Smoothing and Regression Splines

Group 1: Pariente Antonio, Bosch Guillem, Ebner Lena

29/11/2023

Smoothing and Regression Splines

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:

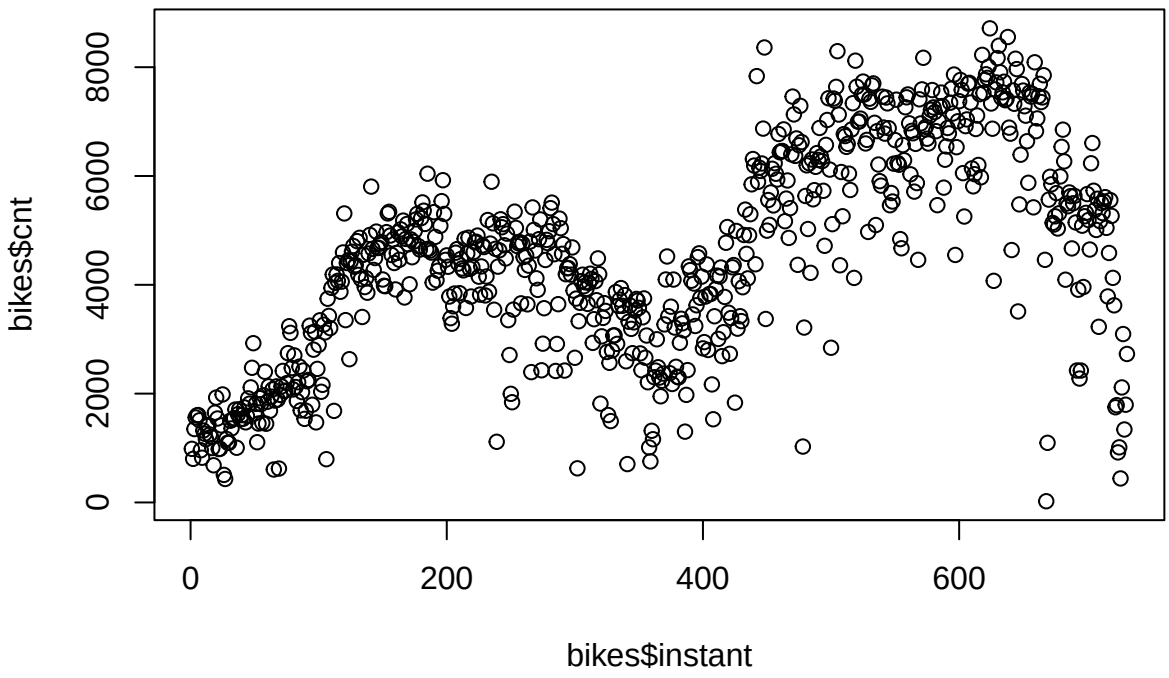
- `instant`: row index, going from 1 to 731.
- `yr`: year (0: 2011, 1:2012).
- `dayyr`: day of the year (from 1 to 365 for 2011, and from 1 to 366 for 2012).
- `weekday`: day of the week (0 for Sunday, 1 for Monday, ..., 6 for Saturday).
- `workingday`: if day is neither weekend nor holiday is 1, otherwise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

1.

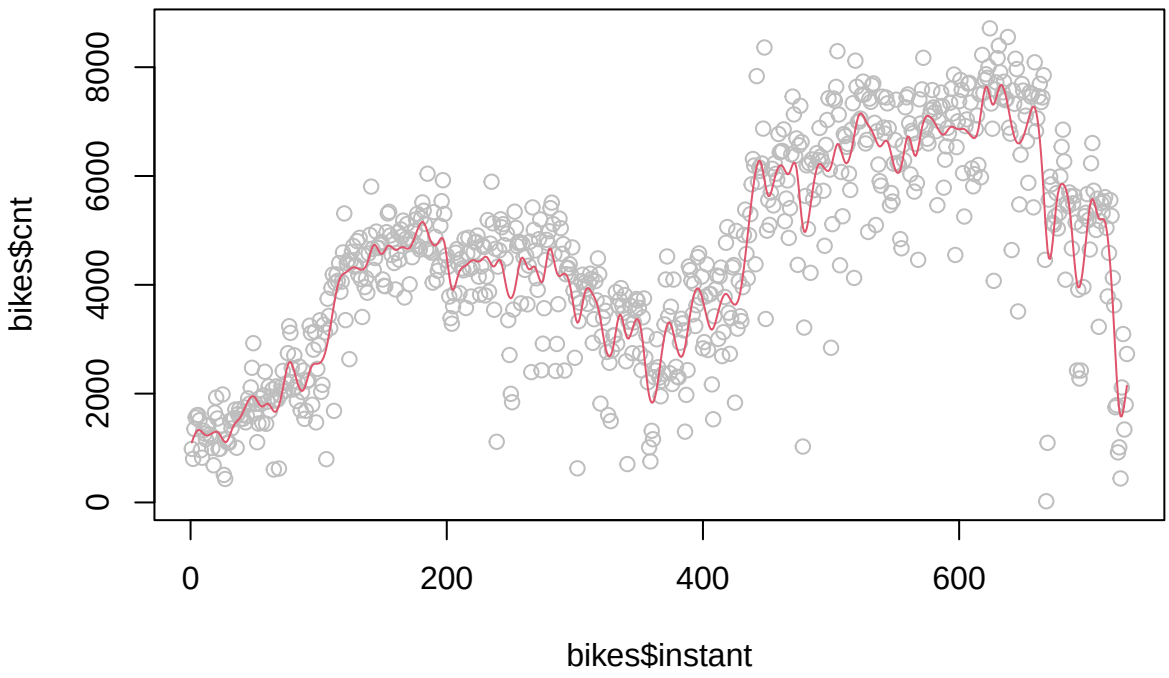
Consider the nonparametric regression of `cnt` as a function of instant. Estimate the regression function m`(instant)` of `cnt` as a function of `instant` using a cubic regression spline estimated with the R function `smooth.spline` and choosing the smoothing parameter by Generalized Cross Validation.

