8 Splines

8.1 The problem

Often relationships are not linear, but linear models can be used to model non-linear relationships.

We will look at three methods:

1. Transformations
2. Polynomials
3. Splines

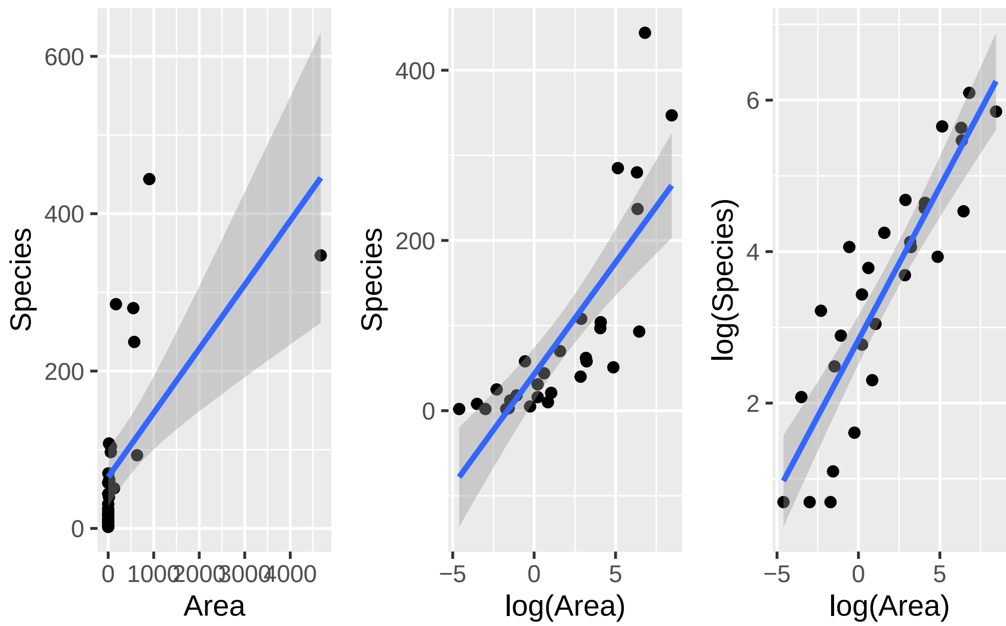
Much of this chapter is currently based on: https://statistics4ecologists-v3.netlify.app/04-nonlinearmodels

8.2 Transformations

We will use the `gala` data set to illustrate this. This gives the number of plant species on 29 Galapagos islands.

```
gala <- read_rds(here::here("data/gala.rds"))

p1 <- ggplot(gala, aes(Area, Species)) + geom_point() +
  geom_smooth(method = "lm")
p2 <- ggplot(gala, aes(log(Area), Species)) + geom_point() +
  geom_smooth(method = "lm")
p3 <- ggplot(gala, aes(log(Area), log(Species))) + geom_point() +
  geom_smooth(method = "lm")
p1 + p2 + p3
```

```
`geom_smooth()` using formula = 'y ~ x'
`geom_smooth()` using formula = 'y ~ x'
`geom_smooth()` using formula = 'y ~ x'
```

