Moving Beyond Linearity

ACTL3142 & ACTL5110 Statistical Machine Learning for Risk Applications

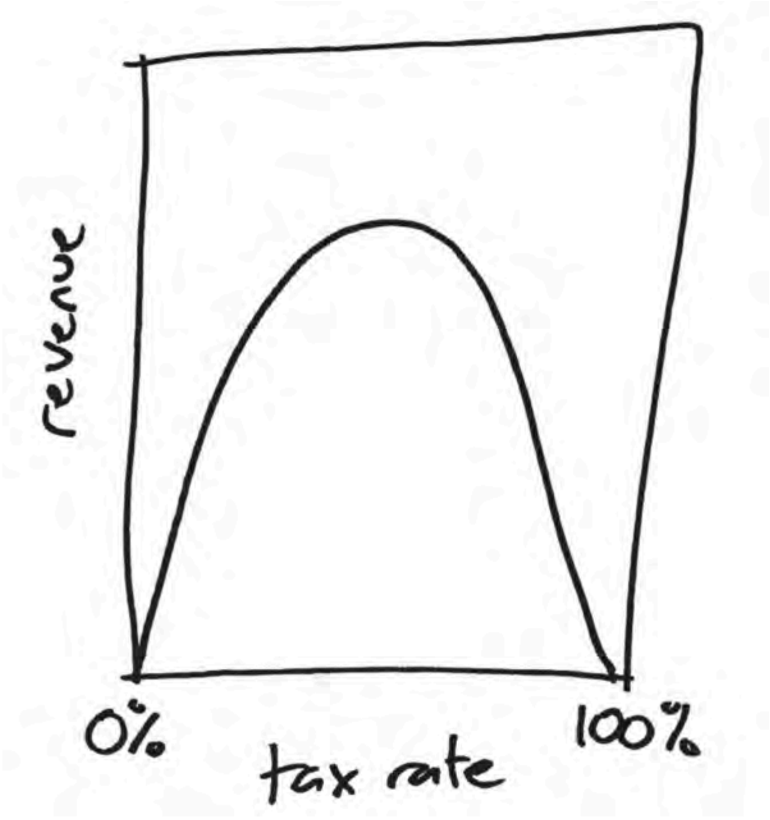
Some of the figures in this presentation are taken from "An Introduction to Statistical Learning, with applications in R" (Springer, 2013) with permission from the authors: G. James, D. Witten, T. Hastie and R. Tibshirani

Lecture Outline

- **Linearity & Nonlinearity**
- Data Science Starts With Data
- Linear Regression
- Polynomial Regression
- Step Functions
- Regression Splines
- Smoothing Splines
- Local Regression
- Generalised Additive Models (GAMs)

Nonlinear curves

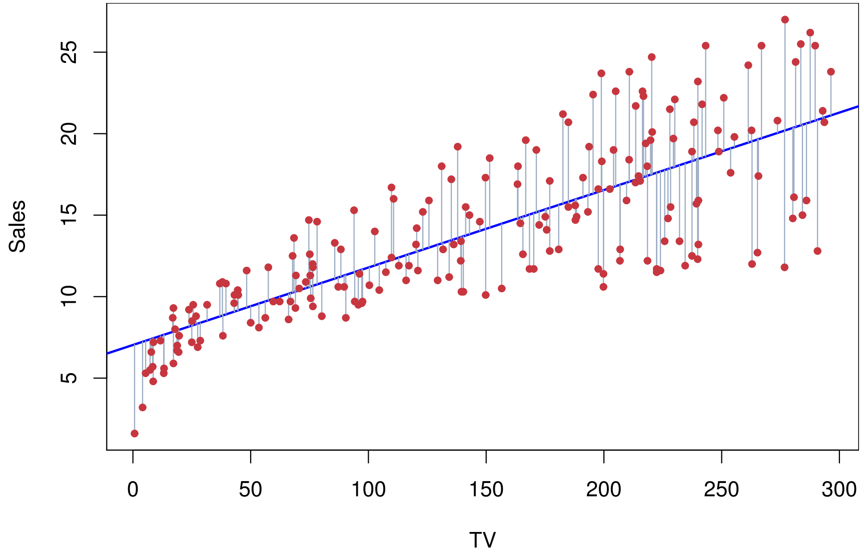
> The legend of the Laffer curve goes like this: Arthur Laffer, then an economics professor at the University of Chicago, had dinner one night in 1974 with Dick Cheney, Donald Rumsfeld, and *Wall Street Journal* editor Jude Wanniski at an upscale hotel restaurant in Washington DC. They were tussling 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

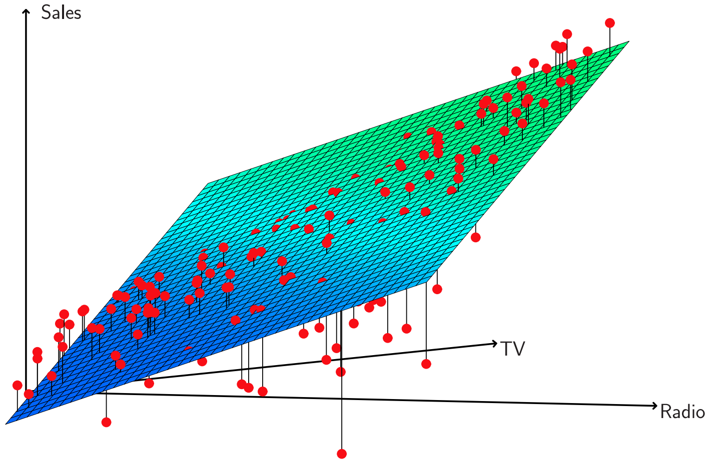
One predictor vs multiple predictors

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



Linear regression

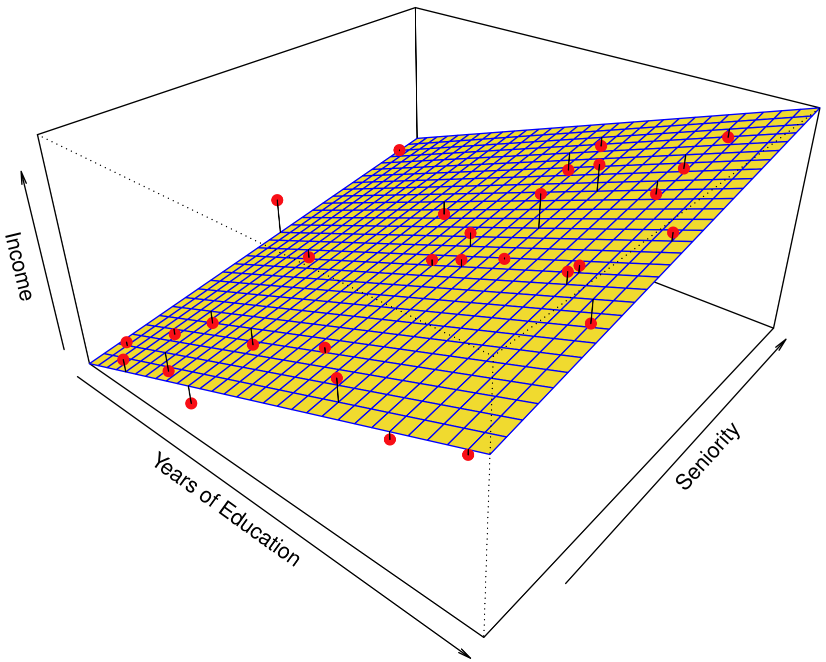
$$\texttt{sales} \approx \beta_0 + \beta_1 \times \texttt{TV} + \beta_2 \times \texttt{radio}$$



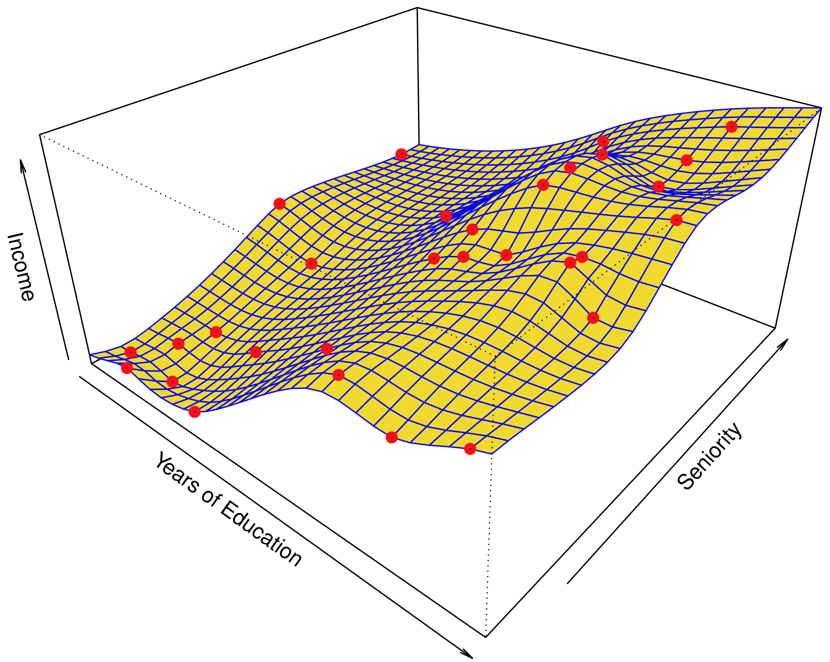
Multiple linear regression

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.

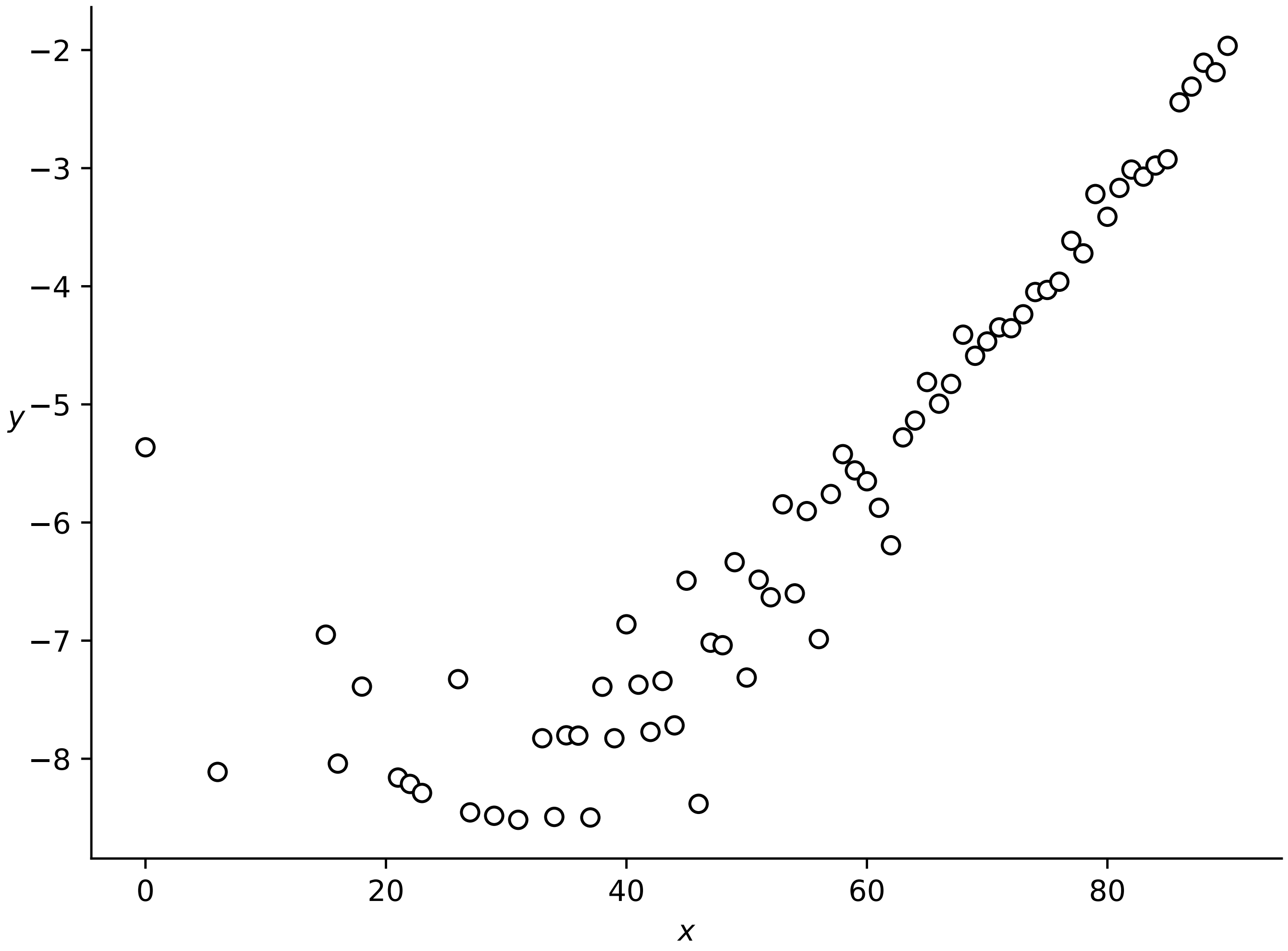
Moving beyond linearity

> 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

In-class demonstration



Some mystery data

Instructions

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

Lecture Outline

- Linearity & Nonlinearity
- **Data Science Starts With Data**
- Linear Regression
- Polynomial Regression
- Step Functions
- Regression Splines
- Smoothing Splines
- Local Regression
- Generalised Additive Models (GAMs)

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
```

Setup & importing the data

R | Python