a) Which is the value of the chosen penalty parameter $?
b) Which is the corresponding equivalent number of degrees of freedom `df`?
c) How many knots have been used?
d) Give a graphic with the scatter plot and the estimated regression function $\hat{m}$`(instant)`

```
# load data
load("./bikes.Washington.Rdata")

plot(bikes$instant, bikes$cnt)
```




```
m.hat.instant <- smooth.spline(bikes$instant, bikes$cnt, cv=FALSE, keep.stuff = TRUE)

plot(bikes$instant, bikes$cnt, col="grey")
lines(m.hat.instant,col=2)
```




```
print(paste0("lambda: ",m.hat.instant$lambda))
```

```
## [1] "lambda: 1.00503770328225e-07"
```

```
print(paste0("df: ",m.hat.instant$df))
```

```
## [1] "df: 93.3409050669671"
```

```
print(paste0("number of knots used: ", m.hat.instant$fit$nk))
```

```
## [1] "number of knots used: 136"
```

2.

The script `IRWLS logistic regression.R` includes the definition of the function `logistic.IRWLS.splines` performing nonparametric logistic regression using splines with a `IRWLS` procedure. The basic syntax is the following:

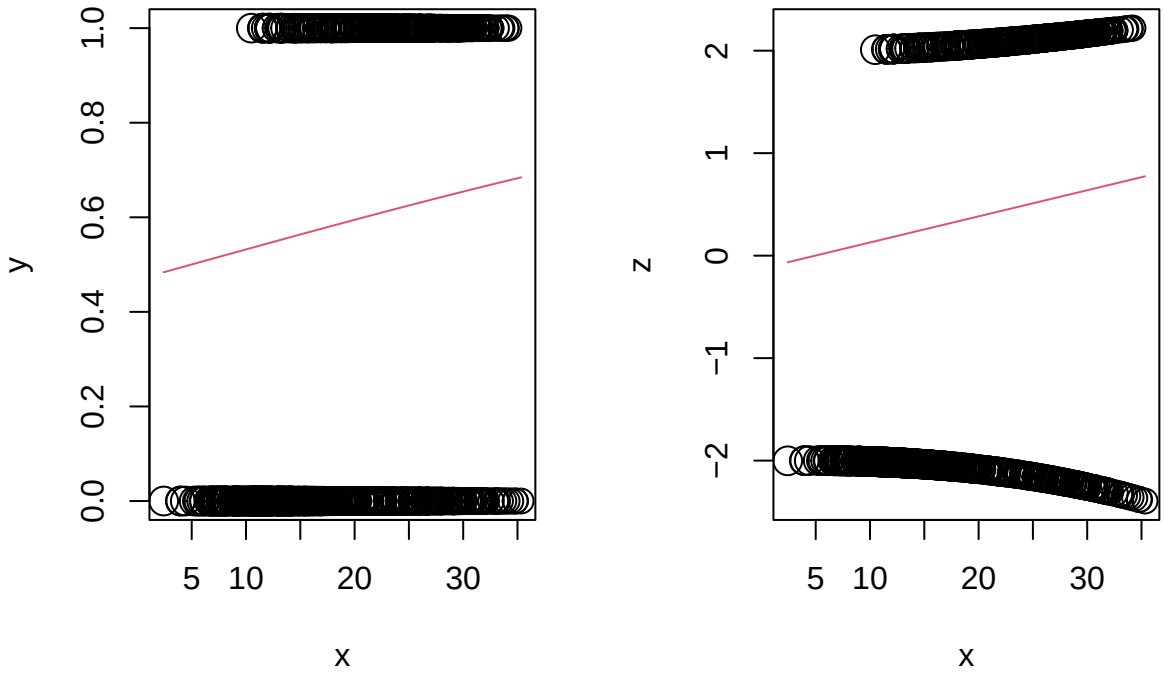
```
logistic.IRWLS.splines(x=..., y=..., x.new=..., df=..., plts=TRUE)
```

