Smoothing and Regression Splines

Ignasi Mañé, Antoni Company & Chiara Barbi

19 de marzo de 2019

Introduction

The file bikes.Washington.Rdata contains information on the bike-sharing rental service in Washington D.C., USA, corresponding to years 2011 and 2012. This file contains only one data frame, bikes, with 731 rows (one for each day of years 2011 and 2012, that was a leap year) and 9 columns:

- instant: row index, going from 1 to 731.
- yr: year (0: 2011, 1:2012).
- dayyr: day of the year (from 1 to 365 for 2011, and from 1 to 366 for 2012).
- weekday: day of the week (0 for Sunday, 1 for Monday, . . . , 6 for Saturday).
- workingday: if day is neither weekend nor holiday is 1, otherwise is 0.
- temp: temperature in Celsius.
- hum: humidity in %.
- windspeed: wind speed in miles per hour.
- cnt: count of total rental bikes. In this exam we consider this variable as continuous.

In the following chunk we will call the libraries used throughout the assignment.

```r
library(ggplot2)
library(splines)
library(tidyverse)
library(splines)
```

Nonparametric regression of cnt as a function of instant

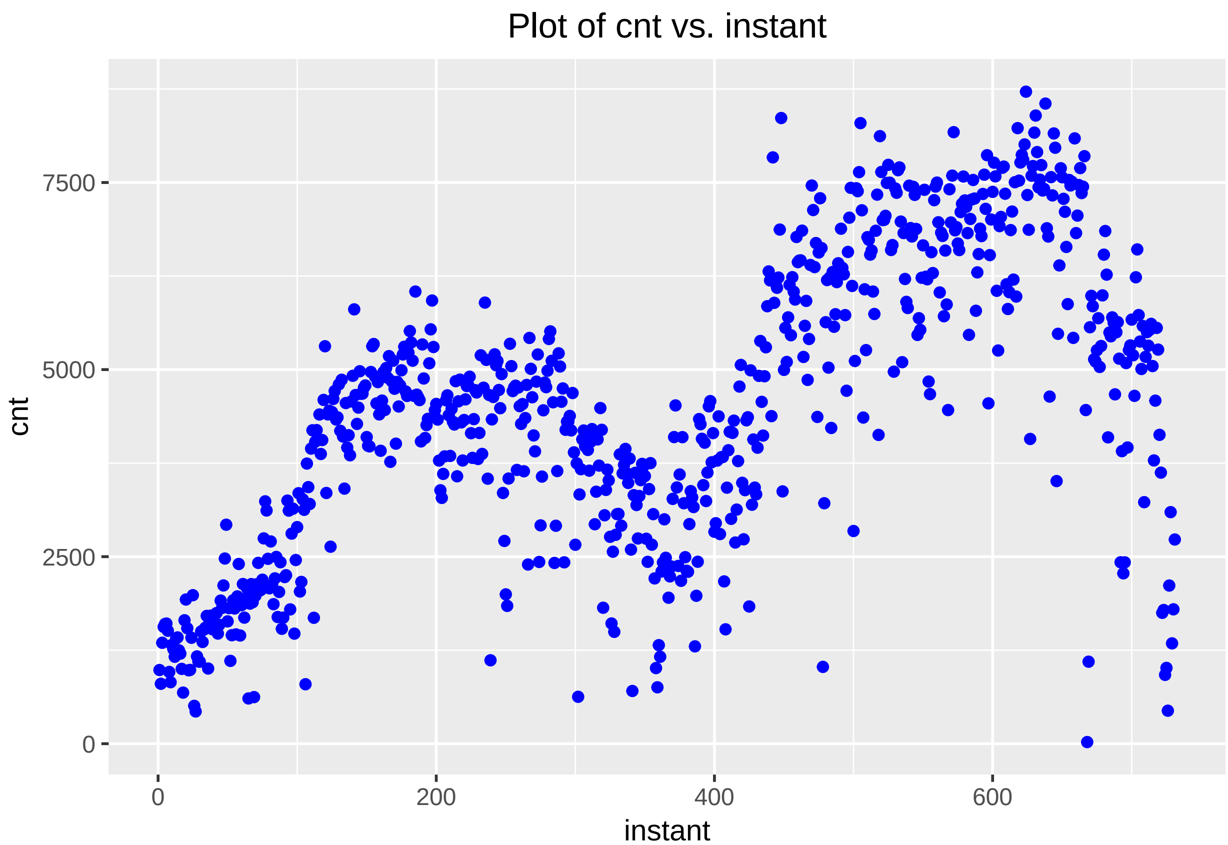
1. Consider the nonparametric regression of cnt as a function of instant. Estimate the regression function m(instant) of cnt as a function of instant using a cubic regression splines estimated with the R function smooth.splines and choosing the smoothing parameter by Generalized Cross Validation.

