Statistical Learning

Code

(*You've seen this before!*)

Christophe Bontemps & Patrick Jonsson - SIAP[1]

Introduction

This course is designed for current or future *data scientists* working in an NSO, or in any statistical office.

> **Data Scientist**: "Person who is **better** at statistics than any software engineer and **better** at software than any statistician." (J. Wills)

The term *machine learning* is used because the computer (more accurately the algorithm) figures out the model from the data. There are many related words: statistical learning, big data, nonparametric estimation, AI, high-dimensional models, …

In fact there are **two main learning problems:**

1. When we observe both the outcome **y** and regressors (also called *features*) **x** the analysis is called *Supervised learning*. This is in fact a regression type of model with two possible denominations

- Regression: y is continuous
- Classification: y is categorical

2. When we do not observe **y** but only several **x** the analysis is called *Unsupervised learning* This is also called *Cluster Analysis*: e.g. determine five types of consumers given many socio-economic measures.

Tasks

Data science involves several tasks, some of them can take a lot of time in the professionnal life of a data scientist.

- Data collection: usually done beforehand, so you don't have to do it yourself
- Data organization: when you get data in several formats, from several sources, …
- Data cleaning: removing duplicates, spotting missing data and errors, …
- Data visualization
- Data analysis

Tools

- R and the *caret* package https://topepo.github.io/caret/index.html (https://topepo.github.io/caret/index.html). Caret can deal with 238 "models": we are going to use some of them.
- We are not going to use Python, but *Scikit-Learn* and *PyCaret* are two main tools in Python
- There are interfaces to call Python from inside R, the *reticulate* package allows to do this from inside a Markdown document.
- If you work in data science, you will have to keep up to date. The internet is a good source, with sites such as https://www.kdnuggets.com/ (https://www.kdnuggets.com/) and https://towardsdatascience.com/ (https://towardsdatascience.com/) Look at https://www.kaggle.com/ (https://www.kaggle.com/) for competitions and solutions posted by participants.

Statistical Learning In practice

One specific feature is that in most cases the data is "big":

- usually n is relatively large
- sometimes $\dim(x)$ is also relatively large compared to n
- in addition, machine learning typically requires to estimate several models, several times
- in practice, this requires computing power and takes time

> In the course, we will deal with "not so big" data for practical purposes

Let's start with an example

We will begin by a look at consumption data and focus on relations between **total expenses** of a household and the **share of a good** in total consumption. Here we will study food share in SouthAfrica[2].

Hide

```
# We load the data from the "Data" folder where we store raw data
SouthAfrica <-read.csv2("https://www.unsiap.or.jp/on_line/ML/MLData/ML_SouthAfrica.csv")
# ltexp is expenditure in log
# The variables "zij" are dummies for families with "i" adults and "j" kids.
```

Now we select **only singles** households and reorder according to expenditure variable.

Hide

```
Singles <- SouthAfrica[SouthAfrica$z10==1,4:5]
#Singles Subset
Singles <- Singles[order(Singles$ltexp),1:2]
#Reorder so that log expenditure is in increasing order
FoodShr <- Singles$FoodShr
ltexp   <- Singles$ltexp
MyData <- data.frame(FoodShr,ltexp)
```

Hide

```
datasummary_skim(MyData, type = "numeric")
```

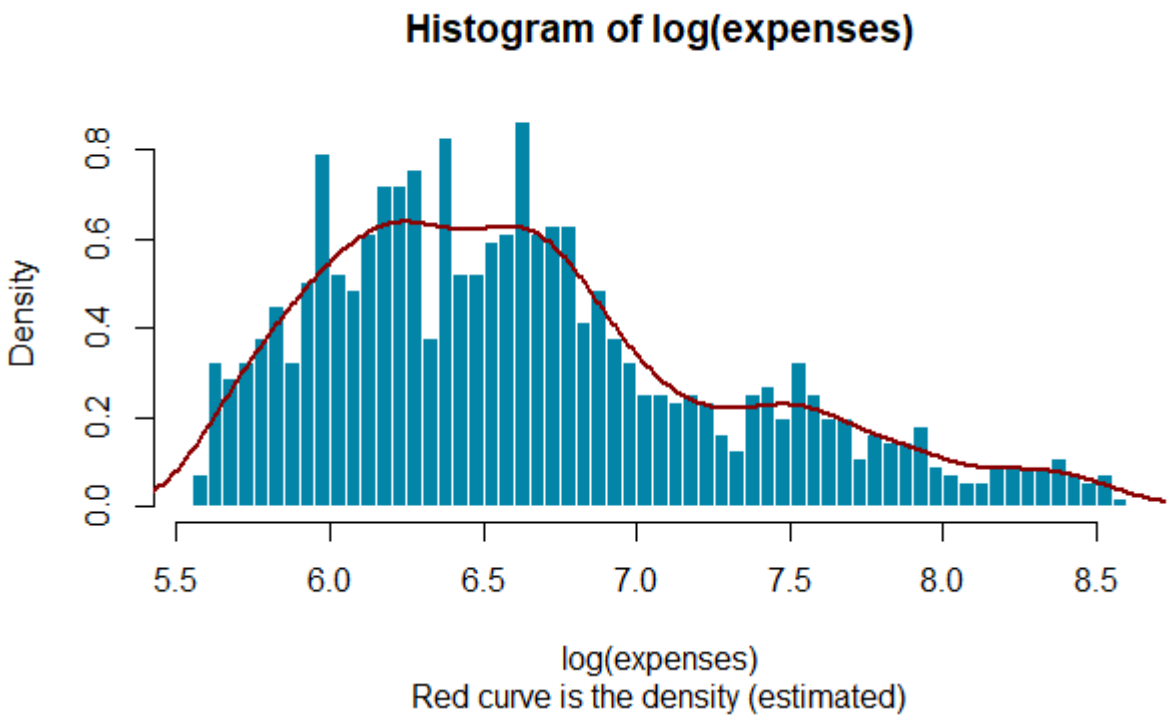
	Unique (#)	Missing (%)	Mean	SD	Min	Median	Max	
FoodShr	88	0	0.4	0.2	0.0	0.4	0.9	
ltexp	265	0	6.6	0.7	5.6	6.5	8.6	

Histograms and Densities

Let's first look at the data.