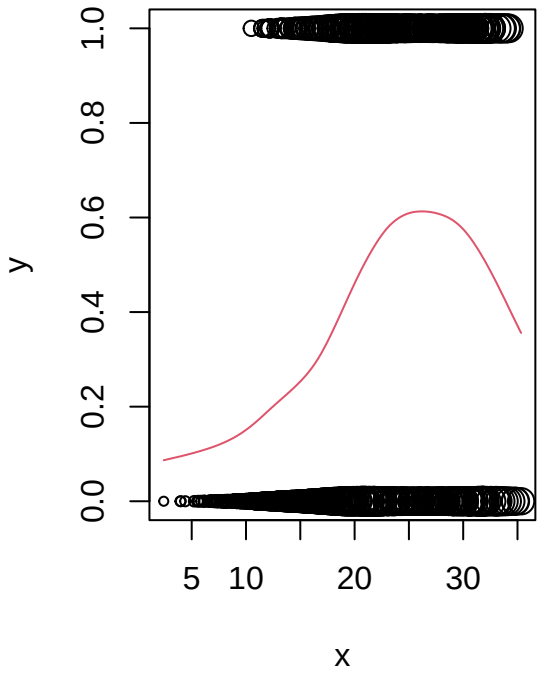
where the arguments are the explanatory variable `x`, the 0-1 response variable `y`, the vector `x.new` of new values of variable `x` where we want to predict the probability of `y` being 1 given that `x` is equal to `x.new`, the equivalent number of parameters (or model degrees of freedom) `df`, and the logical `plts` indicating if plots are desired or not.

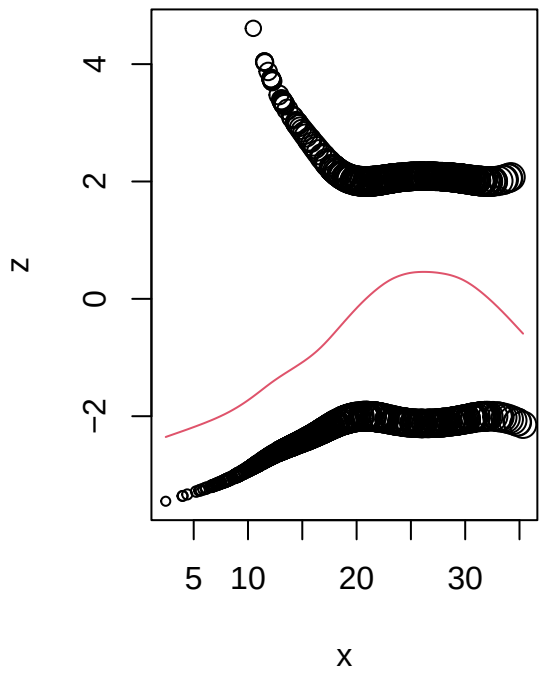
Define a new variable `cnt.5000` taking the value 1 for days such that the number of total rental bikes is larger than or equal to 5000, on 0 otherwise.

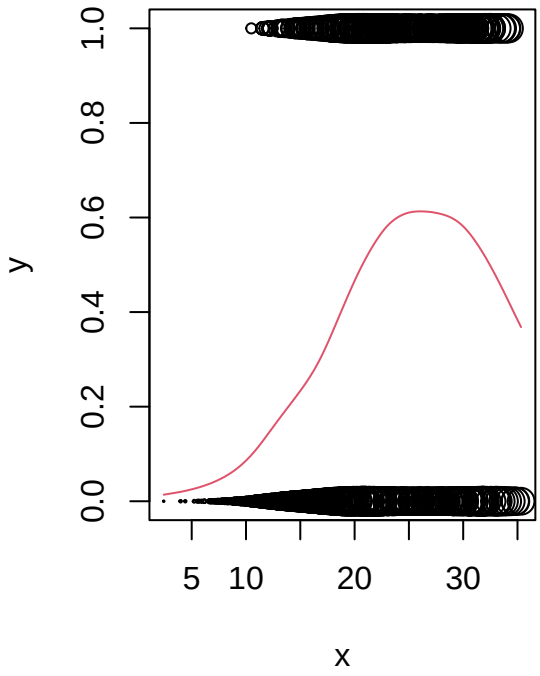
a) Use the function `logistic.IRWLS.splines` to fit the non-parametric binary regression `cnt.5000` as a function of the temperature, using `df=6`. In which range of temperatures is Pr`(cnt >= 5000|temp)` larger than 0,5?