First, we will proceed by uploading the data to the R environment and defining two auxiliary parameters for the variables cnt and instant.

```r
load("bikes.Washington.RData")
cnt <- bikes$cnt
instant <- bikes$instant
```

Once we have uploaded the data we will proceed with a simple representation of the cnt as a function of instant to see the behavior of the parameters.

```r
ggplot(bikes)+geom_point(aes(instant,cnt), col='blue')+
  ggtitle(label="Plot of cnt vs. instant")+
  theme(plot.title = element_text(hjust = 0.5))
```



Once we have seen the data, we will proceed with the questions for this exercise.

a) Which is the value of the chosen penalty parameter λ?

We have chosen λ by applying the generalized cross-validation for the data to analyse, to obtain its value we applied the function smooth.spline() and we imposed cv=FALSE to do the generalized cross-validation.

```r
s.gcv <- smooth.spline(x = instant,y = cnt,cv=FALSE)
```

Where the optimal value for λ is:

```r
s.gcv$lambda
```

```
## [1] 1.005038e-07
```

b) Which is the corresponding equivalent number of degrees of freedom df?

The function smooth.spline() used before also gives us the value for the degrees of freedom, where *df* will be:

```r
s.gcv$df
```

```
## [1] 93.34091
```

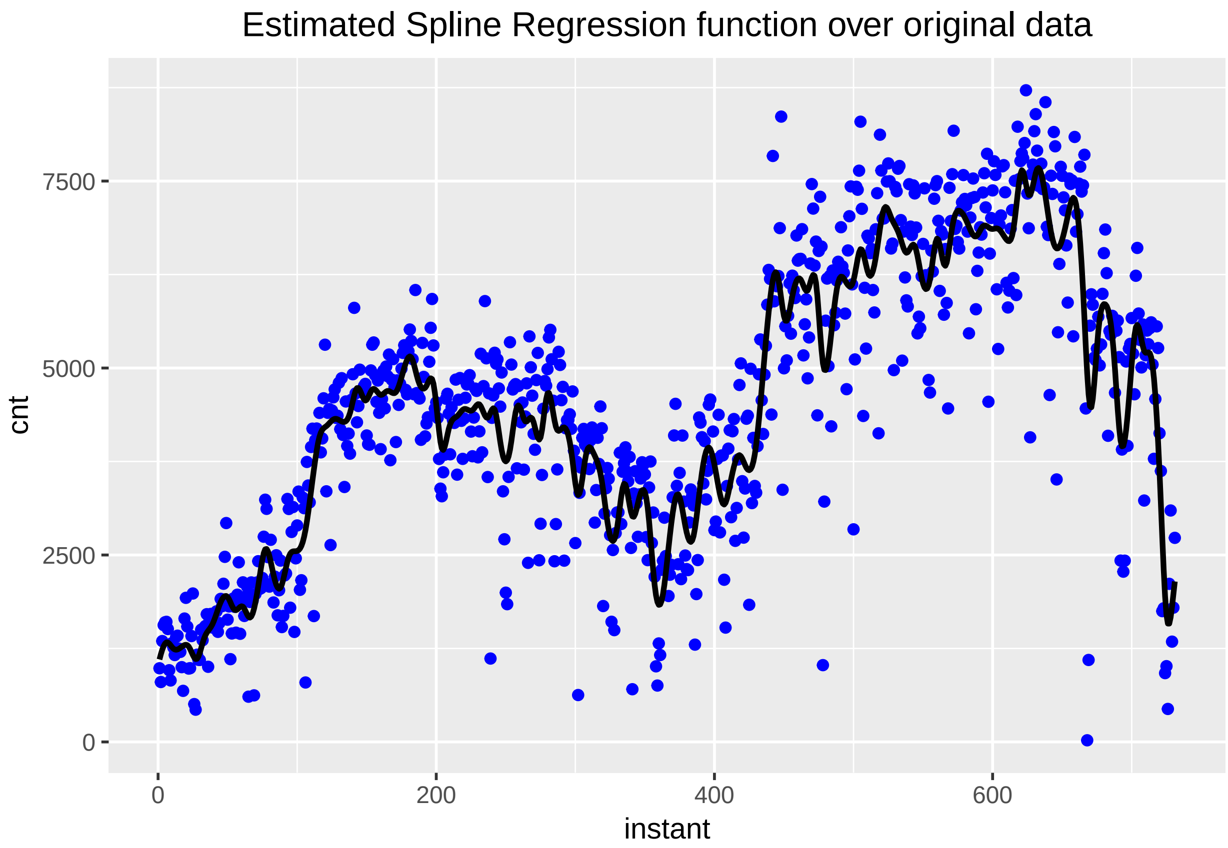
c) How many knots have been used?

```r
s.gcv$fit$nk - 2
```

```
## [1] 134
```

d) Give a graphic with the scatter plot and the estimated regression function m(instant).

```r
ggplot(bikes)+geom_point(aes(instant,cnt), col='blue')+
  ggtitle(label="Estimated Spline Regression function over original data")+
  theme(plot.title = element_text(hjust = 0.5))+
  geom_line(data = as.data.frame(s.gcv$x,s.gcv$y),
            aes(x =s.gcv$x,y =s.gcv$y),color="black",size=1)
```



e) Estimate now m(instant) by unpenalized regression splines combining the R functions bs and lm, using the knots my.knots <- quantile(instant,((1:n.knots)-.5)/n.knots) where n.knots is the previous value of df minus 4.

```
n.knots <- s.gcv$df-4
my.knots <- quantile(instant,((1:n.knots)-0.5)/n.knots)
my.knots <- my.knots[-c(1,length(my.knots))]
```

