Machine Learning for Biostatistics

Module 5

Jaroslaw Harezlak & Armando Teixeira-Pinto

2025-08-08

Contents

Beyond Linearity **5**
Introduction 5
Dataset used in the examples 5
Slides from the videos 6

1 Polynomial Regression **7**
1.1 Introduction 7
1.2 Readings 8
1.3 Practice session 8
1.4 Exercises 15

2 Piecewise Regression and Splines **17**
2.1 Introduction 17
2.2 Readings 18
2.3 Practice session 19
2.4 Exercises 29

3 Smoothing splines **33**
3.1 Introduction 33
3.2 Readings 34
3.3 Practice session 34
3.4 Exercises 41

Beyond Linearity

Introduction

This module will cover methods to explore non-linear effects of numerical predictors on the outcome.

By the end of this module you should be able to:

1. Identify approaches to model non-linear effects
2. Implement linear and polynomial piecewise regression
3. Understand the difference between polynomial splines, b-splines and natural splines
4. Fit a GLM with different splines
5. Use smoothing splines to approximate non-linear effects
6. Integrate smoothing splines in modeling strategies using generalised additive models

Dataset used in the examples

The dataset **triceps** is available in the `MultiKink` package. You may `install.packages("MultiKink")`, load the library (`library(MultiKink)`) and then run `data("triceps")`.

The data are derived from an anthropometric study of 892 females under 50 years in three Gambian villages in West Africa. There are 892 observations on the following 3 variables:

- age - Age of respondents.
- lntriceps - Log of the triceps skinfold thickness.
- triceps - Triceps skinfold thickness.

The data SA_heart.csv is retrospective sample of males in a heart-disease high-risk region of the Western Cape, South Africa. There are roughly two controls per case of CHD.

Many of the CHD positive men have undergone blood pressure reduction treatment and other programs to reduce their risk factors after their CHD event. In some cases the measurements were made after these treatments. These data are taken from a larger dataset, described in Rousseauw et al, 1983, South African Medical Journal.

The data contains 462 observations on the following 10 variables.

- sbp - systolic blood pressure
- tobacco - cumulative tobacco (kg)
- ldl - low density lipoprotein cholesterol
- adiposity - a numeric vector
- famhist - family history of heart disease, a factor with levels Absent Present
- typea - type-A behavior
- obesity - a numeric vector
- alcohol - current alcohol consumption
- age - age at onset
- chd- response, coronary heart disease (1 - chd, 0 - no chd)

Slides from the videos

You can download the slides used in the videos form Beyond Linearity:

Slides

Chapter 1

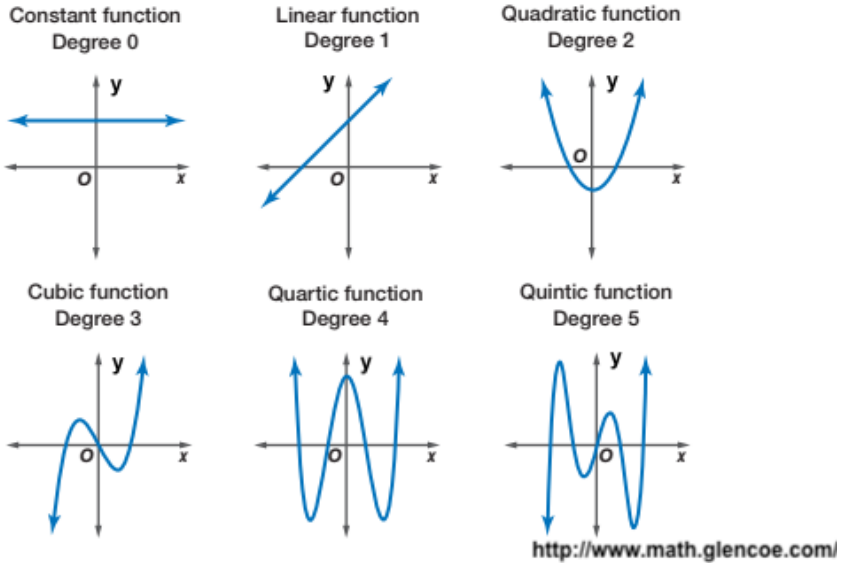
Polynomial Regression

1.1 Introduction

The extension of the linear models $y = \beta_0 + \beta_1 x + \varepsilon$ to include higher degree polynomial terms x^2, x^3, ..., x^p is straightforward. Each additional term can be viewed as another predictor in the regression equation:

$$y = \beta_0 + \beta_1 x + \beta_2 x^2 + \cdots + \beta_p x^p + \varepsilon$$

This allows the fit of more flexible models representing the association between the outcome and some continuous predictors.



However, in practice, we hardly go beyond the degree 3. If the association

between the outcome and predictor is highly non-linear, than it is preferable to use the methods that we will discuss in the next sections.

1.2 Readings

Read the following chapters of *An introduction to statistical learning*:

- 7.1 Polynomial Regression

1.3 Practice session

Task 1 - Fit a cubic model

The dataset **triceps** is available in the `MultiKink` package.

The data contains the measurement of the triceps skin fold of 892 females (variable *triceps*) and we want to model its association with **age**.

First, we will load the data

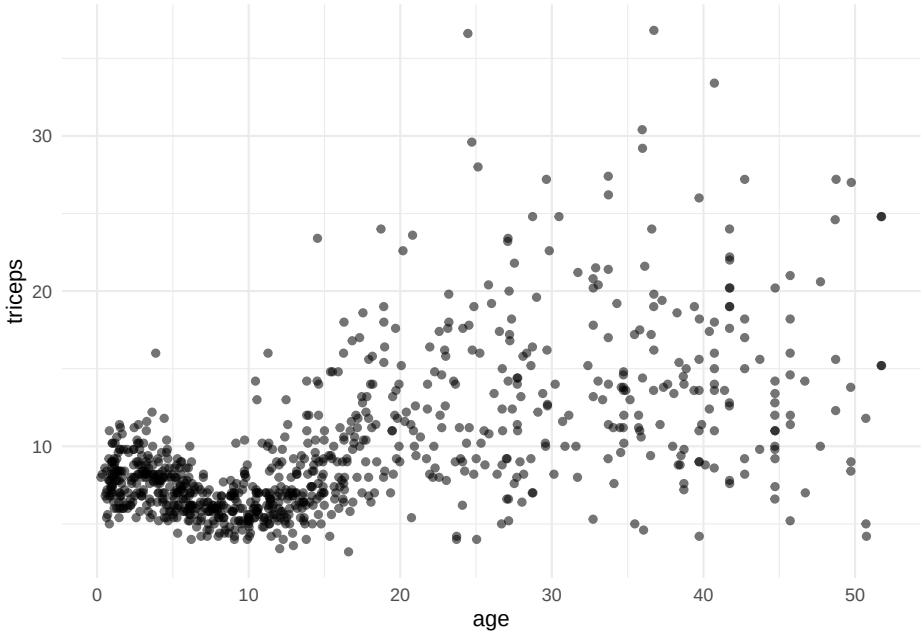
```
#libraries that we will need
#install.packages("MultiKink")
library(MultiKink) #for the data
library(ggplot2)   #for the plots
```

```
## Warning: package 'ggplot2' was built under R version 4.3.3
```

```
set.seed(1974)      #fix the random generator seed

data("triceps")   #load the dataset triceps
                  #notice that the variable of interest
                  #it is also called triceps. Don't get
                  #confused!
```

And plot the scatter for **triceps** and **age**

```
 #simple scatter
  #we can store the scatter in an object
  #to use it later
 tri.age.plot <- ggplot(triceps, aes(x=age, y=triceps)) +
                 geom_point(alpha=0.55, color="black") +
                 theme_minimal()
 tri.age.plot
```



To fit a cubic model we can write all the terms using the `I()` function to evaluate x^2 and x^3, otherwise R will not use the quadratic and cubic terms:

```
model.cubic <- lm(triceps~age + I(age^2) + I(age^3),
                  data=triceps)
summary(model.cubic)
```

```
##
## Call:
## lm(formula = triceps ~ age + I(age^2) + I(age^3), data = triceps)
##
## Residuals:
##      Min       1Q   Median       3Q      Max
## -11.5832  -1.9284  -0.5415   1.3283  24.4440
##
## Coefficients:
##               Estimate Std. Error t value Pr(>|t|)
## (Intercept)  8.004e+00  3.831e-01  20.893  < 2e-16 ***
## age         -3.157e-01  7.721e-02  -4.089 4.73e-05 ***
## I(age^2)     3.101e-02  3.964e-03   7.824 1.45e-14 ***
## I(age^3)    -4.566e-04  5.612e-05  -8.135 1.38e-15 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 3.868 on 888 degrees of freedom
## Multiple R-squared:  0.3836, Adjusted R-squared:  0.3815
```

```
## F-statistic: 184.2 on 3 and 888 DF,  p-value: < 2.2e-16
```