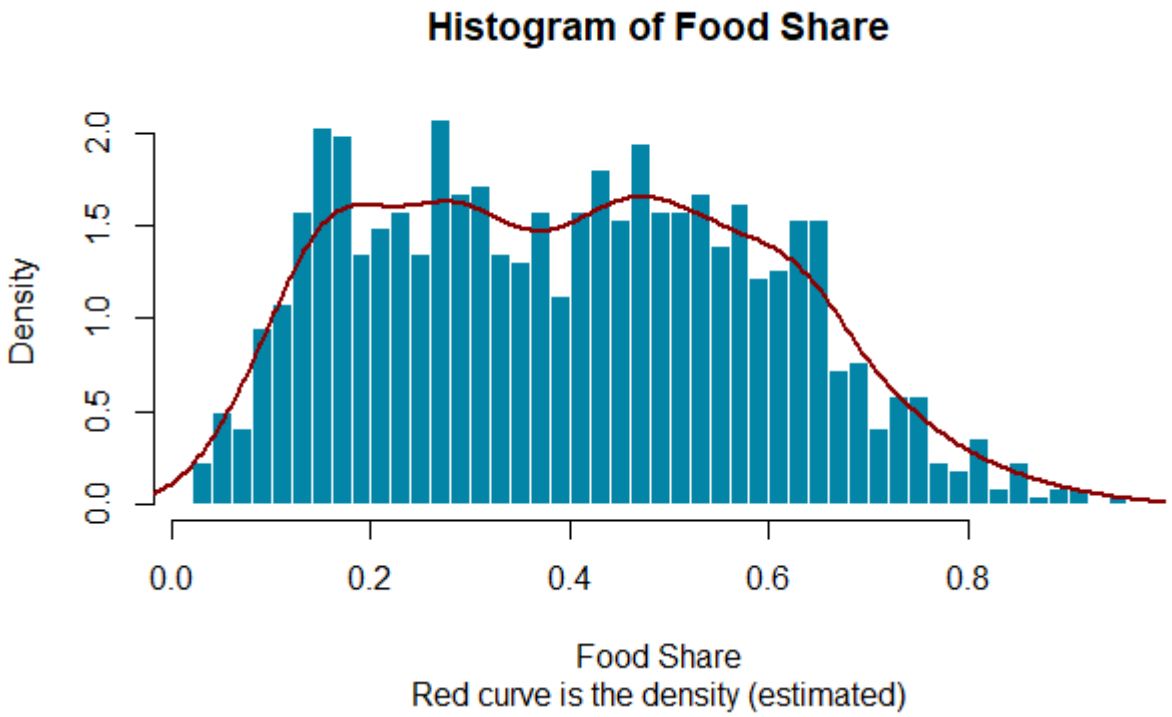
Hide

```
hist(ltexp,prob=T,breaks=50,
     main = "Histogram of log(expenses)",
     sub = "Red curve is the density (estimated)",
     xlab = "log(expenses)" ,
     col = SIAP.color, border = "white")
lines(density(ltexp,na.rm = TRUE),lwd=2,col = "darkred")
```



Hide

```
hist(FoodShr,prob=T,breaks=50,
     main = "Histogram of Food Share",
     sub = "Red curve is the density (estimated)",
     xlab = "Food Share" ,
     col = SIAP.color, border = "white")
lines(density(FoodShr,na.rm = TRUE),lwd=2,col = "darkred")
```

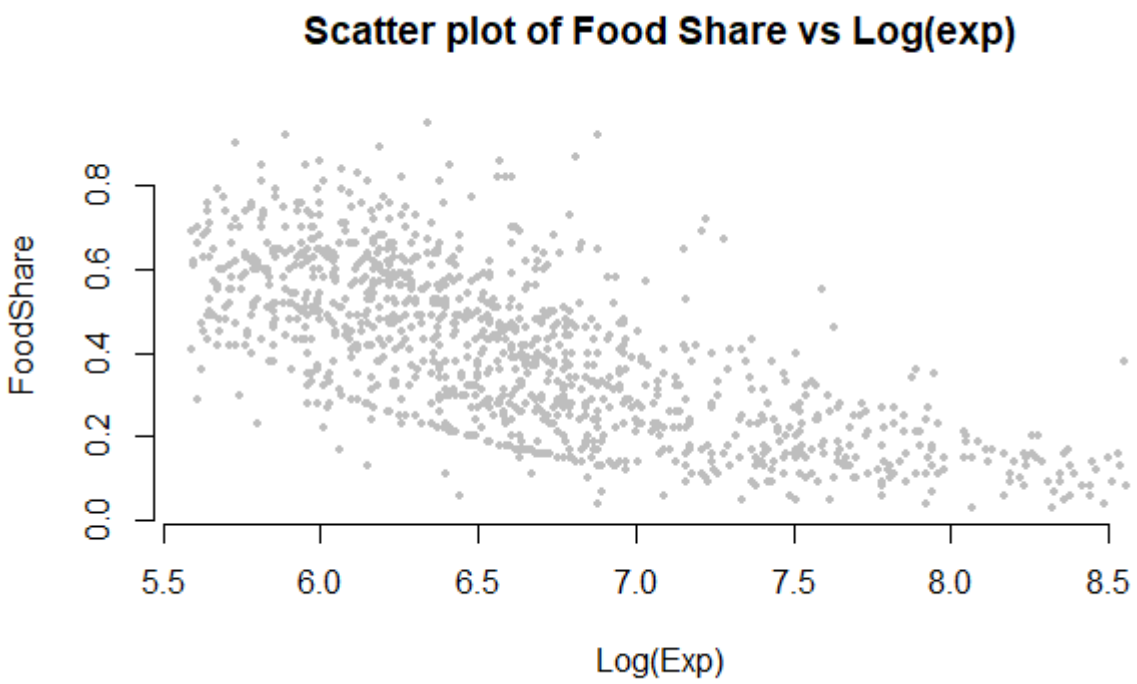


Plot

This is the plot of our observations.

Hide

```
plot(FoodShr~ltexp, main="Scatter plot of Food Share vs Log(exp)",
     xlab="Log(Exp)",ylab = "FoodShare",
    pch=19, cex = 0.5,col = "grey", frame.plot = FALSE )
```



Exporing the relationships: It is all about $f(\cdot)$

We may be interested in the relation between y = Food Share and x = log(Expenses) and thus in the expression

$$y = f(x) + \varepsilon$$

In essence, statistical learning refers to a set of approaches for estimating $f(\cdot)$

James, Witten,Hastie & Tibshirani (2021)

How to estimate $f(\cdot)$?