Once the data is prepared, we can proceed computing the cubic B-Spline basis used for the posterior lm() of cnt as a function of the basis.

```
basis <- bs(x=instant,knots = my.knots,intercept = TRUE,degree = 3)
```

Then, we finally proceed to compute the lm().

```
lm.bs <- lm(cnt~basis-1)
```

If we summarize the results from the lm() we can see that:

```
summary(lm.bs)
```

```
##
## Call:
## lm(formula = cnt ~ basis - 1)
```

```
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -4856.9  -360.8   112.5   502.8  2487.2
##
## Coefficients:
##         Estimate Std. Error t value Pr(>|t|)
## basis1     965.9      686.5   1.407 0.159899
## basis2    1732.7      968.8   1.789 0.074166 .
## basis3     923.5      935.2   0.988 0.323760
## basis4    1453.0      700.5   2.074 0.038448 *
## basis5     995.0      692.0   1.438 0.150982
## basis6    1544.7      689.7   2.240 0.025443 *
## basis7    2087.8      689.0   3.030 0.002542 **
## basis8    1673.5      688.8   2.430 0.015389 *
## basis9    1932.5      688.8   2.806 0.005171 **
## basis10   1365.0      688.7   1.982 0.047925 *
## basis11   3449.6      688.7   5.009 7.10e-07 ***
## basis12   1159.2      688.7   1.683 0.092857 .
## basis13   3193.8      688.7   4.637 4.28e-06 ***
## basis14   1993.5      688.7   2.894 0.003928 **
## basis15   3863.9      688.7   5.610 3.01e-08 ***
## basis16   4217.9      688.7   6.124 1.59e-09 ***
## basis17   4443.4      688.7   6.452 2.19e-10 ***
## basis18   4029.1      688.7   5.850 7.84e-09 ***
## basis19   5052.5      688.7   7.336 6.69e-13 ***
## basis20   4322.1      688.7   6.275 6.43e-10 ***
## basis21   4996.5      688.7   7.255 1.17e-12 ***
## basis22   4278.5      688.7   6.212 9.41e-10 ***
## basis23   5335.0      688.7   7.746 3.73e-14 ***
## basis24   4773.7      688.7   6.931 1.02e-11 ***
## basis25   5053.7      688.7   7.338 6.61e-13 ***
## basis26   4176.0      688.7   6.063 2.28e-09 ***
## basis27   3992.6      688.7   5.797 1.06e-08 ***
## basis28   4526.3      688.7   6.572 1.03e-10 ***
## basis29   4549.3      688.7   6.605 8.35e-11 ***
## basis30   4147.4      688.7   6.022 2.91e-09 ***
## basis31   4999.5      688.7   7.259 1.13e-12 ***
## basis32   2851.3      688.7   4.140 3.94e-05 ***
## basis33   5162.2      688.7   7.495 2.21e-13 ***
## basis34   3974.4      688.7   5.771 1.23e-08 ***
## basis35   4184.0      688.7   6.075 2.13e-09 ***
## basis36   4630.0      688.7   6.723 3.96e-11 ***
## basis37   4272.6      688.7   6.204 9.90e-10 ***
## basis38   3457.3      688.7   5.020 6.71e-07 ***
```