```r
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))
```

```r
lux
```

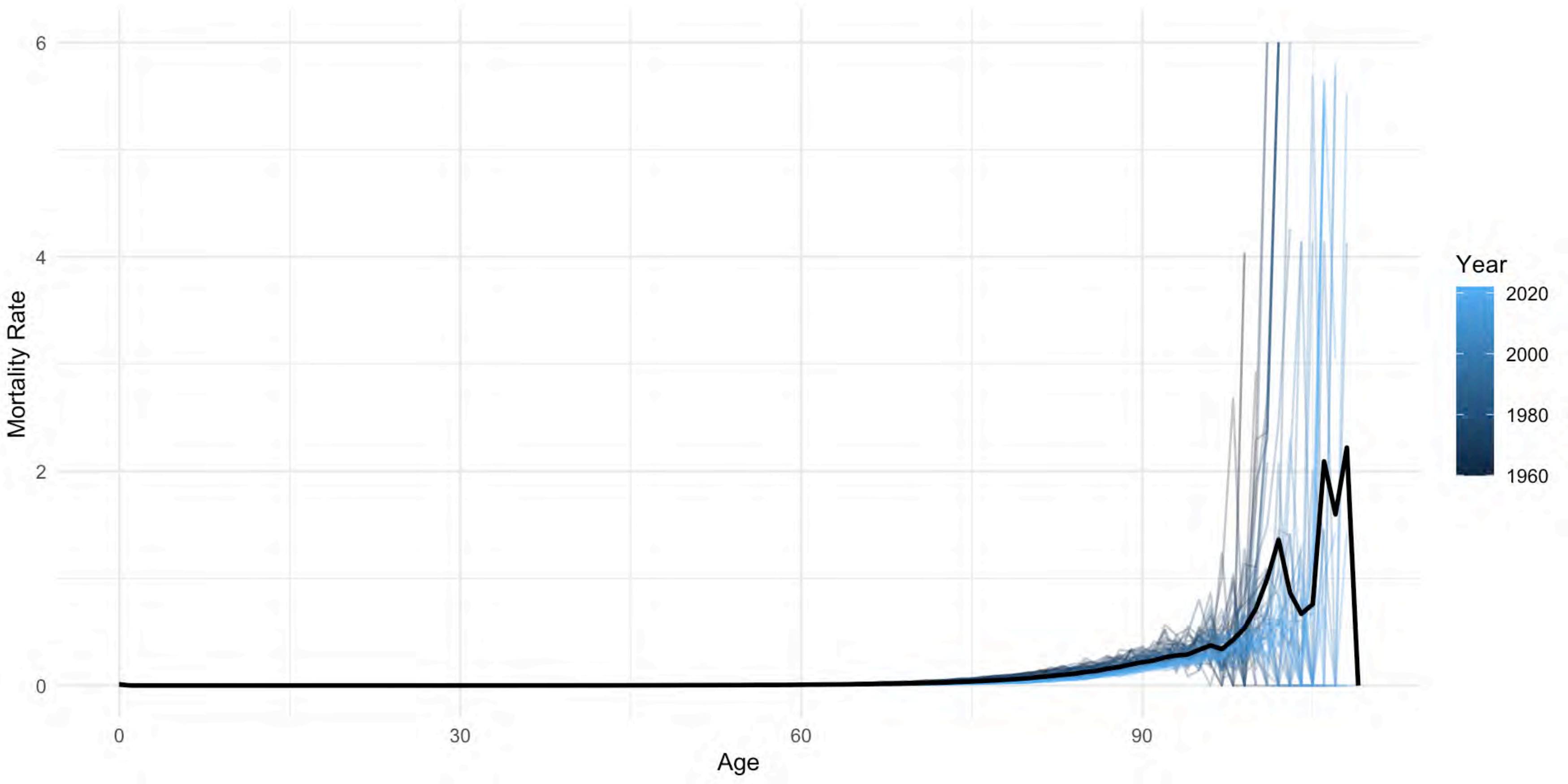
```
# A tibble: 6,930 × 3
     age  year        mx
   <int> <int>     <dbl>
 1     0  1960 0.0239
 2     1  1960 0.00169
 3     2  1960 0.00171
 4     3  1960 0.00126
 5     4  1960 0.000844
 6     5  1960 0.000873
 7     6  1960 0.000443
 8     7  1960 0
 9     8  1960 0.000951
10     9  1960 0
# ℹ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

```r
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

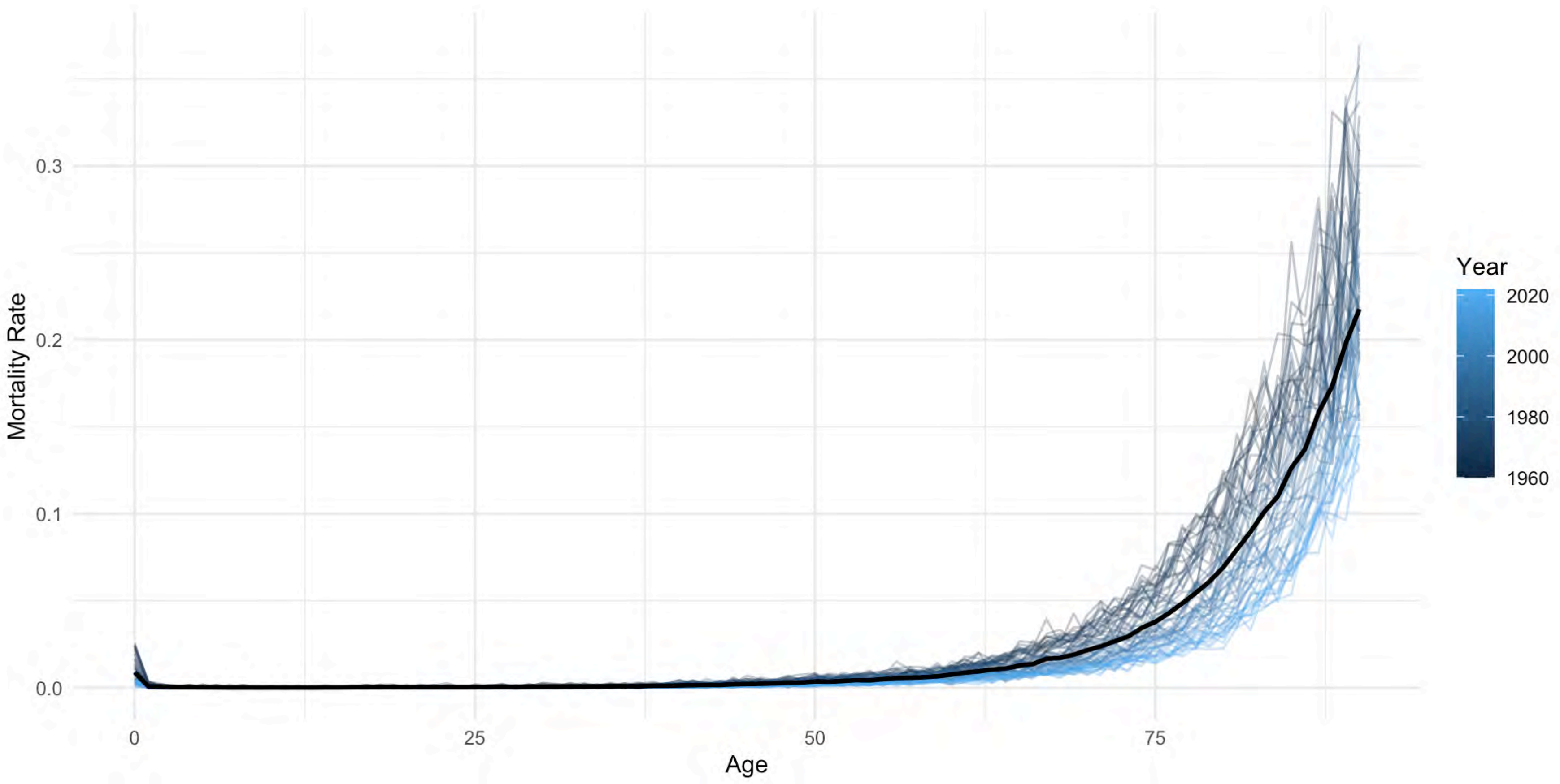
R | Python



Mortality (zoom in)

R | Python

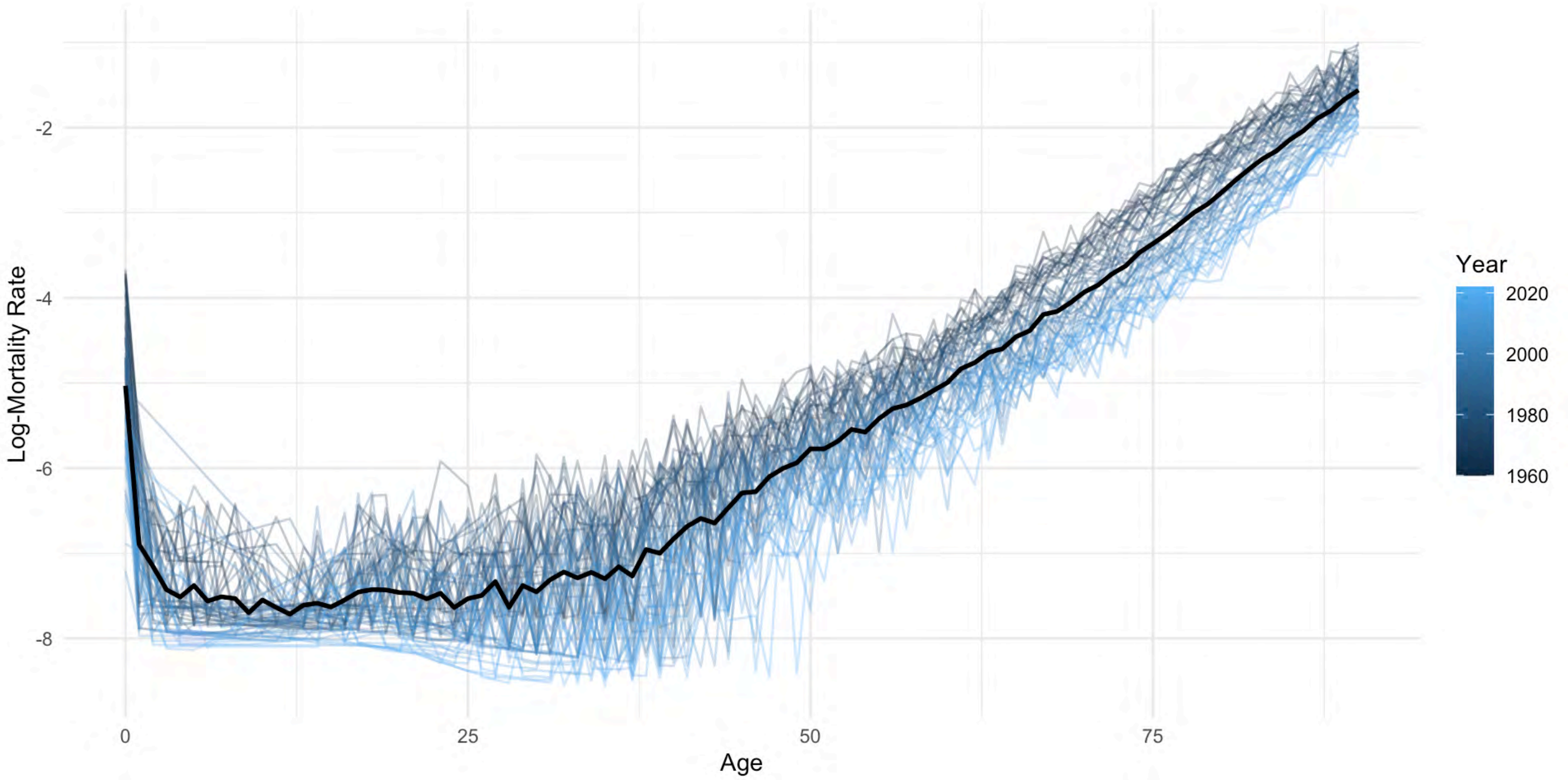
```r
lux <- lux %>% filter(age <= 90)
```



Log-mortality

R Python

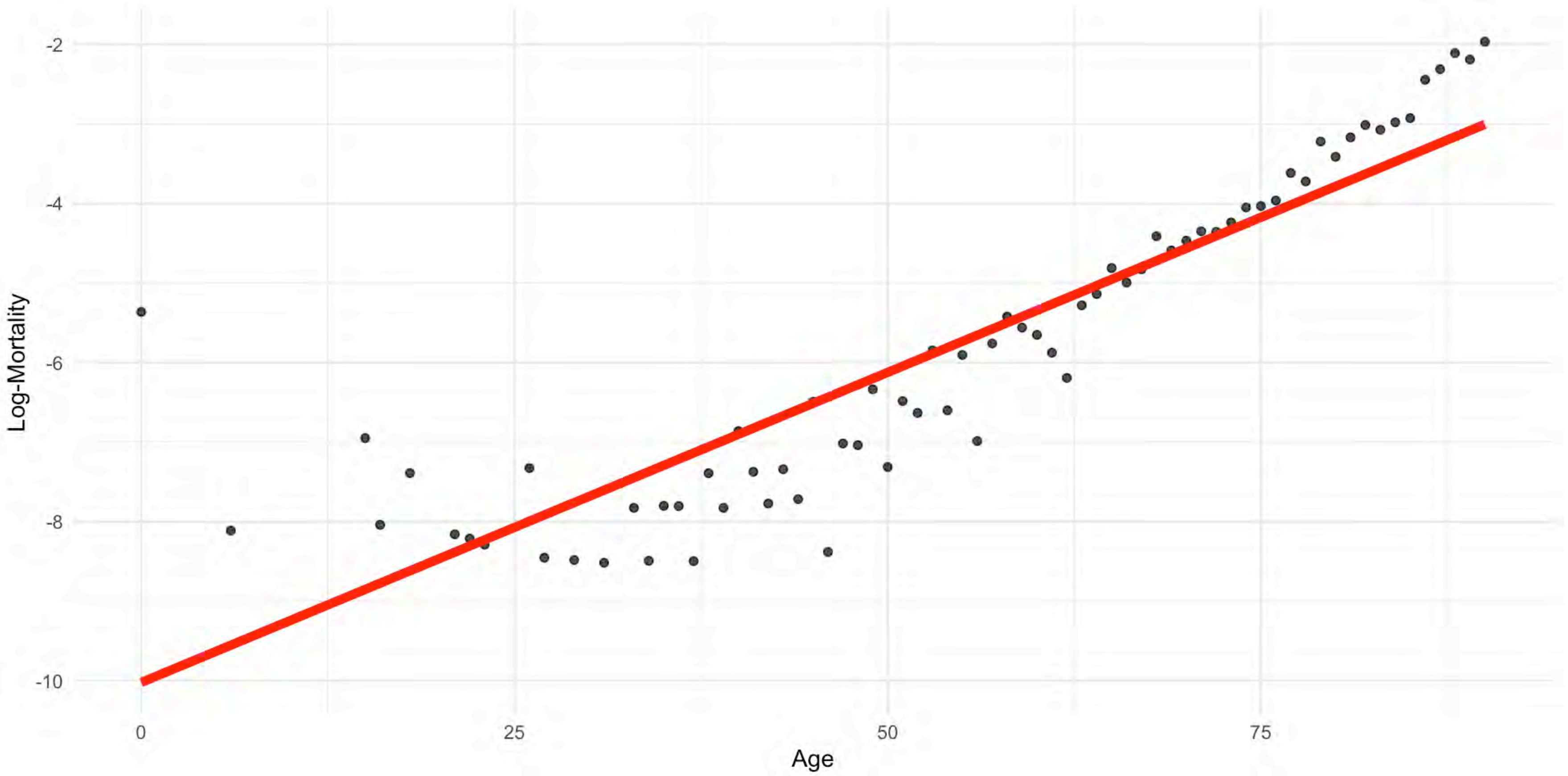
```
lux$log_mx <- log(lux$mx)
lux <- lux[lux$log_mx != -Inf, ]
```



Linear regression

R | Python

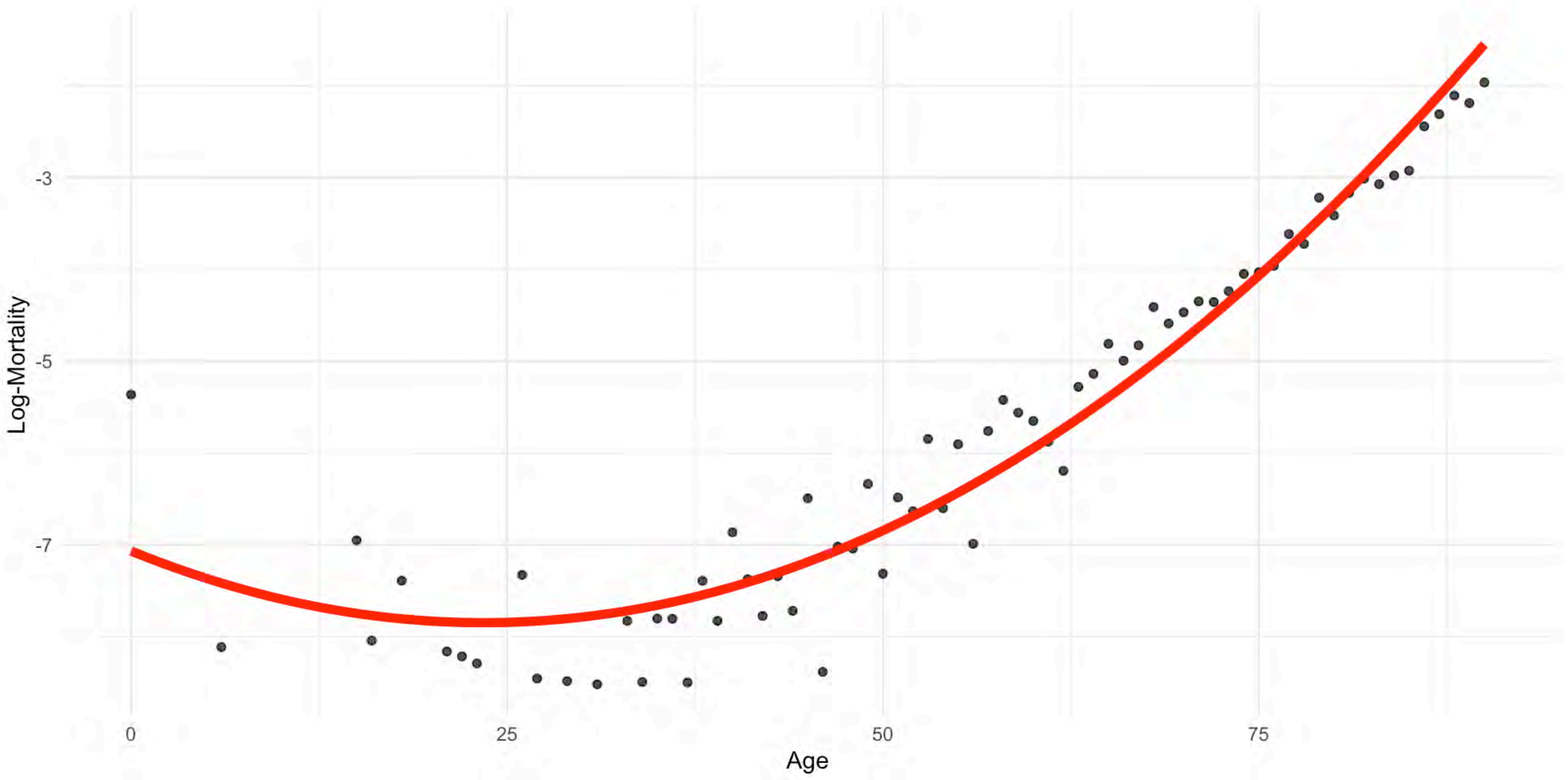
```r
lux_2020 <- lux %>% filter(year == 2020)
model_lr <- lm(log_mx ~ age, data = lux_2020)
```



Quadratic regression

R | Python

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

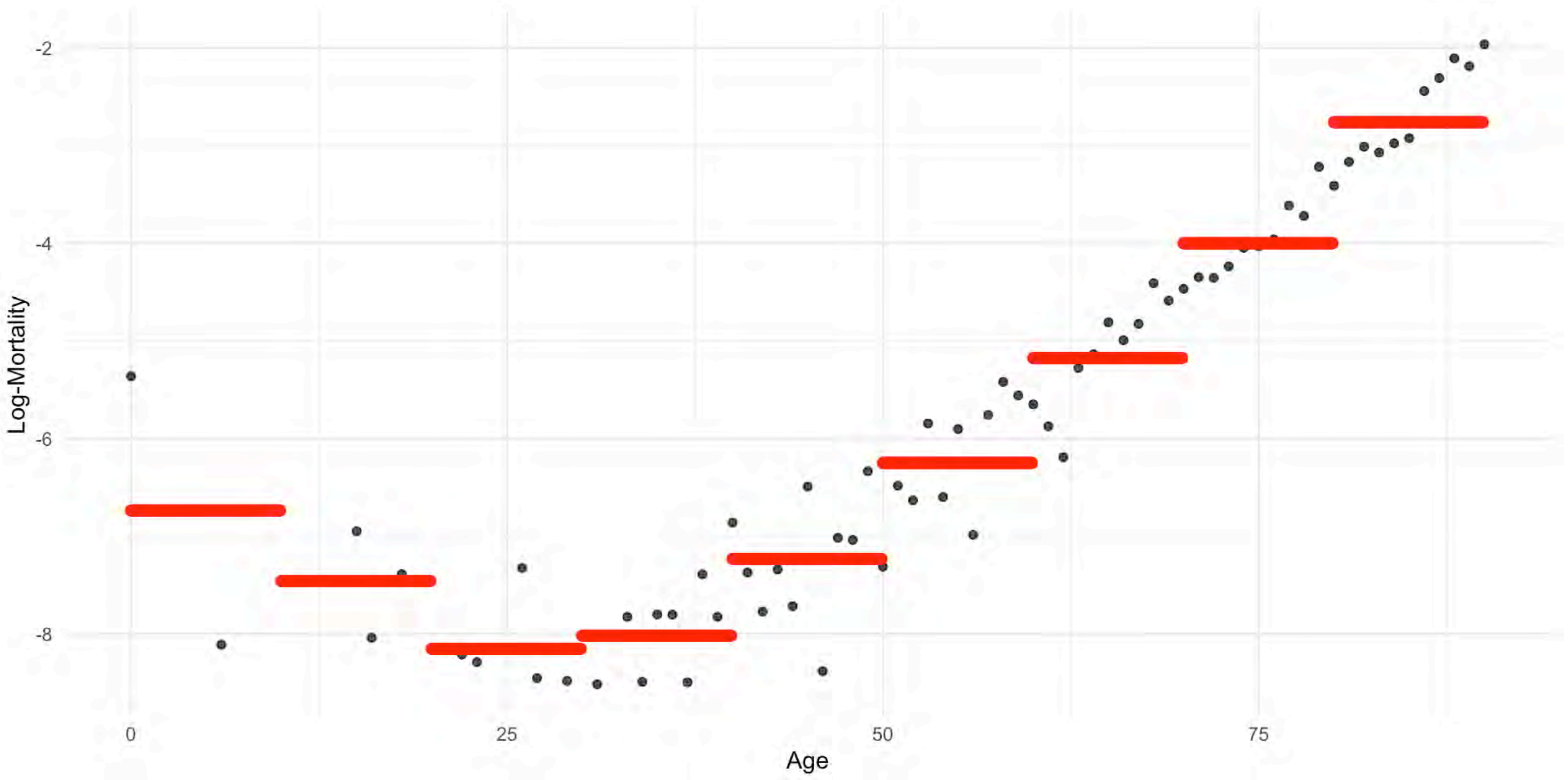


UNSW SYDNEY

Step function regression

R | Python

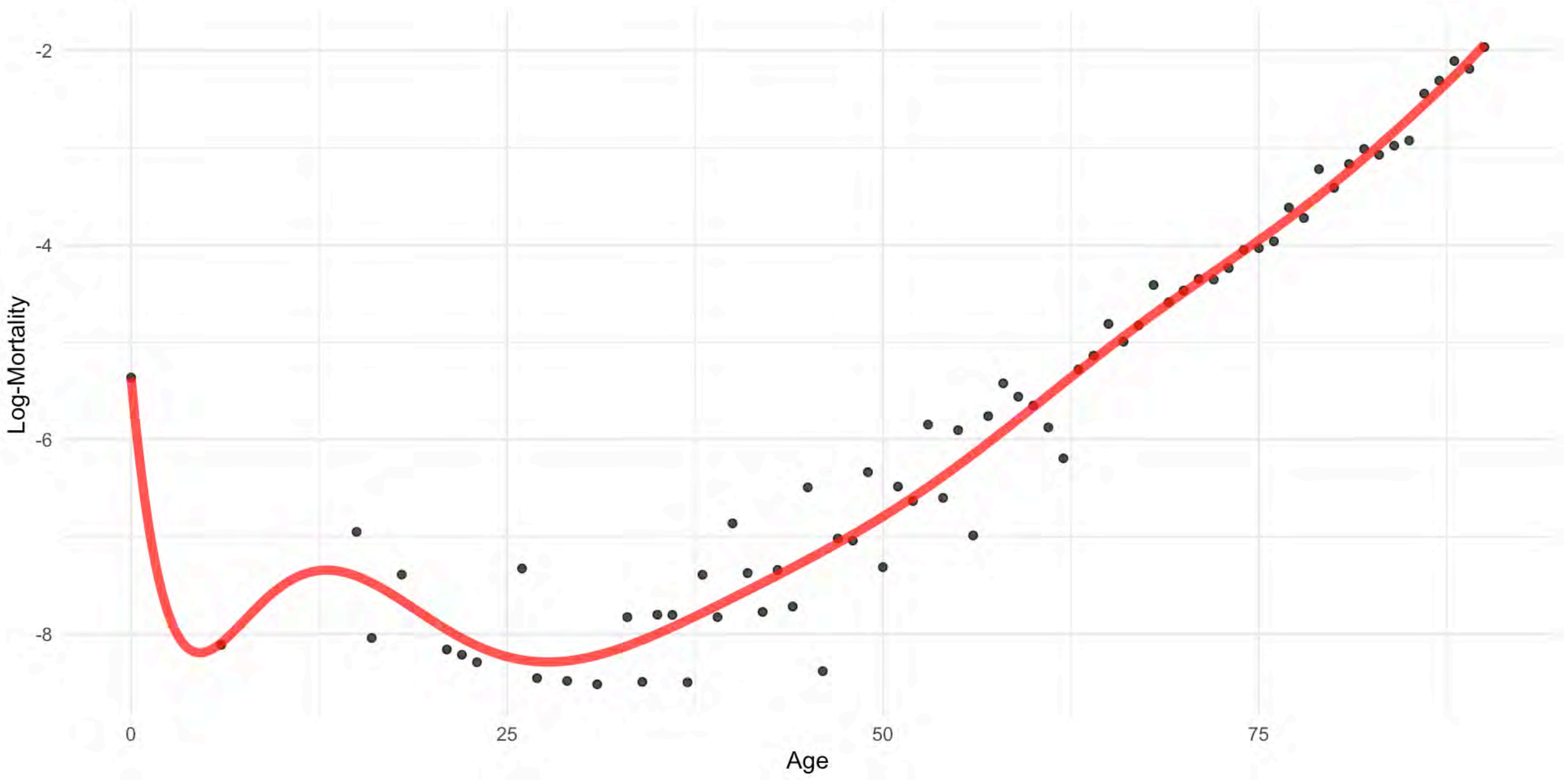
```r
model_step <- lm(log_mx ~ cut(age, seq(0, 90, 10), right=F), data = lux_2020)
```



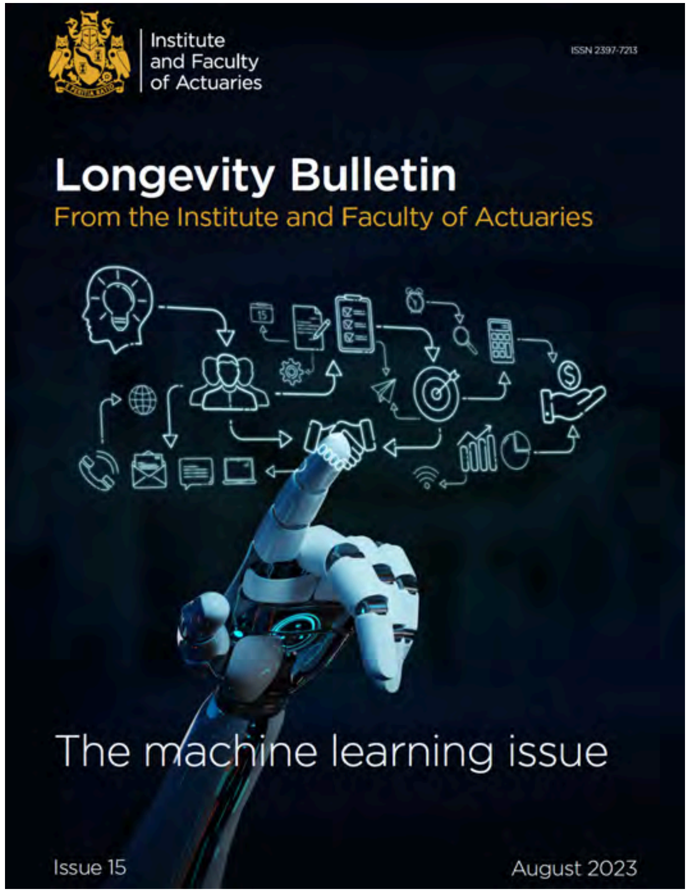
Regression spline

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)

Take **ACTL3141/ACTL5104** for mortality modelling, **ACTL3143/ACTL5111** for actuarial AI

Lecture Outline

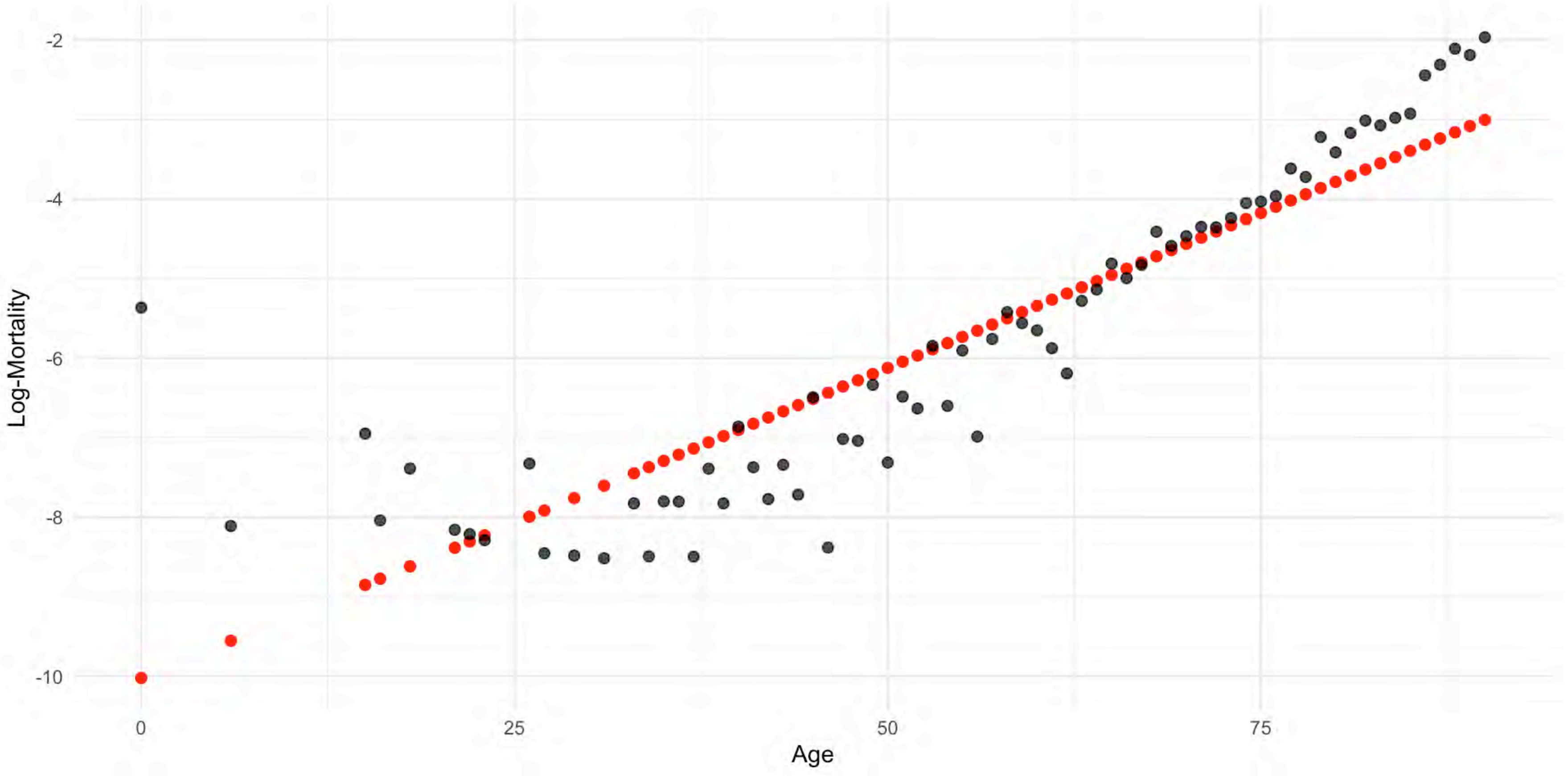
- Linearity & Nonlinearity
- Data Science Starts With Data
- **Linear Regression**
- Polynomial Regression
- Step Functions
- Regression Splines
- Smoothing Splines
- Local Regression
- Generalised Additive Models (GAMs)

UNSW
SYDNEY

Plotting the fitted values

R | Python

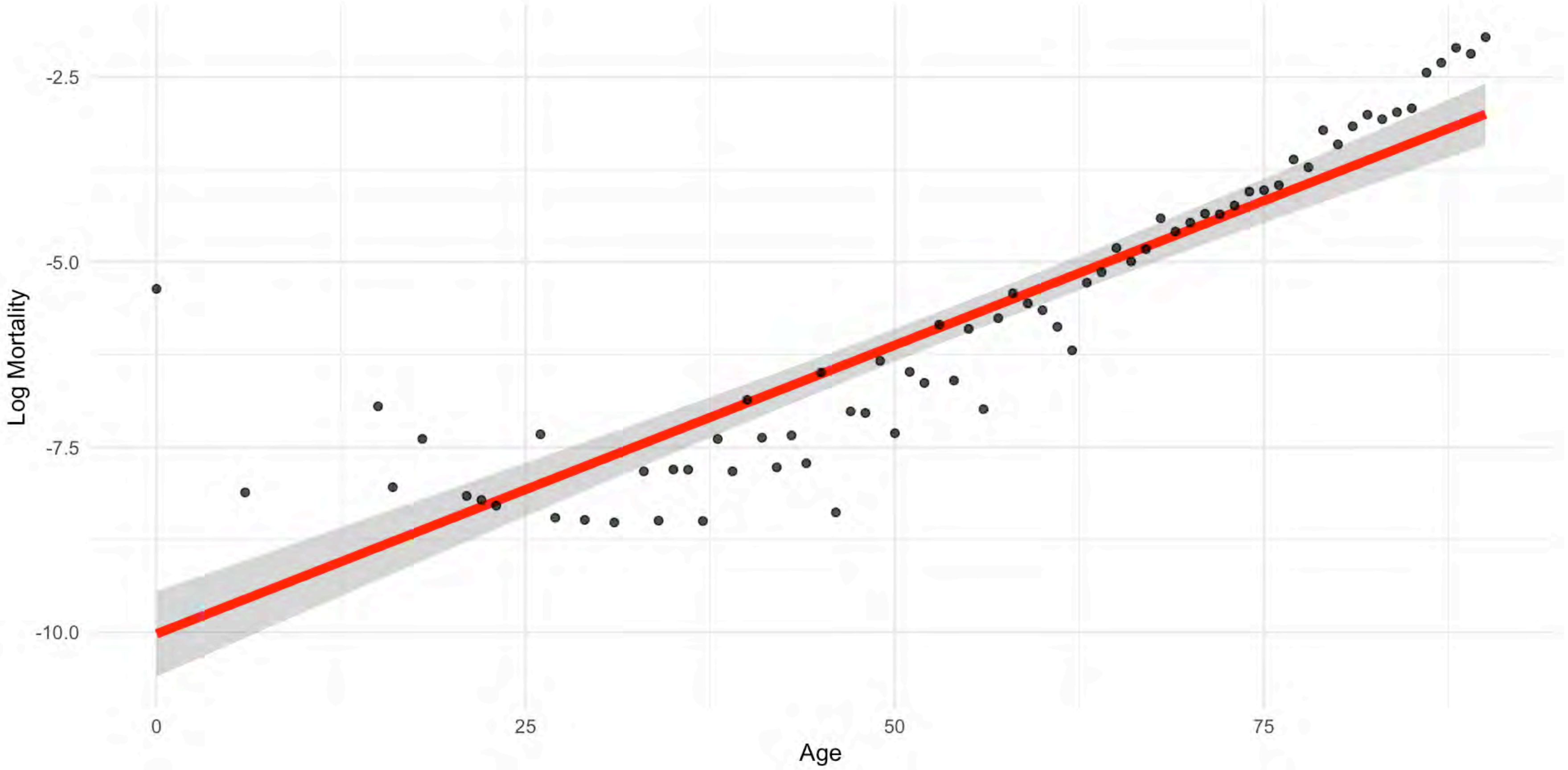
```r
ggplot(lux_2020, aes(x = age, y = log_mx)) + theme_minimal() +
  geom_point(aes(y = predict(model_lr)), color = "red", size = 2) +
  geom_point(alpha = 0.75, size = 2) + labs(x = "Age", y = "Log-Mortality")
```



Linear regression with error bars

R | Python

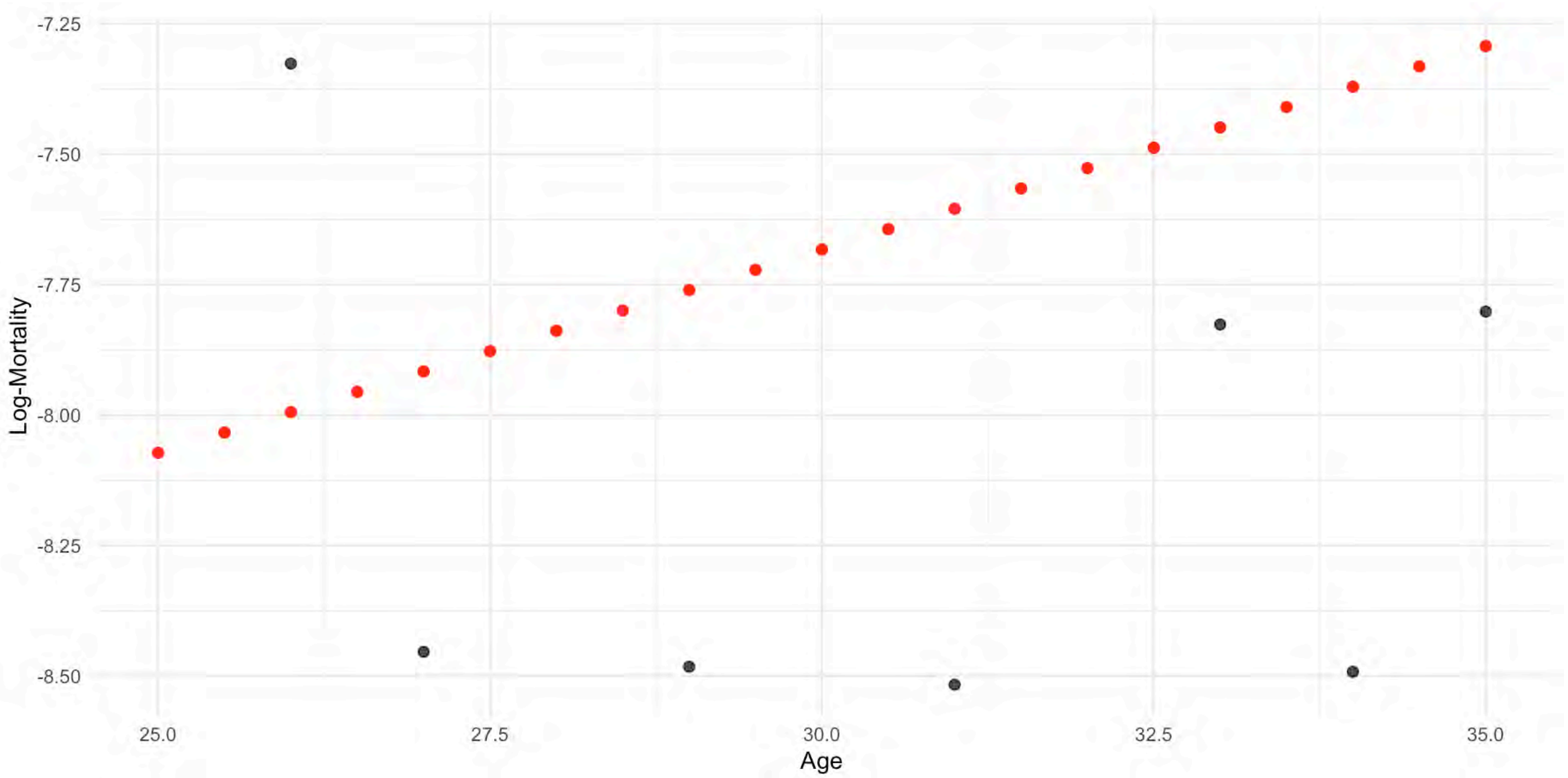
```r
ggplot(lux_2020, aes(x = age, y = log_mx)) + theme_minimal() +
  geom_smooth(method = "lm", formula = y ~ x, color = "red", linewidth=2) +
  geom_point(alpha = 0.75) + labs(x = "Age", y = "Log Mortality")
```



Interpolating

R | Python

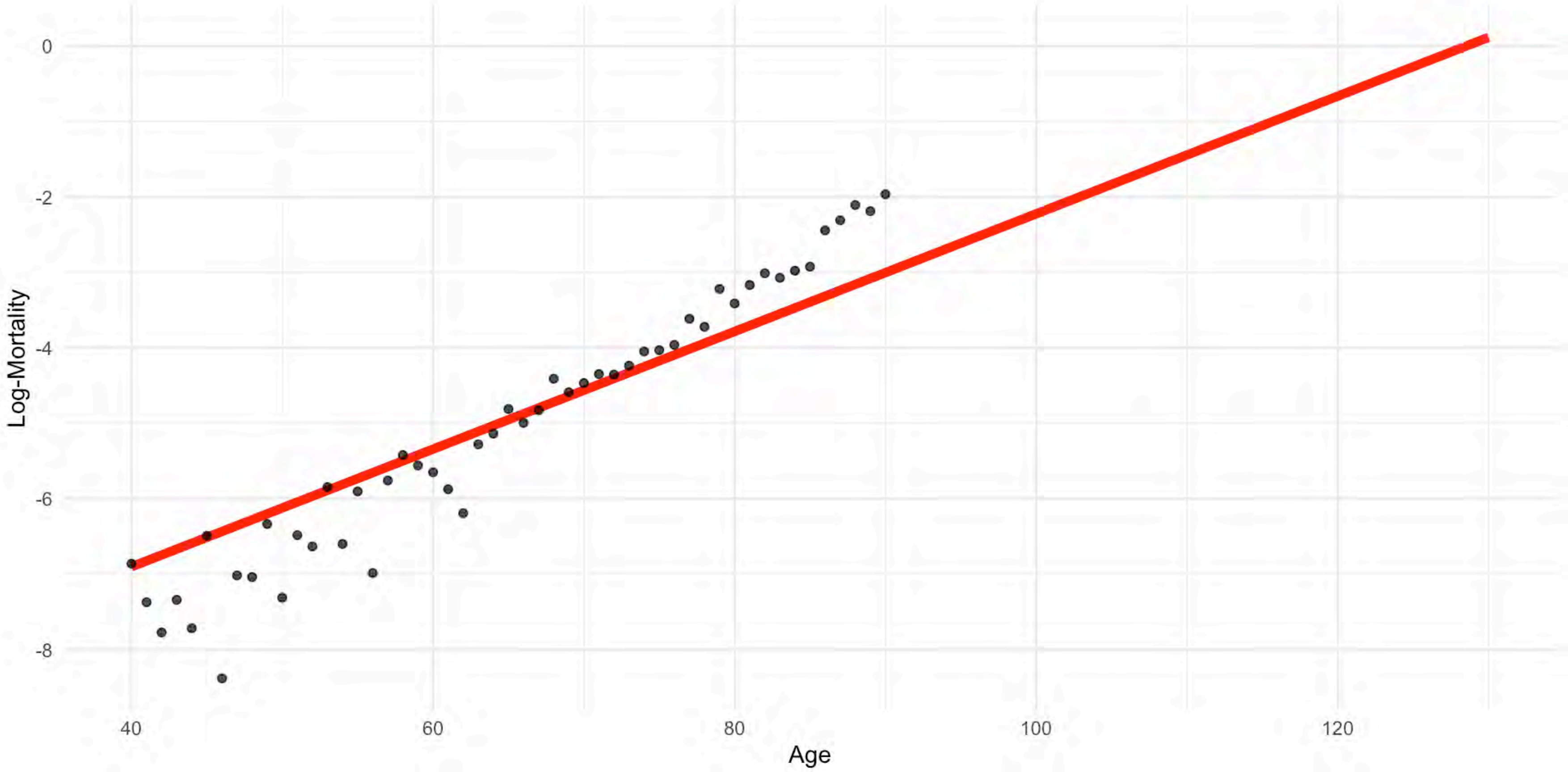
```r
df_grid <- data.frame(age = seq(25, 35, by = 0.5))
df_grid$log_mx <- predict(model_lr, newdata = df_grid)
```



Extrapolating

R | Python

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

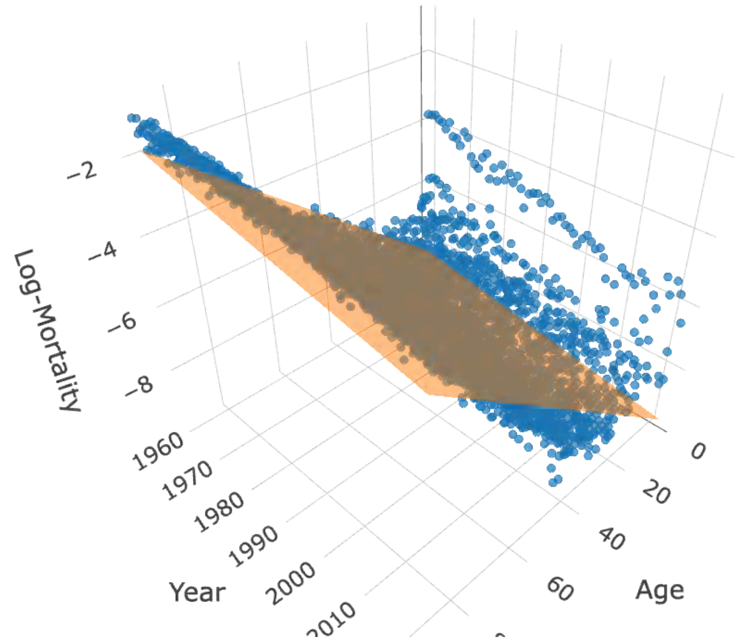
```r
new_point <- data.frame(year = 2040, age = 20)
predict(linear_model, newdata = new_point)
```