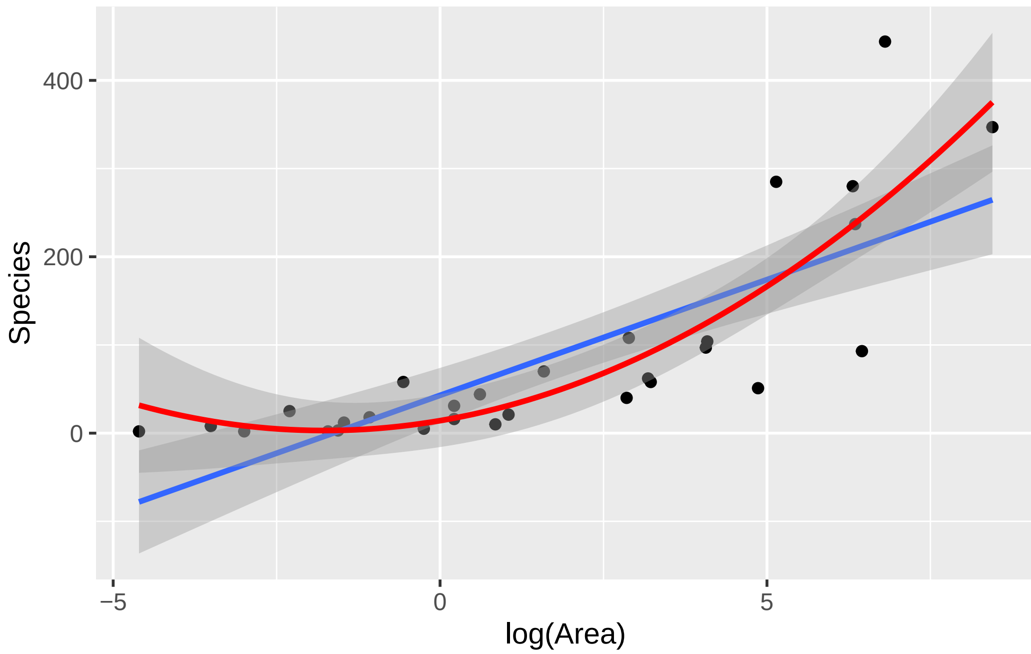


8.3 Polynomials

We can introduce some flexibility to the models using polynomials.

```
ggplot(gala, aes(log(Area), Species)) + geom_point() +
  geom_smooth(method = "lm") +
  geom_smooth(method="lm", formula= y ~ poly(x, 2), se = TRUE, col = "red")
```

```
`geom_smooth()` using formula = 'y ~ x'
```



How can we fit such a model?

```
m1 <- lm(Species ~ log(Area), data = gala)
m2a <- lm(Species ~ log(Area) + I(log(Area)^2), data = gala)
m2b <- lm(Species ~ poly(log(Area), 2, raw = TRUE), data = gala)
coef(m2a)
```

```
   (Intercept)      log(Area) I(log(Area)^2)
     14.152955      12.622562       3.564096
```

```
coef(m2b)
```

```
                    (Intercept) poly(log(Area), 2, raw = TRUE)1
                      14.152955                       12.622562
poly(log(Area), 2, raw = TRUE)2
                       3.564096
```

8.4 Basis functions

The model that we used here is

$$\hat{y}_i = \hat{\beta}_0 + \hat{\beta}_1 x_i + \hat{\beta}_2 x_i^2$$

This can be rewritten as a basis function

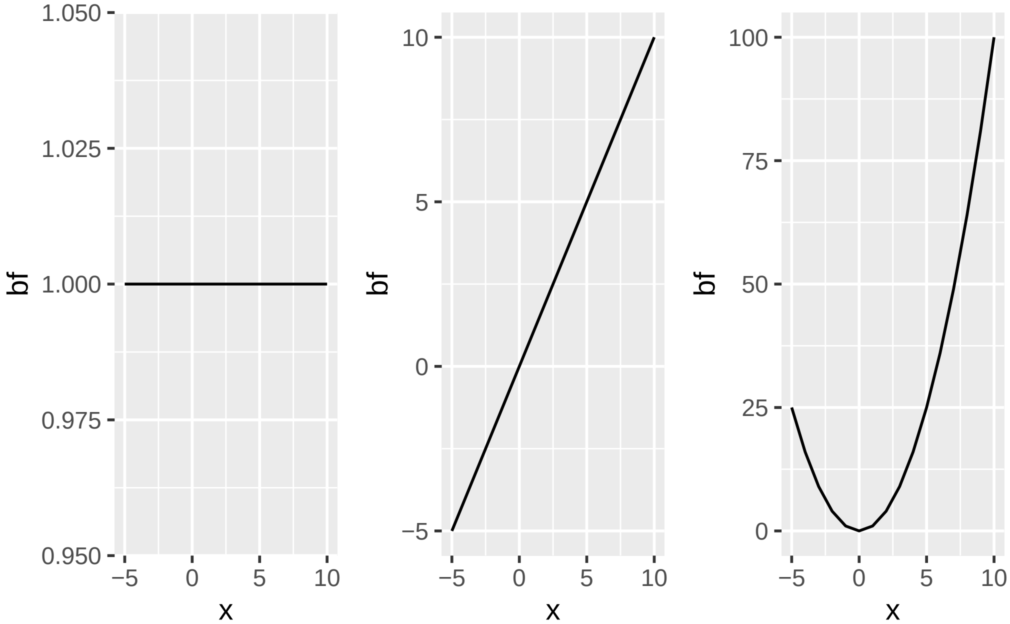
$$y_i = \sum_{j=0}^{p} \beta_j b_j(x_i)$$

```
X <- model.matrix(Species~ poly(log(Area),2, raw=TRUE), data=gala)
head(X)
```

```
  (Intercept) poly(log(Area), 2, raw = TRUE)1 poly(log(Area), 2, raw = TRUE)2
1           1                       3.2224694                     10.38430878
2           1                       0.2151114                      0.04627291
3           1                      -1.5606477                      2.43562139
4           1                      -2.3025851                      5.30189811
5           1                      -2.9957323                      8.97441185
6           1                      -1.0788097                      1.16383029
```