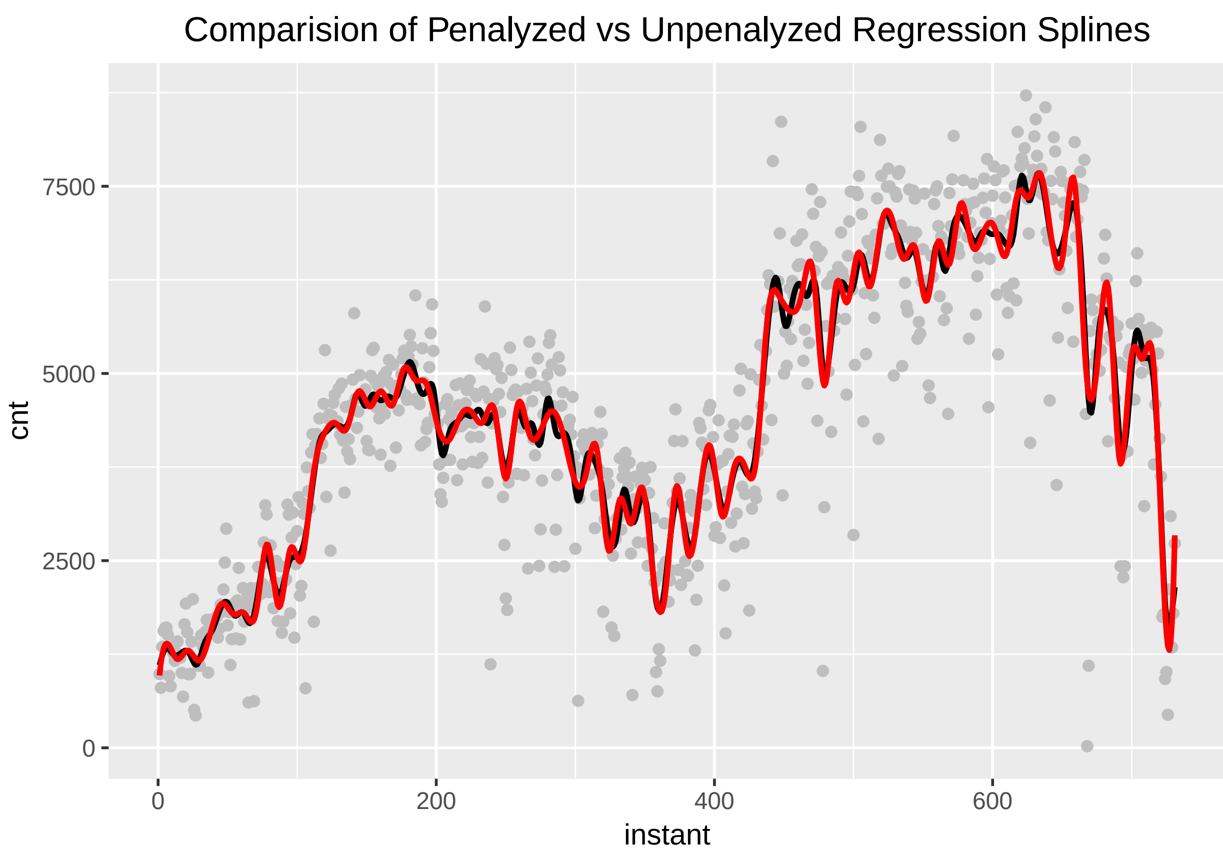
```
## basis39   3425.8       688.7   4.974 8.44e-07 ***
## basis40   4657.2       688.7   6.762 3.07e-11 ***
## basis41   1820.0       688.7   2.642 0.008431 **
## basis42   3899.1       688.7   5.661 2.27e-08 ***
## basis43   2490.0       688.7   3.615 0.000324 ***
## basis44   4104.1       688.7   5.959 4.19e-09 ***
## basis45   1926.3       688.7   2.797 0.005316 **
## basis46   1503.9       688.7   2.184 0.029356 *
## basis47   4373.9       688.7   6.351 4.06e-10 ***
## basis48   1965.0       688.7   2.853 0.004469 **
## basis49   3334.5       688.7   4.842 1.62e-06 ***
## basis50   4566.3       688.7   6.630 7.14e-11 ***
## basis51   2511.7       688.7   3.647 0.000287 ***
## basis52   3968.0       688.7   5.761 1.30e-08 ***
## basis53   3850.8       688.7   5.591 3.34e-08 ***
## basis54   3206.4       688.7   4.656 3.93e-06 ***
## basis55   6318.6       688.7   9.174  < 2e-16 ***
## basis56   6038.3       688.7   8.767  < 2e-16 ***
## basis57   5775.4       688.7   8.385 3.22e-16 ***
## basis58   5836.6       688.7   8.474  < 2e-16 ***
## basis59   7165.0       688.7  10.403  < 2e-16 ***
## basis60   3724.1       688.7   5.407 9.05e-08 ***
## basis61   6961.8       688.7  10.108  < 2e-16 ***
## basis62   5387.9       688.7   7.823 2.14e-14 ***
## basis63   7172.5       688.7  10.414  < 2e-16 ***
## basis64   5624.7       688.7   8.167 1.69e-15 ***
## basis65   7268.4       688.7  10.553  < 2e-16 ***
## basis66   7196.9       688.7  10.450  < 2e-16 ***
## basis67   6207.6       688.7   9.013  < 2e-16 ***
## basis68   7133.1       688.7  10.357  < 2e-16 ***
## basis69   5331.6       688.7   7.741 3.86e-14 ***
## basis70   7333.9       688.7  10.648  < 2e-16 ***
## basis71   5895.0       688.7   8.559  < 2e-16 ***
## basis72   7839.7       688.7  11.383  < 2e-16 ***
## basis73   6370.6       688.7   9.250  < 2e-16 ***
## basis74   6943.0       688.7  10.081  < 2e-16 ***
## basis75   7146.2       688.7  10.376  < 2e-16 ***
## basis76   6139.9       688.7   8.915  < 2e-16 ***
## basis77   7796.8       688.7  11.321  < 2e-16 ***
## basis78   7059.9       688.7  10.251  < 2e-16 ***
## basis79   8078.5       688.7  11.729  < 2e-16 ***
## basis80   6662.0       688.7   9.673  < 2e-16 ***
## basis81   6013.4       688.7   8.731  < 2e-16 ***
## basis82   8782.5       688.7  12.752  < 2e-16 ***
## basis83   4281.4       688.7   6.217 9.16e-10 ***
```

```
## basis84   4713.6      688.6   6.846 1.79e-11 ***
## basis85   7412.7      688.1  10.772  < 2e-16 ***
## basis86   2374.4      686.7   3.458 0.000581 ***
## basis87   6015.9      681.7   8.825  < 2e-16 ***
## basis88   4729.4      665.1   7.111 3.09e-12 ***
## basis89   6648.8      918.9   7.236 1.33e-12 ***
## basis90  -1205.7      957.2  -1.260 0.208288
## basis91   2838.5      650.9   4.361 1.51e-05 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 883.6 on 640 degrees of freedom
## Multiple R-squared:  0.9716, Adjusted R-squared:  0.9675
## F-statistic: 240.3 on 91 and 640 DF,  p-value: < 2.2e-16
```

