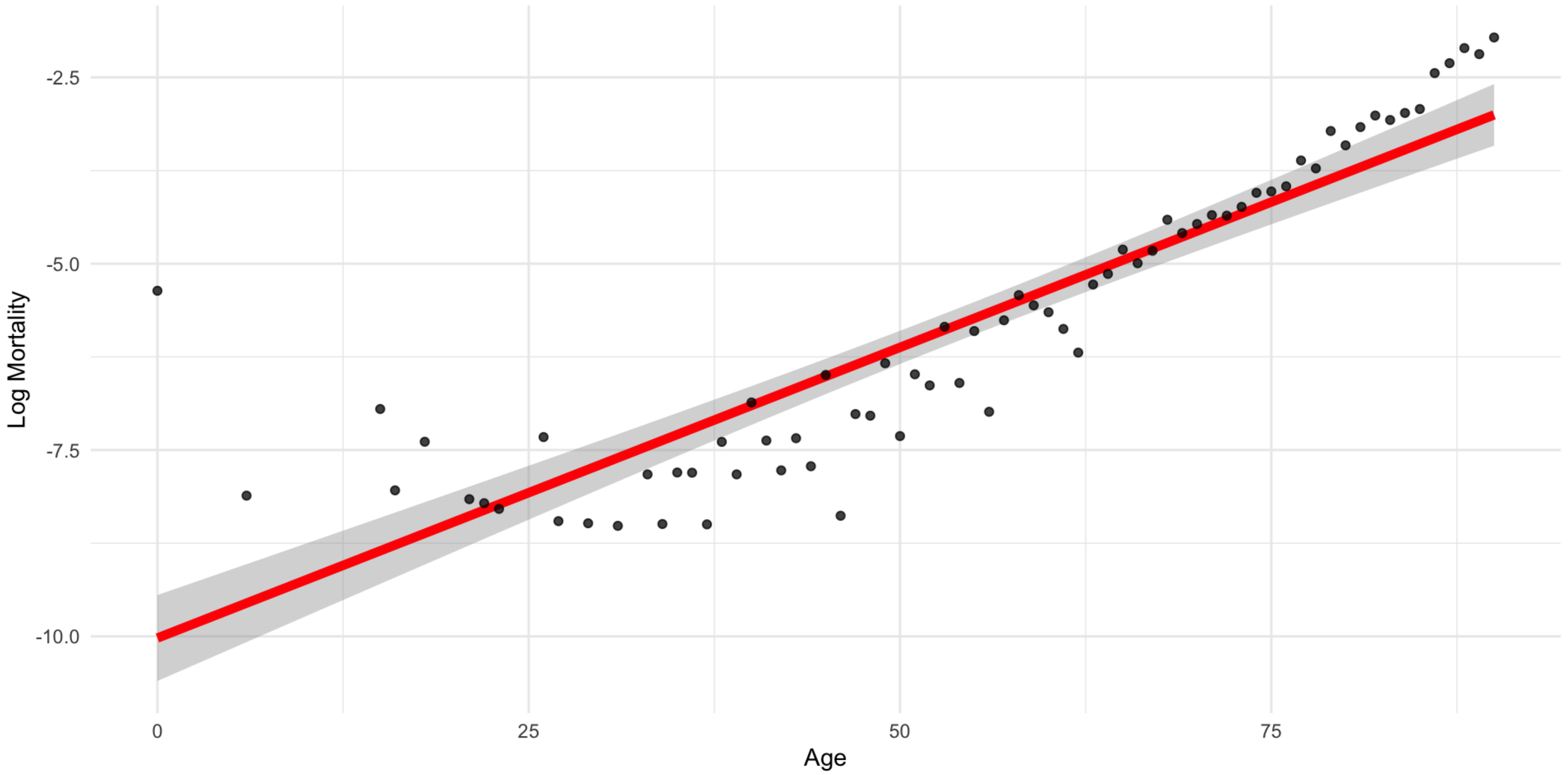
$$\mathbf{X}^\top\mathbf{X}\boldsymbol{\beta} = \mathbf{X}^\top\boldsymbol{y}$$

$$\therefore \boldsymbol{\beta} = (\mathbf{X}^\top\mathbf{X})^{-1}\mathbf{X}^\top\boldsymbol{y}$$

Linear regression with error bars

R | Python

```
ggplot(lux_2020, aes(x = age, y = log_mx)) + theme_minimal() +
  geom_smooth(method = "lm", formula = y ~ x, color = "red", linewidth=2) +
  geom_point(alpha = 0.75) + labs(x = "Age", y = "Log Mortality")
```



Quadratic regression with error bars

```r
ggplot(lux_2020, aes(x = age, y = log_mx)) + theme_minimal() +
  stat_smooth(method = "lm", formula = y ~ poly(x, 2), color = "red", linewidth=2) +
  geom_point(alpha = 0.75) + labs(x = "Age", y = "Log-Mortality")
```