```
    1
-8.66
```

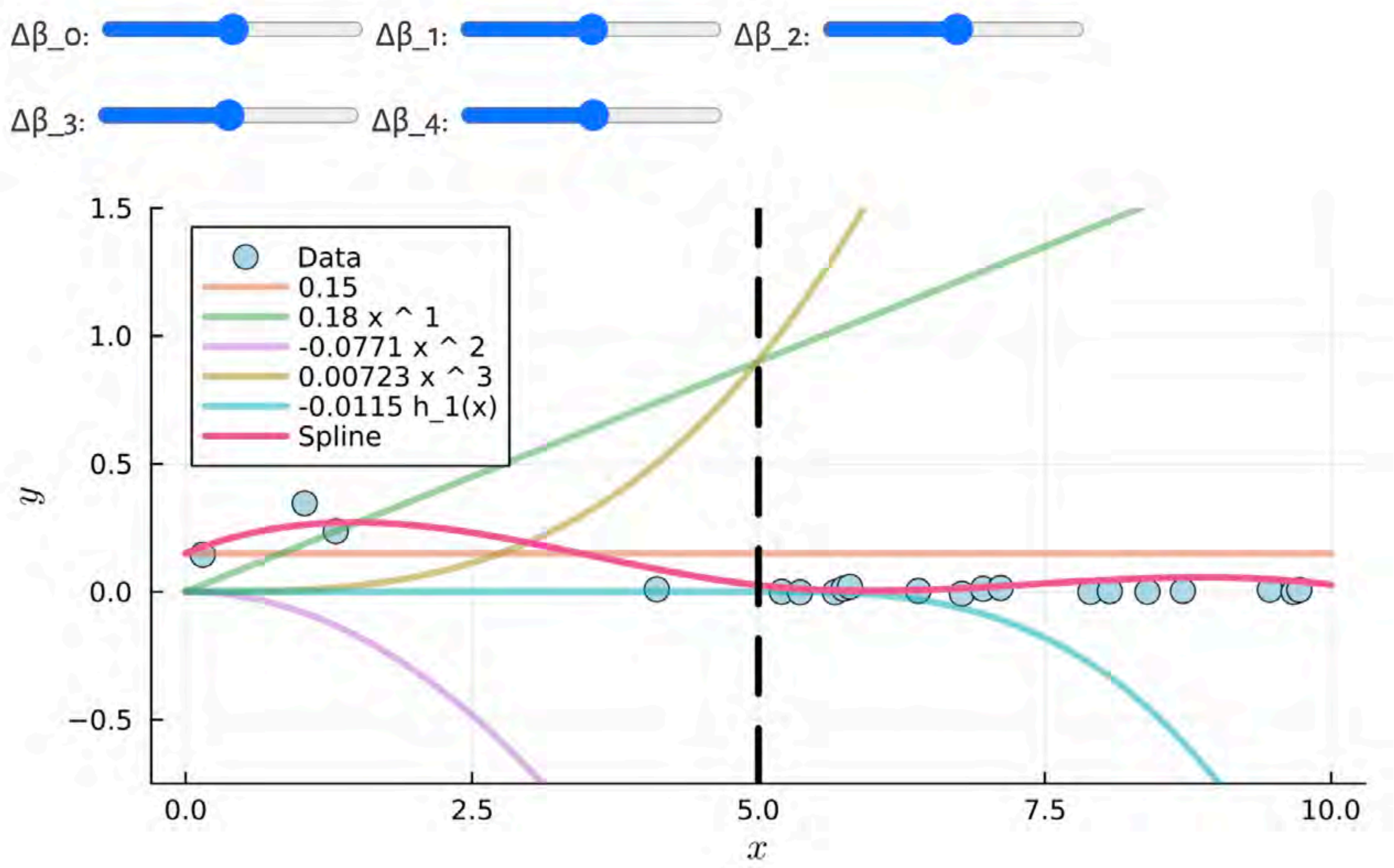
```r
coef(linear_model)
```

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

Fitted multiple linear regression



Link to interactive notebook



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

Lecture Outline

- Linearity & Nonlinearity
- Data Science Starts With Data
- Linear Regression
- **Polynomial Regression**
- Step Functions
- Regression Splines
- Smoothing Splines
- Local Regression
- Generalised Additive Models (GAMs)

Polynomial regression

Extend the standard linear model

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

to

$$Y_i = \beta_0 + \beta_1 x_i + \beta_2 x_i^2 + \cdots + \beta_d x_i^d + \varepsilon_i$$

- 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 <- data.frame(age = lux_2020$age, age2 = lux_2020$age^2, log_mx = lux_2020$log_mx)
head(df_pr)
```