Linear Model

Let's begin with a simple linear model.

$$y = x'\beta + \varepsilon$$

Hide

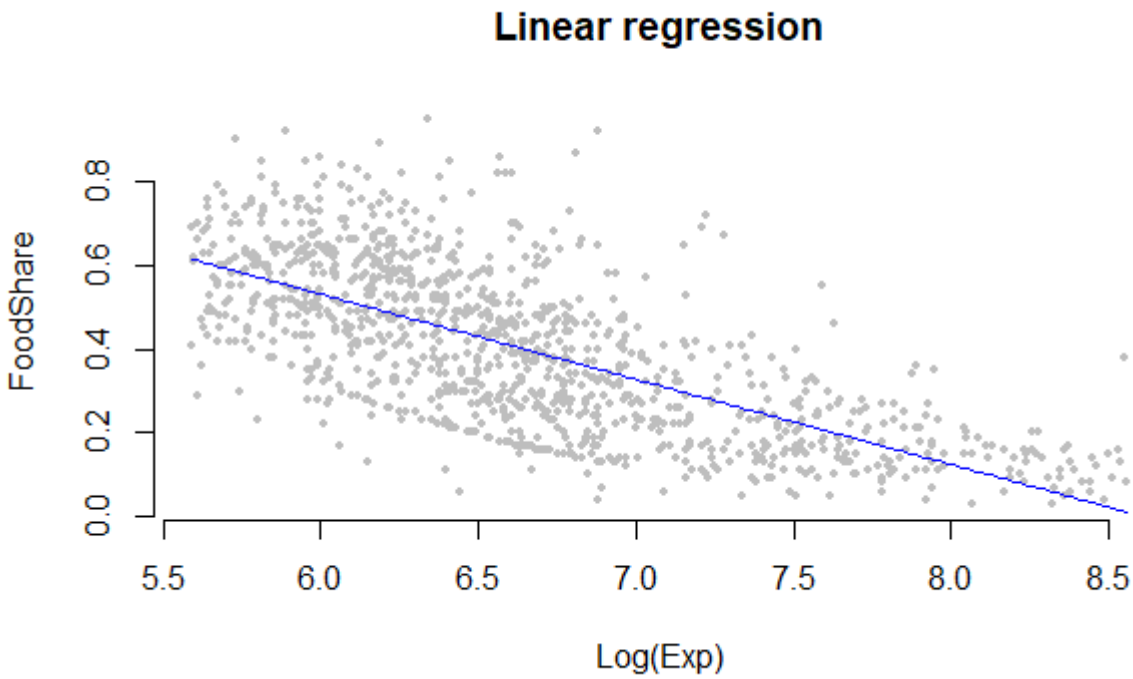
```
lmFood <- lm(FoodShr~ltexp)

# Table
xtable(summary(lmFood)) %>%
  kable(digits=2) %>%
  kable_styling()
```

	Estimate	Std. Error	t value	Pr(>|t|)
(Intercept)	1.75	0.04	41.09	0
ltexp	-0.20	0.01	-31.84	0

Here is the estimated regression line. Note how it is constructed in practice.

Hide

```
plot(ltexp, FoodShr, type="n",
     main="Linear regression",
     xlab="Log(Exp)", ylab = "FoodShare",
    pch=19, cex = 0.5,col = "grey", frame.plot = FALSE )
points(ltexp,FoodShr,
    pch=19, cex = 0.5,col = "grey" )

lmFood <- lm(FoodShr ~ poly(ltexp, degree=1,raw=TRUE))
newx <- seq(from=min(ltexp),to=max(ltexp),
            length.out = 200)

lines(newx, predict(lmFood, data.frame(ltexp = newx)),
      col = "blue")
```



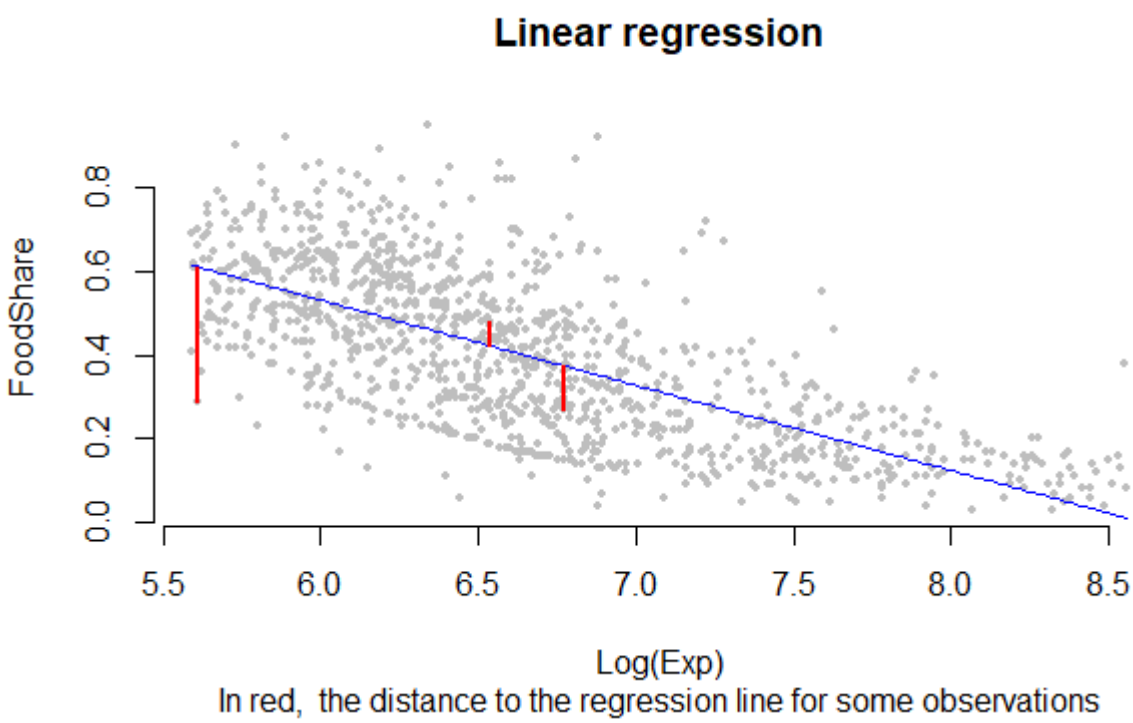
How is this line computed?

