Introduction to Machine Learning

Working Group “Computational Statistics” – Bernd Bischl et al.

Splines

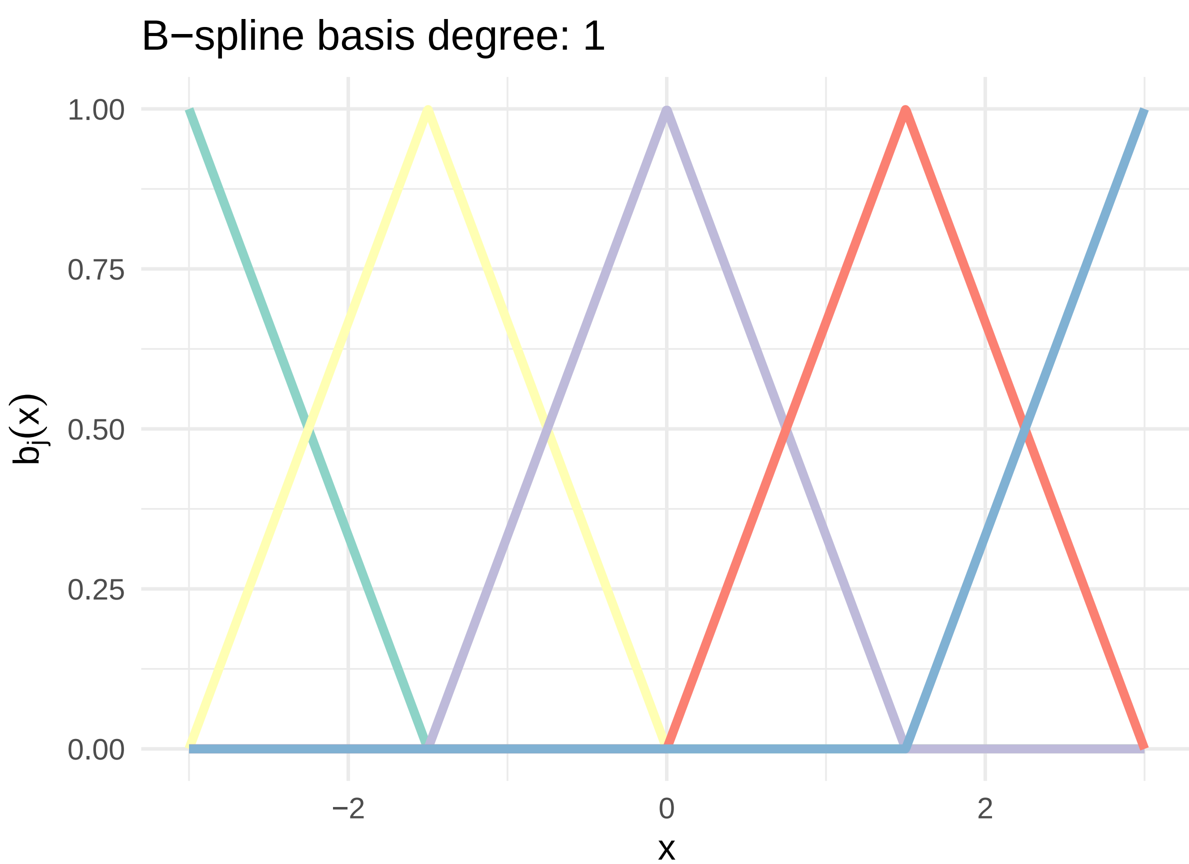
Basic idea

A very important idea in machine learning consists in transforming features of interest or creating new features from the data. Here we will take a look at so-called splines and spline basis functions.
With splines, we can model certain functional relationships arbitrarily well. To do so, we re-represent the features as evaluations of a spline basis. We can treat these new features as additional features *of an extended linear model*. Using empirical risk minimization, we can then get an estimate for the regression coefficients θ_j associated with these spline basis features $b_j(x), j = 1, \dots, K$, which determine the shape of the spline function, i.e., the functional relationship $f(x)$ between the feature x and the target y:

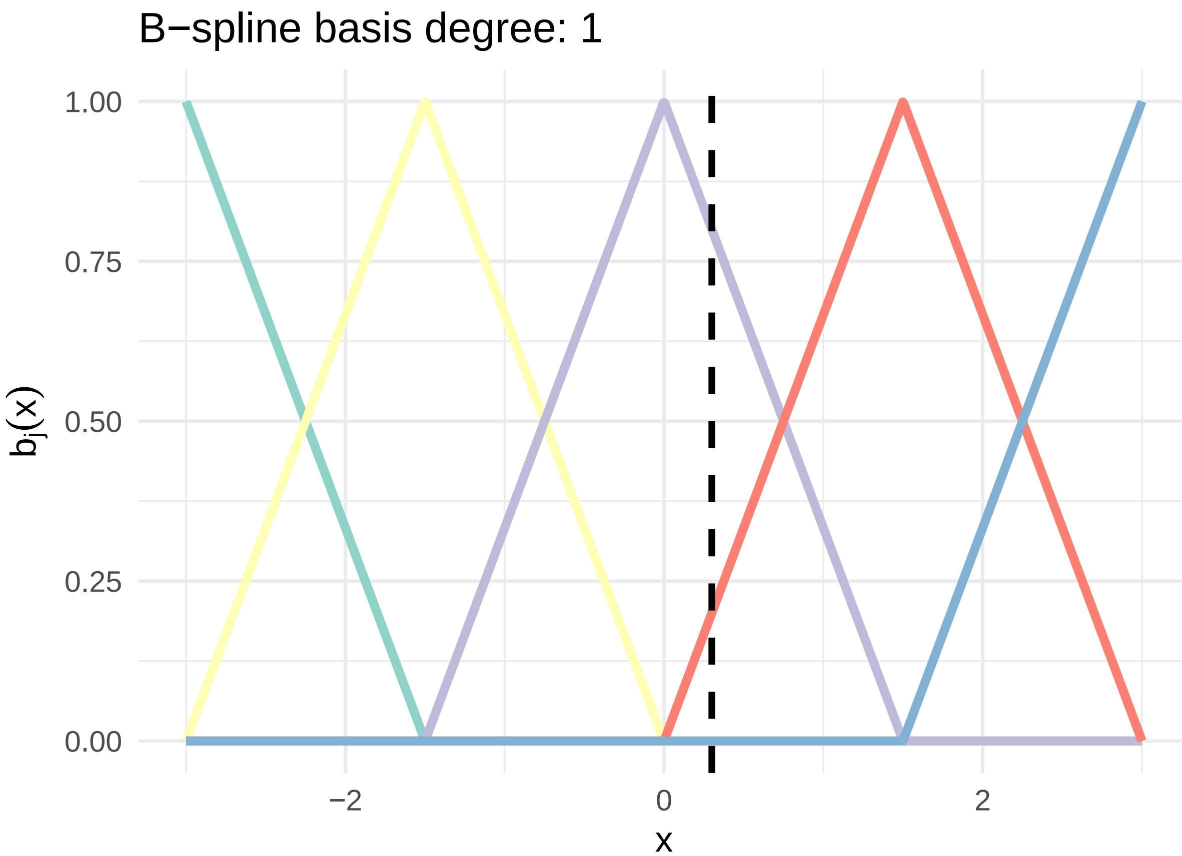
$$f(x) = \sum_{j=1}^{K} b_j(x)\theta_j,$$

B-splines

One of the most commonly used spline bases is the so-called B-spline basis, which consists of locally defined polynomials. To use splines we have to fill our domain with knots $t_0, \dots, t_{m+1}$, which will serve as supporting points for the spline. For example we can use 5 equidistant knots with a piecewise polynomial degree of $l = 1$:



The number of B-spline basis function K is $K = m + l + 1$. From our machine learning perspective, this means that we are projecting the values of the feature x into a $(3 + 1 + 1 = 5)$-dimensional vector space with entries $\boldsymbol{b}(x) = (b_1(x), b_2(x), \dots, b_5(x))$ in our little example, s.t. it holds e.g. for $x = 0.3$:

