

Parametric Splines

MATH 4939

2026-03-07

Hand-Made Parametric Splines

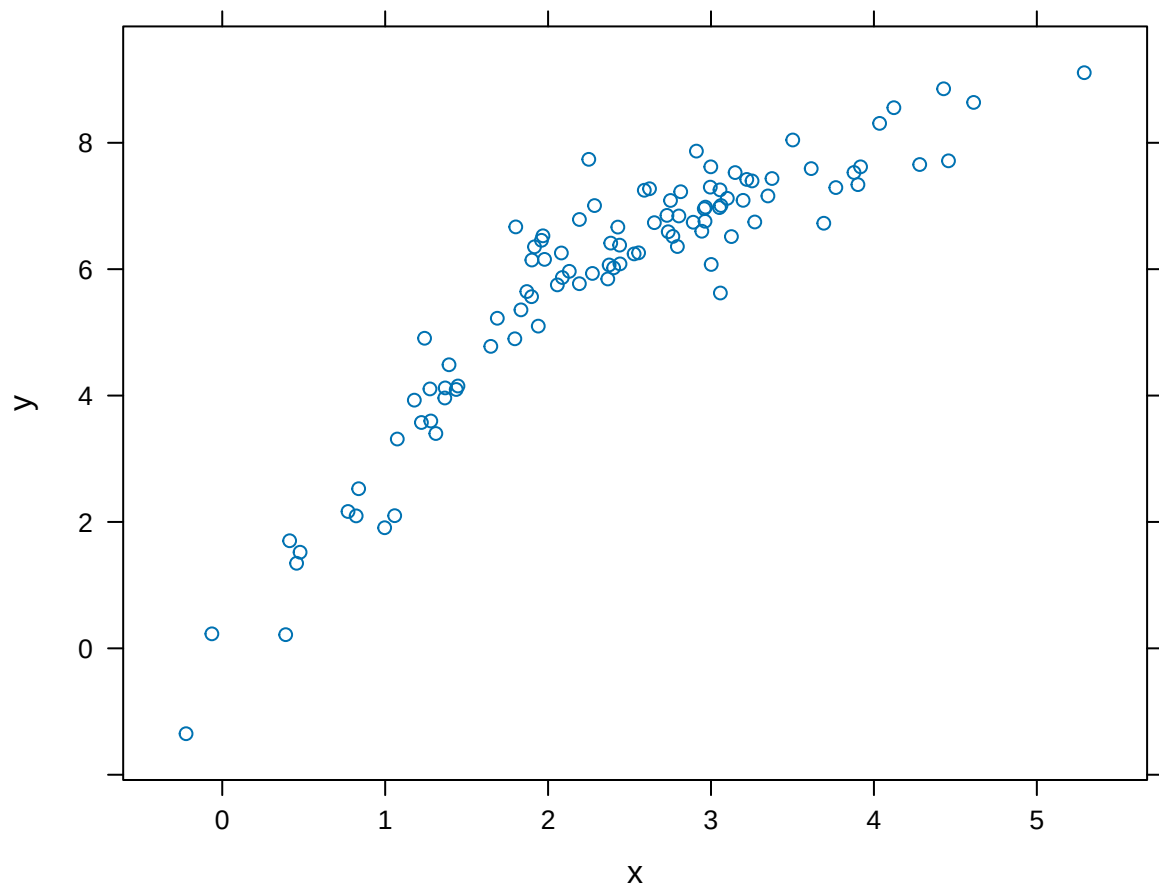
What do you do in regression when a straight line isn't appropriate?