You may recall that the regression is obtained by trying to find the line defined by the equation $\beta_0 + \beta_1 x$ that *fits* the data and *minimize the vertical distance between a point and the estimated line*. In other words, we are looking for β_0 and β_1, the *parameters* of the line, such that the sum of all distances for all points is minimized. The vertical distance of any y_i to the line for a particular point i is simply:

$$\left(y_i - (\beta_0 + \beta_1 x_i)\right)^2$$

Hide

```
# Setting the plot with points
plot(ltexp, FoodShr, type="n",
     main="Linear regression",
     sub = "In red,  the distance to the regression line for some observations",
     xlab="Log(Exp)", ylab = "FoodShare",
    pch=19, cex = 0.5,col = "grey", frame.plot = FALSE )
points(ltexp,FoodShr,
    pch=19, cex = 0.5,col = "grey" )

# Estimation of the linear regression model
lmFood <- lm(FoodShr ~ poly(ltexp, degree=1,raw=TRUE))

# Plotting the regression line for a sequence of points
newx <- seq(from=min(ltexp),to=max(ltexp),
            length.out = 200)
lines(newx, predict(lmFood, data.frame(ltexp = newx)),
      col = "blue")

# Plotting the vertical distances in red
i <- 6                # Here we take the distance for the 6th point
segments(ltexp[i] , FoodShr[i], ltexp[i], lmFood$coefficients[1] + lmFood$coefficients[2]* ltexp[i]
         , col = "red", lw=2)
# Vertical distances for some other points
segments(ltexp[555] , FoodShr[555], ltexp[555],
         lmFood$coefficients[1] + lmFood$coefficients[2]* ltexp[555],
         col = "red", lw=2)
segments(ltexp[725] , FoodShr[725], ltexp[725],
         lmFood$coefficients[1] + lmFood$coefficients[2]* ltexp[725],
         col = "red", lw=2)
```



Therefore, the regression line can be found by minimizing the *residual sum of squares* (RSS, see below) that is by solving the following optimization problem:

- find β_0 and β_1 such that:

$$Min_{\ (\beta_0,\beta_1)} \quad \sum_{i=1}^{n} \left(y_i - (\beta_0 + \beta_1 x_i)\right)^2$$

  is **minimal**.

The regression line is found as a solution of an **optimization** problem

In this case when $f(\cdot)$ is linear, an analytical solution exist (the equation can be written explicitly).

Goodness of fit: R^2

We can compute one of the favorite measures of adjustment: the R^2 that measures how close the data are to the fitted regression line. We use the sum of squared distances of the observations Y_i to the regression line, or *residual sum of squares* (RSS), as compared to the *total sum of the squares* (TSS, measured as the sum of the distances of the observations Y_i to their mean. So:

$$TSS = \sum_{i=1}^{n} (y_i - \bar{y})^2$$

and

$$RSS = \sum_{i=1}^{n} (y_i - \widehat{f}(x_i))^2$$

The definition of R-squared is then: $R^2 = \frac{TSS-RSS}{TSS}$ It is simply the explained (by the regression) variation of the outcome variable y divided by the total variation of the outcome variable. It can be noted that $R^2 = 1 - \frac{RSS}{TSS}$ and that the goodness of fit is perfect when equal to 1.

Hide

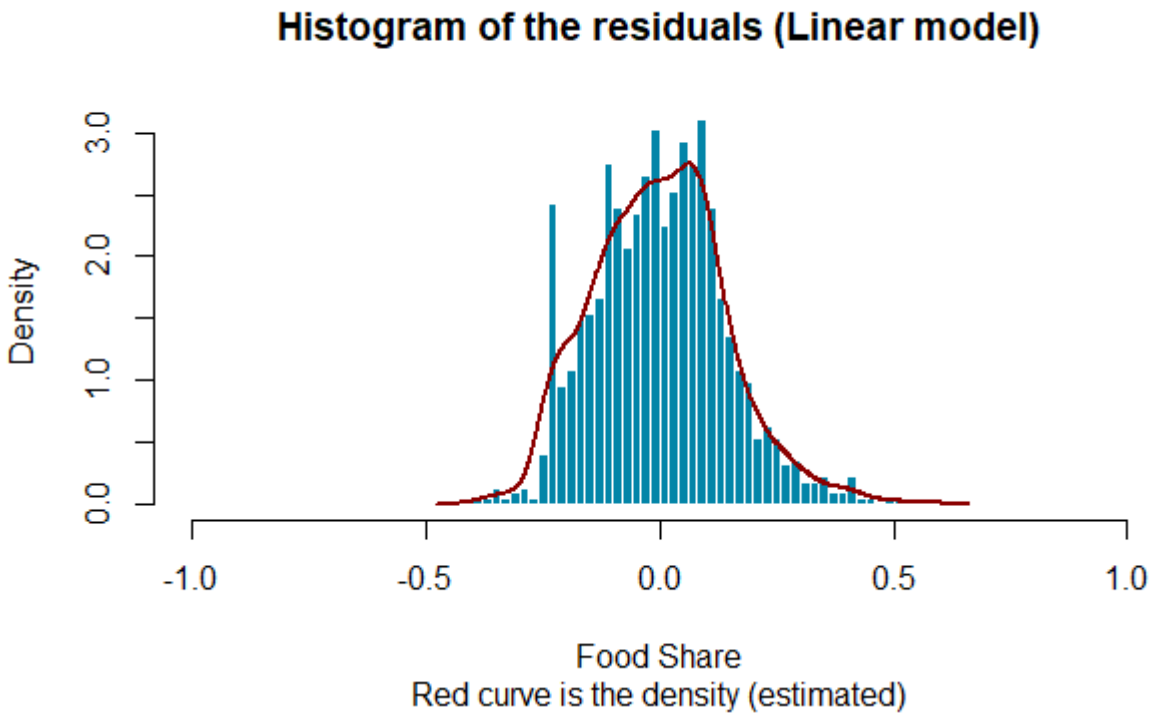
```
summary(lmFood)$adj.r.squared
```

```
## [1] 0.4776279
```