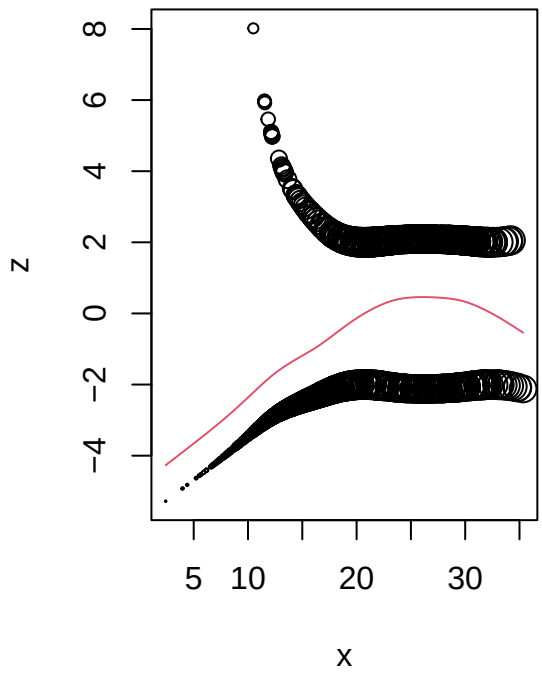
```
source("IRWLS_logistic_regression.R")
bikes$cnt.5000 <- ifelse(bikes$cnt>=5000,1,0)

pred<-seq(0,40,length.out=1000)
bikes<-bikes[order(bikes$temp),]
fit<-logistic.IRWLS.splines(x=bikes$temp, y=bikes$cnt.5000, x.new=pred, df=6, plts=TRUE)
```