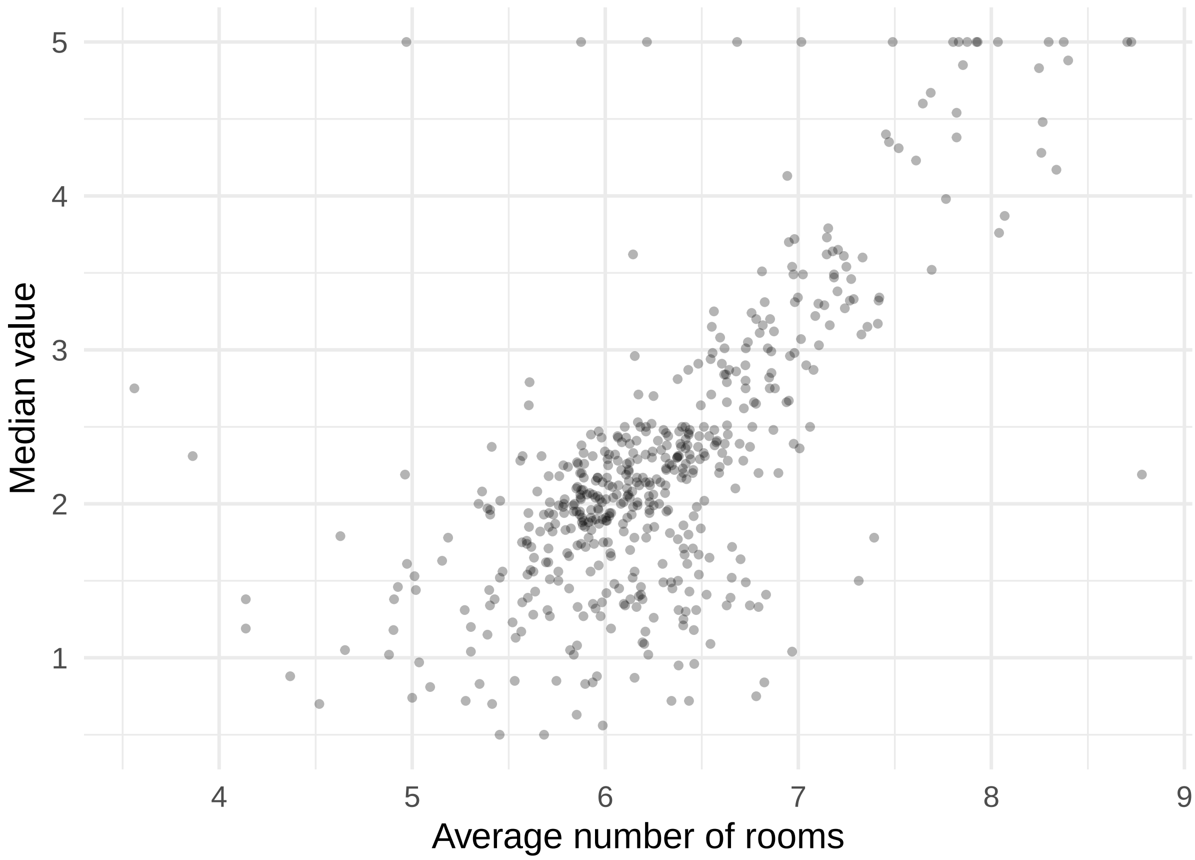



```
## [1] "b(x) = (0, 0, 0.8, 0.2, 0)"
```