Observing the residuals

Hide

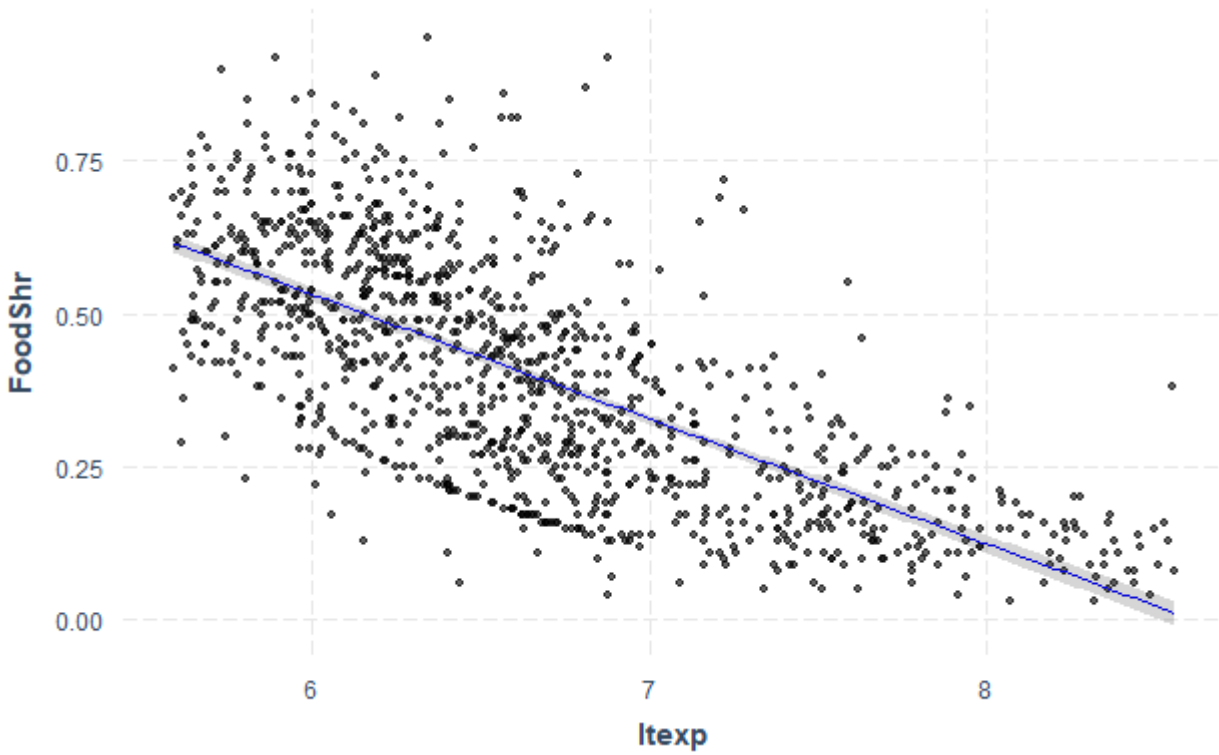
```
lmFood.res = resid(lmFood)
# We now plot the residual against the observed values of the variable FoodShr.
hist(lmFood.res,prob=T,breaks=50,
     main = "Histogram of the residuals (Linear model)",
     sub = "Red curve is the density (estimated)",
     xlab = "Food Share" , xlim=c(-1,1),
     col = SIAP.color, border = "white")
lines(density(lmFood.res,na.rm = TRUE),lwd=2,col = "darkred")
```



The *jtools* package allows to simply obtain a similar plot, with confidence interval around the regression line.

Hide

```
library(jtools)
effect_plot(lmFood, data = Singles,
            pred = ltexp,
            interval = TRUE, robust = "HC1",
            colors = "blue", line.thickness = 0.7,
            plot.points = TRUE, pch = ".",
            point.size = 1)
```



Polynomial Models

We can try to have a better model by introducing some non linearity. This can be done using polynomials of the unique regressor x. Here we define a polynomial model of order 2, or *quadratic model*:

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

.

Hide

```
lmFood2 <- lm(FoodShr ~ poly(ltexp, degree=2,raw=TRUE))
summ(lmFood2, robust = "HC1")$coeftable %>%
  kable(digits=2) %>%
  kable_styling()
```

	Est.	S.E.	t val.	p
(Intercept)	3.25	0.34	9.55	0
poly(ltexp, degree = 2, raw = TRUE)1	-0.64	0.10	-6.55	0
poly(ltexp, degree = 2, raw = TRUE)2	0.03	0.01	4.56	0

Let us see if the adjustment, in terms of the R^2 has been better:

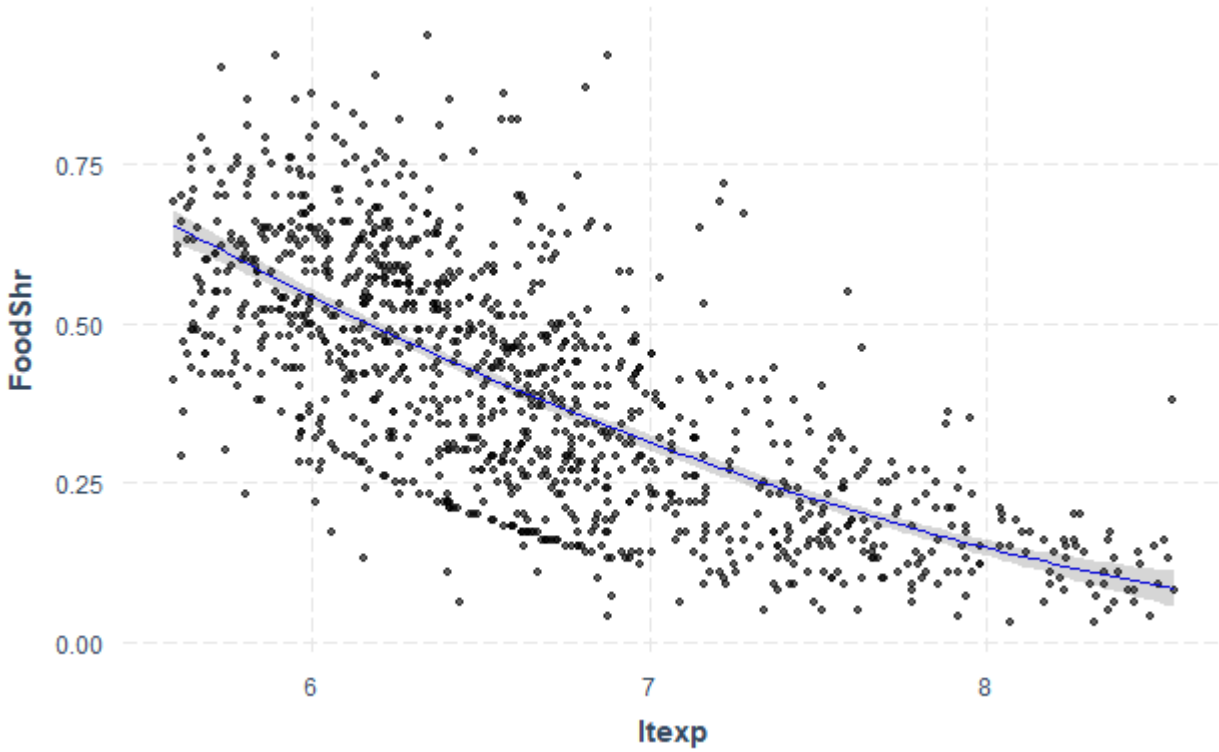
Hide

```
summary(lmFood2)$adj.r.squared
```

```
## [1] 0.4845056
```