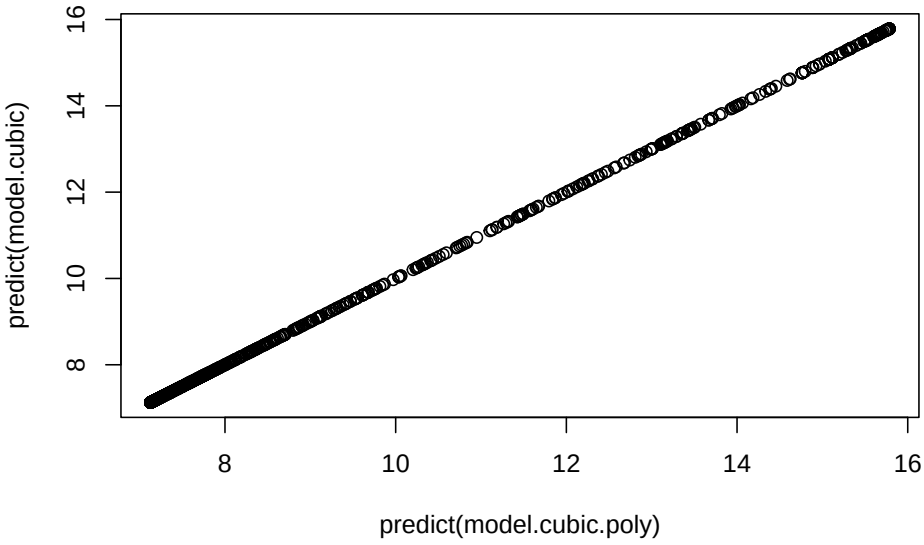
Another option is to use the `poly()` function. Note, however, that the this function fits a **linear transformation** of the terms x, x^2, x^3. This is for numerical stability given that those three terms will be highly correlated. Thus, the regression coefficients will not be the same but the model is just a reparameterisation of the one above and its predictions are exactly the same.

```
model.cubic.poly <- lm(triceps~poly(age,3),
                       data=triceps)
summary(model.cubic.poly)
```

```
##
## Call:
## lm(formula = triceps ~ poly(age, 3), data = triceps)
##
## Residuals:
##      Min       1Q   Median       3Q      Max
## -11.5832  -1.9284  -0.5415   1.3283  24.4440
##
## Coefficients:
##               Estimate Std. Error t value Pr(>|t|)
## (Intercept)     9.7024     0.1295  74.911  < 2e-16 ***
## poly(age, 3)1  85.2594     3.8682  22.041  < 2e-16 ***
## poly(age, 3)2  -3.1638     3.8682  -0.818    0.414
## poly(age, 3)3 -31.4683     3.8682  -8.135 1.38e-15 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 3.868 on 888 degrees of freedom
## Multiple R-squared:  0.3836, Adjusted R-squared:  0.3815
## F-statistic: 184.2 on 3 and 888 DF,  p-value: < 2.2e-16
```

If you look at the predictions of both model, the results are exactly the same

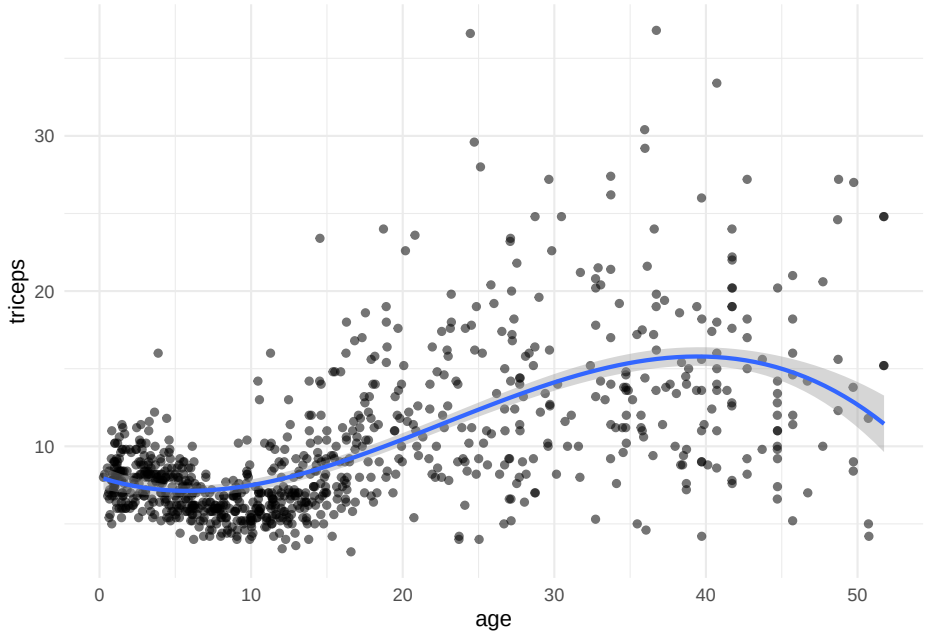
```
plot(predict(model.cubic.poly), predict(model.cubic))
```



We can also fit and plot the cubic model using `ggplot`. We will use the initial scatter plot an add the fitted curve

```
#plots
tri.age.plot +
  stat_smooth(method = "lm",
              formula = y~poly(x,3,raw=T), size = 1)
```

```
## Warning: Using `size` aesthetic for lines was deprecated in ggplot2 3.4.0.
## i Please use `linewidth` instead.
## This warning is displayed once every 8 hours.
## Call `lifecycle::last_lifecycle_warnings()` to see where this warning was
## generated.
```

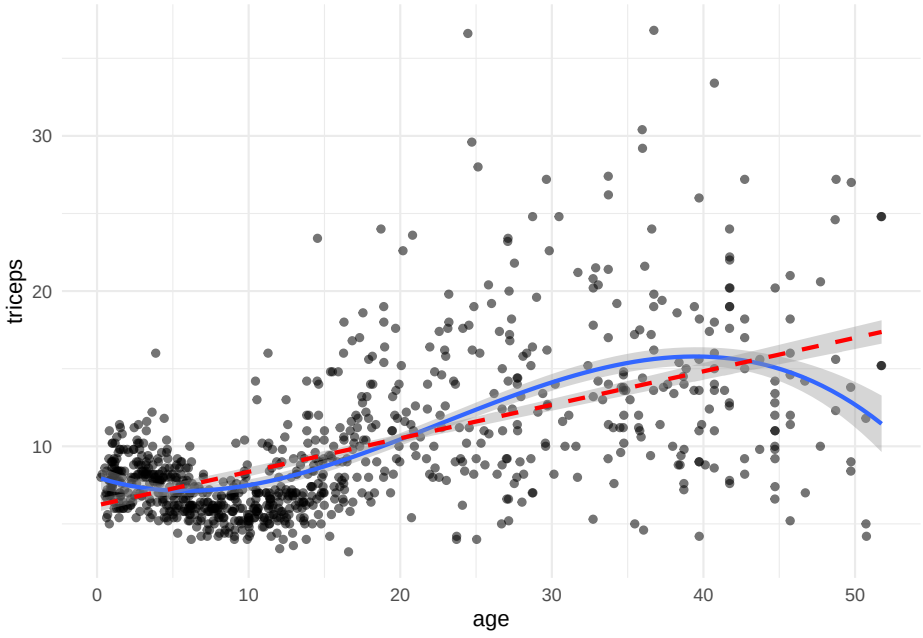


TRY IT YOURSELF:

1) Add a linear fit to the plot above

See the solution code

```
tri.age.plot +
  stat_smooth(method = "lm",
              formula = y~poly(x,3,raw=T), size = 1) +
    stat_smooth(method = "lm",
              formula = y~poly(x,1,raw=T), lty = 2, col = "red" , size = 1)
```



Task 2 - Mean Squared Error for the quadratic model

We will use the same dataset and the same variables as in TASK 1 but now we want to compute the cross-validated MSE for the quadratic model

There are multiple ways of doing this. We will take advantage of the easy implementation of cross-validation in the `caret` package. We will do 10-fold cross-validations and repeat it 10 times:

```
library(caret)
```

```
## Loading required package: lattice
```

```
## Registered S3 method overwritten by 'future':
##   method               from
##   all.equal.connection parallelly
```

```
set.seed(1234)

#repeated CV for the MSE
trC.lm <- trainControl(method = "repeatedcv",
                       number = 10,          #10-fold cross-validation
                       repeats = 10)         #10 times

#function to fit a polynomial model of degree x
pol.model <- train(triceps ~ poly(age,2),
                       data = triceps,
```

```
                        method = "lm",
                        trControl = trC.lm)

 #this is the root mean squared error
  pol.model$results[2]
```

```
##       RMSE
## 1 3.988524
```

TRY IT YOURSELF:

1) Calculate the MSE (or the root mean squared error) for the model using a degree 4 polynomial, through cross-validation

See the solution code

```
set.seed(1001)

#repeated CV for the MSE
trC.lm <- trainControl(method = "repeatedcv",
                        number = 10,          #10-fold cross-validation
                        repeats = 10)         #10 times

#function to fit a polynomial model of degree x
pol.model <- train(triceps ~ poly(age,4),
                        data = triceps,
                        method = "lm",
                        trControl = trC.lm)

 #this is the root mean squared error
  pol.model$results[2]
```

```
##       RMSE
## 1 3.782918
```

2) Calculate the MSE (or the root mean squared error) for the models using polynomials from degree 1 (linear) up to 10

See the solution code

```
set.seed(1001)
   #repeated CV for the MSE
  trC.lm <- trainControl(method = "repeatedcv",
                          number = 10,
                          repeats = 10)

  #function to fit a polynomial model of degree x
  my.pol.f <- function(x) {
    xx<-poly(triceps$age, x, raw=T)                    #design matrix with age,
```

```
                                                    #age^2, ..., age^10
  new.data  <- cbind(triceps=triceps$triceps, xx)   #dataset with the added
                                                    #poly terms
  pol.model <- train(triceps ~ .,                   #the . uses all the
                     data = new.data,               #predictors
                     method = "lm",
                     trControl = trC.lm)

  RMSE.cv = pol.model$results[2]
}

  #RMSE
t(sapply(1:10, my.pol.f))                       #applies the function
                                                #to poly degrees 1 to 10
```

1.4 Exercises

Solve the following exercise:

1) The dataset SA_heart.csv contains on coronary heart disease status (variable *chd*) and several risk factors including the cumulative tobacco consumption *tobacco*.

a) Fit a logistic model for *chd* using the predictor *tobacco* (as a linear effect) and compute its AIC

b) Plot the fitted curve in a)

c) Fit a logistic model for *chd* allowing a cubic effect of the predictor *tobacco* and compute its AIC.