Example

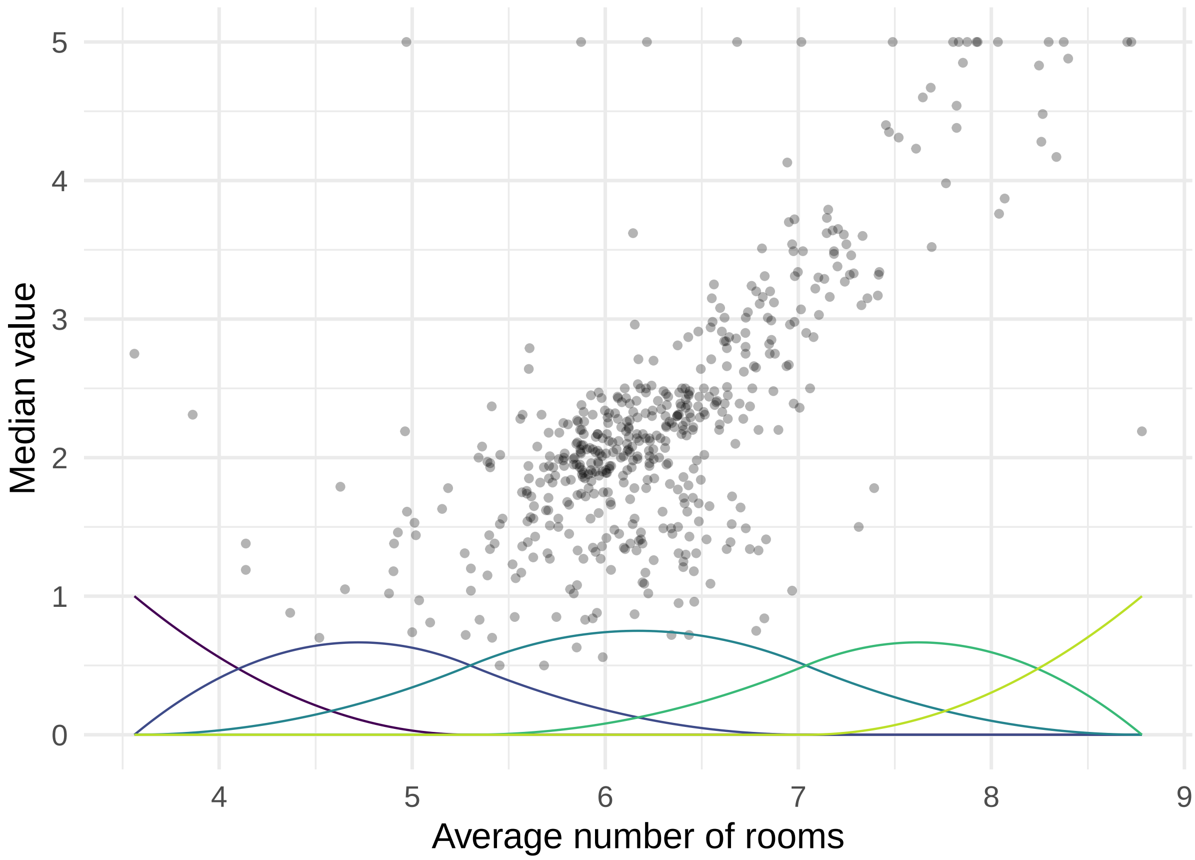
Let's look at some real data, where we want to model a functional relationship:

```
library(mlbench)
library(ggplot2)
library(splines)
data(BostonHousing2)
boston_housing <- BostonHousing2
boston_housing$medv <- boston_housing$medv/10 # rescale for nicer plots

(medv_rooms_plot <- ggplot() +
    geom_point(data = boston_housing, aes(x = rm, y = medv), alpha = 0.3) +
    labs(x = "Average number of rooms",
         y = "Median value"))
```



Now let's say we want to use 5 piecewise polynomials of degree 2:

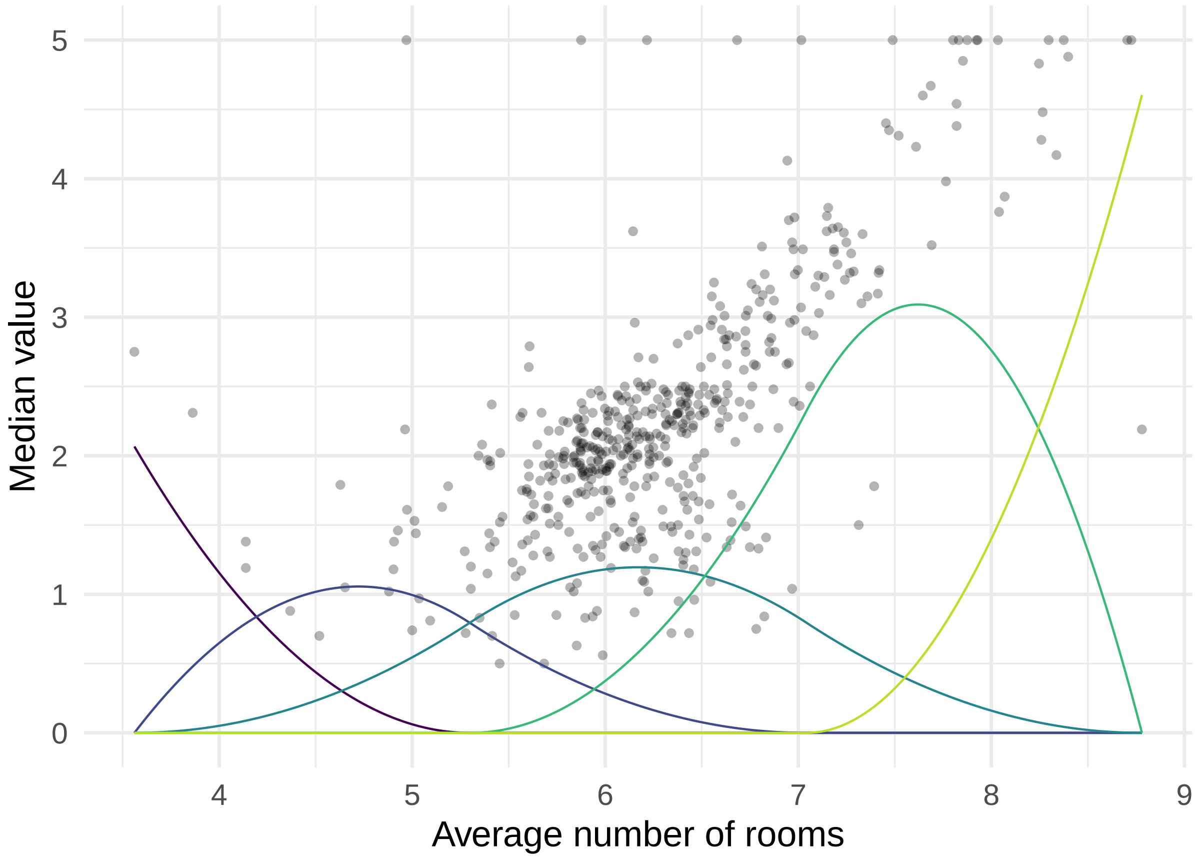
```
library(reshape2)
poly_deg <- 2
num_bfuns <- 5
num_data <- 1000 # number of points we want to use for plotting

rm_min <- min(boston_housing$rm)
rm_max <- max(boston_housing$rm)

rooms <- seq(rm_min, rm_max, length.out = num_data)

bbasis_plot <- bs(rooms,
                  df = num_bfuns ,
                  degree = poly_deg,
                  intercept = TRUE)

plot_data <- melt(data.frame(cbind(bbasis_plot, rooms)),
                  id = "rooms")

medv_rooms_plot +
  geom_line(data = plot_data, aes(x = rooms, y = value, color = variable)) +
  theme(legend.position = "none")
```



Train a linear model using the 5 new features corresponding to the B-spline basis evaluations:

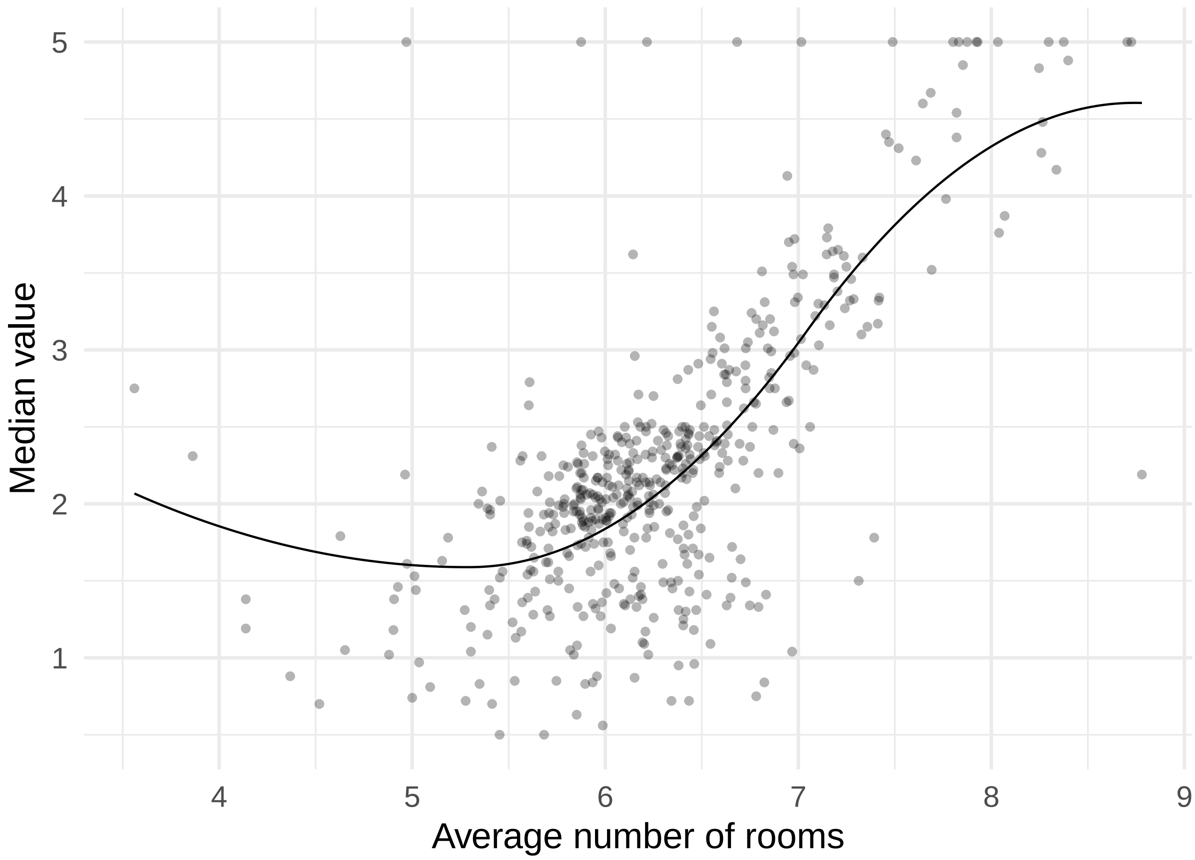
```
# since we want to use the same transformation, we have to specify the knots and
# piecewiese polynomial degree, we have used before
bbasis_data <- data.frame(
    bs(
      x = boston_housing$rm,
      Boundary.knots = attr(bbasis_plot, "Boundary.knots"),
      knots = attr(bbasis_plot, "knots"),
      degree = poly_deg,
      intercept = TRUE
    )
  )
bbasis_data$medv <- boston_housing$medv

# estimate a linear model for transformed features.
# exclude the intercept as it is contained in the B-splines basis functions.
lm_bs <- lm(medv ~ . - 1, data = bbasis_data)

# use estimated coefficients of linear model to re-scale the B-spline features:
for (i in 1:ncol(bbasis_plot)) {
  bbasis_plot[, i] = bbasis_plot[, i] *
    lm_bs$coefficients[i]
}

plot_data <-
```

```
  melt(data.frame(cbind(bbasis_plot, rooms)), id = "rooms")

medv_rooms_plot +
  geom_line(data = plot_data, aes(x = rooms, y = value, color = variable)) +
  theme(legend.position = "none")
```



The estimated function relationship is simply the sum of the scaled features:

```
function_estimate <- data.frame(x = rooms)

function_estimate$y <- rowSums(bbasis_plot)

medv_rooms_plot + geom_line(data = function_estimate, aes(x = rooms, y = y)) +
  theme(legend.position = "none")
```