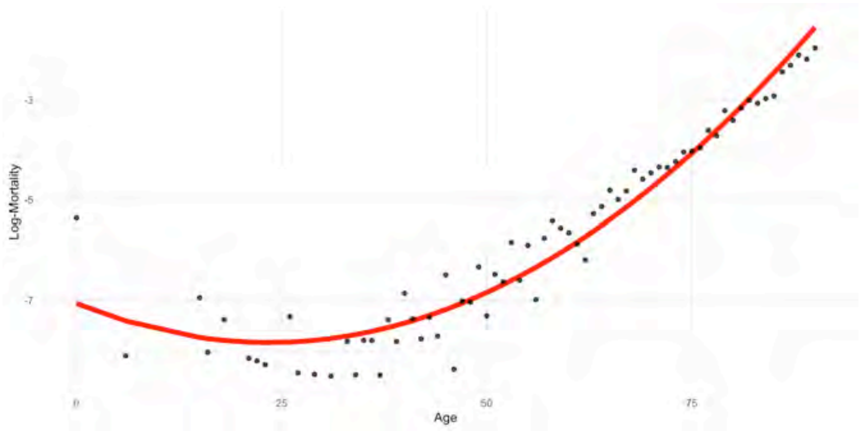
```
  age age2    log_mx
1   0    0 -5.363176
2   6   36 -8.111728
3  15  225 -6.949619
4  16  256 -8.040959
5  18  324 -7.389022
6  21  441 -8.159519
```

```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.

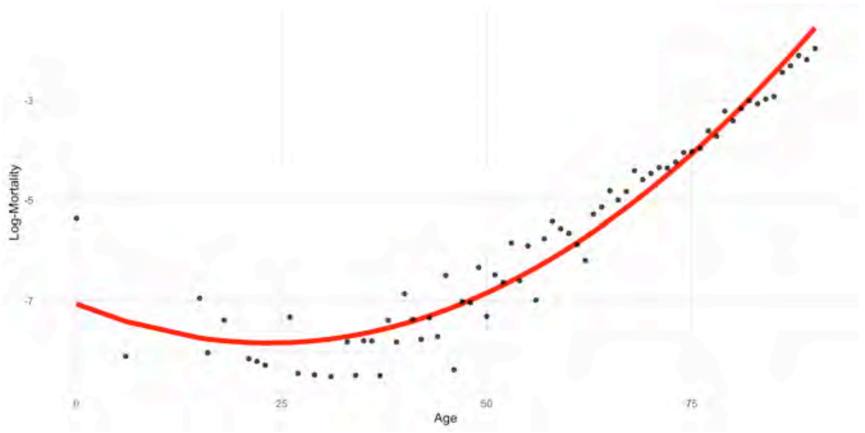
The `poly` function

```r
df_pr <- data.frame(age = lux_2020$age, log_mx = lux_2020$log_mx)
head(df_pr)
```

```
  age    log_mx
1   0 -5.363176
2   6 -8.111728
3  15 -6.949619
4  16 -8.040959
5  18 -7.389022
6  21 -8.159519
```

```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 
```



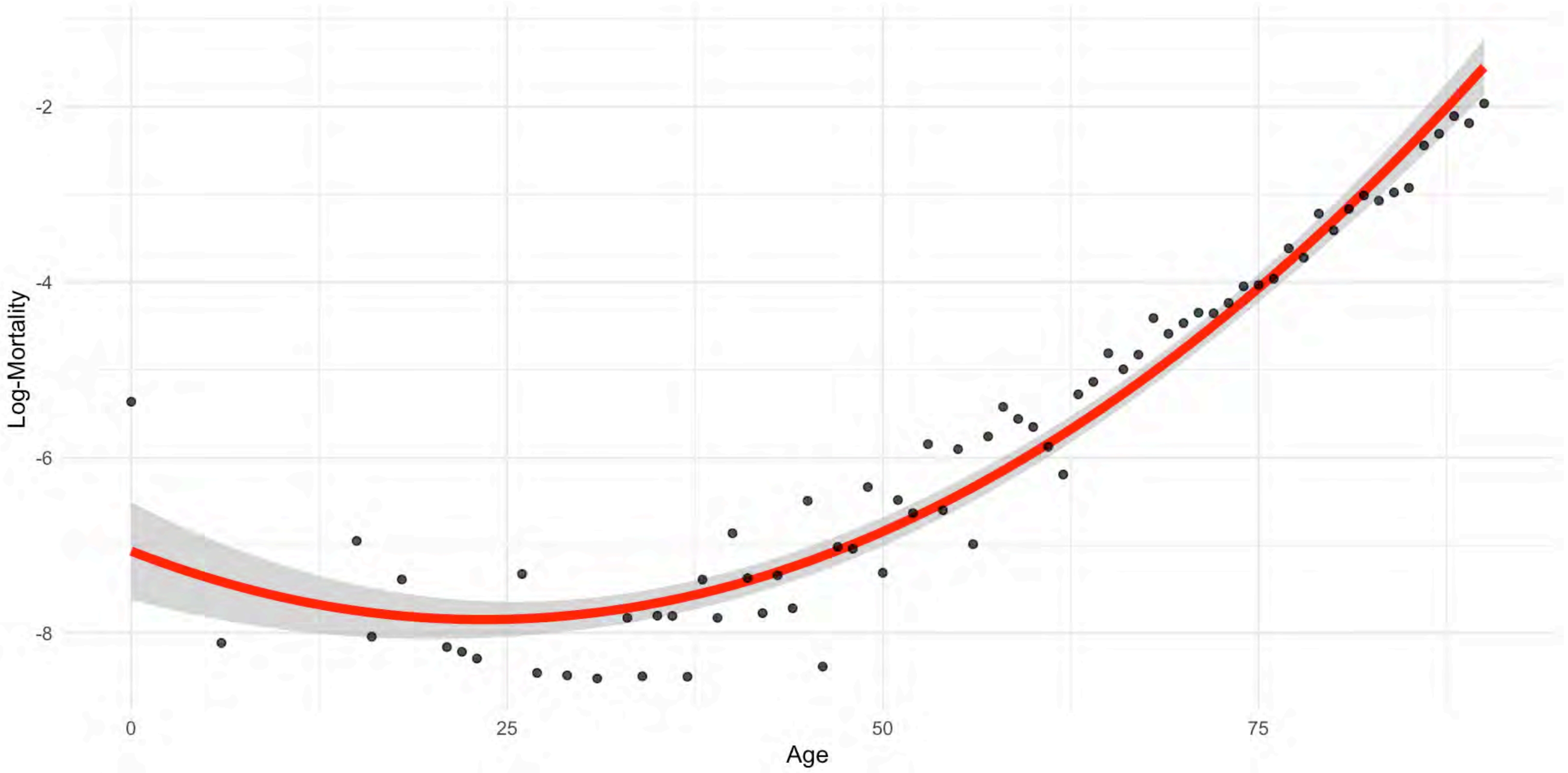
Now we *can't* put in `age^2` as a separate variable.

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

```
        1 
-7.829633 
```

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")
```



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)
```

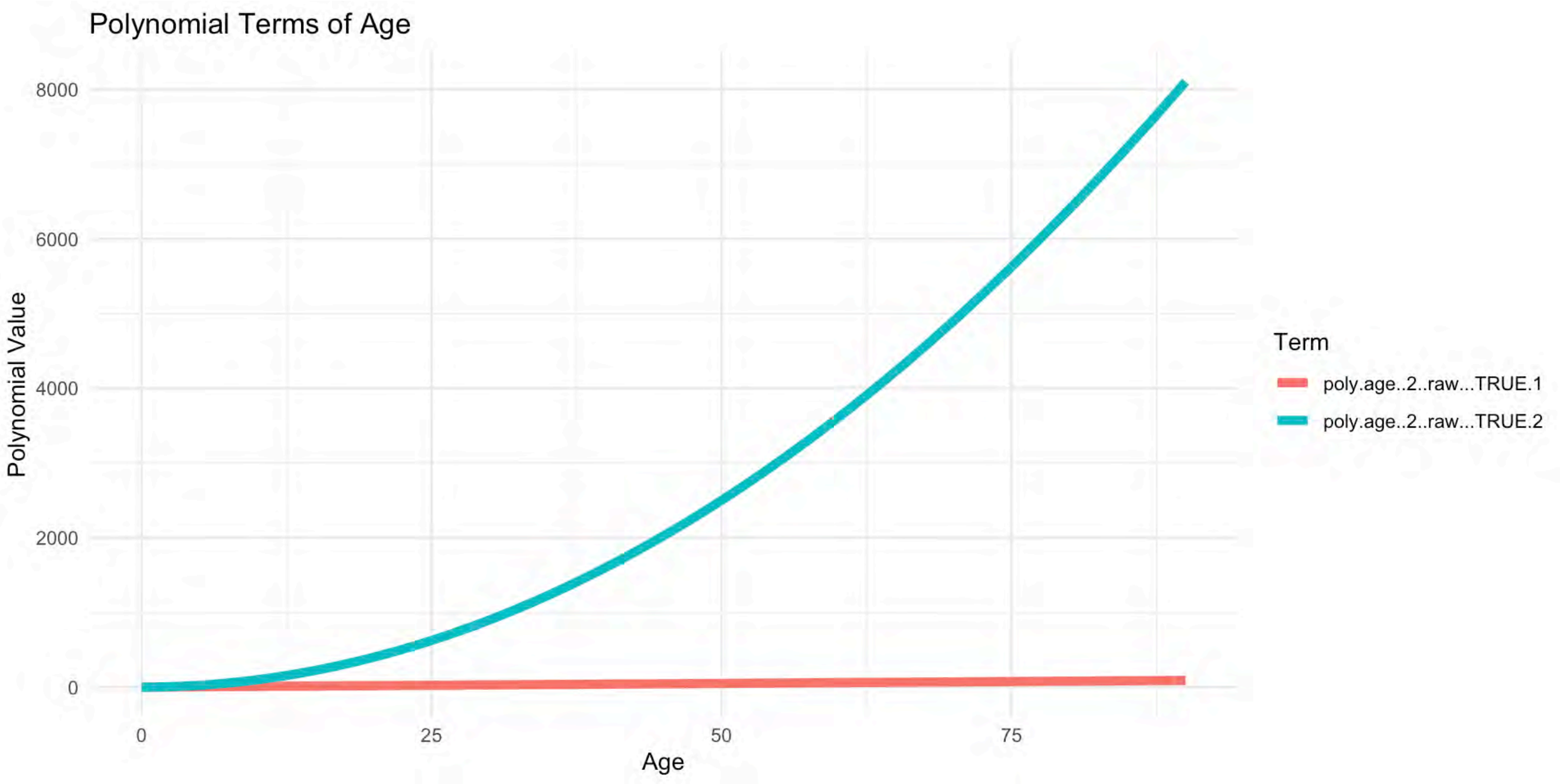
```
  (Intercept) poly(age, 2)1 poly(age, 2)2
1           1   -0.03020513    0.03969719
2           1   -0.02961226    0.03744658
3           1   -0.02901939    0.03524373
4           1   -0.02842652    0.03308866
5           1   -0.02783365    0.03098136
6           1   -0.02724077    0.02892183
```