The model adjusted is highly significant, and only a few bases could be discarded for having a p-value under the 0.05.

f) Give a graphic with the scatter plot and the two estimated regression functions.

```r
ggplot(bikes)+geom_point(aes(instant,cnt), col='grey')+
  ggtitle(label="Comparision of Penalyzed vs Unpenalyzed Regression Splines")+
  theme(plot.title = element_text(hjust = 0.5))+
  geom_line(data = as.data.frame(s.gcv$x,s.gcv$y),aes(x =s.gcv$x,y =s.gcv$y),color="bla
  geom_line(aes(x=instant,y=lm.bs$fitted.values),color="red",size=1)
```



Nonparametric logistic regression using splines

2. Nonparametric logistic regression using splines with a IRWLS procedure. The script IRWLS logistic regression.R includes the definition of the function logistic.IRWLS.splines performing non-parametric logistic regression using splines with a IRWLS procedure. The basic syntax is the following: logistic.IRWLS.splines(x=. . . , y=. . . , x.new=. . . , df=. . . , plts=TRUE) where the arguments are the explanatory variable x, the 0-1 response variable y, the vector x.new of new values of variable x where we want to predict the probability of y being 1 given that x is equal to x.new, the equivalent number of parameters (or model degrees of freedom) df, and the logical plts indicating if plots are desired or not. Define a new variable cnt.5000 taking the value 1 for days such that the number of total rental bikes is larger than or equal to 5000, on 0 otherwise.

a) Use the function logistic.IRWLS.splines to fit the non-parametric binary regression cnt.5000 as a function of the temperature, using df=6. In which range of temperatures is Pr(cnt >= 5000|temp) larger than 0,5?

First we have to upload the script "IRWLS_logistic_regression.R" in order to get the function logistic.IRWLS.splines()

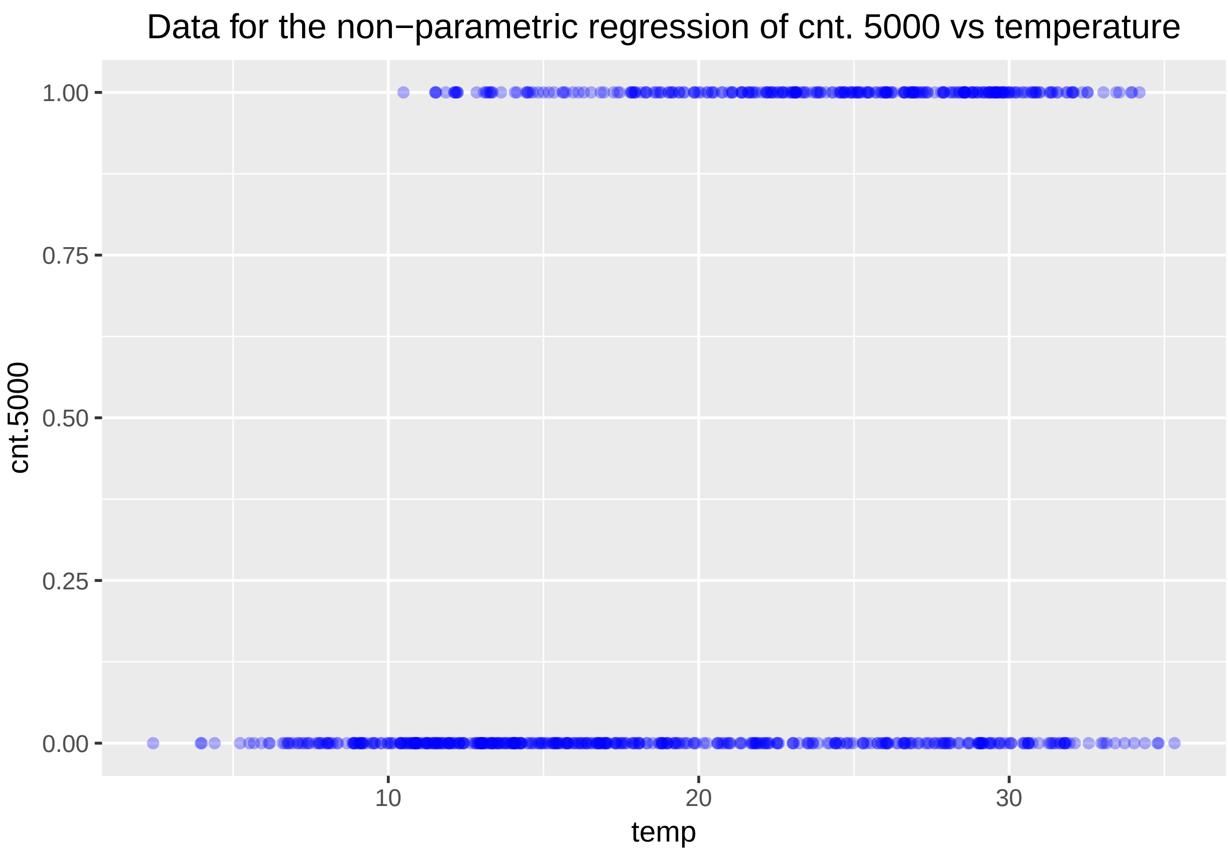
```r
source("IRWLS_logistic_regression.R")
```