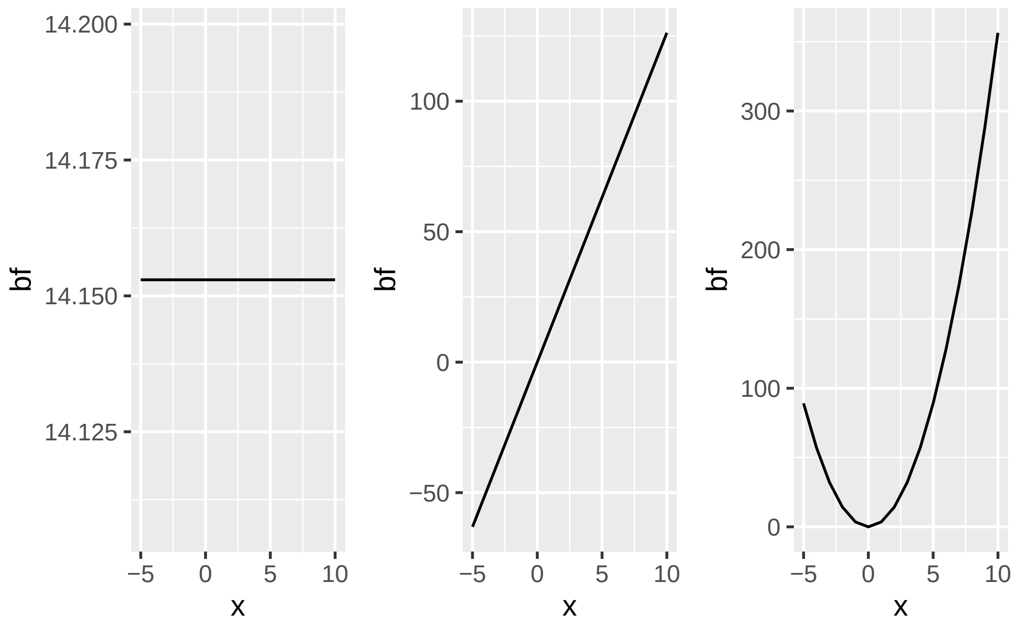
The three basis functions are

```
p1 <- ggplot(tibble(x = -5:10, bf = x^0), aes(x, bf)) +
  geom_line()
p2 <- ggplot(tibble(x = -5:10, bf = x^1), aes(x, bf)) +
  geom_line()
p3 <- ggplot(tibble(x = -5:10, bf = x^2), aes(x, bf)) +
  geom_line()

p1 + p2 + p3
```