d) Plot the fitted curve in c)

e) Compute the cross-validated ROC of the models a) and c) (use the `caret` package)

See the solution code for e)

```
library(caret)
set.seed(2001)
SA_heart <- read.csv("https://www.dropbox.com/s/cwkw3p91zyizcqz/SA_heart.csv?dl=1")

# caret will give an error for factors coded as 0 and 1
# because it uses the factors names to create
# names of internal variables. This way it is better
#to use an outcome variable with strings as the factor names
SA_heart$chd.f <- ifelse(SA_heart$chd ==1,
                         "chd",
```

```
                          "nochd")

#sets the control for 10-fold cross-validation, 10 times
# the classProbs = TRUE and summaryFunction = twoClassSummary
# store the information to compute the area under the ROC
trC.lm <- trainControl(method = "repeatedcv",
                       number = 10,
                       repeats = 10,
                       classProbs = TRUE,                  #necessary for
                       summaryFunction = twoClassSummary) #the AUC ROC

 #linear effect
 roc.l <- train(form = chd.f  ~ tobacco,
                data = SA_heart,
                method = "glm",
                family = "binomial",
                trControl = trC.lm,
                metric = "ROC")
 #cubic effect
 roc.c <- train(form = chd.f  ~ poly(tobacco,3),
                data = SA_heart,
                method = "glm",
                family = "binomial",
                trControl = trC.lm,
                metric = "ROC")

 roc.l
 roc.c
```

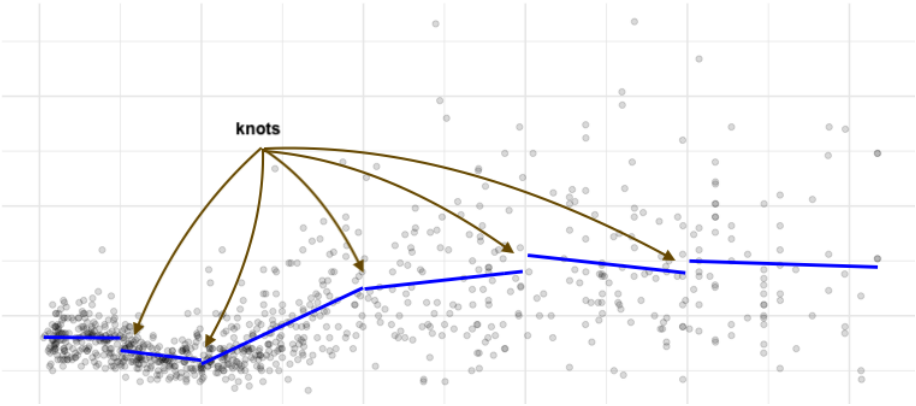
f) Which model would you prefer?

Chapter 2

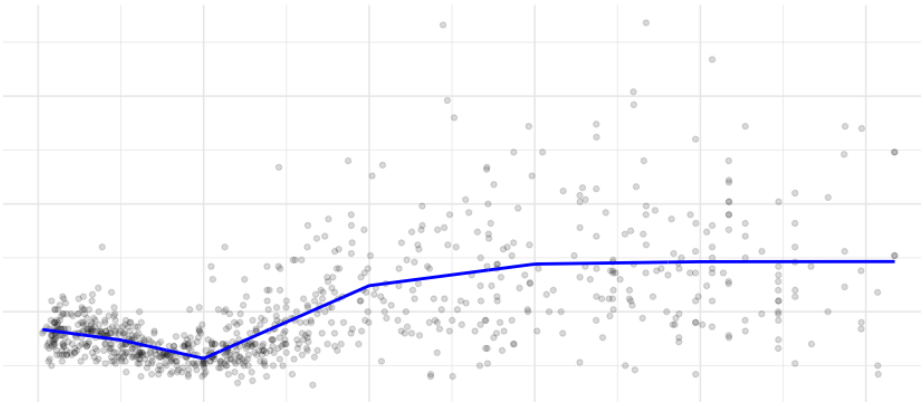
Piecewise Regression and Splines

2.1 Introduction

An alternative to fitting all data points with a single polynomial curve, is to fit segments to different parts of the data, with breakpoints (knots) at predetermined places.



We can further require continuity, meaning that the segments have to be connected



Again, the knots need to be specified and the regression equation becomes:

$$y = \beta_0 + \beta_1 x + \beta_2 (x - k_1)_+ + \beta_3 (x - k_2)_+ + ... + \beta_6 (x - k_p)_+ + \varepsilon$$

where

$$(x - k)_+ = \begin{cases} 0, & \text{if } x < k \\ x - k, & \text{if } x \geq k \end{cases}$$

The approach above may be extended to use polynomial segments. For example, using cubic segments, the model would become

$$y = \beta_0 + \beta_1 x + \beta_2 x^2 + \beta_3 x^3 + \beta_4 (x - k_1)_+ + \beta_5 (x - k_1)_+^2 + \beta_6 (x - k_1)_+^3 + ... + \varepsilon$$

The above model will be a smooth curve within the intervals bounded by the knots, but the "connection" between the segments will not be smooth. To force smoothness over the entire fitted curve we can restrict the the model above to only include the cubic components:

$$y = \beta_0 + \beta_1 x + \beta_2 x^2 + \beta_3 x^3 + \beta_4 (x - k_1)_+^3 + \beta_5 (x - k_2)_+^3 + \beta_6 (x - k_3)_+^3 + ... + \varepsilon$$

This will guarantee that the fitted curve is differentiable, with no sharp changes in the direction. This is called a cubic spline.

An improvement of the fitting of splines in the boundary of the data is achieved by using linear fitting before the first knot and after the last one.

2.2 Readings

Read the following chapters of *An introduction to statistical learning*:

- 7.2 Step Functions
- 7.3 Basis Functions
- 7.4 Regression Splines

2.3 Practice session

Task 1 - Fit a piecewise linear regression

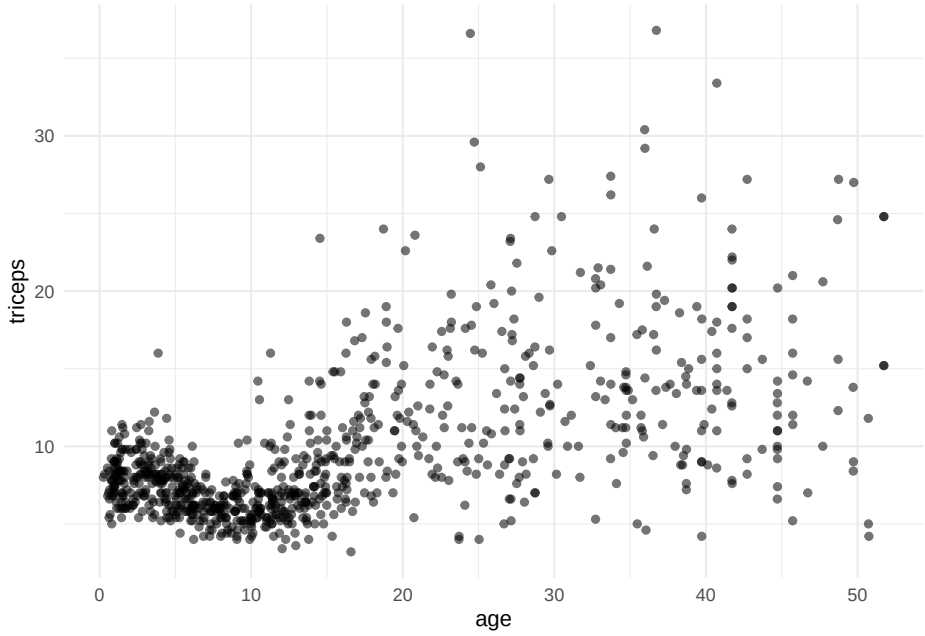
We will continue the example using the dataset **triceps** available in the `MultiKink` package. The data contains the measurement of the triceps skin fold of 892 females (variable *triceps*) and we want to model its association with **age**, using piecewise linear regression with knots at 5,10,20,30 and 40.

First, we will load the data

```
#libraries that we will need
#install.packages("MultiKink")
library(MultiKink) #for the data
library(ggplot2)   #for the plots
set.seed(1974)     #fix the random generator seed

data("triceps")    #load the dataset triceps
                   #notice that the variable of interest
                   #it is also called tricets. Don't get
                   #confused!
```

And plot the scatter for **triceps** and **age**

```
 #simple scatter
  #we can store the scatter in an object
  #to use it later
 tri.age.plot <- ggplot(triceps, aes(x=age, y=triceps)) +
                 geom_point(alpha=0.55, color="black") +
                 theme_minimal()
 tri.age.plot
```