- Transformation informed by knowledge of the dynamics of the process (start here)
- Box-Cox transformations?
 - How to choose transformations without the risk of overfitting?
 - * Cross-validation: see e.g. **cv** package in R
- Polynomials: They go wild at the ends (Runge phenomenon)
- Piecewise polynomials
- Hand-made parametric splines
- Generalized parametric splines (in `spida2`)
- Non-parametric splines: **mgcv** package

Hand-made splines

```
# Generate some sample data
{
  set.seed(1838)
  dd <- data.frame(x = 2.5 + rnorm(100))
  dd <- within(
    dd,
    {
      y <- ifelse(x < 2, 3 * x, 6 + (x - 2)) + .5 * rnorm(x)
    }
  )
  dd <- rbind(dd, c(x = 2, y = NA))
}

{
  dd <- dd[order(dd$x),]
  library(lattice)
  library(latticeExtra)
  p <- xyplot(y ~ x, dd)
  p
}
```



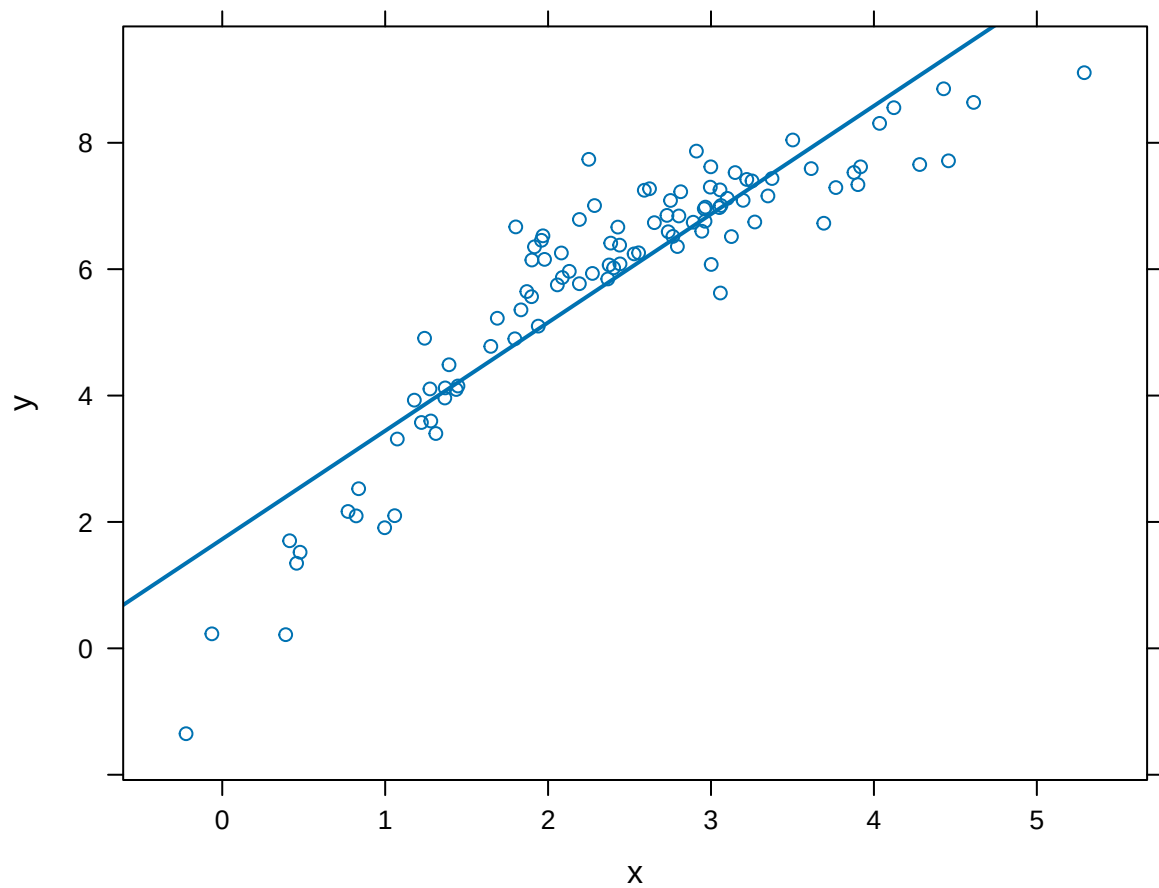
Fitting line

```
fit_line <- lm(y ~ x, dd)

pred <- data.frame(x = seq(-1, 6, .01))

pred $ fit_line <- predict(fit_line, pred)

(
  p2 <- p + xyplot(fit_line ~ x , pred, type = 'l', lwd = 2)
)
```



Suppose we either know or decide to use a **knot** at $x = 2$.