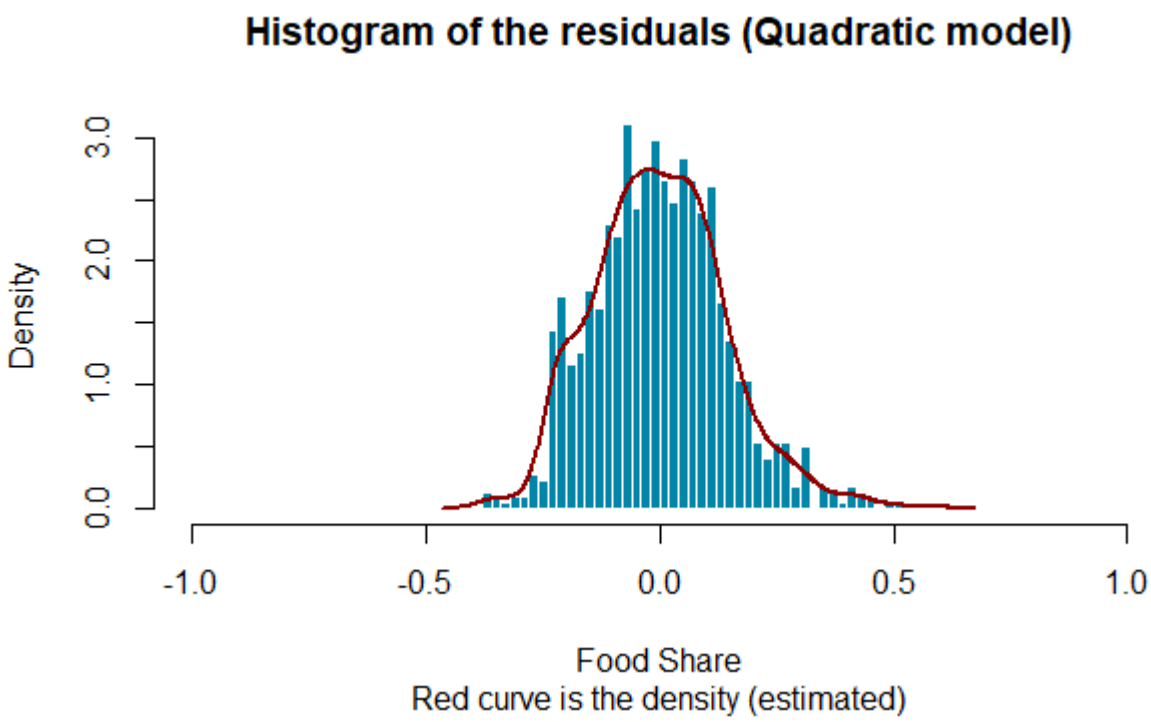
Hide

```
effect_plot(lmFood2, data = Singles,
            pred = ltexp,
            interval = TRUE, robust = "HC1",
            colors = "blue", line.thickness = 0.7,
            plot.points = TRUE, pch = ".",
            point.size = 1)
```



Hide

```
lmFood2.res = resid(lmFood2)
# We now plot the residual against the observed values of the variable FoodShr.
hist(lmFood2.res,prob=T,breaks=50,
     main = "Histogram of the residuals (Quadratic model)",
     sub = "Red curve is the density (estimated)",
     xlab = "Food Share" , xlim=c(-1,1),
     col = SIAP.color, border = "white")
lines(density(lmFood2.res,na.rm = TRUE),lwd=2,col = "darkred")
```



Let's try with more degrees in our polynomial model (*Cubic model*):

$$y = \beta_0 + \beta_1 x + \beta_2 x^2 + \beta_3 x^3 + \varepsilon$$

.

Hide

```
lmFood3 <- lm(FoodShr ~ poly(ltexp, degree=3,raw=TRUE))
summ(lmFood3, robust = "HC1")$coeftable %>%
  kable(digits=2) %>%
  kable_styling()
```

	Est.	S.E.	t val.	p
(Intercept)	-8.74	2.97	-2.95	0
poly(ltexp, degree = 3, raw = TRUE)1	4.61	1.28	3.59	0
poly(ltexp, degree = 3, raw = TRUE)2	-0.73	0.18	-3.96	0
poly(ltexp, degree = 3, raw = TRUE)3	0.04	0.01	4.17	0