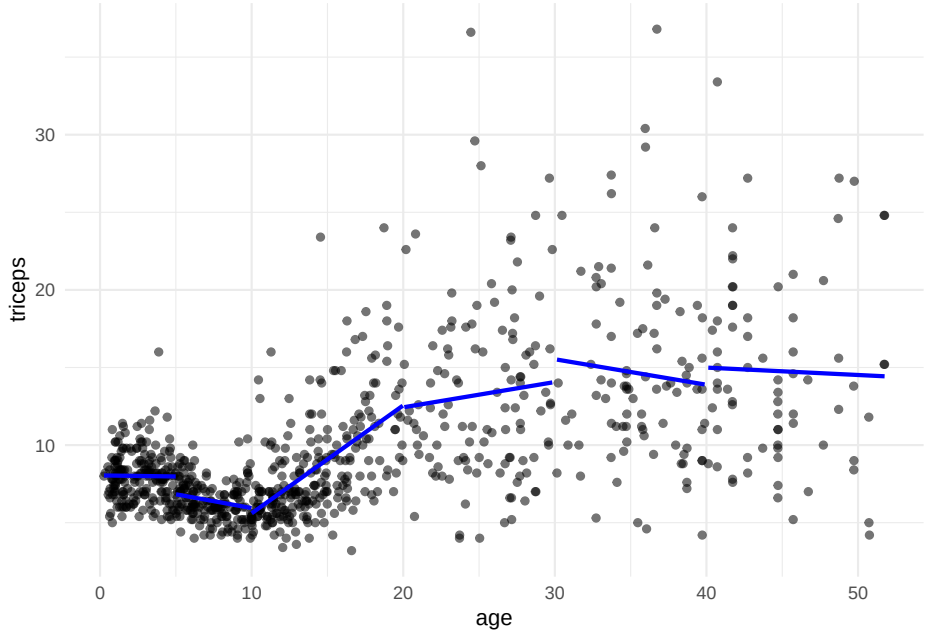
We will fit linear models within the intervals defined by the knots. The `predict()` will give us the fitted lines.

```
###Piecewise regression

pred1 <- predict(lm(triceps~age,
                    data = triceps[triceps$age<5,]))
pred2 <- predict(lm(triceps~age,
                    data = triceps[triceps$age >=5 & triceps$age<10,]))
pred3 <- predict(lm(triceps~age,
                    data = triceps[triceps$age>=10 & triceps$age<20,]))
pred4 <- predict(lm(triceps~age,
                    data = triceps[triceps$age>=20 & triceps$age<30,]))
pred5 <- predict(lm(triceps~age,
                    data = triceps[triceps$age>=30 & triceps$age<40,]))
pred6 <- predict(lm(triceps~age,
                    data = triceps[triceps$age>=40,]))
```

And we can now add the segments to the scatter plot

```
tri.age.plot +
  geom_line(data=triceps[triceps$age<5,],
            aes(y = pred1, x=age), size = 1, col="blue") +
  geom_line(data=triceps[triceps$age >=5 & triceps$age<10,],
            aes(y = pred2, x=age), size = 1, col="blue") +
  geom_line(data=triceps[triceps$age>=10 & triceps$age<20,],
```

```
            aes(y = pred3, x=age), size = 1, col="blue") +
  geom_line(data=triceps[triceps$age>=20 & triceps$age<30,],
            aes(y = pred4, x=age), size = 1, col="blue") +
  geom_line(data=triceps[triceps$age>=30 & triceps$age<40,],
            aes(y = pred5, x=age), size = 1, col="blue") +
  geom_line(data=triceps[triceps$age>=40,],
            aes(y = pred6, x=age), size = 1, col="blue")
```



We can restrict the segments to be connected, i.e., to fit a continuous line. The model is

$$y = \beta_0 + \beta_1 x + \beta_2 (x - k_1)_+ + \beta_3 (x - k_2)_+ + ... + \beta_6 (x - k_p)_+ + \varepsilon$$

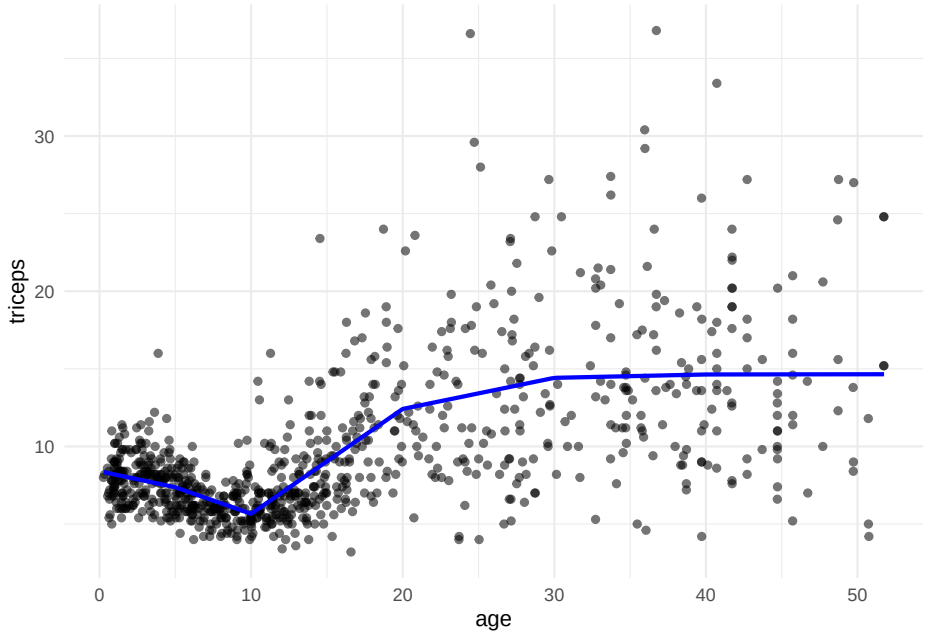
where

$$(x-k)_+ = \begin{cases} 0, & \text{if } x < k \\ x - k, & \text{if } x \geq k \end{cases}$$

So, we need to add the terms $(x - k)$ when $x \geq k$. We will do this by adding $I((age - k) * (age >= k))$ terms to the linear model. Note that $(age >= k)$ is a logical statement that will be 0 $(FALSE)$ of 1 $(TRUE)$ and I() evaluates that all expression.

```
pred7 <- predict(lm(triceps~ age + I((age-5)*(age>=5)) +
                                   I((age-10)*(age >= 10)) +
                                   I((age-20)*(age >= 20)) +
```

```
                                    I((age-30)*(age >= 30)) +
                                   I((age-40)*(age >= 40)),
                    data = triceps))

tri.age.plot +
  geom_line(data=triceps,
            aes(y = pred7, x=age), size = 1, col="blue")
```

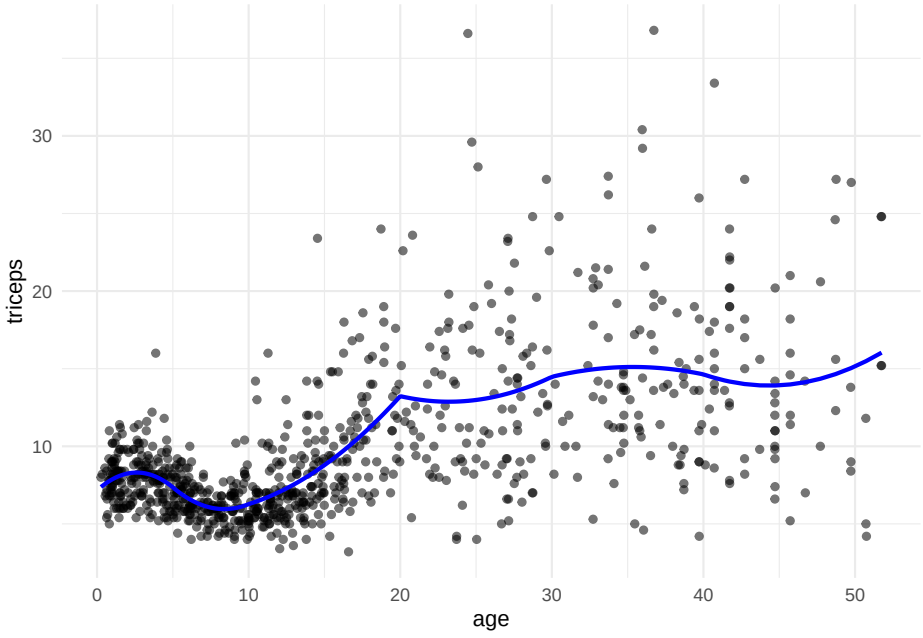


TRY IT YOURSELF:

1) Using the same knots as above, fit a quadratic piecewise regression

See the solution code

```
pred.quad <- predict(lm(triceps~ age + I(age^2) +
                    I((age-5)*(age>=5)) + I((age-5)^2*(age>=5)) +
                    I((age-10)*(age >= 10)) + I((age-10)^2*(age>=10)) +
                    I((age-20)*(age >= 20)) + I((age-20)^2*(age>=20)) +
                    I((age-30)*(age >= 30)) + I((age-30)^2*(age>=30)) +
                    I((age-40)*(age >= 40)) + I((age-40)^2*(age>=40)),
                    data = triceps))

tri.age.plot +
  geom_line(data=triceps,
            aes(y = pred.quad, x=age), size = 1, col="blue")
```



Task 2 - Fit a natural cubic spline

We will the same dataset **triceps** as in TASK 1 to fit a natural cubic spline for the association of **age** and **triceps**.