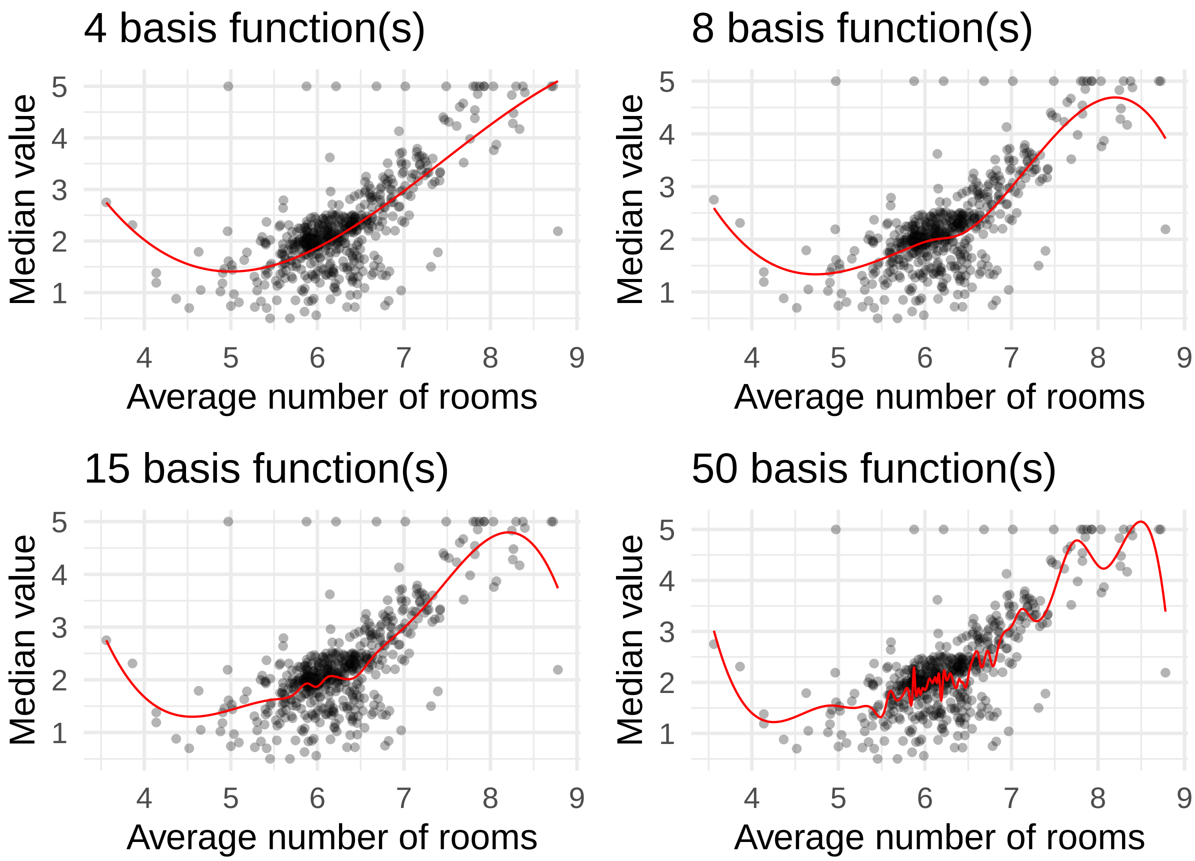
Wrap these steps up in one plot function:

```
# function that plots the b-spline fit
plot_bs_fit <- function(data,
                        x, y,
                        poly_deg,
                        num_bfuns,
                        x_title="x", y_title="y") {
  num_data <- 1000
  bspline <- bs(
      data[[x]],
      df = num_bfuns,
      degree = poly_deg,
      intercept = TRUE
    )
  bspline_data <-  data.frame(bspline)
  bspline_data$y <- data[[y]]
  # estimate a linear model for transformed features (without intercept)
  lm_bs <- lm(y ~ . -1, data = bspline_data)

  plot_data <- data.frame(x = seq(
      min(data[[x]]),
      max(data[[x]]),
      length.out = num_data
    ))
  # scale and add up (i.e. use matrix product)
```

```
  plot_data$y <- bs(
      plot_data$x,
      knots = attr(bspline, "knots"),
      degree = poly_deg,
      Boundary.knots = attr(bspline, "Boundary.knots"),
      intercept = TRUE
    ) %*% lm_bs$coefficients

  ggplot() +
    geom_point(data = data, aes_string(x = x, y = y), alpha = 0.3) +
    geom_line(data = plot_data, aes(x = x, y = y), color = "red") +
    labs(
      title = paste(as.character(num_bfuns), "basis function(s)"),
      x = x_title,#,
      y = y_title#
    )
}
```

Vary number of basis functions for cubic B-splines:

```
library(ggplot2)
library(gridExtra)

poly_deg <- 3
num_bfuns <- c(4, 8, 15, 50)
ggplot_list <- lapply(num_bfuns, function(x)
    plot_bs_fit(data = boston_housing, x = "rm", y = "medv", poly_deg = poly_deg,
                num_bfuns = x, x_title="Average number of rooms",
            y_title = "Median value"))
do.call(grid.arrange, ggplot_list)
```



- We really only fitted a simple linear model, but instead of the original feature we used transformed features.
- We observe that the fit strongly depends on the number of basis functions/knots that we choose.
- For higher number of basis functions/knots a phenomenon called overfitting, where the model fits the observed data very well, but does not generalize well on unseen data, can be seen which will be discussed in chapter 3.