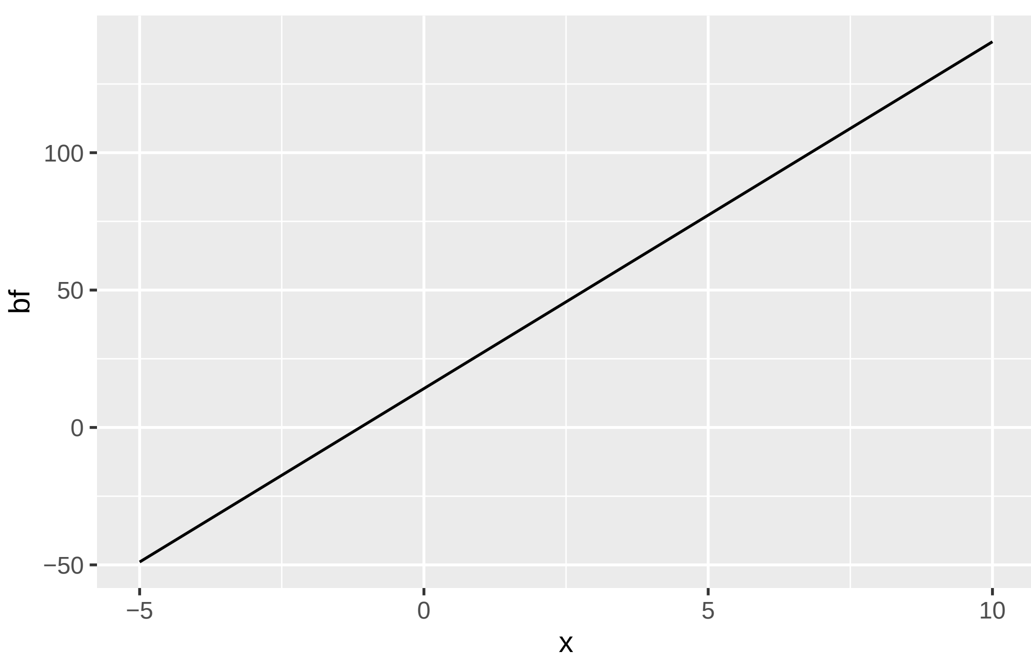


Now, we can scale each basis vector with its coefficient:

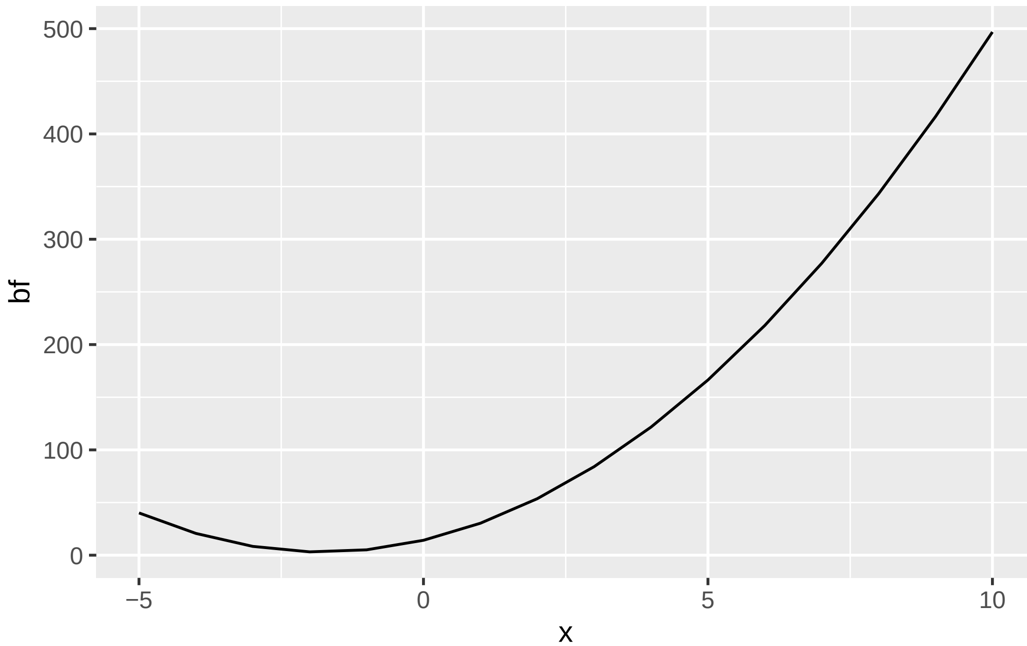
```
p1 <- ggplot(tibble(x = -5:10, bf = x^0 * coef(m2b)[1]),
             aes(x, bf)) + geom_line()
p2 <- ggplot(tibble(x = -5:10, bf = x^1 * coef(m2b)[2]),
             aes(x, bf)) + geom_line()
p3 <- ggplot(tibble(x = -5:10, bf = x^2 * coef(m2b)[3]),
             aes(x, bf)) + geom_line()

p1 + p2 + p3
```




```
ggplot(tibble(x = -5:10, bf = x^0 * coef(m2b)[1] +
                x^1 * coef(m2b)[2]),
             aes(x, bf)) + geom_line()
```



```
ggplot(tibble(x = -5:10, bf = x^0 * coef(m2b)[1] +
                x^1 * coef(m2b)[2] +
                x^2 * coef(m2b)[3]),
             aes(x, bf)) + geom_line()
```



We can apply this to polynomials of order D, with

- $D = 1$: linear
- $D = 2$: quadratic
- $D = 3$: cubic

8.5 Splines

Splines are piecewise polynomials that are connected at predefiend **knots**.