The function `bs()` in the `splines` package generates the B-spline basis matrix for a polynomial spline, and the function `ns()` in the same library generates the B-spline basis matrix matrix for a natural cubic spline (restriction that the fitted curve linear at the extremes). We will compare both.

```
library(splines)
library(MultiKink) #for the data
library(ggplot2)   #for the plots
set.seed(1974)     #fix the random generator seed

data("triceps")    #load the dataset triceps
                   #notice that the variable of interest
                   #it is also called triceps. Don't get
                   #confused!

#linear model with the natural cubic splines function
cub.splines.bs <- lm(triceps ~ bs(age, knots = c(5,10,20,30,40)),
                     data=triceps)
summary(cub.splines.bs)
```

```
##
```

```
## Call:
## lm(formula = triceps ~ bs(age, knots = c(5, 10, 20, 30, 40)),
##     data = triceps)
##
## Residuals:
##      Min       1Q   Median       3Q      Max
## -11.5234  -1.6912  -0.2917   1.1356  23.0922
##
## Coefficients:
##                                       Estimate Std. Error t value Pr(>|t|)
## (Intercept)                             6.9598     0.9729   7.154 1.77e-12 ***
## bs(age, knots = c(5, 10, 20, 30, 40))1  2.5367     1.7154   1.479   0.1396
## bs(age, knots = c(5, 10, 20, 30, 40))2 -0.3032     0.9629  -0.315   0.7529
## bs(age, knots = c(5, 10, 20, 30, 40))3 -1.9092     1.2993  -1.469   0.1421
## bs(age, knots = c(5, 10, 20, 30, 40))4  7.4056     1.2179   6.081 1.78e-09 ***
## bs(age, knots = c(5, 10, 20, 30, 40))5  6.1050     1.4043   4.347 1.54e-05 ***
## bs(age, knots = c(5, 10, 20, 30, 40))6 10.1770     1.5427   6.597 7.23e-11 ***
## bs(age, knots = c(5, 10, 20, 30, 40))7  3.9428     1.9082   2.066   0.0391 *
## bs(age, knots = c(5, 10, 20, 30, 40))8 10.1473     1.7545   5.784 1.01e-08 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 3.743 on 883 degrees of freedom
## Multiple R-squared:  0.4261, Adjusted R-squared:  0.4209
## F-statistic: 81.94 on 8 and 883 DF,  p-value: < 2.2e-16
```

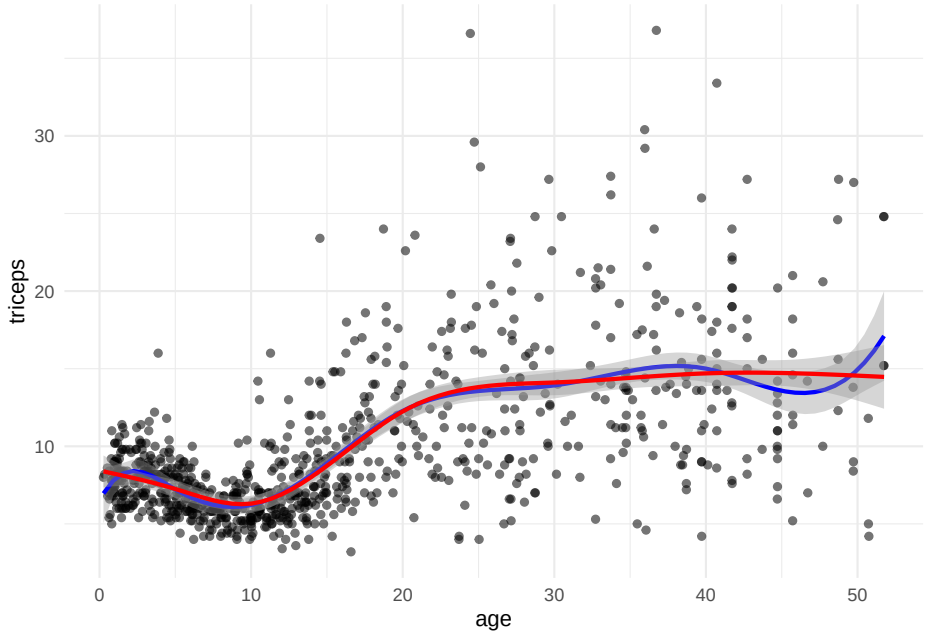
```
cub.splines.ns <- lm(triceps ~ ns(age, knots = c(5,10,20,30,40)),
                     data=triceps)

summary(cub.splines.ns)
```

```
##
## Call:
## lm(formula = triceps ~ ns(age, knots = c(5, 10, 20, 30, 40)),
##     data = triceps)
##
## Residuals:
##      Min       1Q   Median       3Q      Max
## -10.4875  -1.6873  -0.3665   1.1146  22.8643
##
## Coefficients:
##                                       Estimate Std. Error t value Pr(>|t|)
## (Intercept)                             8.3811     0.5219  16.059  < 2e-16 ***
## ns(age, knots = c(5, 10, 20, 30, 40))1 -3.5592     0.6712  -5.303 1.44e-07 ***
## ns(age, knots = c(5, 10, 20, 30, 40))2  5.7803     1.0379   5.569 3.39e-08 ***
## ns(age, knots = c(5, 10, 20, 30, 40))3  5.5118     0.9416   5.853 6.78e-09 ***
```

```
## ns(age, knots = c(5, 10, 20, 30, 40))4   6.9070     0.9050   7.632 5.99e-14 ***
## ns(age, knots = c(5, 10, 20, 30, 40))5   5.4136     1.3783   3.928 9.24e-05 ***
## ns(age, knots = c(5, 10, 20, 30, 40))6   6.6460     1.0829   6.137 1.27e-09 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 3.759 on 885 degrees of freedom
## Multiple R-squared:  0.4199, Adjusted R-squared:  0.416
## F-statistic: 106.8 on 6 and 885 DF,  p-value: < 2.2e-16
```

Notice that are less regression parameters for the natural spline due to the linearity restriction. We can see this in the plot. To plot we could either get predictions from the fitted models or fit the models in the `ggplot` function directly:

```
#simple scatter
tri.age.plot <- ggplot(triceps, aes(x=age, y=triceps)) +
                geom_point(alpha=0.55, color="black") +
                theme_minimal()

 tri.age.plot +
   stat_smooth(method = "lm",
               formula = y~bs(x,knots = c(5,10,20,30,40)),
               lty = 1, col = "blue") +
   stat_smooth(method = "lm",
               formula = y~ns(x,knots = c(5,10,20,30,40)),
               lty = 1, col = "red")
```

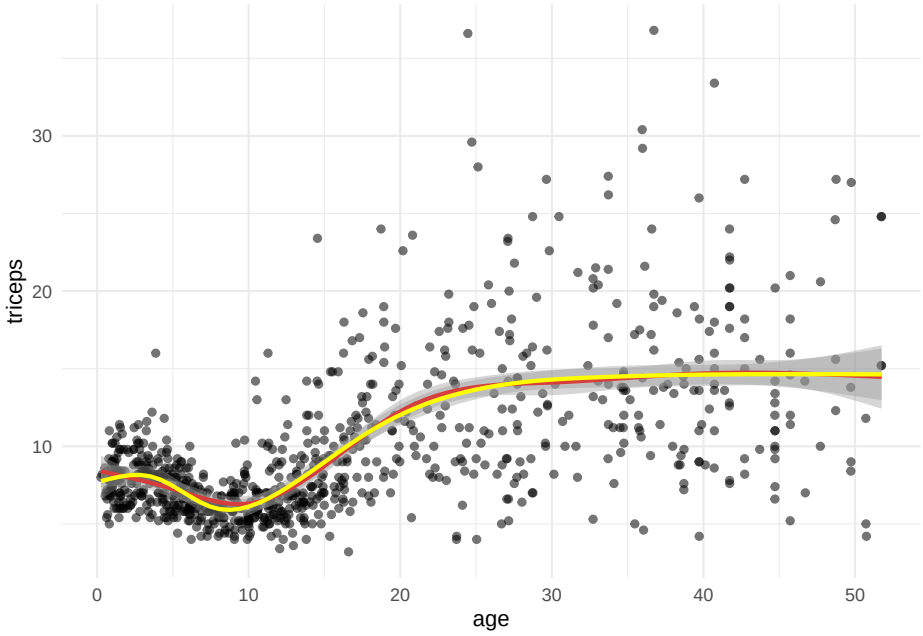


TRY IT YOURSELF:

1) Fit a natural spline with 6 degrees of freedom and compare it with the natural spline using `knots = c(5,10,20,30,40)`. What is the difference?

See the solution code

```
tri.age.plot +
   stat_smooth(method = "lm",
               formula = y~ns(x,knots = c(5,10,20,30,40)),
               lty = 1, col = "red") +
   stat_smooth(method = "lm",
               formula = y~ns(x,df=6),
               lty = 1, col = "yellow")
```




```
#df=6 also chooses 5 knots but the knots
#are based on the quantiles of the data
#in this case the knots are at values:
attr(ns(triceps$age, df=6), "knots")
```

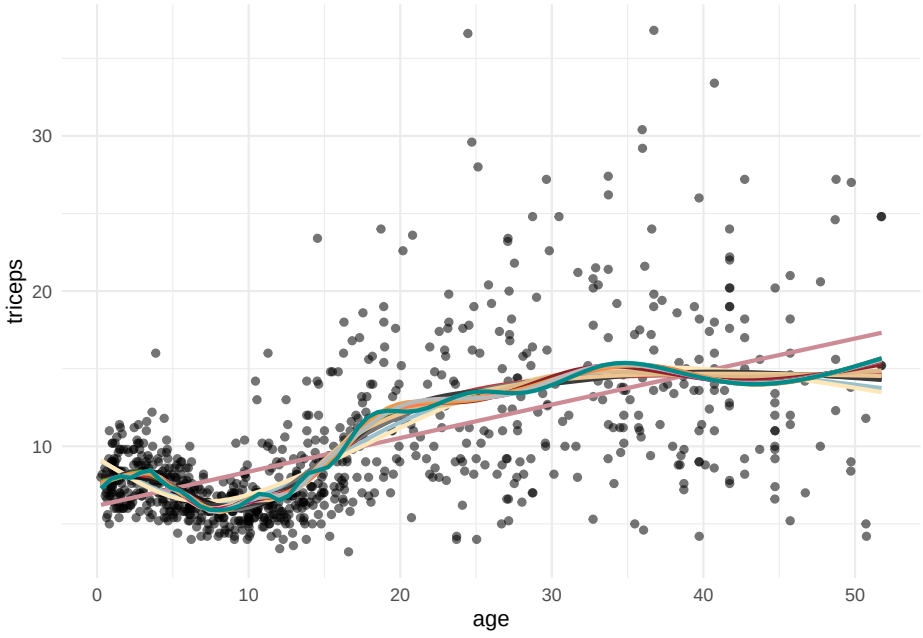
```
## [1]  3.76  7.67 12.21 18.12 32.55
```

2) Calculate the MSE (or the root mean squared error) for the models using natural cubic splines with *df* from 2 (linear model) up to 20. You can use the library `caret`.