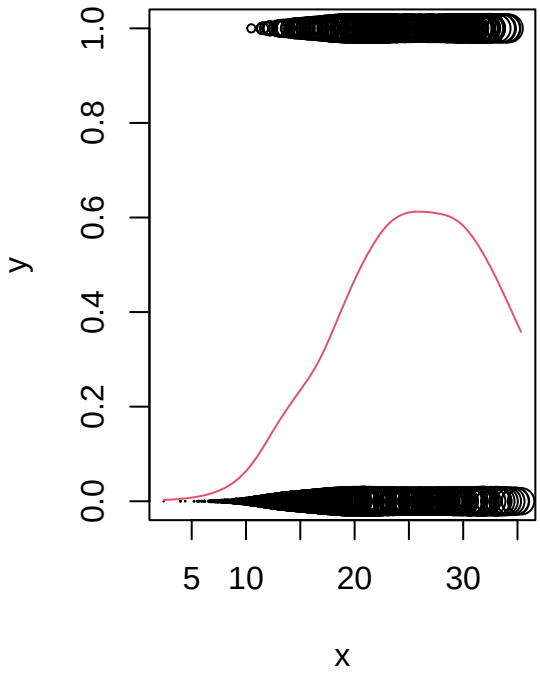


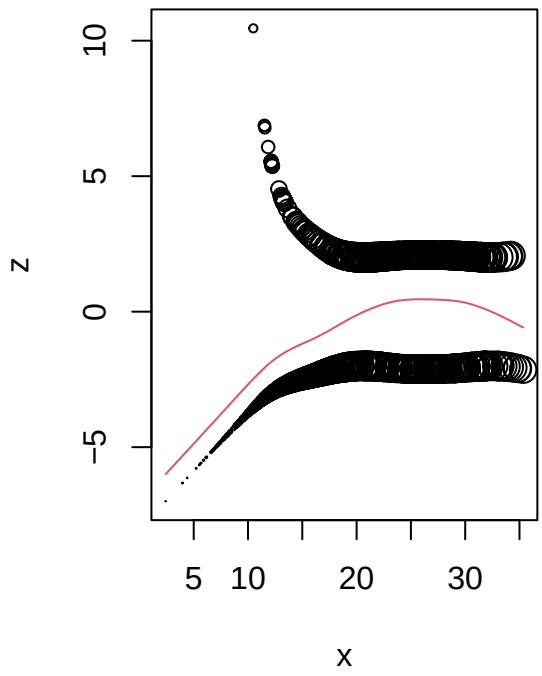

y
0.0
0.2
0.4
0.6
0.8
1.0
5
10
20
30
x

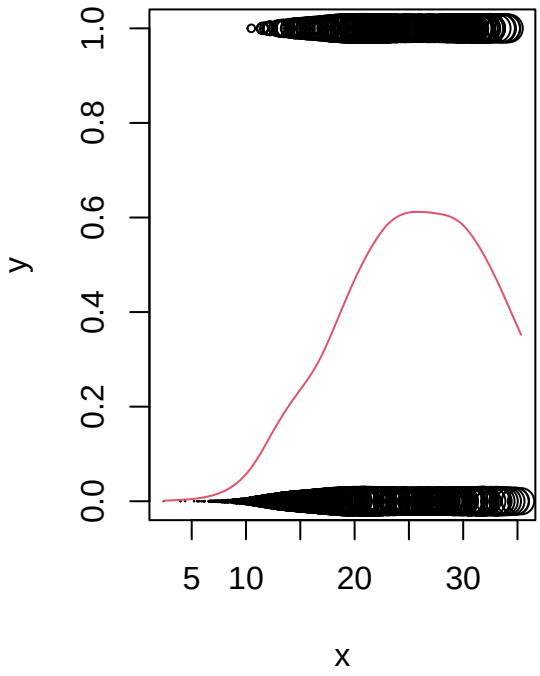

z
−2
0
2
4
5
10
20
30
x

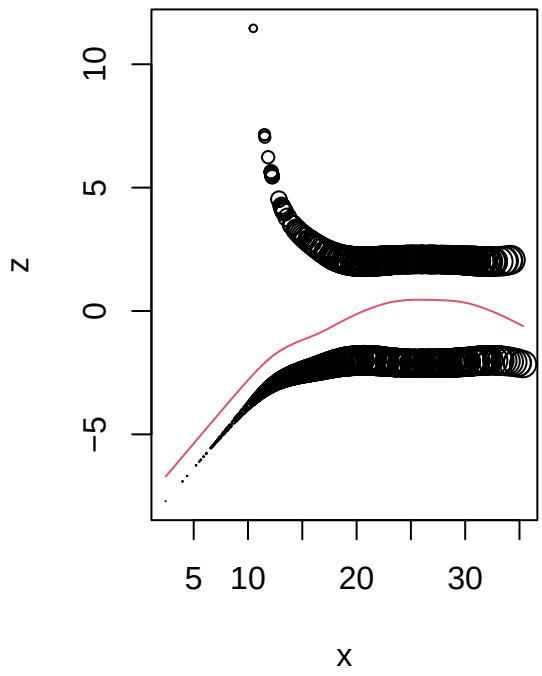

y
0.0
0.2
0.4
0.6
0.8
1.0
5
10
20
30
x

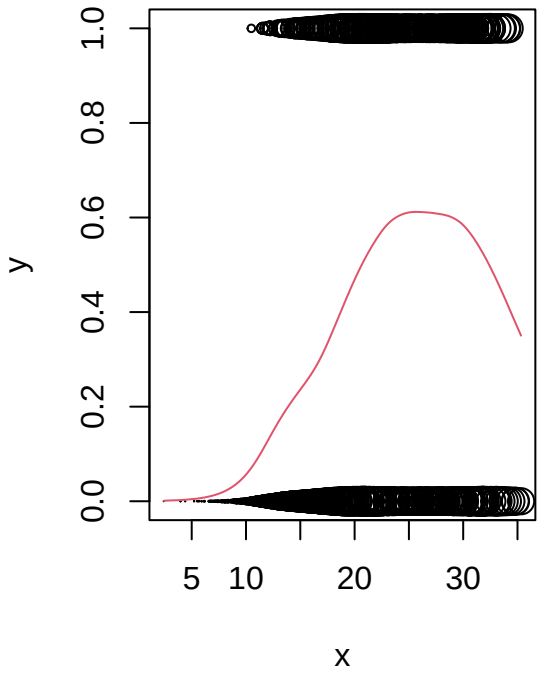

z
−4
−2
0
2
4
6
8
5
10
20
30
x

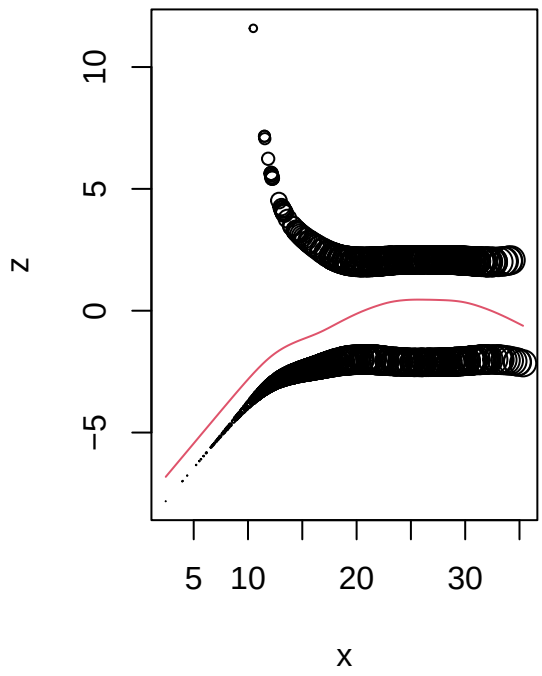

y
0.0
0.2
0.4
0.6
0.8
1.0
5
10
20
30
x

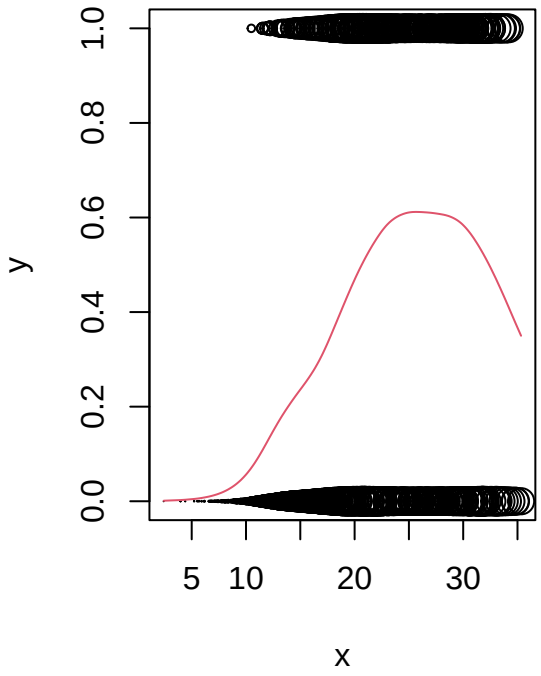

z
−5
0
5
10
5
10
20
30
x

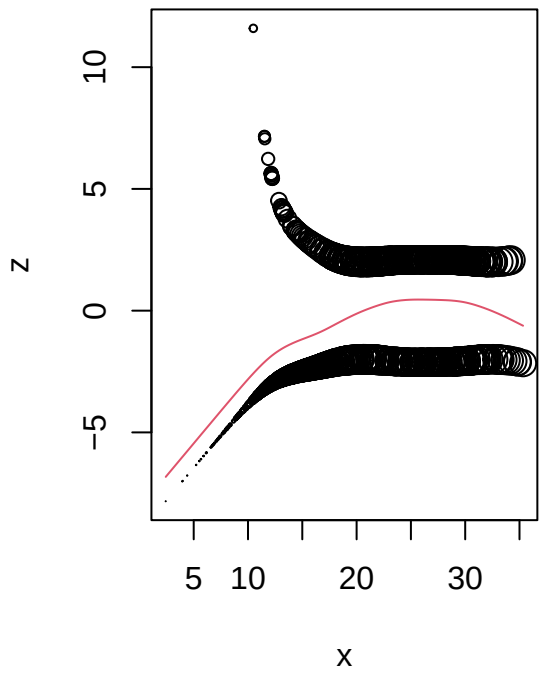

y
0.0
0.2
0.4
0.6
0.8
1.0
5
10
20
30
x


z
−5
0
5
10
5
10
20
30
x

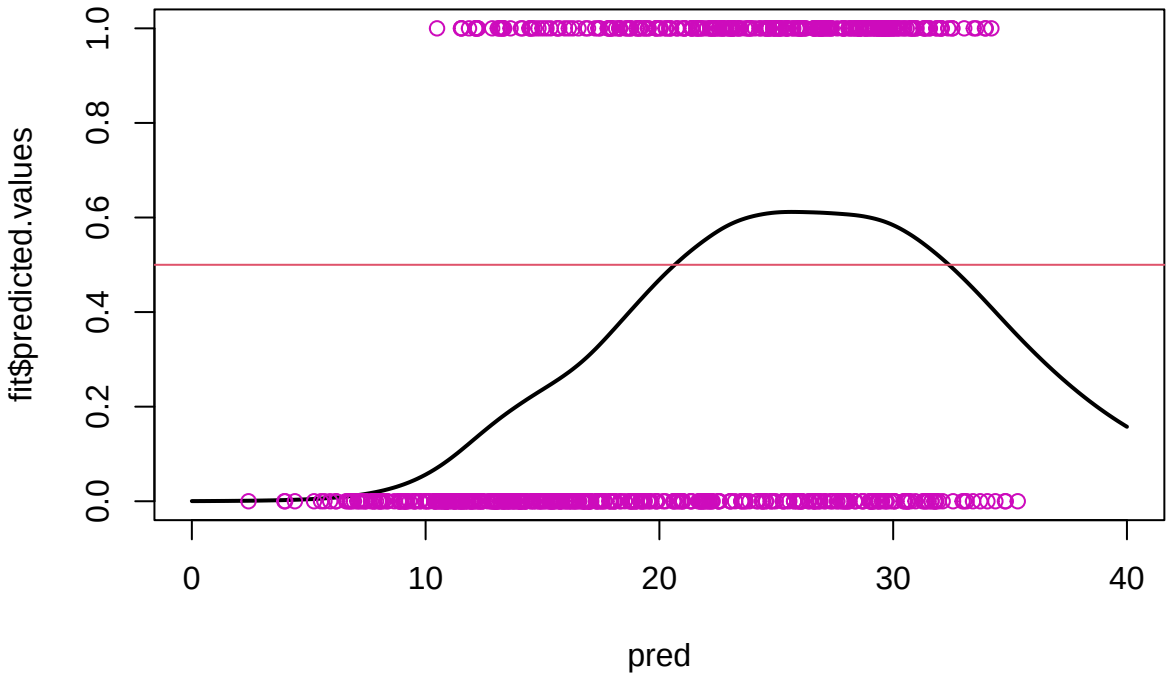

y
0.0
0.2
0.4
0.6
0.8
1.0
5
10
20
30
x


z
−5
0
5
10