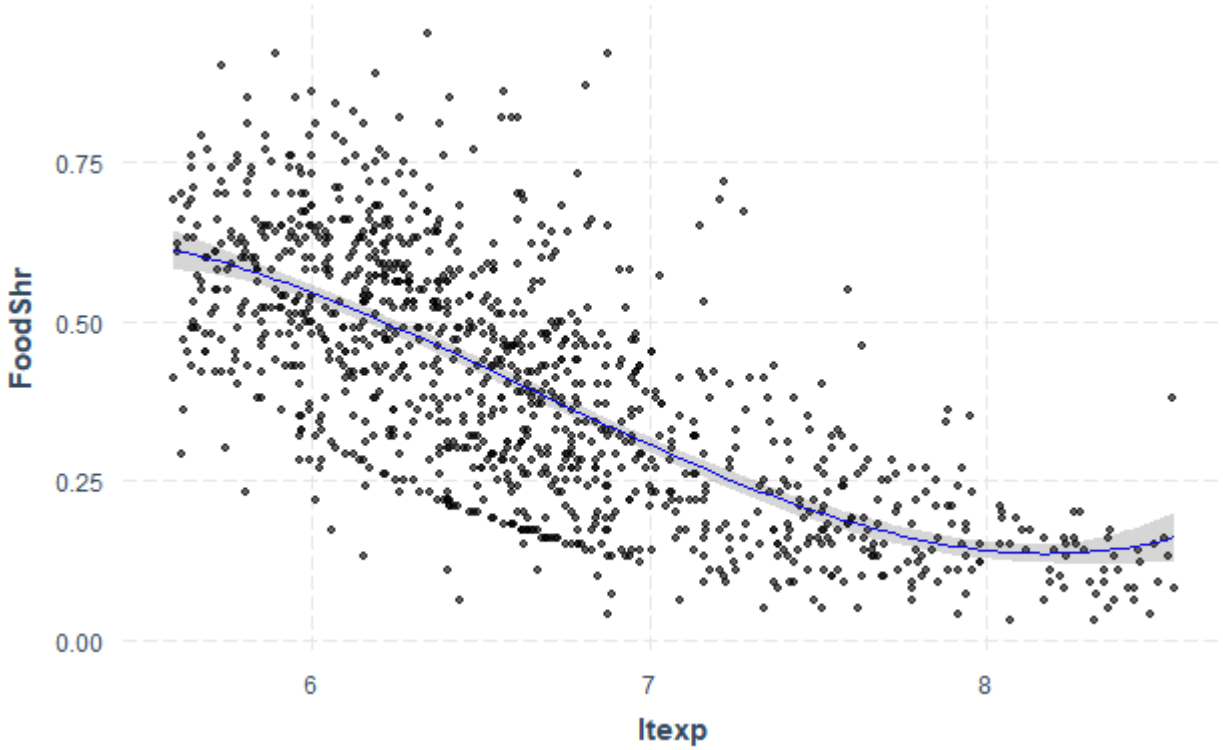
Hide

```
summary(lmFood3)$adj.r.squared
```

```
## [1] 0.489811
```

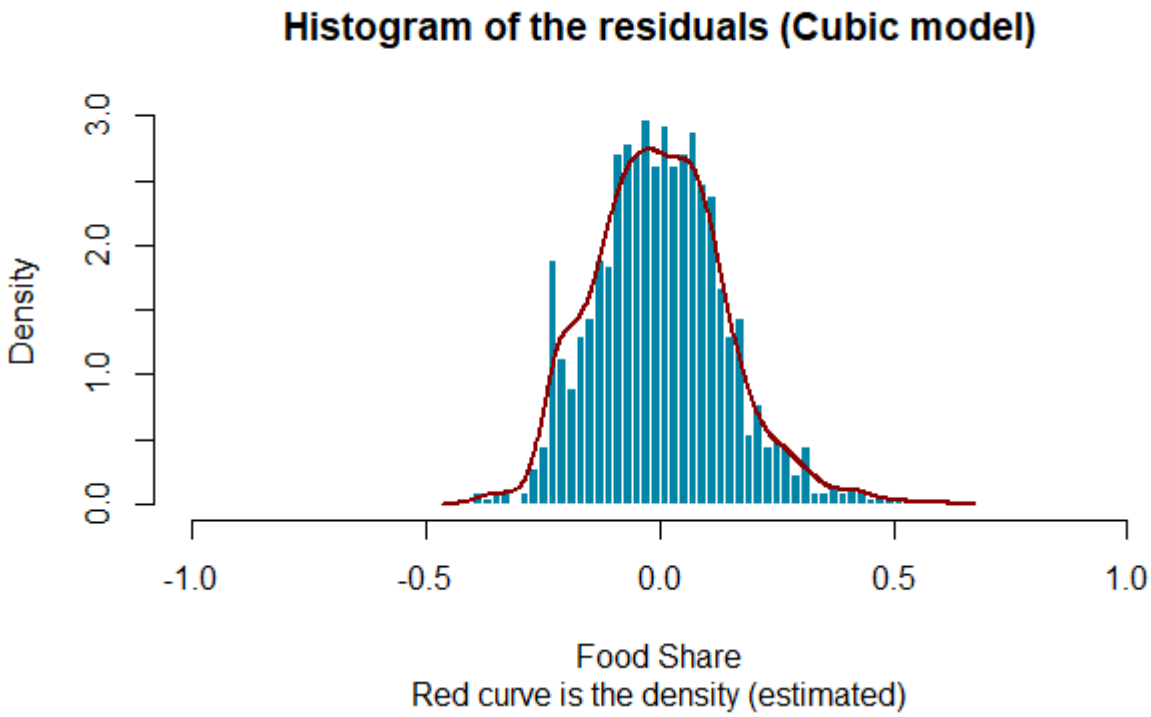
Hide

```
effect_plot(lmFood3, data = Singles,
            pred = ltexp,
            interval = TRUE, robust = "HC1",
            colors = "blue", line.thickness = 0.7,
            plot.points = TRUE, pch = ".",
            point.size = 1)
```