Fit two lines

```
fit_two <- lm(y ~ x * I(x>2), dd, na.action = na.exclude)
```

$$E(Y) = \beta_0 + \beta_1 x + \beta_2 I_{x>2}(x) + \beta_3 x \times I_{x>2}(x)$$

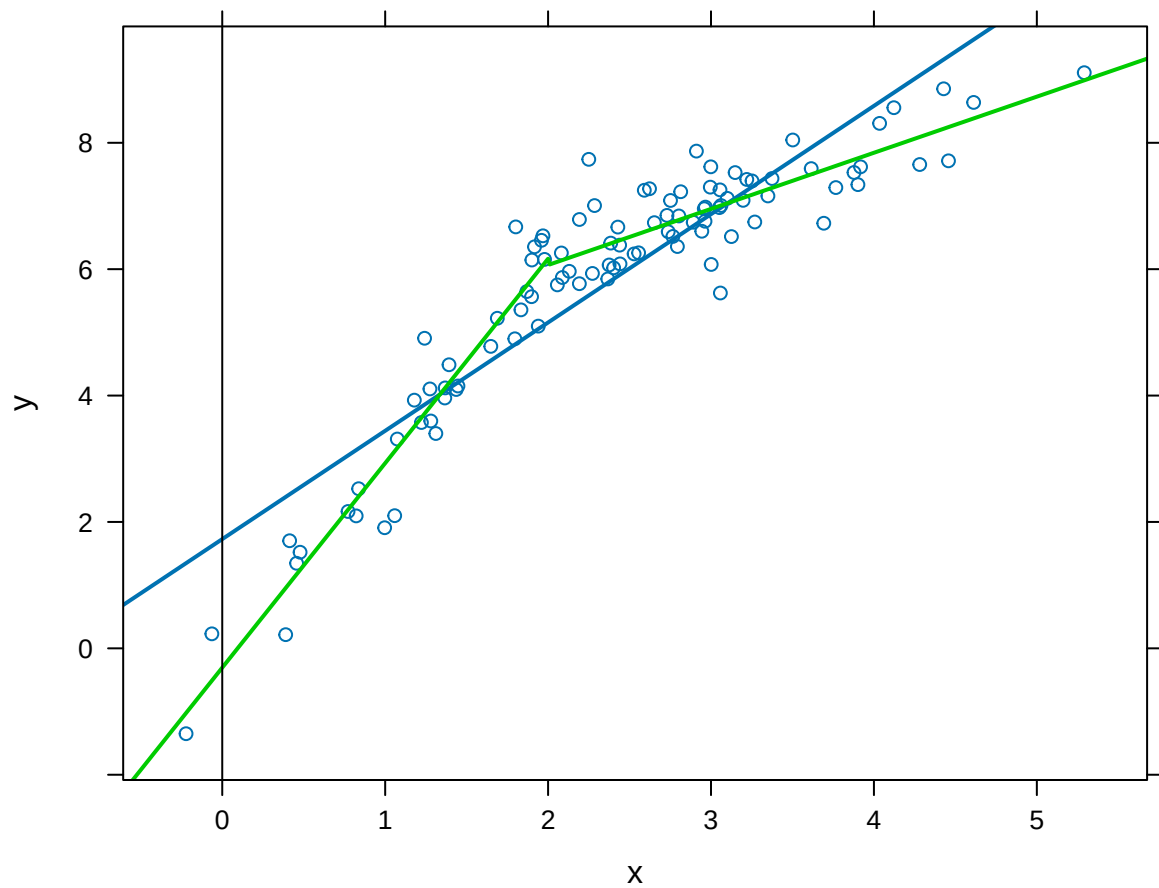
```
summary(fit_two)
```

```
##
## Call:
## lm(formula = y ~ x * I(x > 2), data = dd, na.action = na.exclude)
##
## Residuals:
##      Min       1Q   Median       3Q      Max
## -1.38206 -0.30113 -0.01326  0.28628  1.45034
##
## Coefficients:
##              Estimate Std. Error t value Pr(>|t|)
## (Intercept)   -0.3052     0.1977  -1.544   0.126
## x              3.2355     0.1413  22.902 <2e-16 ***
## I(x > 2)TRUE    4.5951     0.3426  13.414 <2e-16 ***
## x:I(x > 2)TRUE -2.3475     0.1675 -14.017 <2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
```

```
## Residual standard error: 0.4971 on 96 degrees of freedom
## (1 observation deleted due to missingness)
## Multiple R-squared: 0.9422, Adjusted R-squared: 0.9404
## F-statistic: 521.3 on 3 and 96 DF, p-value: < 2.2e-16
```

```
pred$fit_two <- predict(fit_two, pred)

(
  p3 <-
    p2 +
    xyplot(fit_two ~ x, pred, type = 'l', col = '#00CC00', lwd = 2)+
    layer(panel.abline(v = 0))
)
```