5
10
20
30
x






```
r<-pred[fit$predicted.values>0.5]
rr<-range(r)

plot(pred, fit$predicted.values, xlim=c(0,40),ylim=c(0,1), lwd=2, type="l")
points(bikes$temp, bikes$cnt.5000, cex=1, col=6)
abline(h=0.5, col=2)
```



Once the logistic spline regression is fitted, the range of temperatures can be estimated by predicting a large sequence of temperatures and considering the range where the prediction is higher than 0.5. In this case the result was 20.6606607, 32.3523524.

```r
# r
print("Range where predicted temperatures are larger is higher than 0.5:")
```

```
## [1] "Range where predicted temperatures are larger is higher than 0.5:"
```

```r
rr
```

```
## [1] 20.66066 32.35235
```

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

```r
library(groupdata2)     # fold()
set.seed(7777777)
```

```r
n=5
df <- fold(
  bikes,
  k = n,
  num_fold_cols = 1,
```

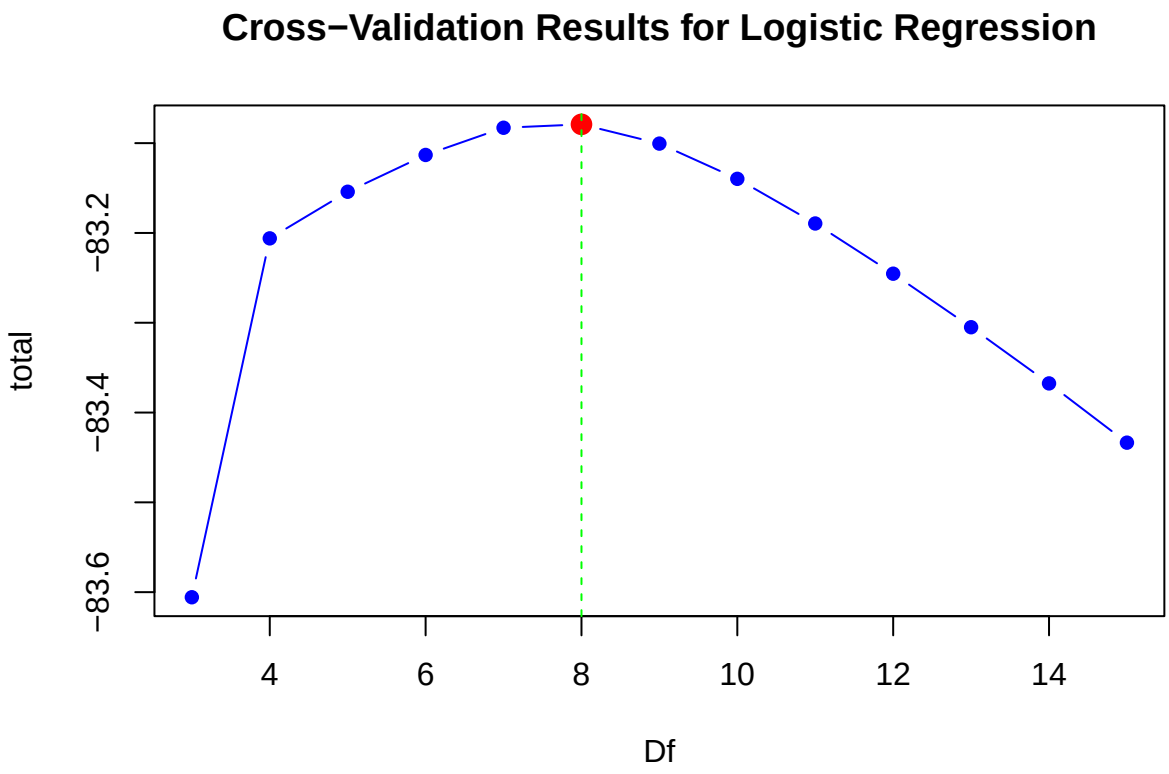
```
  parallel = FALSE #
)


dff=seq(3,15)
total=rep(0,length(dff))
for (j in  dff){
  for ( i in seq(n)){
    fit<-logistic.IRWLS.splines(x=bikes$temp[df$.folds!=i], y=bikes$cnt.5000[df$.folds!=i], x.new=bikes
    total[j - 2] <- total[j - 2] + sum(log(fit$predicted.values) * bikes$cnt.5000[df$.folds == i] +
                                       log(1 - fit$predicted.values) * (1 - bikes$cnt.5000[df$.folds ==
  }
total[j-2]<-total[j-2]/n
}
dff_best<-dff[which.max(total)]
```