```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
```

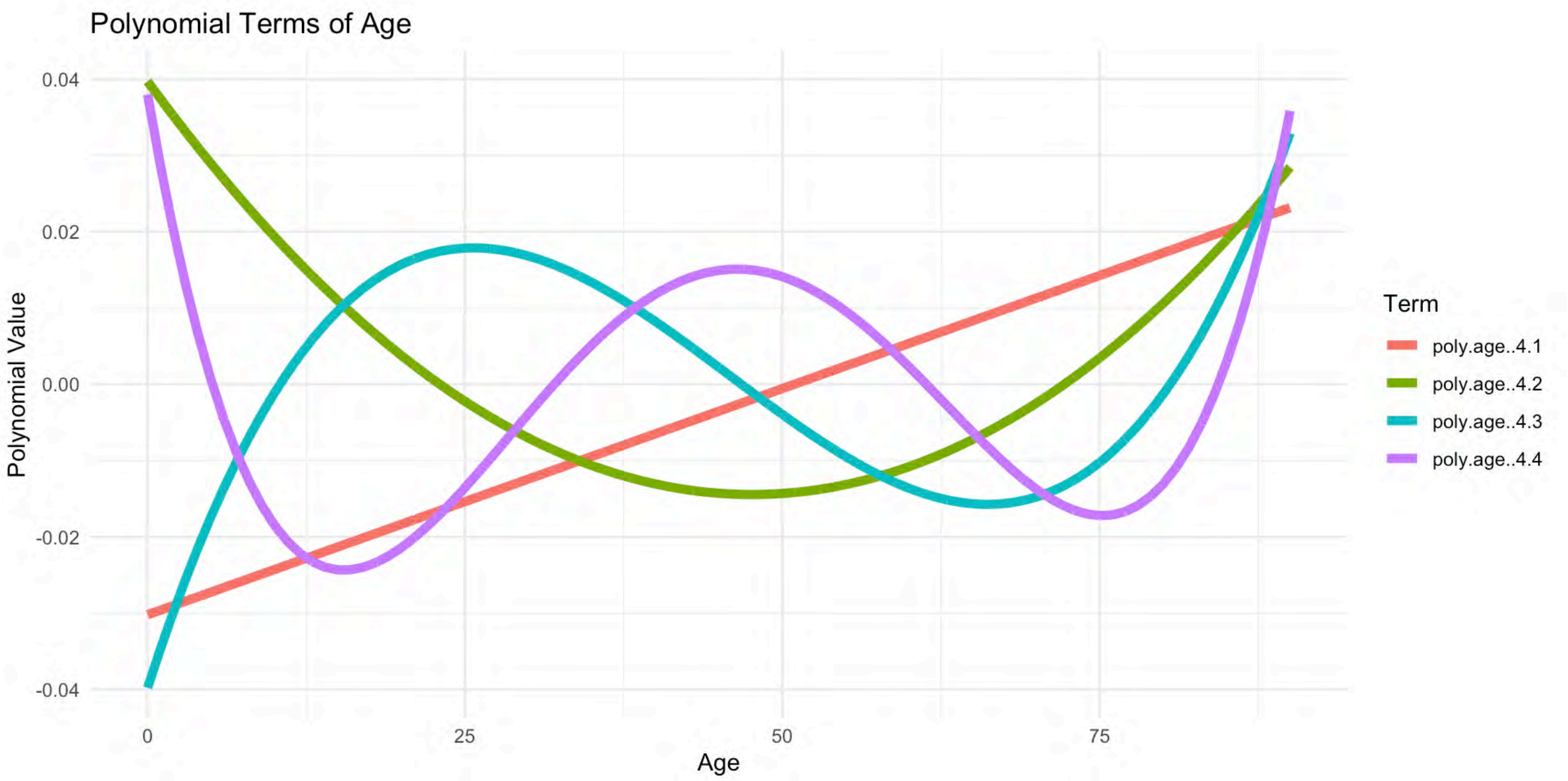
Monomials plotted (raw=TRUE)

```
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? Raw polynomials can be highly correlated

Reciprocal of the condition number is `rcond`.

```
X <- model.matrix(~ poly(age, 10),
    data = lux)
rcond(t(X) %*% X)
```

```
[1] 0.0002087683
```

```
X_raw <- model.matrix(~ poly(age, 10, raw=TRU
    data = lux)
rcond(t(X_raw) %*% X_raw)
```

```
[1] 1.155411e-40
```

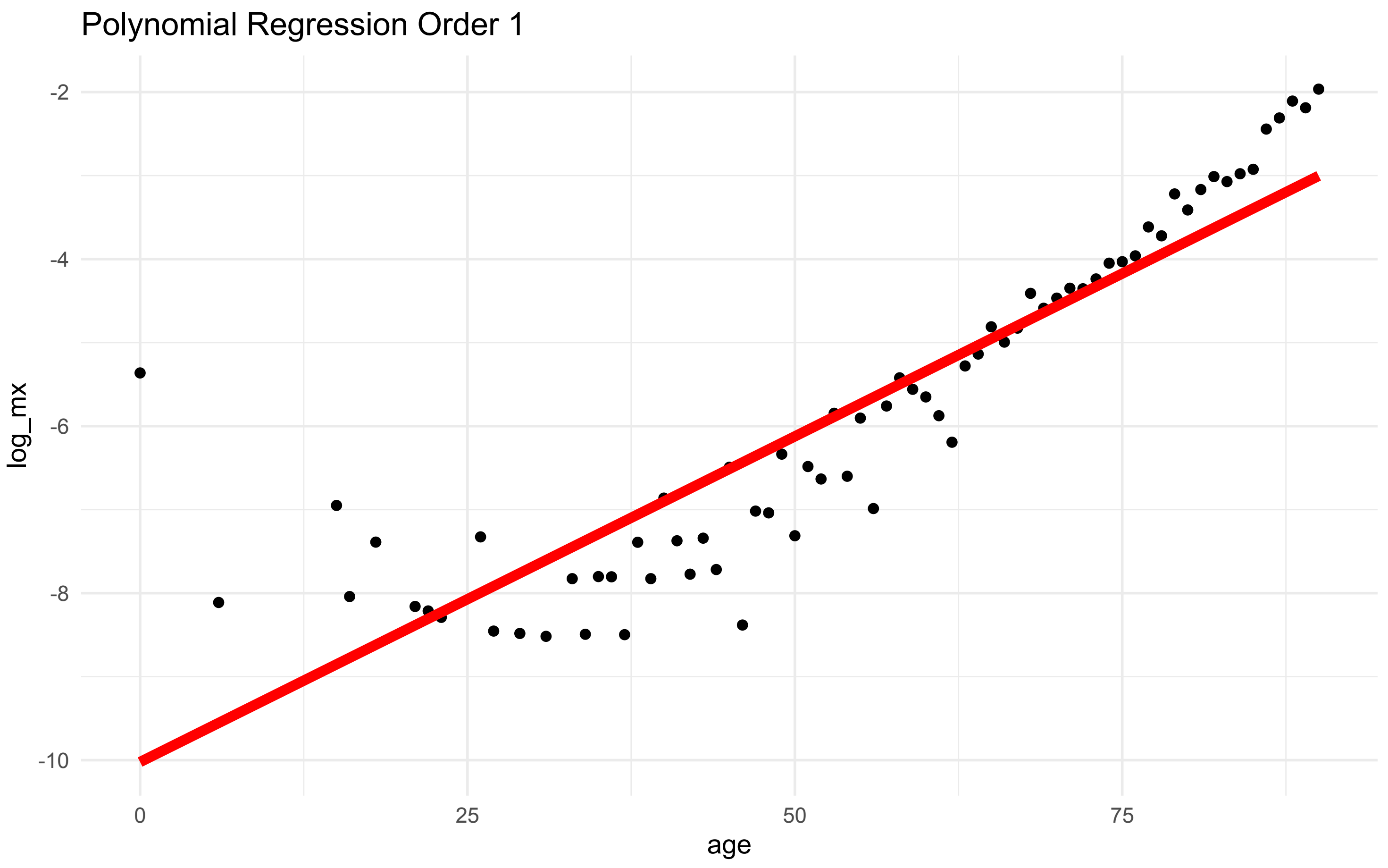
We want it to be close to 1, so that the matrix can be inverted.

```
inv <- solve(t(X) %*% X)
```

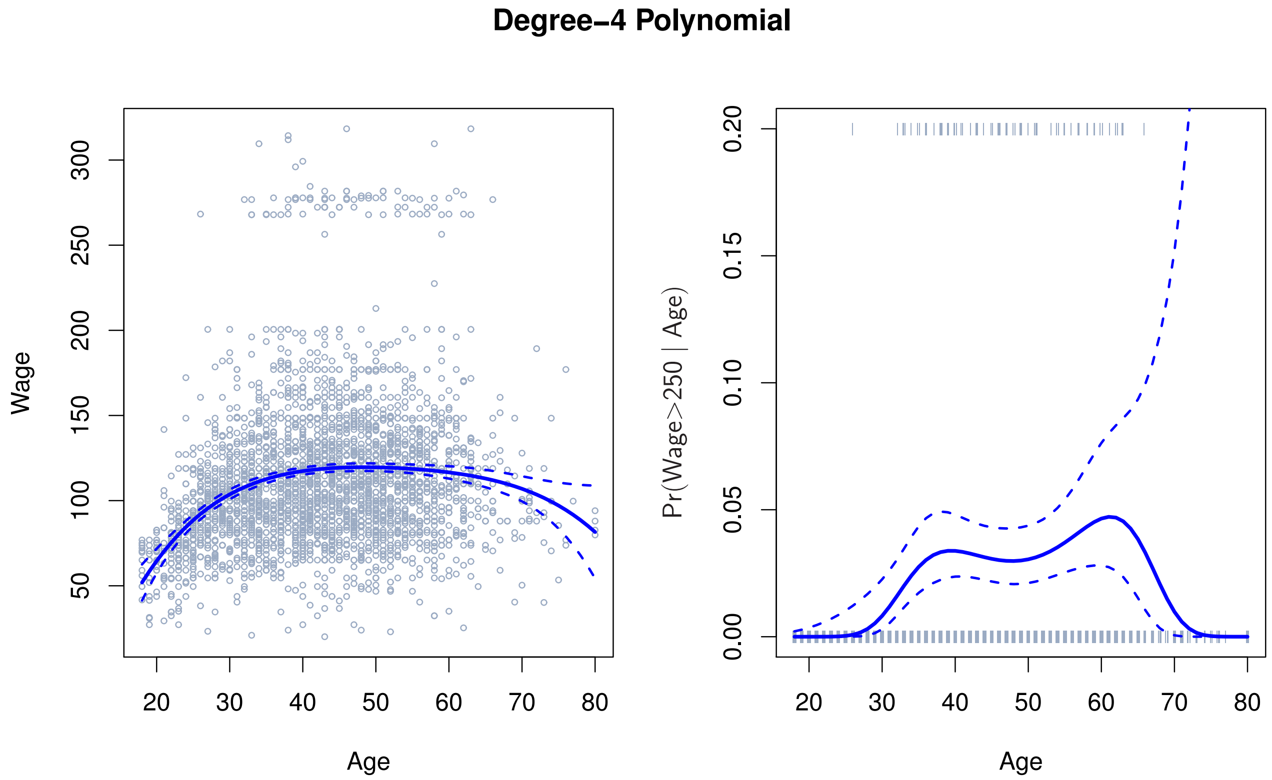
```
inv <- solve(t(X_raw) %*% X_raw)
```

```
Error in solve.default(t(X_raw) %*% X_raw): system
is computationally singular: reciprocal condition
number = 1.15541e-40
```

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.

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

Lecture Outline

- Linearity & Nonlinearity
- Data Science Starts With Data
- Linear Regression
- Polynomial Regression
- **Step Functions**
- Regression Splines
- Smoothing Splines
- Local Regression
- Generalised Additive Models (GAMs)

Step functions

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

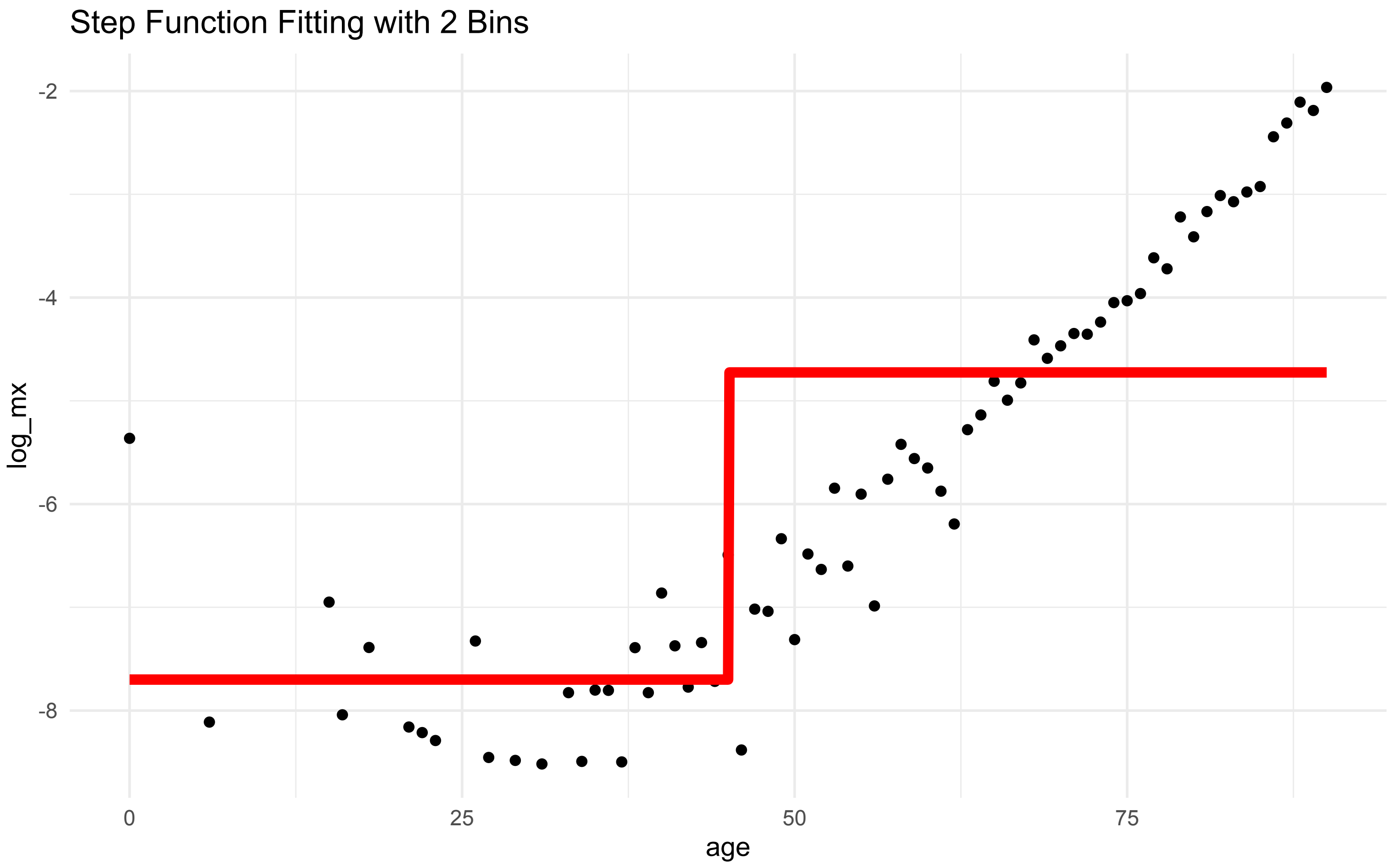
Break up range of x into k distinct regions

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

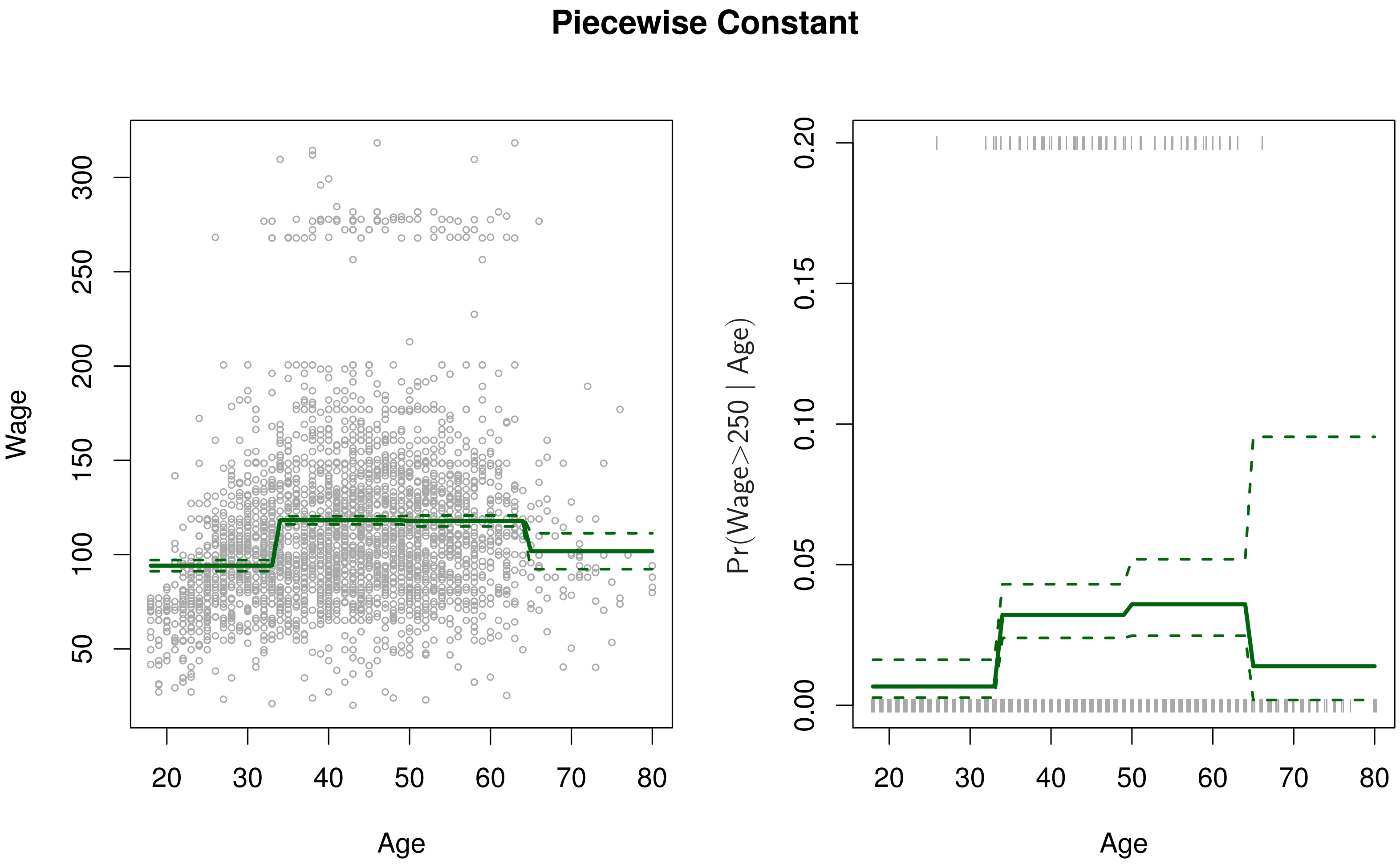
Do a least squares fit on

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

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"

$$\boldsymbol{X} = \begin{bmatrix} 1 & x_{11} & x_{12} & \ldots & x_{1p} \\ 1 & x_{21} & x_{22} & \ldots & x_{2p} \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ 1 & x_{n1} & x_{n2} & \ldots & x_{np} \end{bmatrix} \qquad \boldsymbol{X} = \begin{bmatrix} 1 & b_1(x_1) & b_2(x_1) & \ldots & b_k(x_1) \\ 1 & b_1(x_2) & b_2(x_2) & \ldots & b_k(x_2) \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ 1 & b_1(x_n) & b_2(x_n) & \ldots & b_k(x_n) \end{bmatrix}$$