How to get rid of the discontinuity?

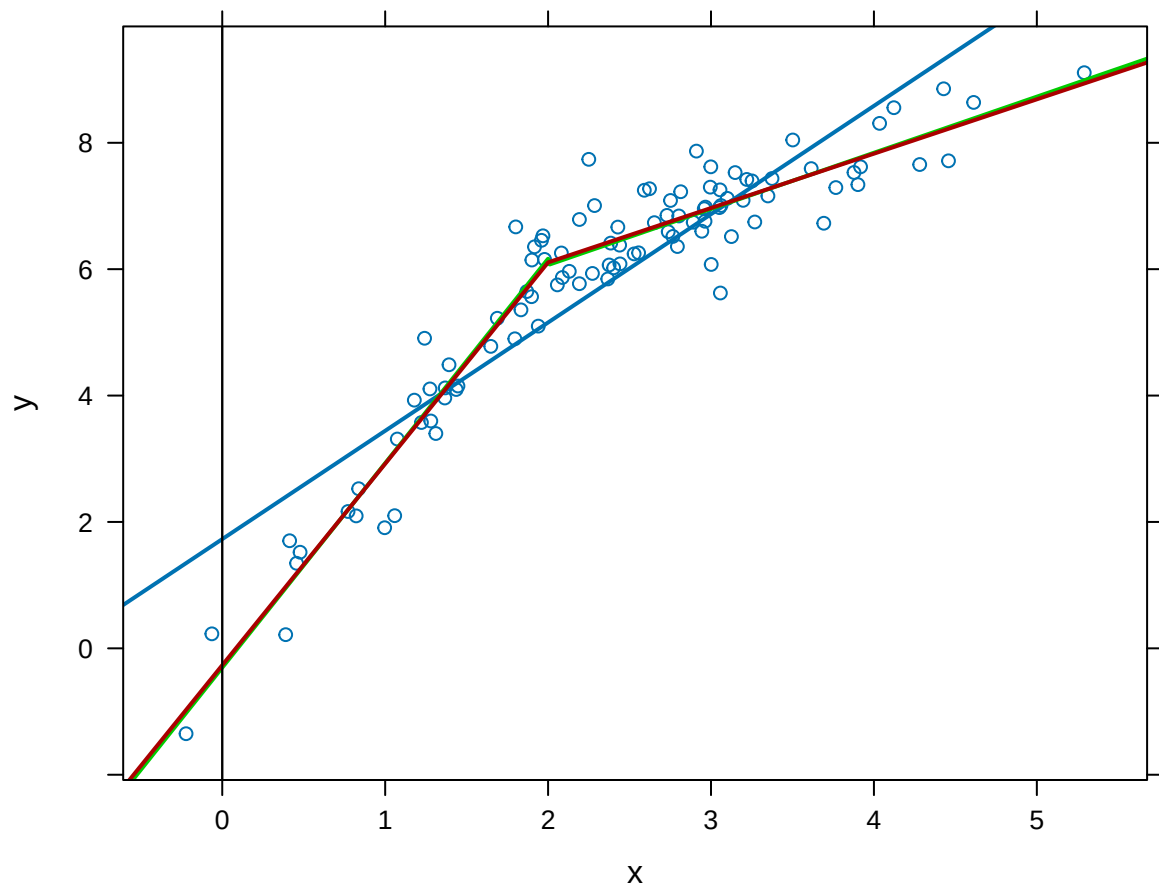
```

plus <- function(x) x * I(x > 0)

fit_linear.spline <- lm(y ~ x + plus(x - 2), dd)

pred$fit_linear.spline <- predict(fit_linear.spline, newdata = pred)
(
  p4 <-
    p3 +
    xyplot(fit_linear.spline ~ x, pred, type = 'l', col = '#AA0000', lwd = 2)+
    layer(panel.abline(v = 0))
)