The best value for **df** was `dff_best`

```
print(paste0("Best choosen df by log-likelihood cv: ", dff_best))
```

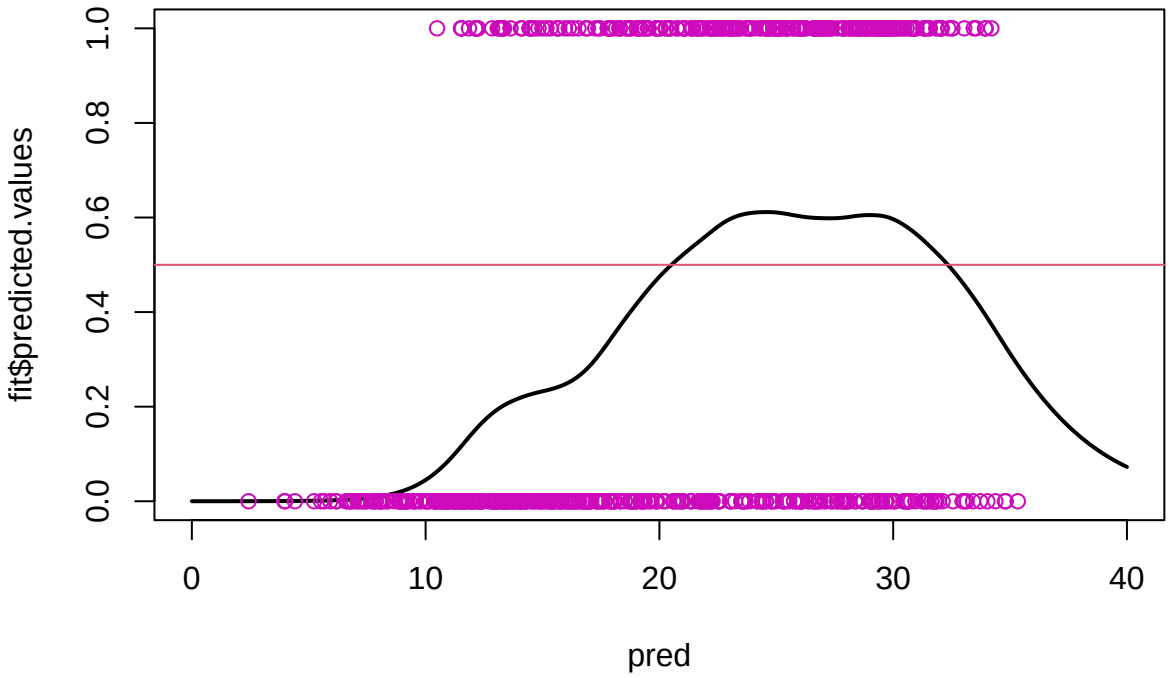
```
## [1] "Best choosen df by log-likelihood cv: 8"
```

```
plot(dff, total, type = "b", pch = 16, col = "blue",
     xlab = "Df",
     main = "Cross-Validation Results for Logistic Regression")

points(dff_best, max(total), col = "red", pch = 16, cex = 1.5)

abline(v = dff_best, col = "green", lty = 2)
```




```
fit<-logistic.IRWLS.splines(x=bikes$temp, y=bikes$cnt.5000, x.new=pred, df=dff_best, plts=FALSE)
```

```
plot(pred, fit$predicted.values, xlim=c(0,40),ylim=c(0,1), lwd=2, type="l")
points(bikes$temp, bikes$cnt.5000, cex=1, col=6)
abline(h=0.5, col=2)
```


fit$predicted.values
0.0
0.2
0.4
0.6
0.8
1.0
0
10
20
30
40
pred