Once we have the function uploaded, we also have to define the parameters for the non-parametric regression, where:

```r
temp <- bikes$temp #x axis
#and the response variable is defined as follows:
cnt.5000 = ifelse(bikes$cnt>=5000,1,0)
```

We will represent a plot in order to see the data for the non-parametric regression:

```r
ggplot(as.data.frame(temp,cnt.5000))+geom_point(aes(temp,cnt.5000), col='blue',alpha=0.
  ggtitle(label="Data for the non-parametric regression of cnt. 5000 vs temperature")+th
```



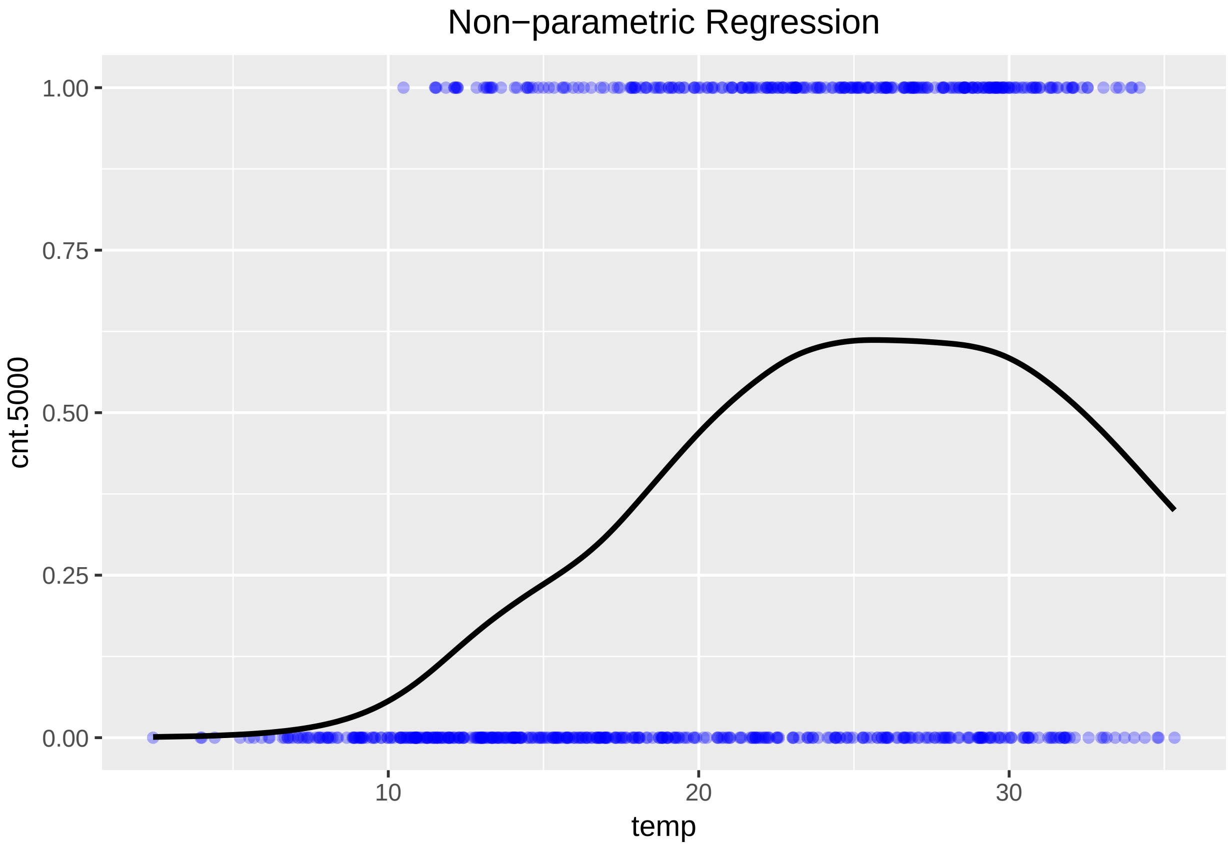
Once we have seen the behavior of the data, we can proceed with the non-parametric regression with the logistic function with df=6.

```r
log.reg <- logistic.IRWLS.splines(y=cnt.5000,x = temp, x.new = temp, df = 6, plts = F)
```