- 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 $j = 1, \ldots, k-1$

Lecture Outline

UNSW
SYDNEY

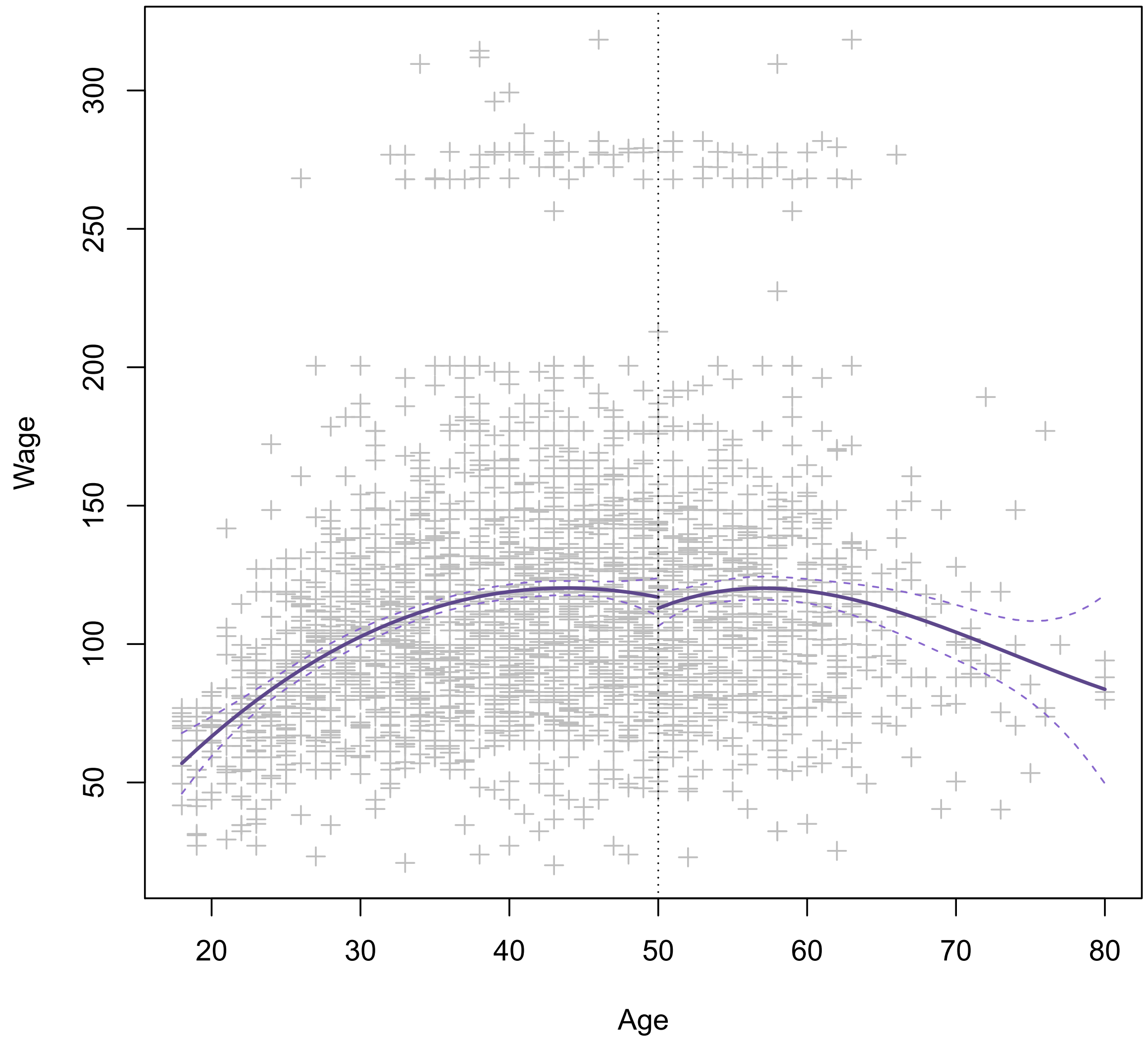
Example: Piecewise cubic regression

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

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

c is a knot: a point of our choosing where the model changes from one to another

Unconstrained cubic regression



Unconstrained cubic regression on `Wage` data

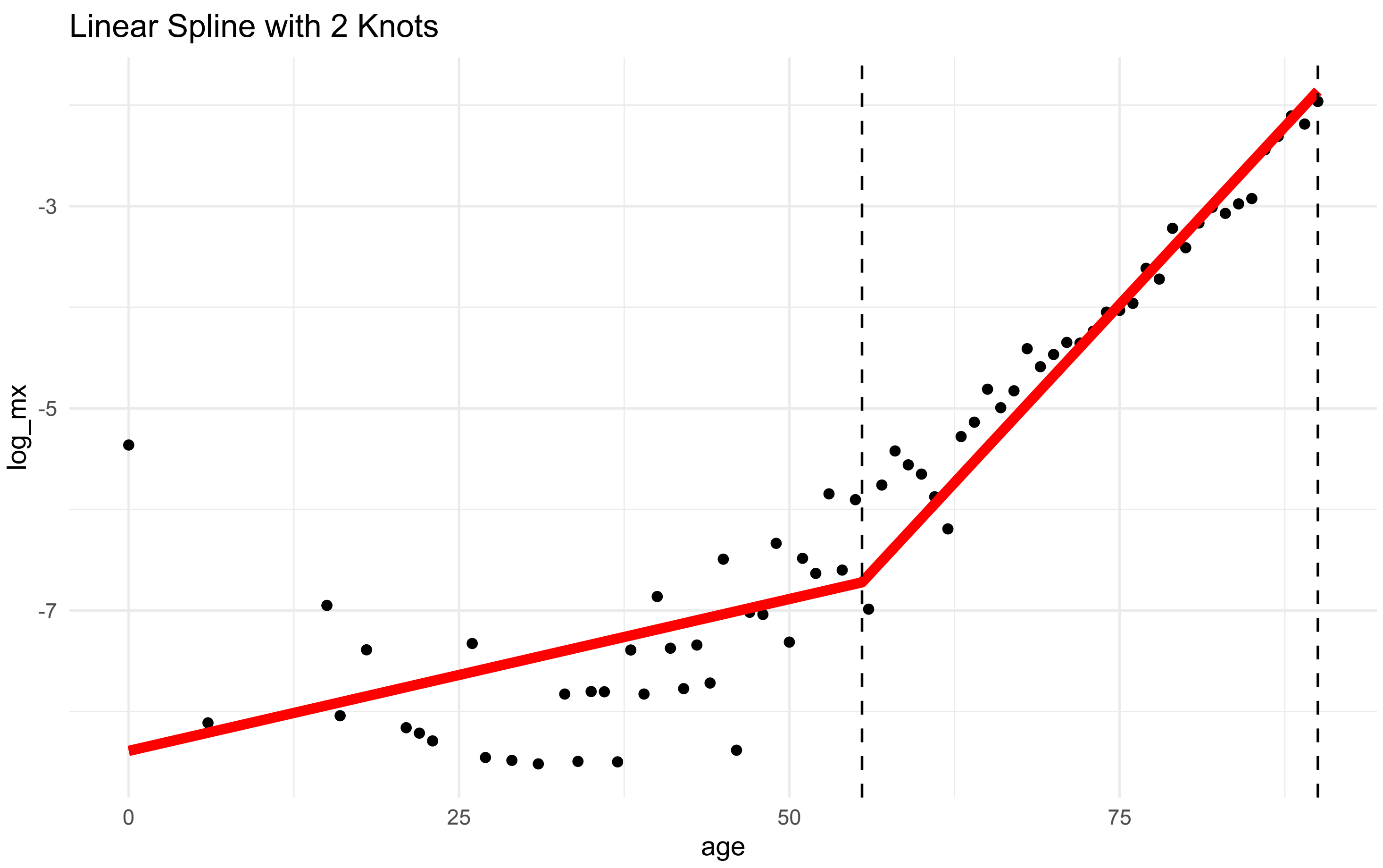
Spline definition

A piecewise polynomial function of degree d is a **spline** if the function is continuous up to the $(d-1)$th derivative at each knot.

- A 1st degree spline is a piecewise linear function which is continuous (i.e. the 0th derivative)
- A 2nd degree spline is a piecewise quadratic function which is continuous and has a continuous derivative
- A 3rd degree spline is a piecewise cubic function which is continuous and has continuous 1st and 2nd derivatives

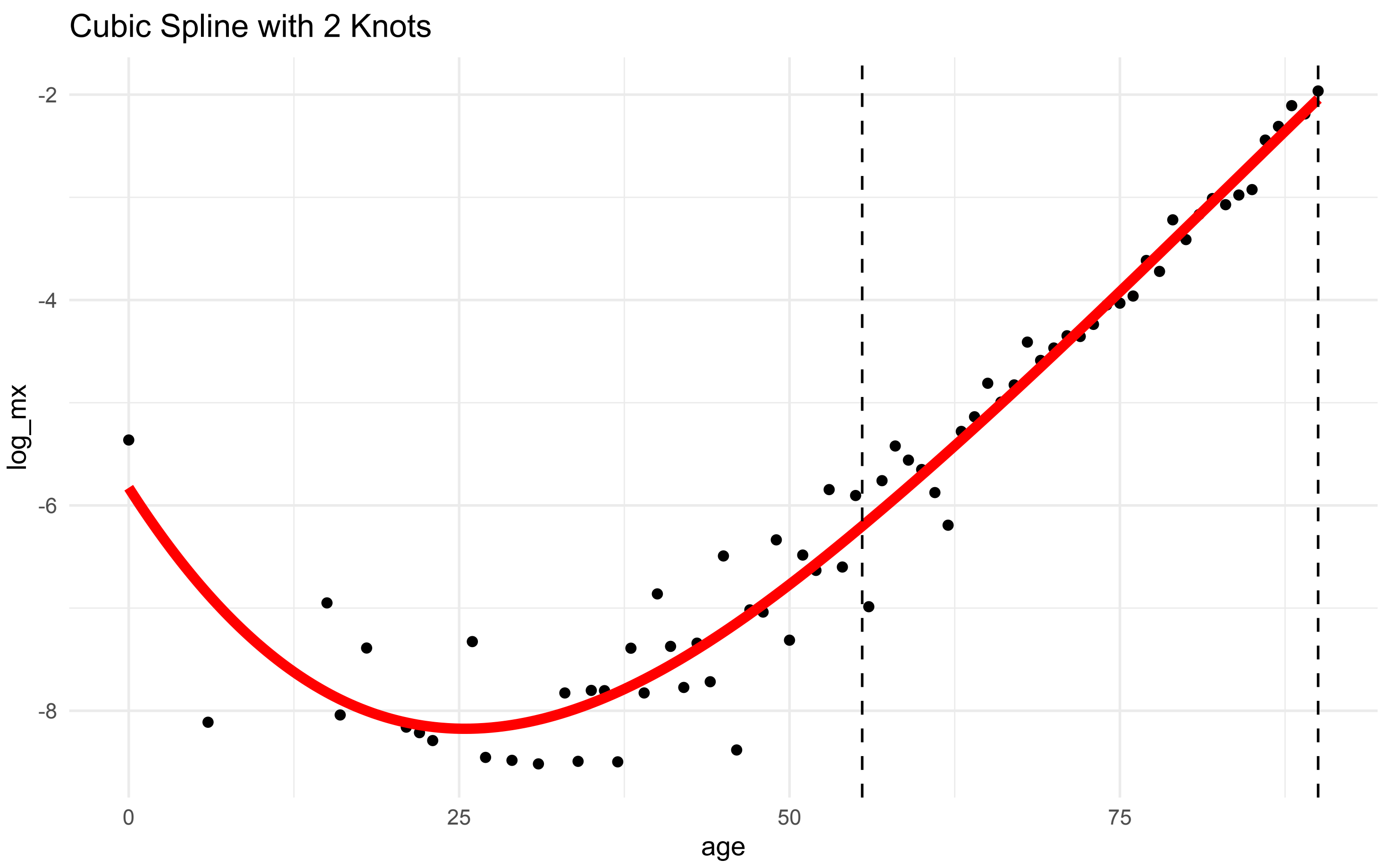
Example: Linear spline

```
model <- lm(log_mx ~ bs(age, degree=1, knots=...), data = lux_2020)
```

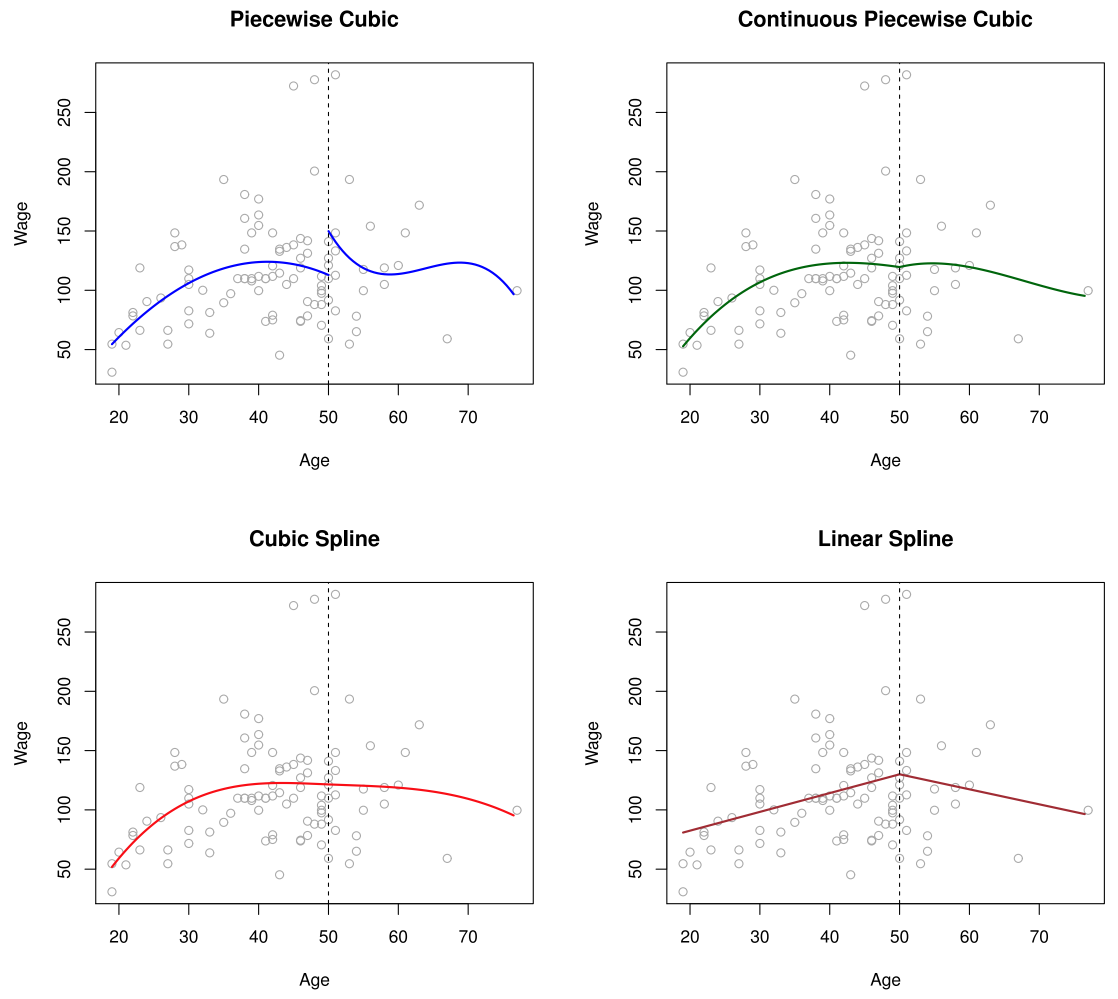


Example: Cubic spline

```r
model <- lm(log_mx ~ bs(age, degree=3, knots=...), data = lux_2020)
```



Examples: Different types of splines



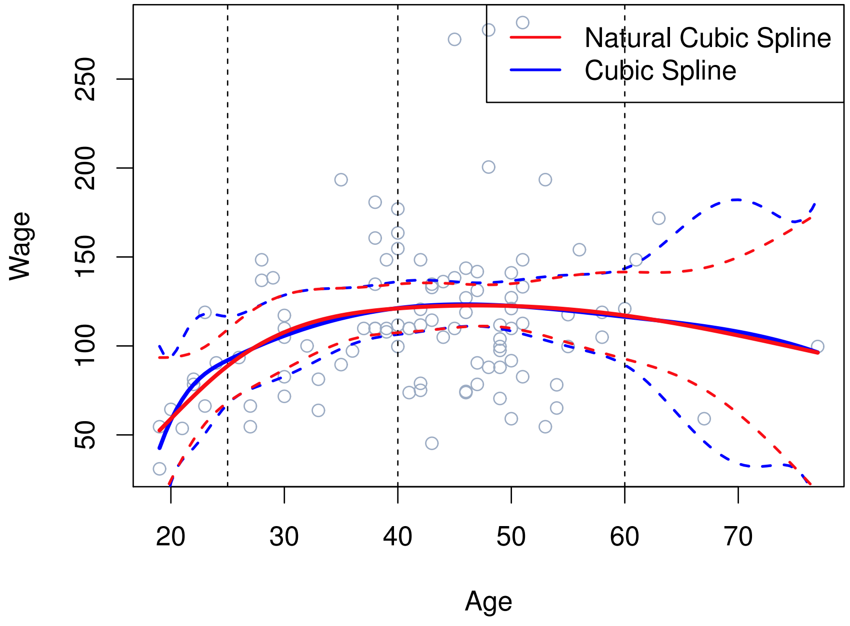
Four varieties of splines fit on a subset of the `Wage` data

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

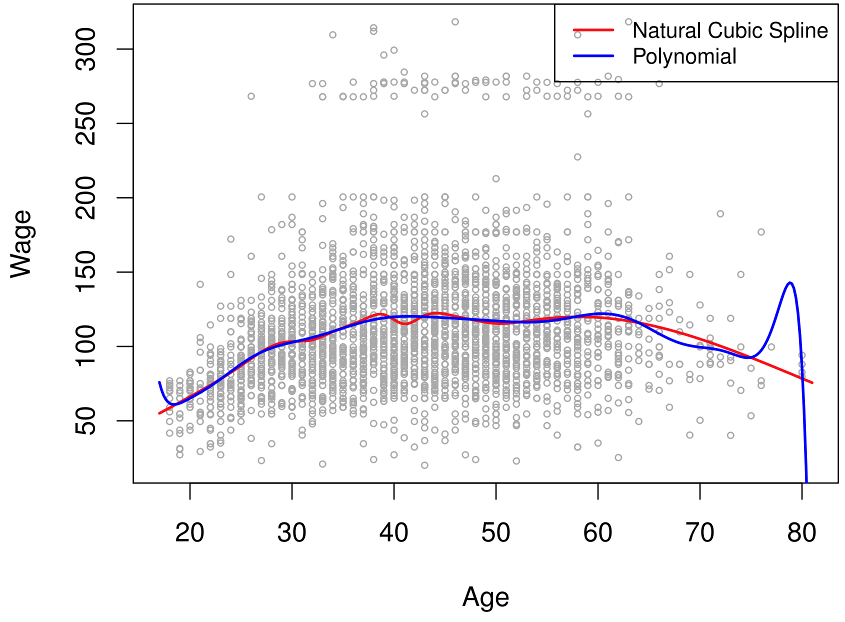
Cubic splines & natural splines

The cubic is preferred as it is the smallest order where the knots are not visible without close inspection.