```



Traditional quadratic spline: knots at 1, 2, 3.5

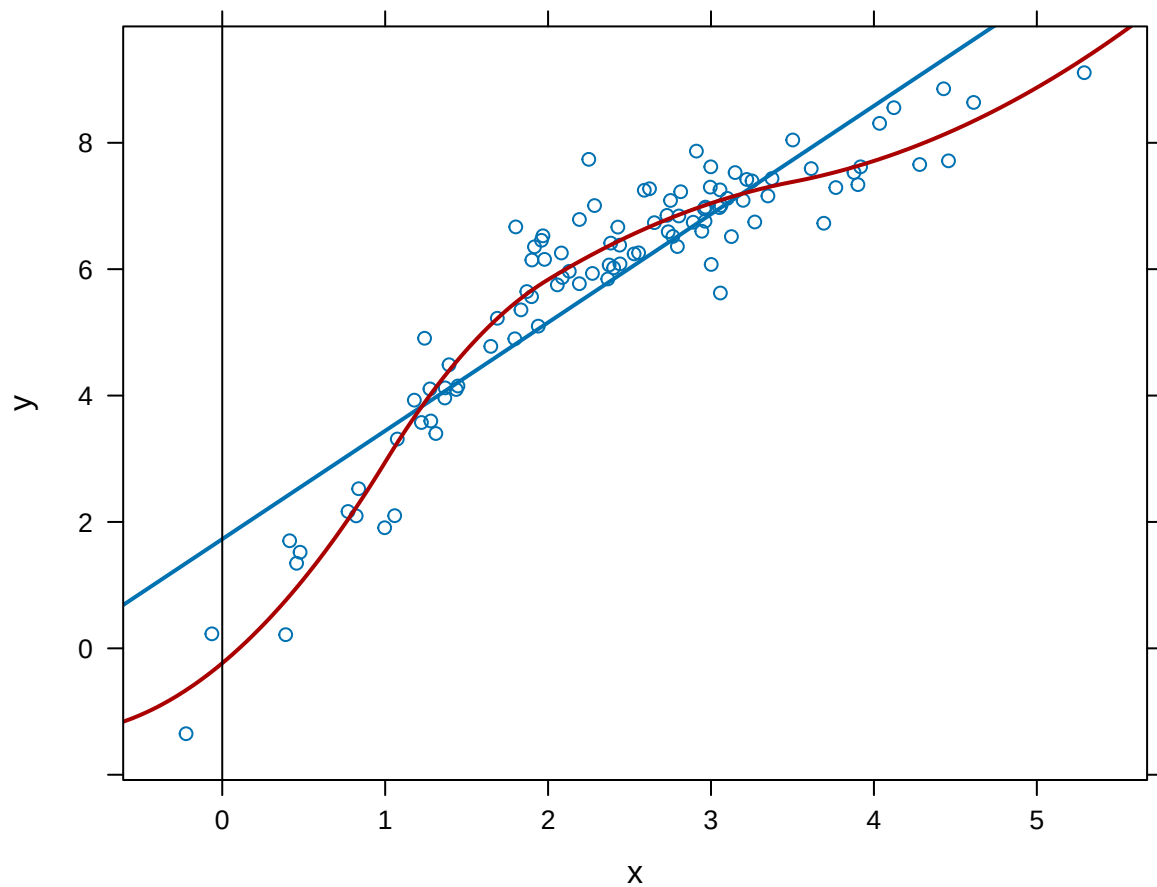
```

fit_quad.spline <- lm(y ~ x + I(x^2) +
                      I(plus(x-1)^2) +
                      I(plus(x-2)^2) +
                      I(plus(x-3.5)^2), dd)

pred$fit_quad.spline <- predict(fit_quad.spline, pred)

(
  p5 <-
    p2 +
    xyplot(fit_quad.spline ~ x, pred, type = 'l', col = '#AA0000', lwd = 2)+
    layer(panel.abline(v = 0))
)