See the solution code

```
library(caret)
set.seed(1001)

#repeated CV for the MSE
trC.lm <- trainControl(method = "repeatedcv",
                       number = 10,
                       repeats = 10)

#function to fit a spline with x degrees of freedom
  my.spline.f <- function(x) {
    #need to construct the model formula
    spline.formula <- as.formula(paste("triceps ~ ns(age, df=",x, ")" ))
    pol.model <- train(spline.formula,
                       data = triceps,
```

```
                        method = "lm",
                        trControl = trC.lm)

    RMSE.cv = pol.model$results[2]                #extracts the RMSE
  }

  #RMSE
  t(sapply(2:20, my.spline.f))                    #Computes the RMSE for splines
                                                  #with df degrees 2 to 20
  ##############################################################################
  #if you want to plot the curves,
  #it is tricky to get ggplot to work
  #within a loop. This is a solutions:
  col.ran <- sample(colours(), 20)                  #colours for the lines
  my.plot<- tri.age.plot                            #scatterplot
  for (i in 2:20){
      #builds the stat_smooth with df=i
      loop_input <-  paste("stat_smooth(method = \"lm\",
                           formula = y~ns(x,df=",i,"),
                           lty = 1, col =\"",col.ran[i],"\",
                           se = FALSE)", sep="")

      #updates the scatter plot with
      #the new spline
      my.plot <- my.plot + eval(parse(text=loop_input))
  }

 my.plot
```



2.4 Exercises

Solve the following exercise:

1) The dataset SA_heart.csv contains on coronary heart disease status (variable *chd*) and several risk factors including the cumulative tobacco consumption *tobacco*.

a) Fit a logistic model for *chd* using the predictor *tobacco* (as a linear effect) and compute its AIC

b) Plot the fitted curve in a)

c) Fit a logistic model for *chd* with a natural cubic spline for the predictor *tobacco*, with *df* 5 and 10. Compute the AIC of the two models.

d) Plot the fitted curves in c)

e) Compute the cross-validated ROC of the models a) and c) (use the `caret` package)

See the solution code for e)

```
library(caret)
library(splines)
set.seed(2001)
SA_heart <- read.csv("https://www.dropbox.com/s/cwkw3p91zyizcqz/SA_heart.csv?dl=1")
```

```
# caret will give an error for factors coded as 0 and 1
# because it uses the factors names to create
# names of internal variables. This way it is better
#to use an outcome variable with strings as the factor names
SA_heart$chd.f <- ifelse(SA_heart$chd ==1,
                         "chd",
                         "nochd")

#sets the control for 10-fold cross-validation, 10 times
# the classProbs = TRUE and summaryFunction = twoClassSummary
# store the information to compute the area under the ROC
trC.lm <- trainControl(method = "repeatedcv",
                       number = 10,
                       repeats = 10,
                       classProbs = TRUE,                  #necessary for
                       summaryFunction = twoClassSummary) #the AUC ROC

 #linear effect
 roc.l <- train(form = chd.f  ~ tobacco,
                data = SA_heart,
                method = "glm",
                family = "binomial",
                trControl = trC.lm,
                metric = "ROC")

 roc.5 <- train(form = chd.f  ~ ns(tobacco,df=5),
                data = SA_heart,
                method = "glm",
                family = "binomial",
                trControl = trC.lm,
                metric = "ROC")
 #cubic effect
 roc.10 <- train(form = chd.f  ~ ns(tobacco,df=10),
                data = SA_heart,
                method = "glm",
                family = "binomial",
                trControl = trC.lm,
                metric = "ROC")

 roc.l
 roc.5
 roc.10
```

f) Which model would you prefer?

2) The dataset fev.csv contains the measurements of forced expiratory volume

(FEV) tests, evaluating the pulmonary capacity in 654 children and young adults.

a) Plot the association between *fev* and *height* and fit a linear model for *fev* using *height* as a predictor

b) Fit a model for *fev* with a cubic b-spline for the predictor *height*, with *df* 5 and 10.

c) Fit a model for *fev* with a natural cubin spline for the predictor *height*, with *df* 5 and 10.

d) Plot the fitted curves for models a), b) and c)

e) compare the cross-validated MSE of the models a), b) and c)

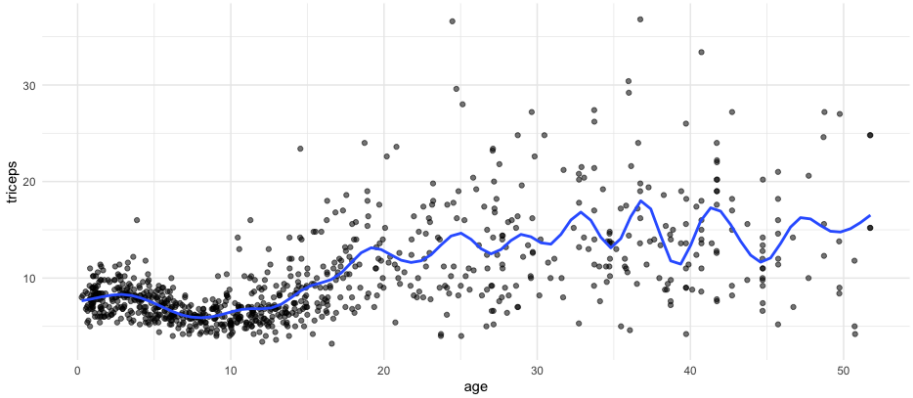
Chapter 3

Smoothing splines

3.1 Introduction

In the previous section we learn how to fit regression splines by specifying the knots and a set of basis function. It should be easy to see that a higher number of knots will lead to a lower MSE because we will be overfitting the features of the curve.

The model below is fitted with natural splines with 25 knots.



Clearly the curve seems to be overfitting the data.

We will use a similar idea to the one used in regularisation (module 4). We select many knots but penalise for the roughness of the fitting.

Remember that the 1st derivative indicates the slope of the curve and the second derivative is the speed of change of the slope. Thus, the second derivative of a curve is associated with the roughness of the curve.

We will then use the second derivative as the penalisation term in the residual sum of squares.

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

The result is a **smoothing spline**. For smoothing splines, the number of knots is not as important given that the penalisation term will handle the roughness.

The animation below, shows the fitting of smoothing splines, with amounts of penalisation (*lambda*), and automatic choice of number of knots given by the `smooth.spline` function in R. The cross validated MSE is also shown.

A large λ results in a smooth curve (a straight line in the limit) and a smaller λ leads to a more rough curve. The optimal λ can be chosen by cross-validation.

The smoothing splines can be incorporated in the generalised linear models framework which is usually referred as **generalised additive models** (GAM). Rather than a linear effect of a predictor, we can have a smoothing spline modeling the association of the predictor with the outcome:

$g\left(E(y|\mathbf{x})\right) = \beta_0 + f_1(x_1) + f_2(x_2) + ... + f_k(x_k)$

Notice that $f_p(x_p)$ can be a linear function $\beta_p x_p$ and if all f's are linear function, the model above is a GLM.

3.2 Readings

Read the following chapters of *An introduction to statistical learning*:

- 7.5 Smoothing Splines
- 7.7 Generalised Additive Model

3.3 Practice session

Task 1 - Fit a smoothing spline

We will continue the example using the dataset **triceps**available in the `MultiKink` package. The data contains the measurement of the triceps skin fold of 892 females (variable *triceps*) and we want to model its association with **age**, using smoothing cubic splines.

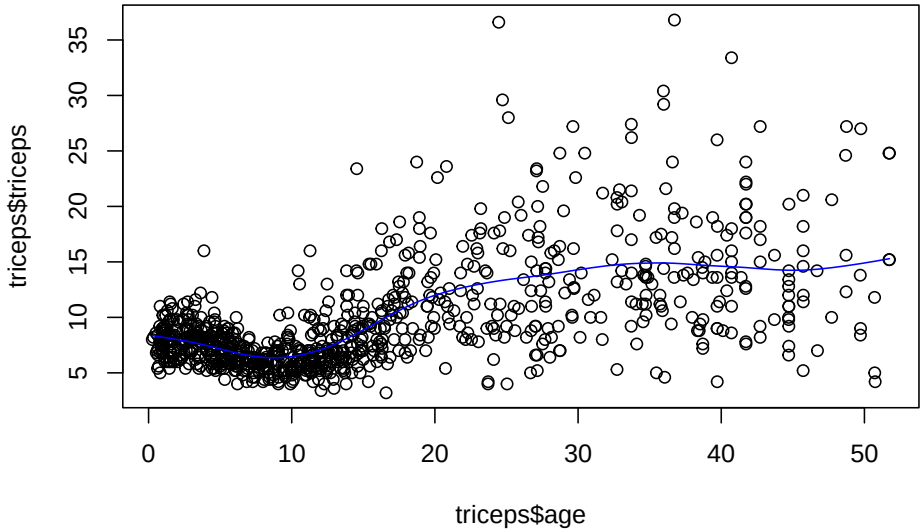
The function `smooth.spline()` fits smoothing cubic splines. We can provide the penalisation and/or number of knots, df, or just use the defaults.

```
library(splines)
library(MultiKink) #for the data
library(ggplot2)   #for the plots
set.seed(1974)     #fix the random generator seed

data("triceps")   #load the dataset triceps
```

```
                    #notice that the variable of interest
                    #it is also called triceps. Don't get
                    #confused!

#smooth spline with automatic number of knots chosen
#and penalisation chosen by leave-one-out CV (this is the
#option cv=T, otherwise generalized' cross-validation is used)
sspline <- smooth.spline(triceps$age,
                         triceps$triceps,
                         cv=TRUE)
```

```
## Warning in smooth.spline(triceps$age, triceps$triceps, cv = TRUE):
## cross-validation with non-unique 'x' values seems doubtful
```

```
plot(triceps$age, triceps$triceps)
lines(sspline, col="blue")
```



*Note: The generalised cross-validation (default), in this case, selects a very low value for λ which results in under smoothing. It is not clear why that is the case.