Let's look at linear splines first, with two knots at 1 and 4.2. We have to set everything before the knot to 0.

```
gala$logarea <- log(gala$Area)
gala$logarea.1 <- ifelse(gala$logarea < 1, 0, gala$logarea - 1)
gala$logarea.4.2 <- ifelse(gala$logarea < 4.2, 0, gala$logarea - 4.2)
```

And then we can use these covariates in a linear model

```r
m3 <- lm(Species ~ logarea + logarea.1 + logarea.4.2, data = gala)
coef(m3)
```

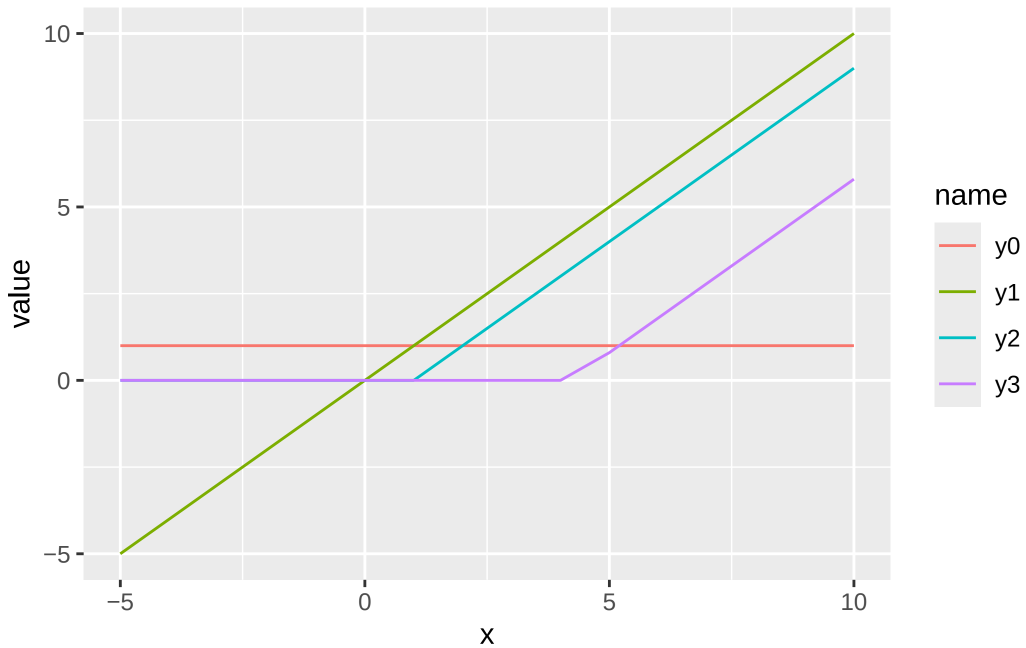
```
(Intercept)     logarea   logarea.1 logarea.4.2
  23.869240    5.212502   17.463683   44.814832
```

We can again calculate the basis functions

```r
bf <- tibble(x = -5:10) |>
  mutate(y0 = 1, y1 = x,
         y2 = ifelse(x < 1, 0, x - 1),
         y3 = ifelse(x < 4.2, 0, x - 4.2)) |>
  pivot_longer(-x)
head(bf, 2)
```

```
# A tibble: 2 x 3
      x name  value
  <int> <chr> <dbl>
1    -5 y0        1
2    -5 y1       -5
```

```r
ggplot(bf, aes(x, value, col = name)) + geom_line()
```