We can extend the idea of a cubic spline to a **natural** cubic spline. It is a spline where outside the boundary knots (extrapolation) the function is linear.



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.

Lecture Outline

- Linearity & Nonlinearity
- Data Science Starts With Data
- Linear Regression
- Polynomial Regression
- Step Functions
- Regression Splines
- **Smoothing Splines**
- Local Regression
- Generalised Additive Models (GAMs)

Smoothing splines

Find a function g which minimises

$$\frac{1}{n}\sum_{i=1}^{n}(y_i - g(x_i))^2 + \lambda \int_{x=-\infty}^{\infty} g''(x)^2 \, \mathrm{d}x$$

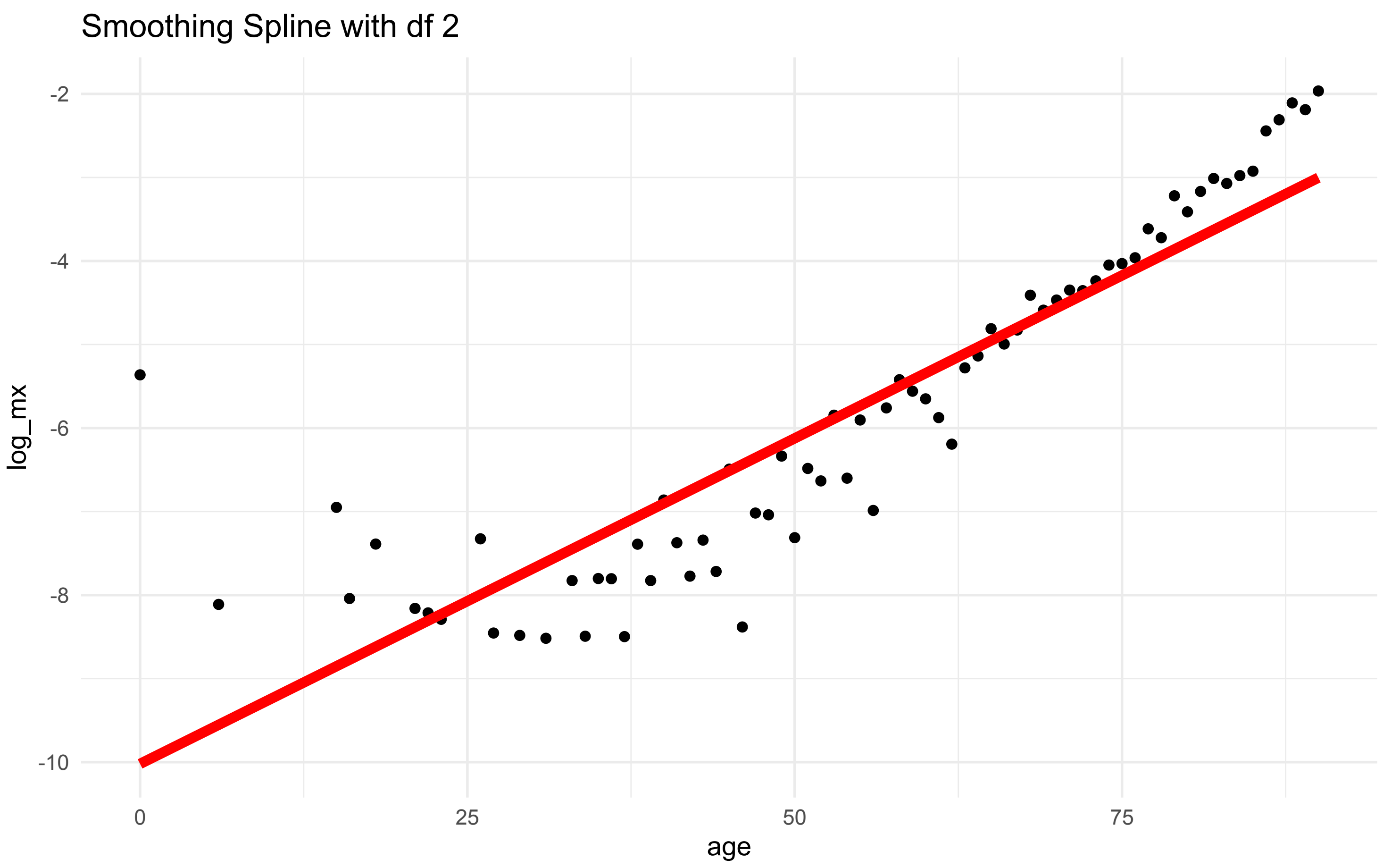
- Goal: fit a function which minimises the MSE whilst still being 'smooth'
- λ is the tuning parameter which penalises a rougher fit
- $\lambda = 0$: g will be very lumpy and will just interpolate all training data points (more flexible: less bias for more variance)
- $\lambda \to \infty$: g will be 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.

Note

For mortality smoothing, US uses smoothing splines, UK uses regression splines.

Example: Smoothing splines

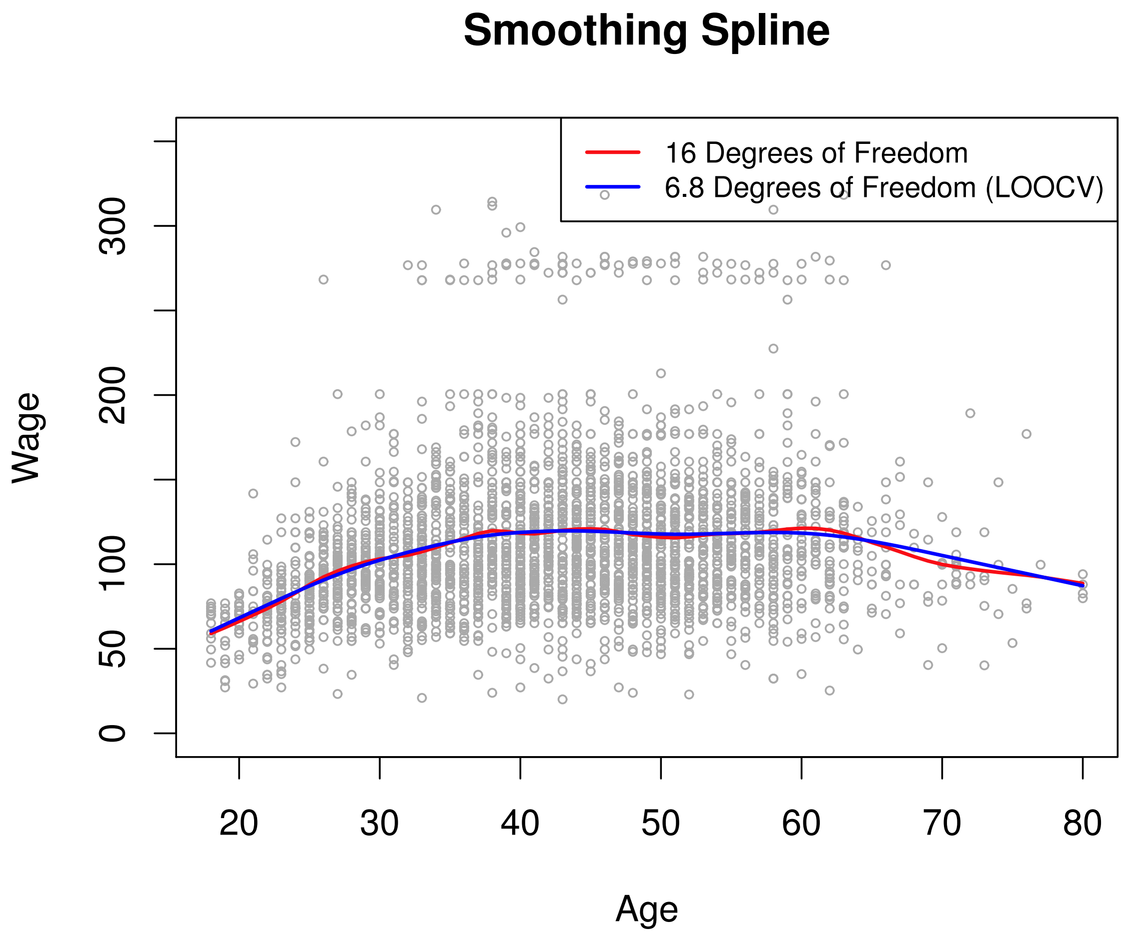
```r
model <- smooth.spline(lux_2020$age, lux_2020$log_mx, df = df)
```



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.
- 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.

Lecture Outline

- Linearity & Nonlinearity
- Data Science Starts With Data
- Linear Regression
- Polynomial Regression
- Step Functions
- Regression Splines
- Smoothing Splines
- **Local Regression**
- Generalised Additive Models (GAMs)

UNSW SYDNEY

Local regression

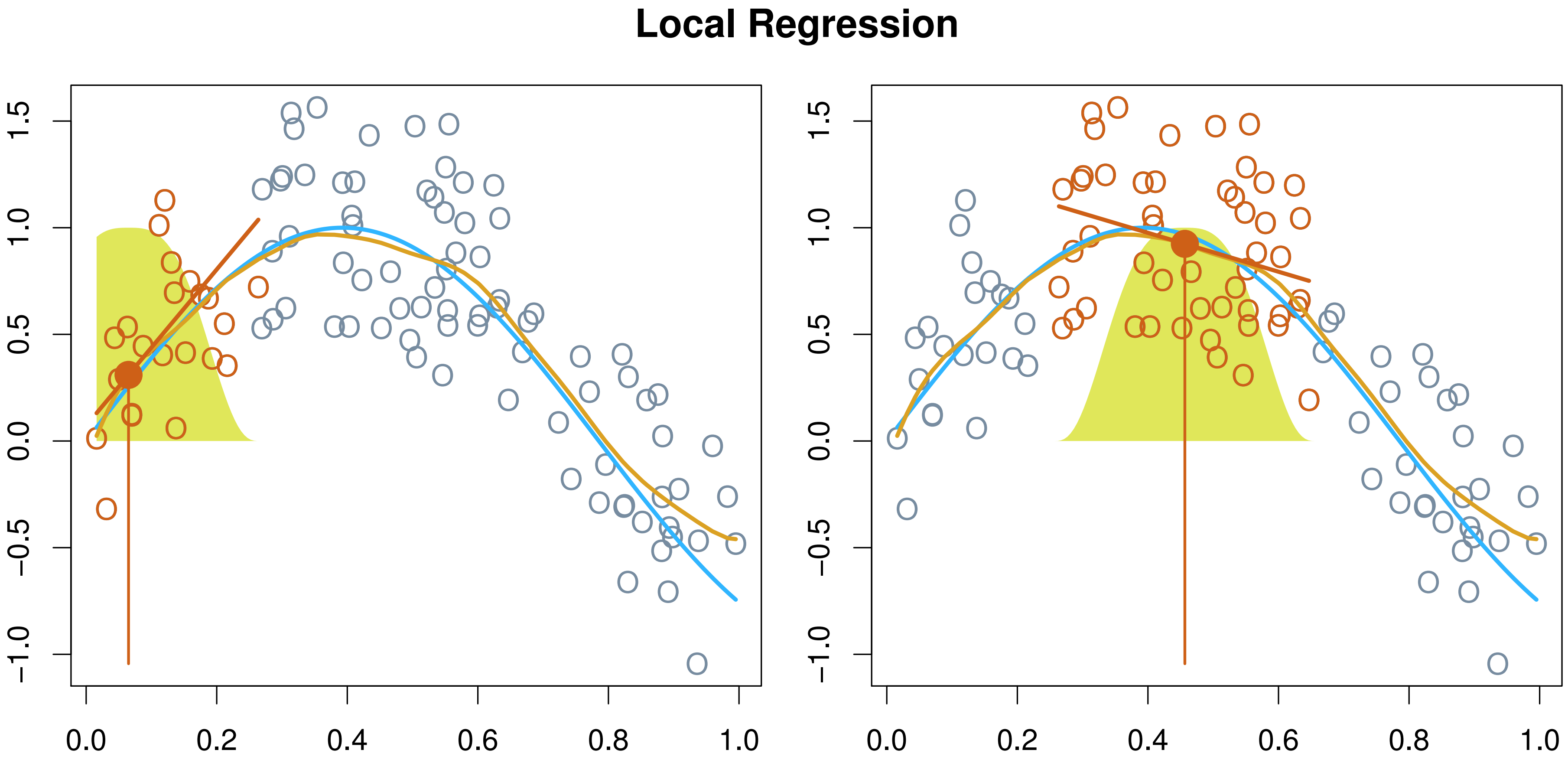
Algorithm

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



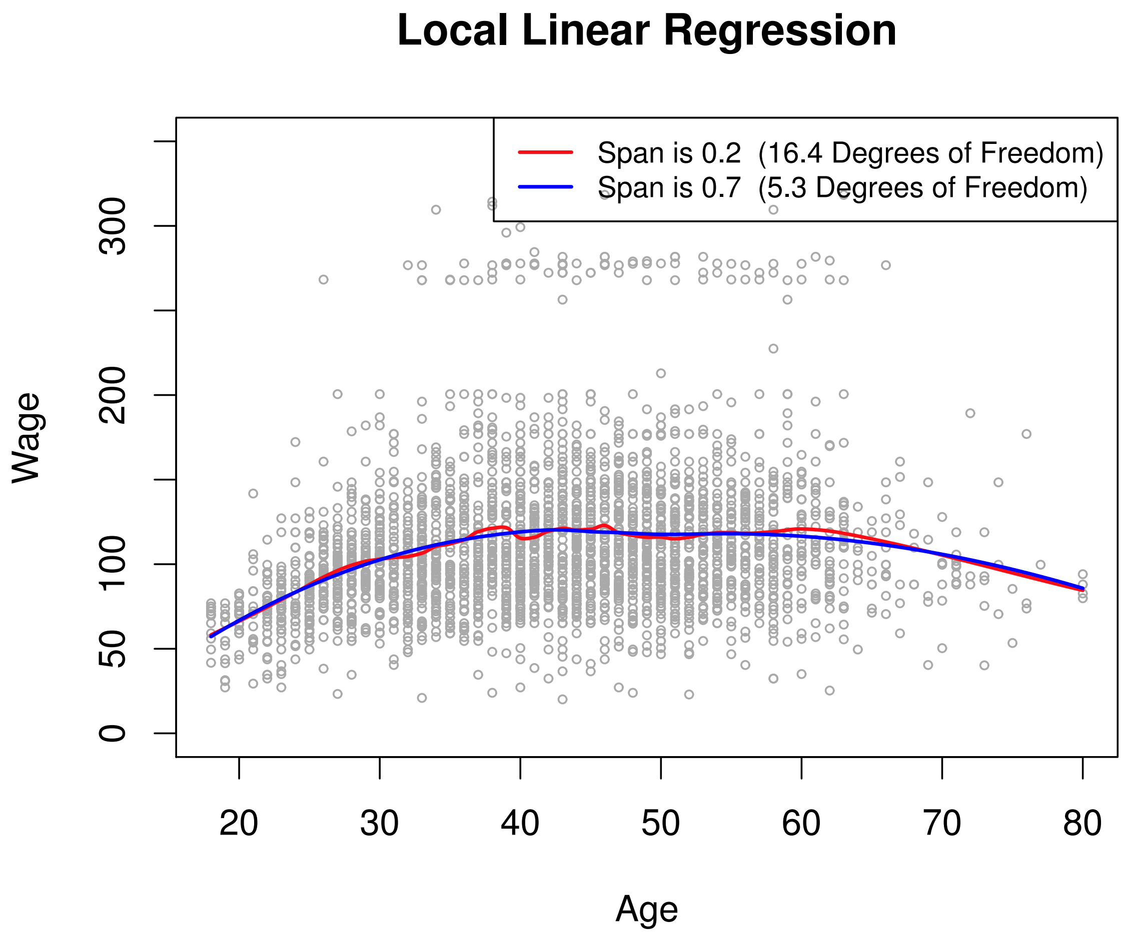
Example of making predictions with local regression at $x \approx 0.05$ and $x \approx 0.45$

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.