```



```
fit_cubic.spline <- lm(y ~ x + I(x^2) + I(x^3) +  
                        I(plus(x-1)^3) +
```

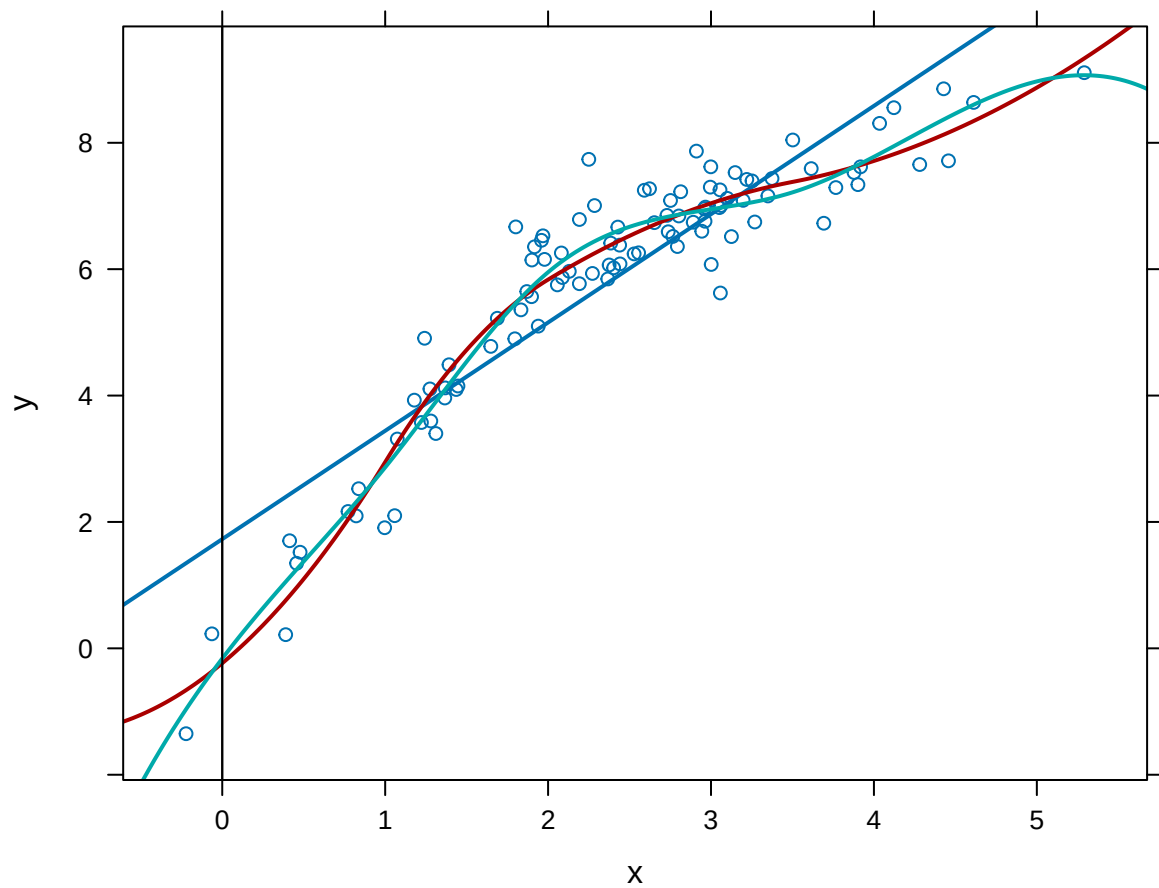
```

      I(plus(x-2)^3) +
      I(plus(x-3.5)^3), dd)

pred$fit_cubic.spline <- predict(fit_cubic.spline, pred)

(
  p6 <-
    p5 +
    xyplot(fit_cubic.spline ~ x, pred, type = 'l', col = '#00AAAA', lwd = 2)+
    layer(panel.abline(v = 0))
)