Let's get the triceps predicted value for the ages of 10 and 30

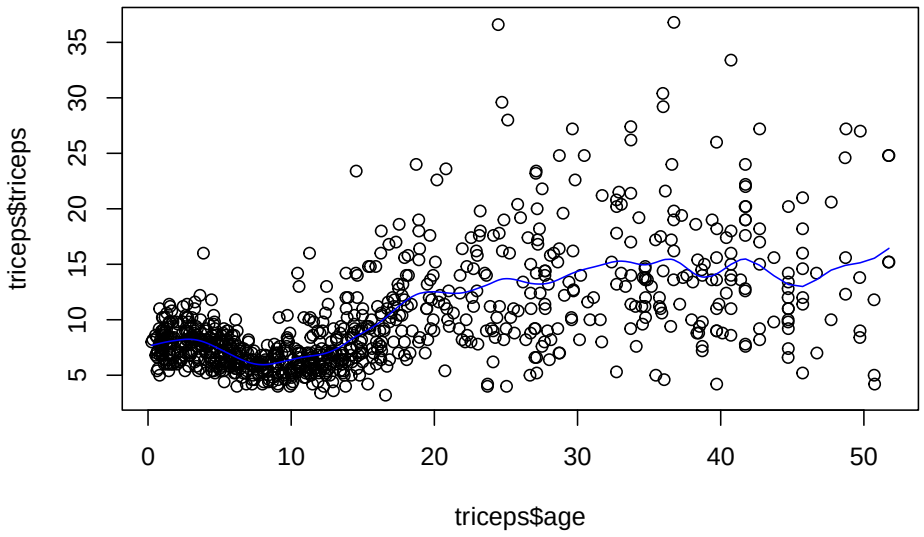
```
predict(sspline, x=c(10,30))
```

```
## $x
## [1] 10 30
##
## $y
## [1]  6.470573 14.290032
```

The predicted values are 6.4705727 and 14.2900322, respectively.

As mentioned before, we could have chosen the amount of penalisation and this would lead to a different smoothing. For example, for

```
sspline <- smooth.spline(triceps$age,
                         triceps$triceps, lambda=.0001)
plot(triceps$age, triceps$triceps)
lines(sspline, col="blue")
```

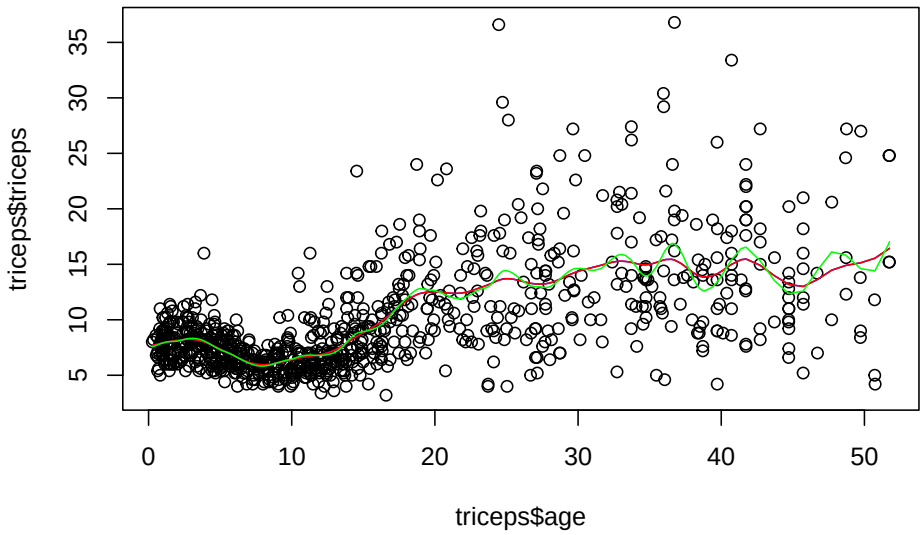


TRY IT YOURSELF:

1) Fit smoothing splines with *df=19* and *df=30* and compare them with the one above. Comment on the results.

See the solution code

```
sspline <- smooth.spline(triceps$age,
                         triceps$triceps, lambda=.0001)
sspline19 <- smooth.spline(triceps$age,
                         triceps$triceps, df=19)
sspline30 <- smooth.spline(triceps$age,
                         triceps$triceps, df=30)
plot(triceps$age, triceps$triceps)
lines(sspline, col="blue")
lines(sspline19, col="red")
lines(sspline30, col="green")
```



The spline with df=19 corresponds to having a lambda=.0001, so the result is the same. If you run `sspline` and `sspline` you can see that the parameters are similar.

The spline with df=30 will be more flexible (more rough).

Task 2 - Fit an additive model

The dataset bmd.csv contains 169 measurement of bone mineral density (variable *bmd*) in men and women of different age. We want to fit a model for *bmd* using *age*, *sex* and *bmi* as predictors.

```
#read the data and compute BMI
bmd.data     <-
  read.csv("https://www.dropbox.com/s/c6mhgatkotuze8o/bmd.csv?dl=1",
           stringsAsFactors = TRUE)
bmd.data$bmi <- bmd.data$weight_kg / (bmd.data$height_cm/100)^2
```

We will use the function `gam()` from the `mgcv` library. Notice that there is another gam function from the gam package (from the help file: "Note that this version of gam is different from the function with the same name in the R library mgcv, which uses only smoothing splines with a focus on automatic smoothing parameter selection via GCV.")

The function `s()` within the model indicates that we want to smoothing spline for that predictor. There are several types of splines implemented in the function. We will use the option basis spline equal to cubic regression splines: `bs="cr"`. The function `s()` has similarities to the `smooth.spline()` but is implemented

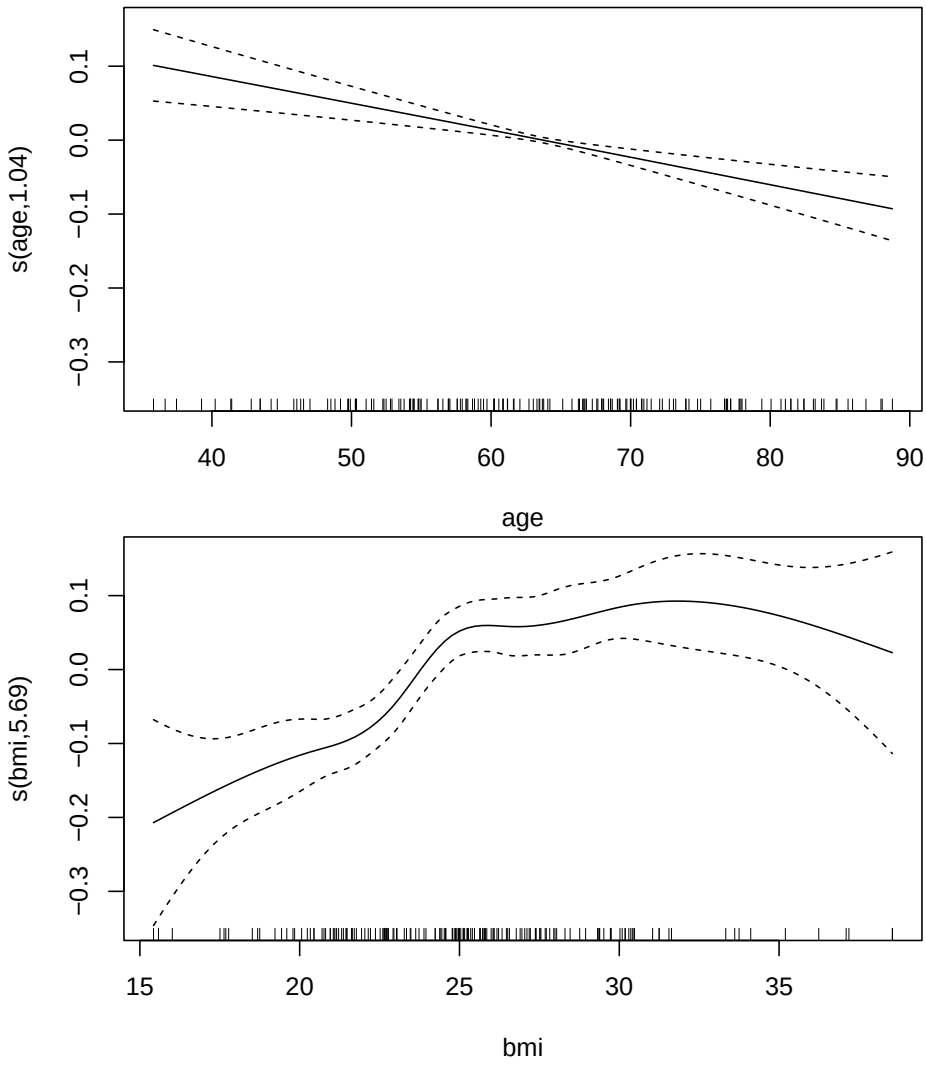
```
#libraries that we will need
library(mgcv)   #package for gam
```

```
set.seed(1974) #fix the random generator seed
bmd.gam <- gam(bmd ~ s(age, bs="cr")+ s(bmi, bs="cr") + sex, data=bmd.data)

summary(bmd.gam)
```

```
##
## Family: gaussian
## Link function: identity
##
## Formula:
## bmd ~ s(age, bs = "cr") + s(bmi, bs = "cr") + sex
##
## Parametric coefficients:
##             Estimate Std. Error t value Pr(>|t|)
## (Intercept)  0.74193    0.01456   50.97  < 2e-16 ***
## sexM         0.08092    0.02064    3.92 0.000131 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Approximate significance of smooth terms:
##          edf Ref.df     F  p-value
## s(age) 1.035  1.070 17.65 3.02e-05 ***
## s(bmi) 5.687  6.611 10.09  < 2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## R-sq.(adj) =  0.381   Deviance explained = 40.9%
## GCV = 0.018099  Scale est. = 0.017165  n = 169
```

The effective number of degrees of freedom for age is approximately 1, which suggests that the effect of age is linear. We can plot the fitted splines for each predictor. Notice, however, that the plot is not in the original scale.