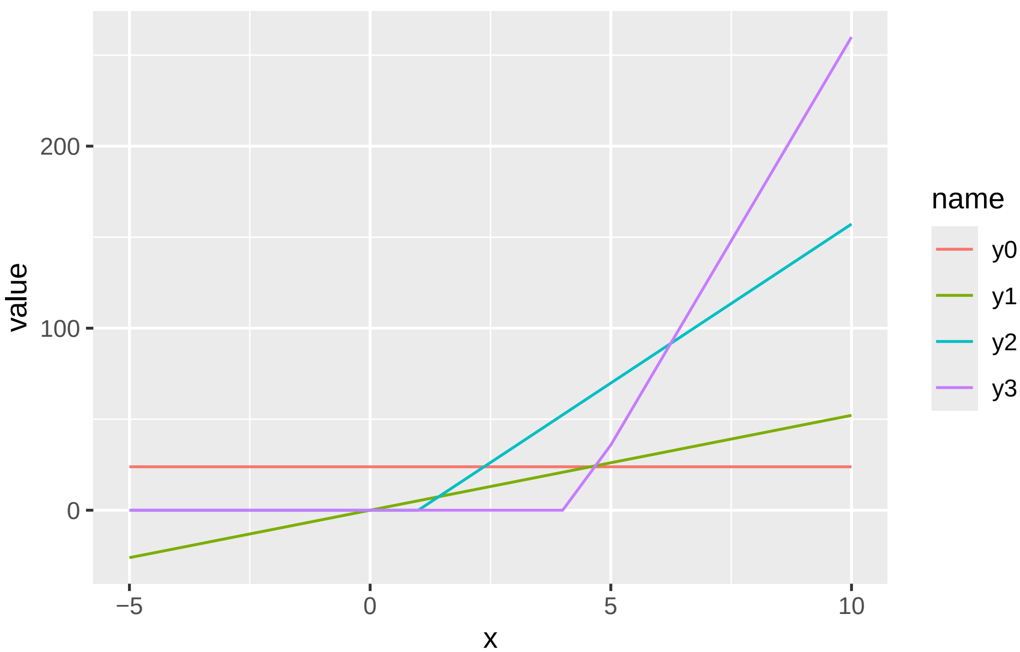


Now lets weight the basis functions with the estimated coefficients of `m3`.

```
bf <- tibble(x = -5:10) |>
  mutate(y0 = 1 * coef(m3)[1],
         y1 = x * coef(m3)[2],
         y2 = ifelse(x < 1, 0, x - 1) * coef(m3)[3],
         y3 = ifelse(x < 4.2, 0, x - 4.2) * coef(m3)[4]) |>
  pivot_longer(-x)
head(bf, 2)
```

```
# A tibble: 2 x 3
      x name  value
  <int> <chr> <dbl>
1    -5 y0     23.9
2    -5 y1    -26.1
```

```
ggplot(bf, aes(x, value, col = name)) + geom_line()
```