Where if we analyse the results of the regression we will see that:

```r
ggplot(as.data.frame(temp,cnt.5000))+
  geom_point(aes(temp,cnt.5000), col='blue',alpha=0.3)+
```

```
  ggtitle(label="Non-parametric Regression")+
  theme(plot.title = element_text(hjust = 0.5))+
  geom_line(data = as.data.frame(temp,log.reg$fitted.values),
            aes(temp,log.reg$predicted.values),color='black',size=1)
```



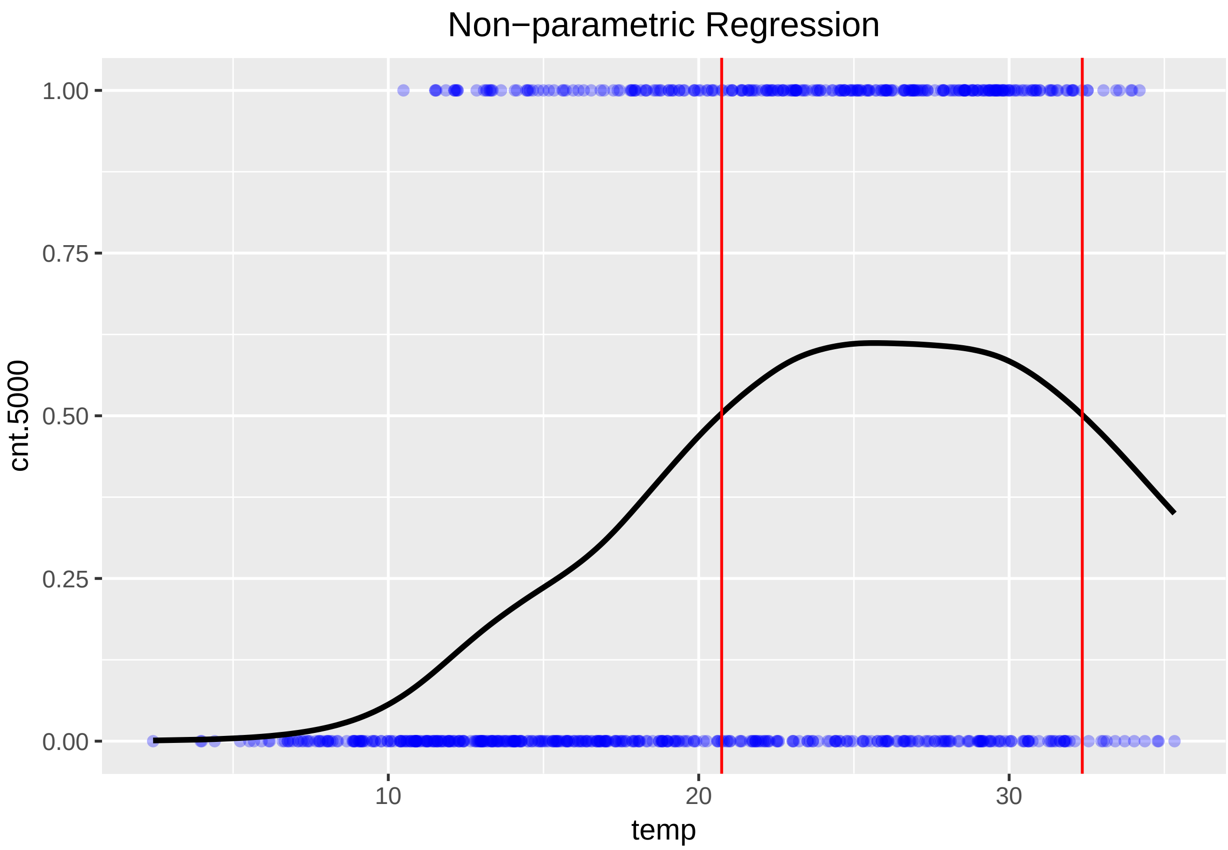
Where the range of the percentage of temperature which are larger than cnt.5000 >= 0.5 is the following:

```
range.0.5 <- range(temp[log.reg$fitted.values>=0.5])
range.0.5
```

```
## [1] 20.73915 32.35585
```

If we analyse the results over the previous plot:

```
ggplot(as.data.frame(temp,cnt.5000))+geom_point(aes(temp,cnt.5000), col='blue',alpha=0.
  ggtitle(label="Non-parametric Regression")+theme(plot.title = element_text(hjust = 0.5
  geom_line(data = as.data.frame(temp,log.reg$fitted.values),
            aes(temp,log.reg$predicted.values),color='black',size=1)+
  geom_vline(xintercept = range.0.5[1],color='red')+
  geom_vline(xintercept = range.0.5[2],color='red')
```



b) (Optional) Choose the parameter df by k-fold cross validation with k = 5 and using df.v = 3:15 as the set of possible values for df.

For this exercise we will use the same parameters as before, but, we also have to define a sequence of the degrees of freedom in order to obtain the optimal df^* by using the 5-fold cross-validation method.

```
df.s <- seq(3,15,by=1)
k <- 5
```

Once we have defined the degrees of freedom and the method of k-fold cross-validation, we will proceed to compute cross-validation in order to obtain the optimal value for the degrees of freedom.