```



Using a 'natural' spline

```

library(spida2)

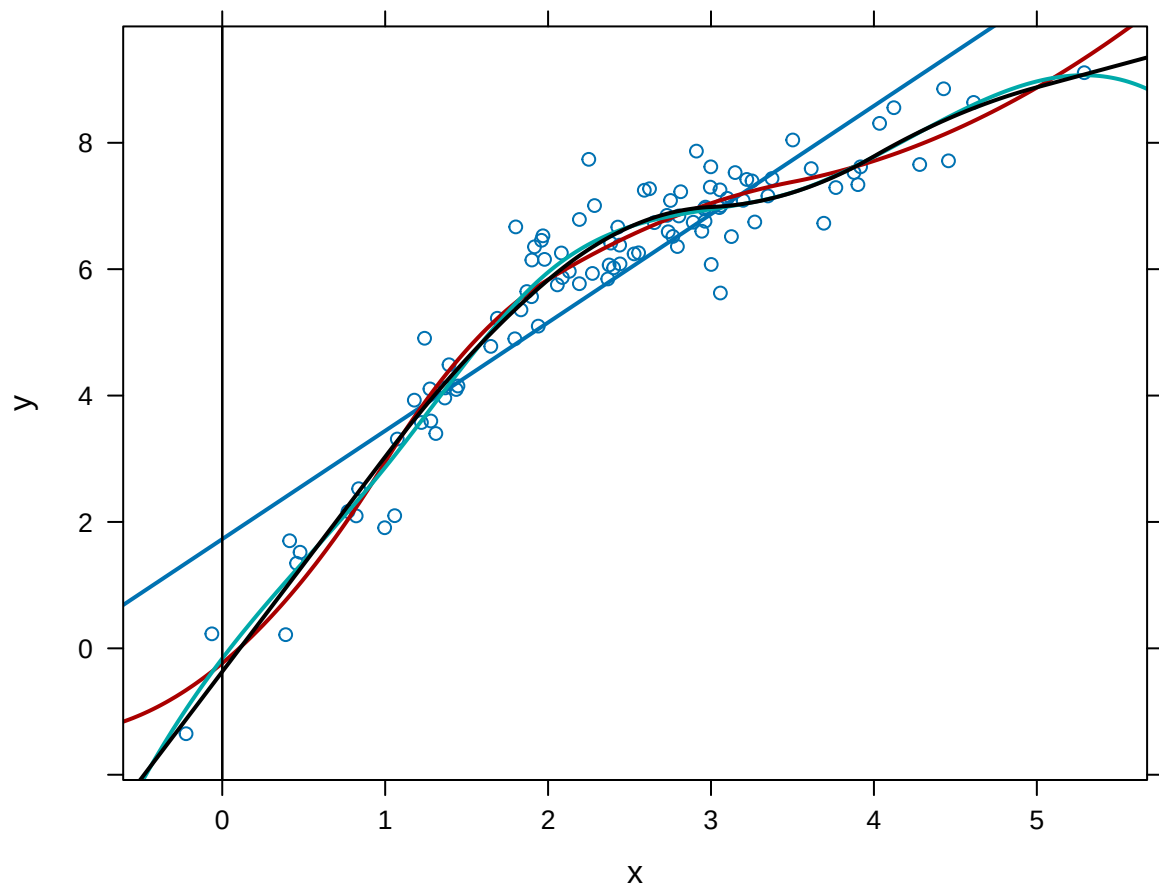
spn <- function(x) gsp(x, 1:5, c(1,2,2,2,2,1), 1)

fit_natural.spline <- lm(y ~ spn(x), dd)

pred$fit_natural.spline <- predict(fit_natural.spline, pred)

(
  p7 <-
    p6 +
    xyplot(fit_natural.spline ~ x, pred, type = 'l', col = 'black', lwd = 2)+
    layer(panel.abline(v = 0))
)

```



But what if we would like to have

- different polynomial degrees in different intervals and
- different degrees of continuity at different knots

See the next example