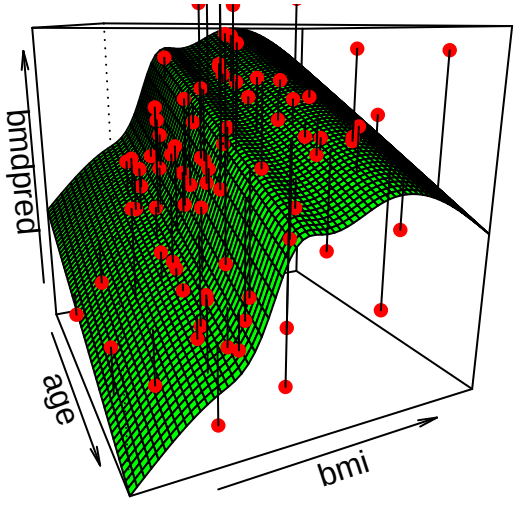
```
plot(bmd.gam)
```



We can also plot the surface fitted for each sex. For example, for women,

```
# Let's create a grid to be used in  persp()
steps <- 60
age <- with(bmd.data,
            seq(min(age), max(age),
                length = steps) )

bmi <-  with(bmd.data,
             seq(min(bmi), max(bmi),
                 length = steps) )
```

```
newdat <- expand.grid(age = age,                #grid
                      bmi = bmi,
                      sex="F")

bmdpred <- matrix(predict(bmd.gam, newdat),
                  steps, steps)                 #predictions

# Now plot it
p <- persp(age, bmi,
           bmdpred,
           theta = 65,                      #angle of the perspective
           col = "green")

# To add the points, you need the same 3d transformation
obs <- with(bmd.data[bmd.data$sex=="F", ],
            trans3d(age, bmi, bmd, p))
pred <- with(bmd.data[bmd.data$sex=="F", ],
             trans3d(age, bmi, fitted(bmd.gam)[bmd.data$sex=="F"], p))


# Add segments to show the points and where they are in 3d
points(obs, col = "red", pch = 16)
segments(obs$x, obs$y, pred$x, pred$y)
```



TRY IT YOURSELF:

1) How would the plot for sex=M compare to the surface above?

See the solution code

The surface would have exactly the same shape but would be separated by the

0.08 units (estimate for the variable sex in the model)

2) Fit a linear model with the same variables and compare the AIC of the linear model with the previous GAM model

See the solution code

```
bmd.glm <- glm(bmd ~ age+bmi+ sex, data=bmd.data)
summary(bmd.glm)
AIC(bmd.glm, bmd.gam)
```

The GAM model seems to fit better because it has a lower AIC.

3.4 Exercises

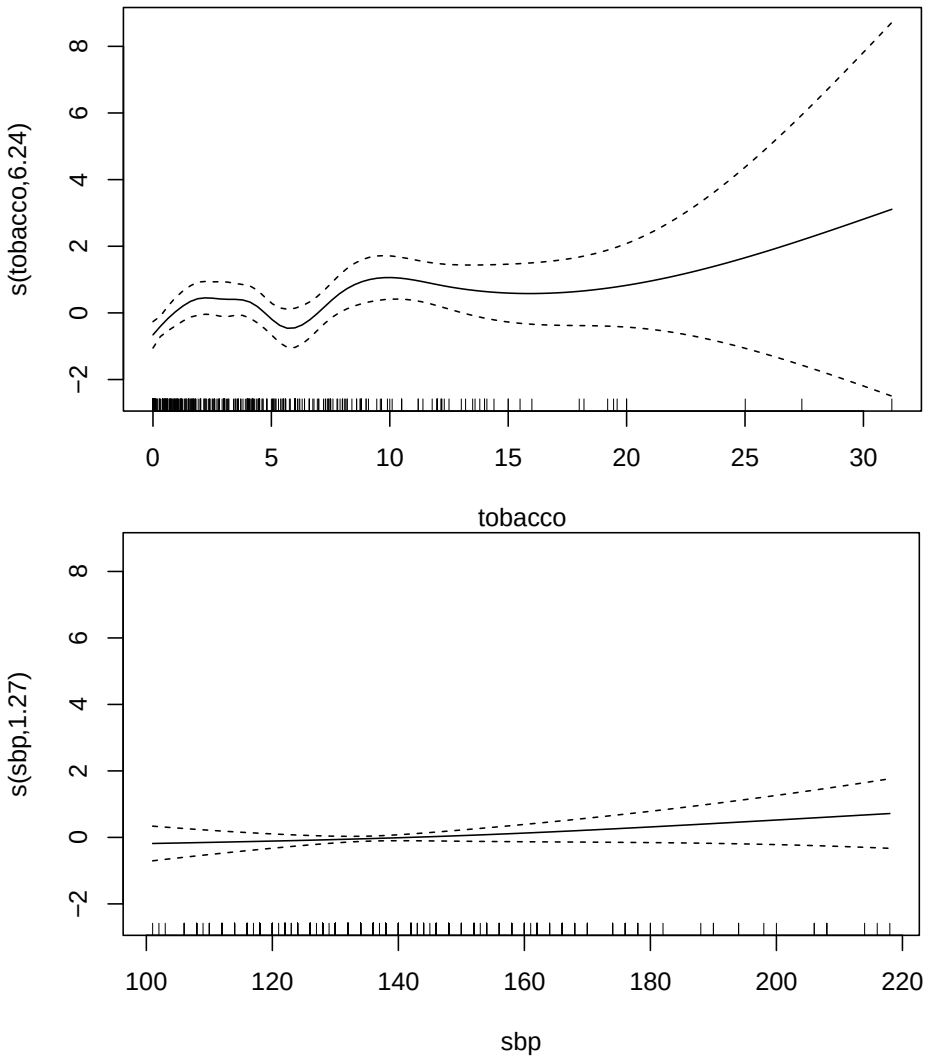
Solve the following exercise:

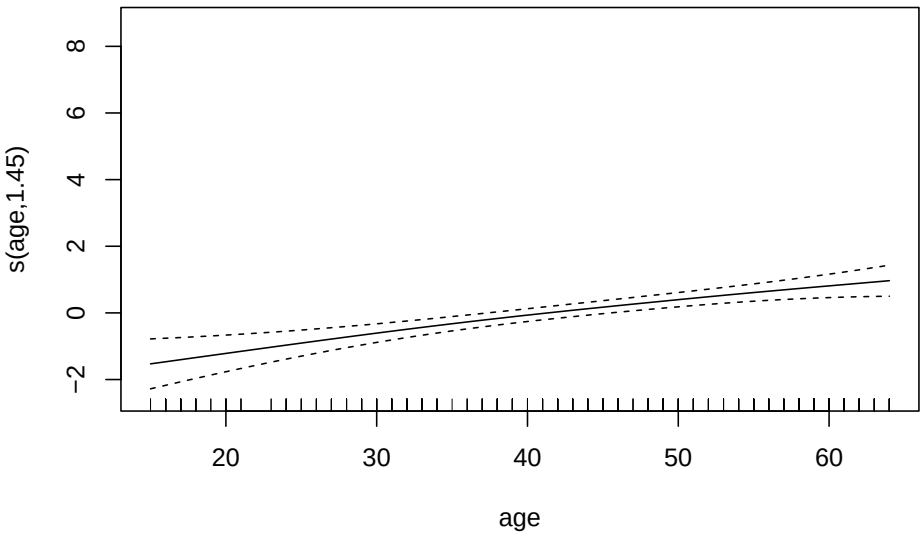
1) The dataset SA_heart.csv contains on coronary heart disease status (variable *chd*) and several risk factors including the cumulative tobacco consumption *tobacco*, systolic *sbp*, and *age*

a) Fit a GAM logistic model for *chd* with splines for the predictor *tobacco*,*sbp* and *age*

See the solution code for a)

```
library(caret)
library(splines)
set.seed(2001)
SA_heart <- read.csv("https://www.dropbox.com/s/cwkw3p91zyizcqz/SA_heart.csv?dl=1")

chd.gam <- gam(chd ~ s(tobacco, bs="cr") +
                 s(sbp, bs="cr") +
                 s(age, bs="cr"),
               family = binomial,
               data=SA_heart)
summary(chd.gam)
plot(chd.gam)
```


s(tobacco,6.24)
−2
0
2
4
6
8
0
5
10
15
20
25
30
tobacco
s(sbp,1.27)
−2
0
2
4
6
8
100
120
140
160
180
200
220
sbp



b) Find the AUC ROC for the model above.

c) Compare the AUC ROC of the GAM model with a logistic regression with linear effects for the predictors.

2) The dataset fev.csv contains the measurements of forced expiratory volume (FEV) tests, evaluating the pulmonary capacity in 654 children and young adults.

a) Plot the association between *fev* and *height* and fit a smoothing spline for *fev* using *height* as a predictor

b) Plot the association between *fev* and *age* and fit a smoothing spline for *fev* using *age* as a predictor

c) Fit a GAM model for *fev* with smoothing slipes for *height* and *age* and also add *sex* to the model.

d) Comment in the results of the fitted GAM model and plot the fitted splines for the predictors