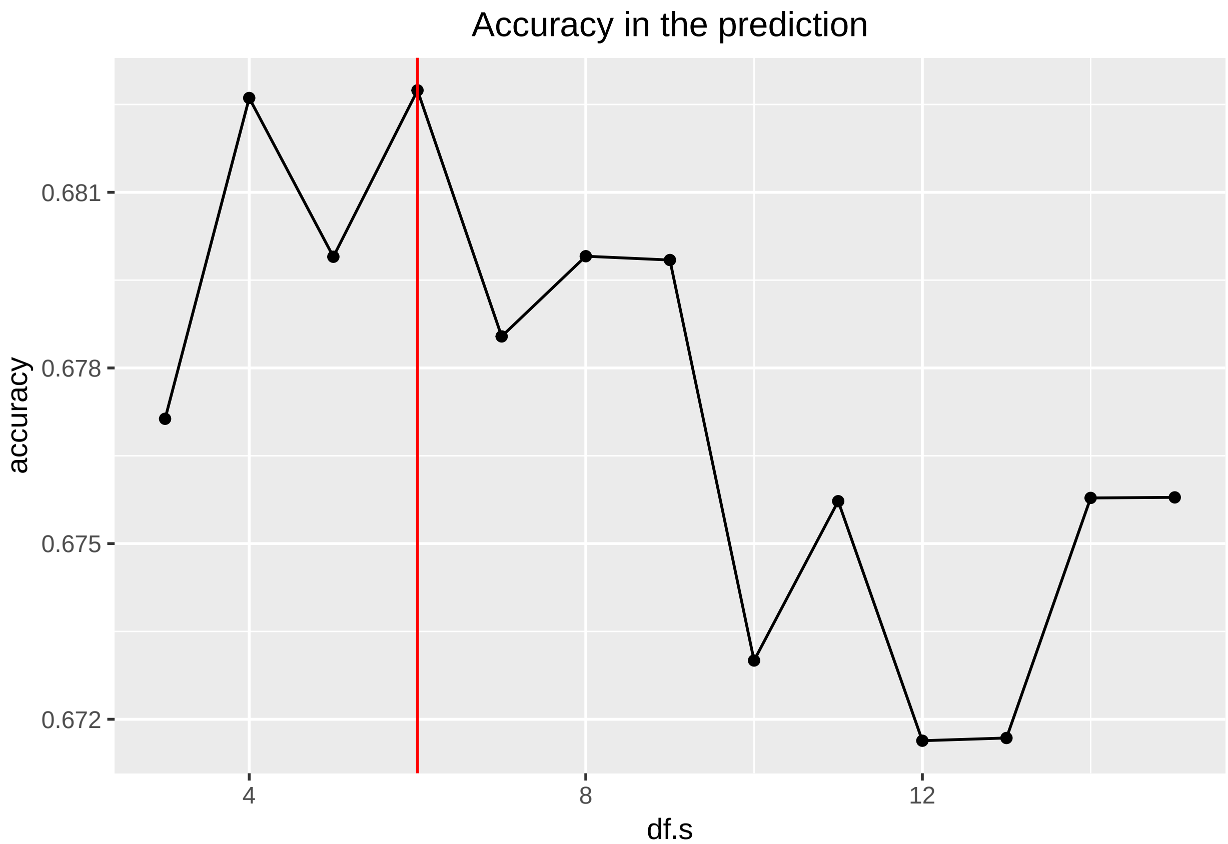
To compute the k-fold cross-validation we have implemented the following code, which mesures the accuracy in the predictions (the same as for the assignment of the delivery ROC) of the regression as a reference.

```
set.seed(1994)
accuracy <- NULL

for (i in 1:length(df.s)){

  #folds definition
```

```r
  folds <- as.numeric(sample(rep(1:k,length.out = length(temp))))
  acc.df.iteration <- NULL

  for (j in 1:k ){

    #validation data
    x.val <- temp[which(folds == j)]
    y.val <- cnt.5000[which(folds == j)]
    #training data
    x.train <- temp[which(folds != j)]
    y.train <- cnt.5000[which(folds != j)]

    #regression estimation for each df value
    log.reg.cv <- logistic.IRWLS.splines(x = x.train,y = y.train,x.new = x.val,df = df.s

    #accuracy in the regression
    acc.table <- table(y.val,ifelse(log.reg.cv$predicted.values>0.5,1,0))
    acc.df.iteration[j] <- (acc.table[1,1]+acc.table[2,2])/sum(acc.table)
  }

  accuracy[i] <- mean(acc.df.iteration)
}
```

We plot the results of the iteration process:

```r
ggplot(as.data.frame(df.s,accuracy))+
  geom_point(aes(df.s,accuracy))+
  geom_line(data=as.data.frame(df.s,accuracy),aes(df.s,accuracy))+
  ggtitle(label="Accuracy in the prediction")+
  theme(plot.title = element_text(hjust = 0.5))+
  geom_vline(xintercept = df.s[which.max(accuracy)],color='red')
```


Accuracy in the prediction
0.681
0.678
0.675
0.672
accuracy
4
8
12
df.s