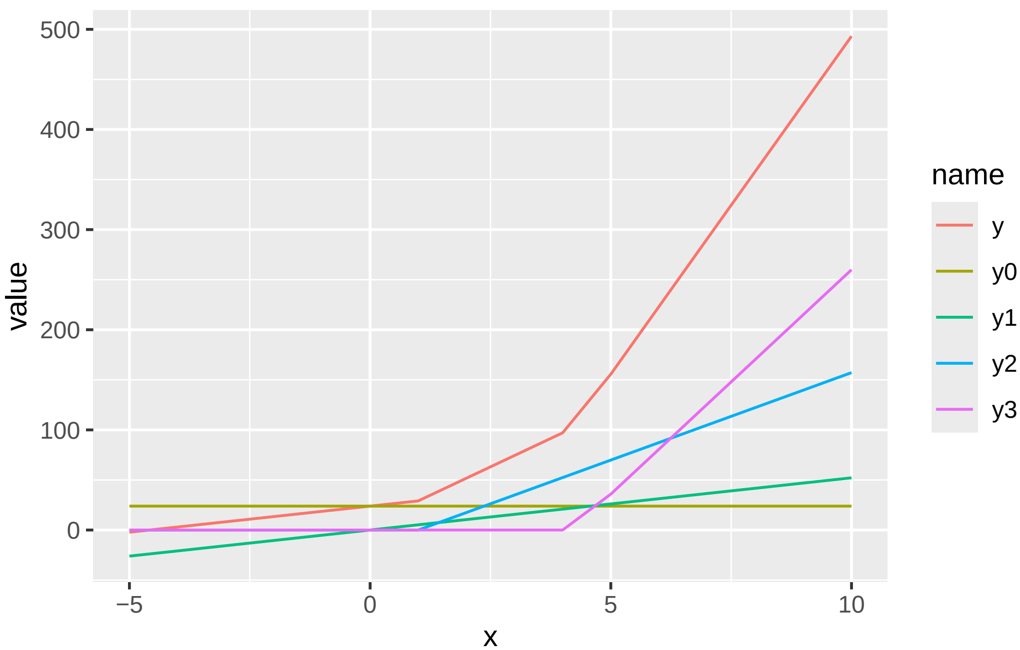


Finally, we can sum the individual lines up

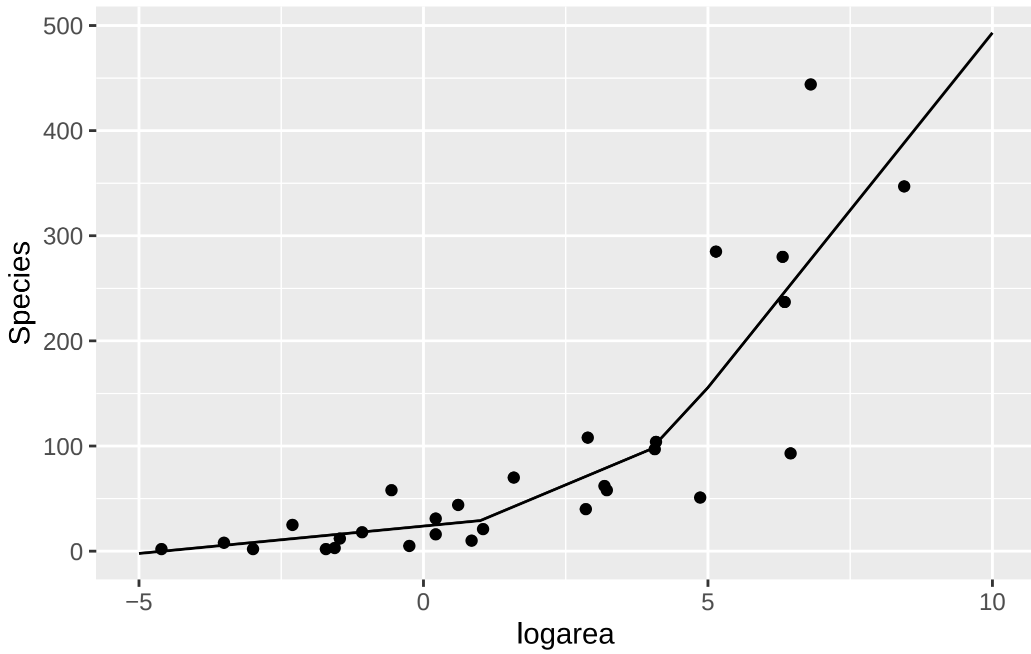
```
bf <- tibble(x = -5:10) |>
  mutate(y0 = 1 * coef(m3)[1],
         y1 = x * coef(m3)[2],
```

```
         y2 = ifelse(x < 1, 0, x - 1) * coef(m3)[3],
         y3 = ifelse(x < 4.2, 0, x - 4.2) * coef(m3)[4],
         y = y0 + y1 + y2 + y3) |>
  pivot_longer(-x)
```

```
ggplot(bf, aes(x, value, col = name)) + geom_line()
```




```
ggplot(gala, aes(logarea, Species)) +
  geom_point() +
  geom_line(aes(x, value), bf |> filter(name == "y"))
```



- Linear splines only ensure that the function is continuous.
- We can use higher order splines (e.g., cubic splines) to ensure that the transitions are smooth.
- To use higher order splines, we can use natural splines or B-splines that are implemented in the `splines` package.

```
library(splines)
head(ns(gala$logarea, 3))
```

```
               1         2          3
[1,]  0.51431980 0.3881911 -0.1468061
[2,]  0.02590396 0.5712523 -0.3539941
[3,] -0.12199115 0.4821792 -0.2988612
[4,] -0.12204050 0.3907838 -0.2422131
[5,] -0.09952908 0.2856200 -0.1770311
[6,] -0.10490254 0.5265790 -0.3263809
```

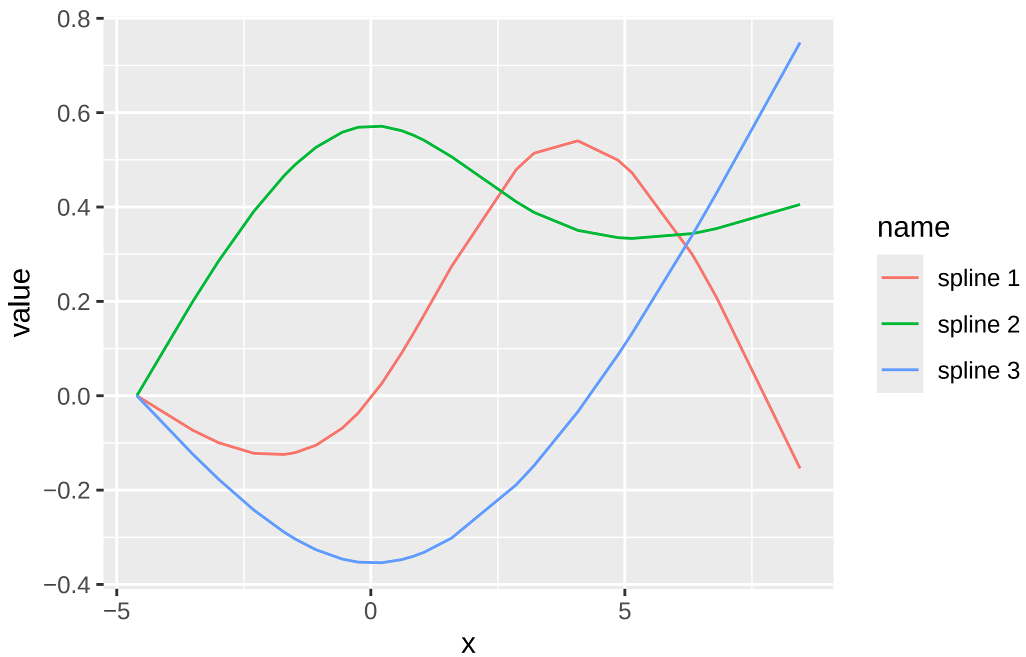
These are again the basis vectors.

```
bf <- ns(gala$logarea, 3) |> as_tibble() |>
  mutate(x = gala$logarea) |>
  pivot_longer(-x) |>
  mutate(name = paste0("spline ", name))
head(bf, 2)
```

```
# A tibble: 2 x 3
      x name     value
  <dbl> <chr>    <ns>
1  3.22 spline 1 0.5143198
2  3.22 spline 2 0.3881911
```