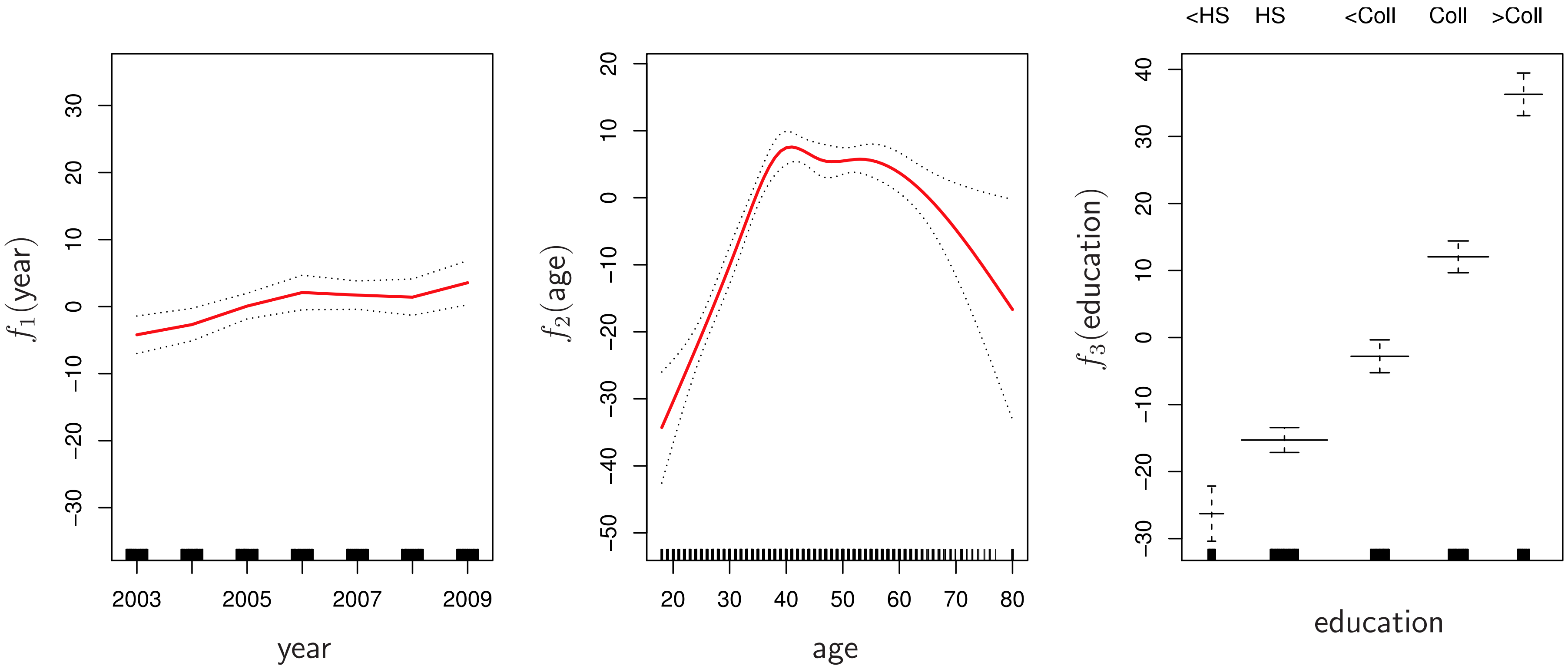
Lecture Outline

GAMs

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

- 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



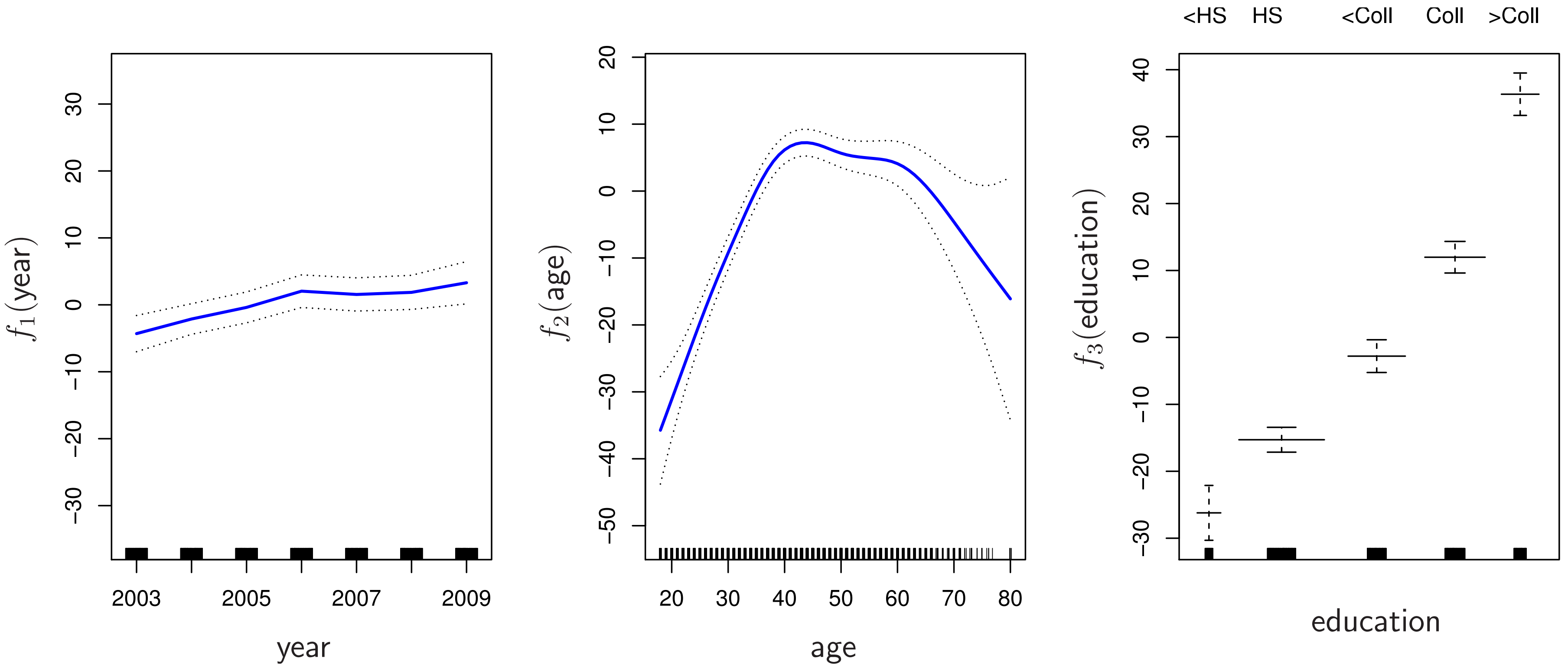
GAM fit using regression splines

- 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.

UNSW SYDNEY

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

Source: James et al. (2021), *An Introduction to Statistical Learning with Applications in R*, Figure 7.12.

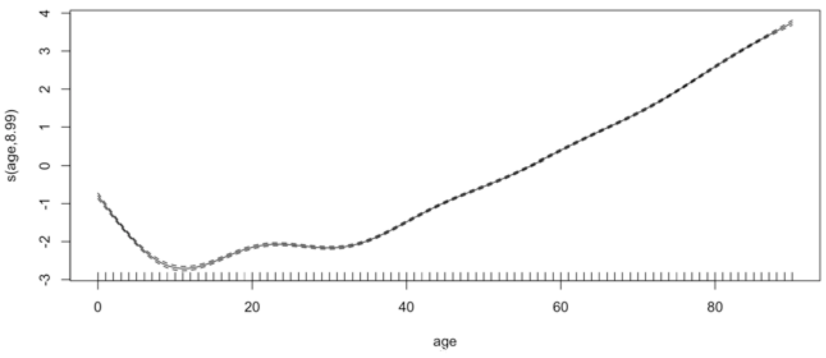
GAMs: pros and cons

- 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. $f_{j,k}(x_j, x_k)$

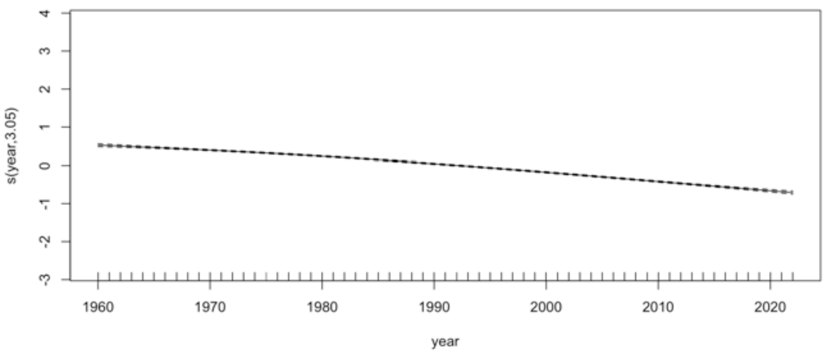
GAMs on Luxembourg data (`mgcv`)

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

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

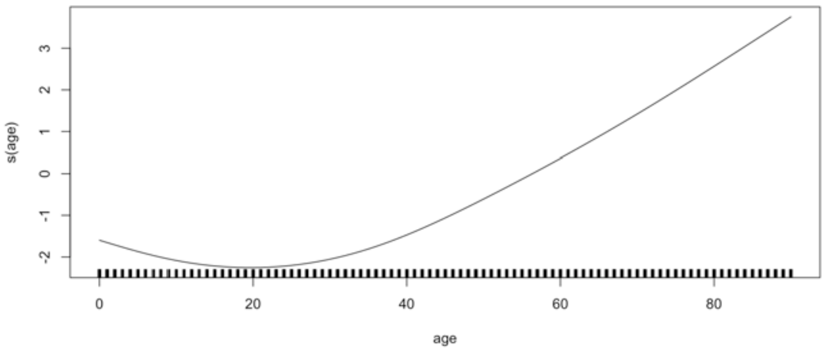


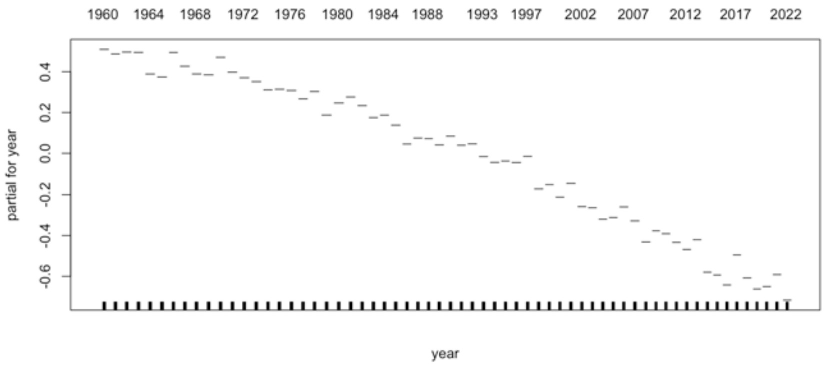

```r
plot(model_gam, select=2)
```

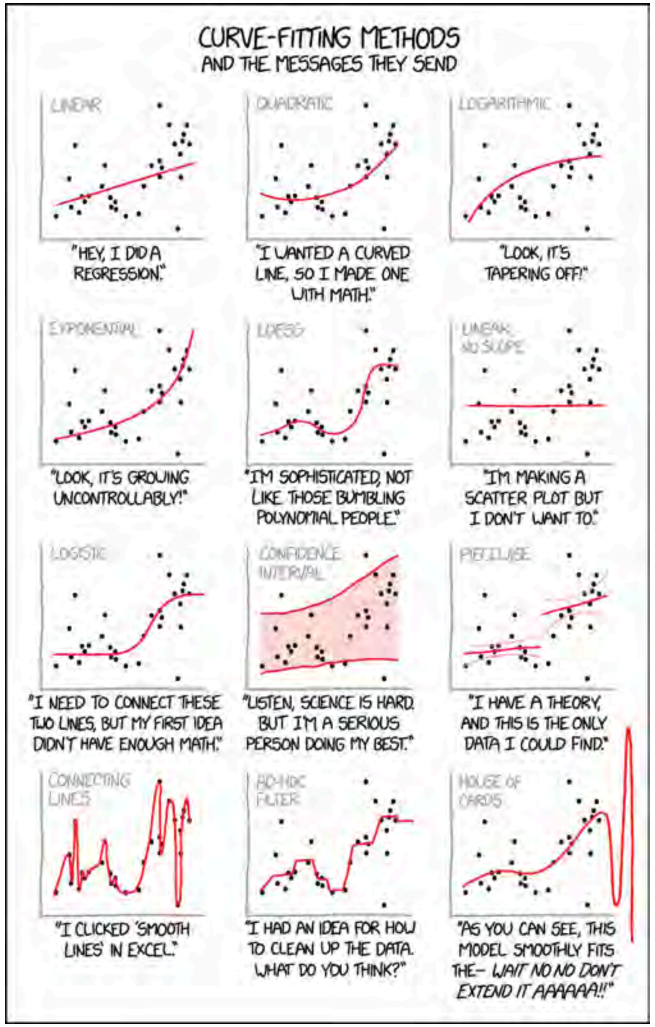


GAMs on Luxembourg data (`gam`)

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





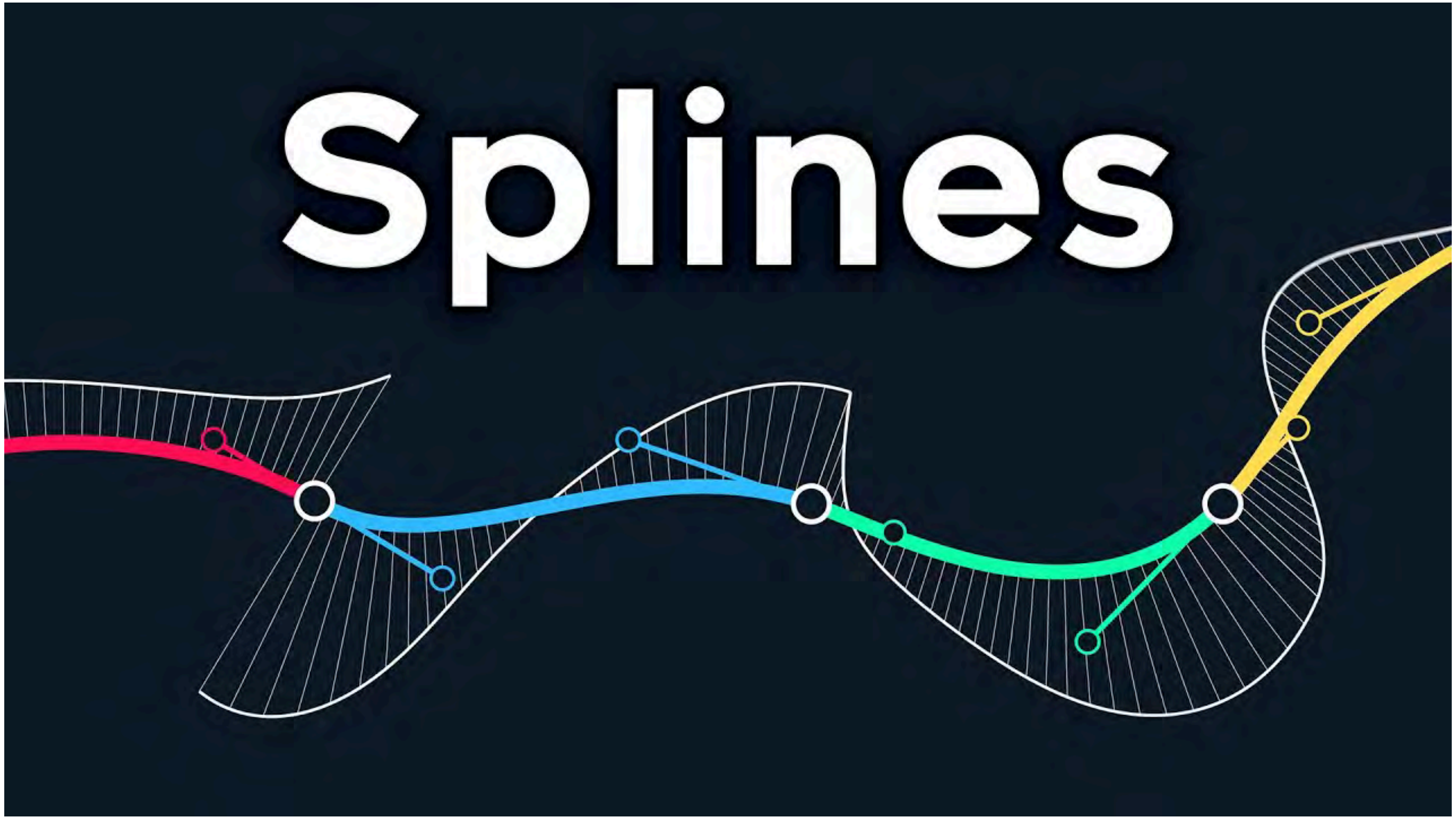


Source: xkcd

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)

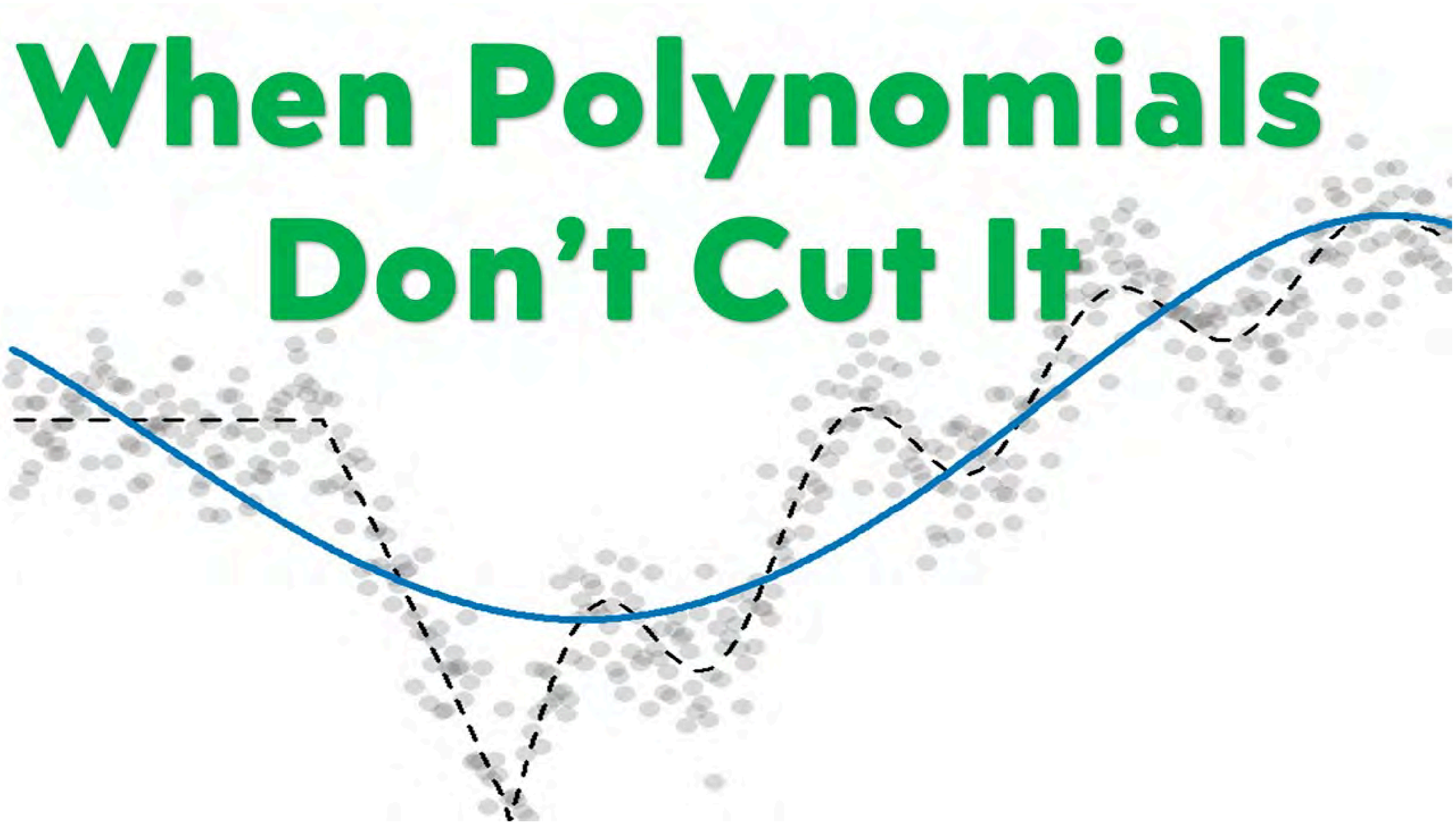
Recommended viewing (splines)



The Continuity of Splines

It won't help with your assessment, it's just very entertaining/interesting.

Recommended viewing (LOESS)



What is LOESS and When Should I Use It?

R Package versions

```r
print(sessionInfo(), locale=FALSE, tzone=FALSE)
```

```
R version 4.4.0 (2024-04-24)
Platform: aarch64-apple-darwin20
Running under: macOS Sonoma 14.5

Matrix products: default
BLAS:   /Library/Frameworks/R.framework/Versions/4.4-arm64/Resources/lib/libRblas.0.dylib
LAPACK: /Library/Frameworks/R.framework/Versions/4.4-arm64/Resources/lib/libRlapack.dylib;  LAPACK version
3.12.0

attached base packages:
[1] splines   stats     graphics  grDevices utils     datasets  methods
[8] base

other attached packages:
 [1] gam_1.22-3      foreach_1.5.2   glue_1.7.0      plotly_4.10.4
 [5] mgcv_1.9-1      nlme_3.1-165    lubridate_1.9.3 forcats_1.0.0
 [9] stringr_1.5.1   dplyr_1.1.4     purrr_1.0.2     readr_2.1.5
[13] tidyr_1.3.1     tibble_3.2.1    ggplot2_3.5.1   tidyverse_2.0.0

loaded via a namespace (and not attached):
 [1] utf8_1.2.4        generics_0.1.3    stringi_1.8.4     lattice_0.22-6
 [5] hms_1.1.3         digest_0.6.36     magrittr_2.0.3    evaluate_0.24.0
```

Python Package versions

```python
from watermark import watermark
print(watermark(python=True, packages="matplotlib,numpy,pandas,seaborn,scipy"))
```

```
Python implementation: CPython
Python version       : 3.11.9
IPython version      : 8.26.0

matplotlib: 3.9.0
numpy     : 1.26.4
pandas    : 2.2.2
seaborn   : 0.13.2
scipy     : 1.11.0
```