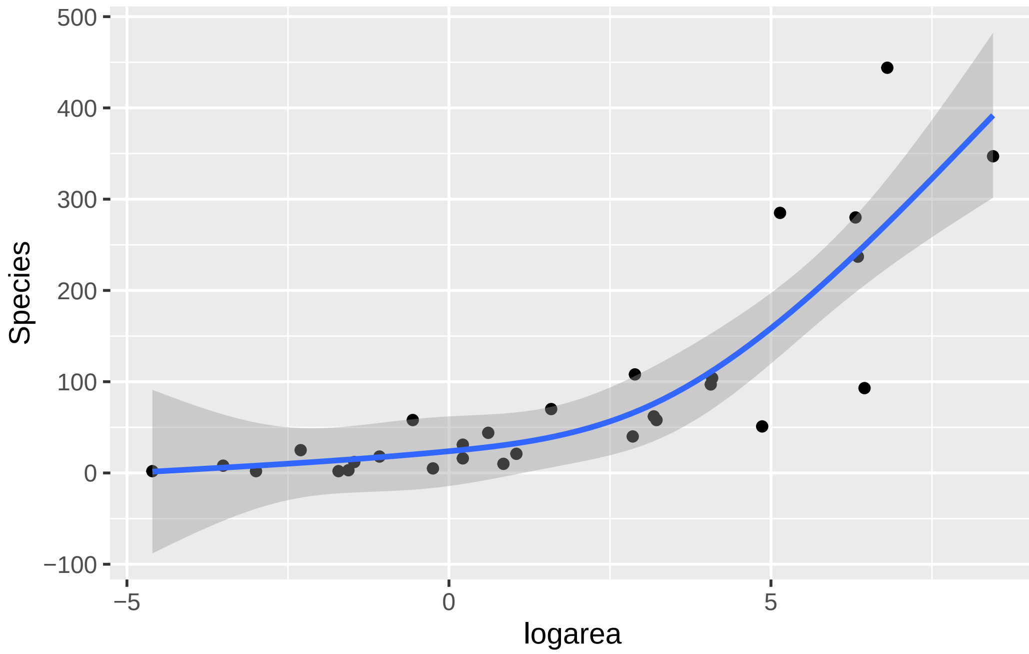
```r
ggplot(bf, aes(x, value, col = name)) +
  geom_line()
```

```
Don't know how to automatically pick scale for object of type
<ns/basis/matrix>. Defaulting to continuous.
```



Adding all up

```r
ggplot(gala, aes(x = logarea, y = Species)) +
  geom_point() +
  geom_smooth(method = "lm", formula = y ~ ns(x, df = 3), se = TRUE)
```



- The locations of the knots can be quantiles (the default) or carefully chosen.
- AIC can be used to select the best number of knots and their locations.
- Use smoothing splines with, where the number of knots is derived from the data within the `mgcv` package.

Exercise 1: Splines

The code, below, reads in data containing estimates of the number of Moose in Minnesota between 2005 and 2020:

```
mnmoose <- data.frame(
  year=2005:2020,
  estimate = c(8160, 8840, 6860, 7890, 7840, 5700,
               4900, 4230, 2760, 4350, 3450, 4020, 3710,
               3030, 4180, 3150))
```

Fit different linear regression (polynomials up to order 5 and natural splines with 3 and 5 knots) model to the data and evaluate whether the assumptions are reasonable.
Plot the data and the model.