Hide

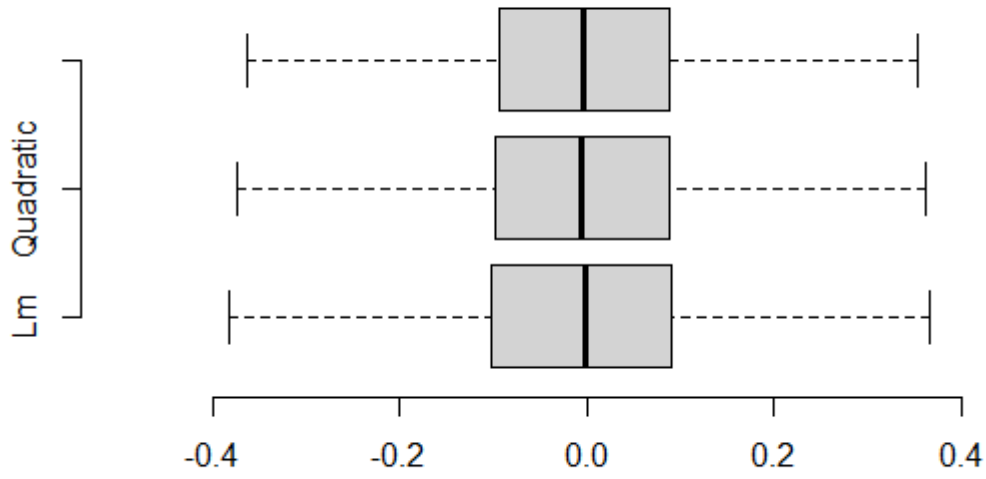
```
lmFood3.res = resid(lmFood3)
# We now plot the residual against the observed values of the variable FoodShr.
hist(lmFood3.res,prob=T,breaks=50,
     main = "Histogram of the residuals (Cubic model)",
     sub = "Red curve is the density (estimated)",
     xlab = "Food Share" , xlim=c(-1,1),
     col = SIAP.color, border = "white")
lines(density(lmFood2.res,na.rm = TRUE),lwd=2,col = "darkred")
```



Comparing the residuals distribution

Hide

```
res.all <- as.data.frame(cbind(lmFood.res, lmFood2.res, lmFood3.res))

boxplot(res.all, ylim = c(-0.5, 0.5),outline=FALSE,
        frame.plot = FALSE, horizontal = TRUE,
        names = c("Lm", "Quadratic", "Cubic") )
```



Trying to find the “right” model, that is the right order of the polynomial function is not so easy.

Nonparametric models

Nearest-Neighbors (k-NN)

We assume that $f(\cdot)$ is *smooth*

- No jumps: continuous
- No kinks: differentiable
- Smooth enough: usually twice differentiable

We want to estimate $f(\cdot)$. We talk about

- **nonparametric regression**, since there is no parameter to be estimated.
- **functional estimation**, since we estimate a function.

Note that we could find a function $f(\cdot)$ that goes through every observation: this is called *interpolation*. There is actually an infinity of such functions, these are defined uniquely only at observations points.

Nearest-neighbors method is close to *moving average*: we estimate $f(x)$ by averaging the y_i corresponding to observations x_i close to x. That's the idea of *smoothing*. Since the x_i are ordered from smallest to largest. We define the estimate of $f(x_i)$ as

$$\widehat{f}(x_i) = \frac{1}{k} \sum_{j \in\, k-nearest\ neighbours\ of\ x_i} y_j$$

k is the number of neighbors of x_i taken into account in estimation. This method is called *k-nearest neighbors* (K-NN for short).

Note: Our estimator should be defined at any point x, even if x is not an observation, so

$$\widehat{f}(x) = \widehat{f}(x_i)$$

where x_i is the closest point to x. So we obtain a step function or *piecewise constant* function.

Playing with k

Here we have 1109 observations in our dataset. We can choose different values for k and see how this affect our estimation of $f(\cdot)$:

> Experiment with different number of nearest-neighbors. What do you get for a small number k? What happens when you increase k? Use the online Shiny application to play with k-nn (https://xtophedataviz.shinyapps.io/KnnExplore/)

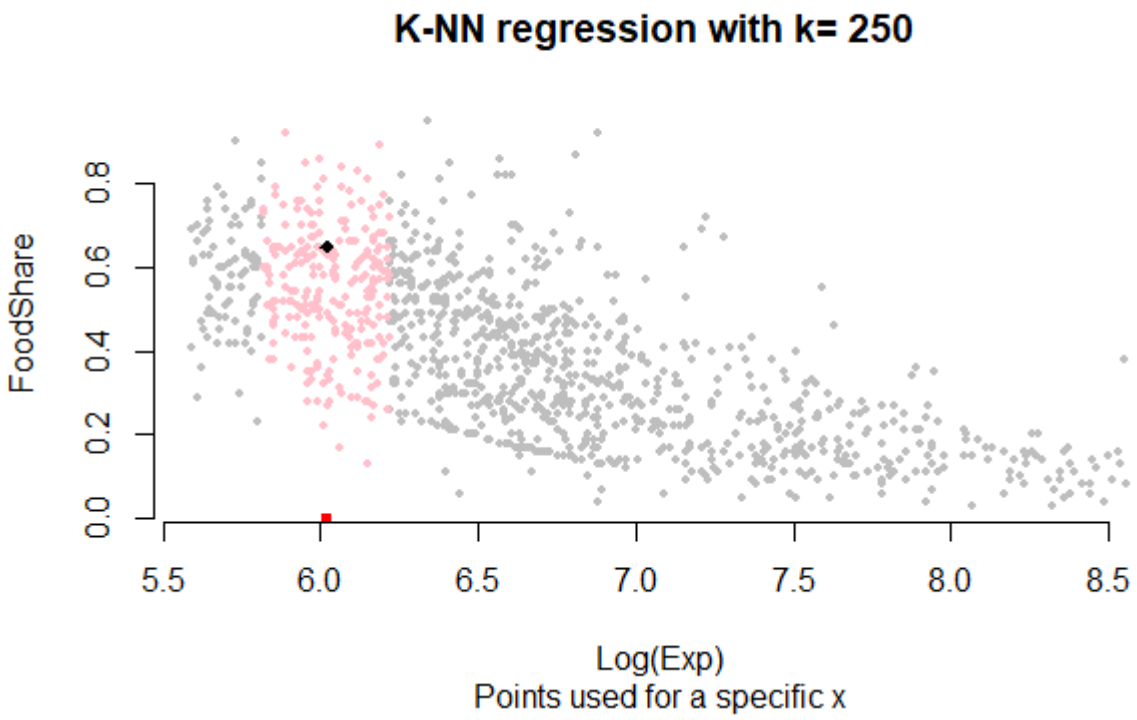
Hide

```
library(caret)

# Change the value of k here!!
k.choice = 250
```

k = 250 (*try with k = 10, 50, 100, 400, or 1000*)

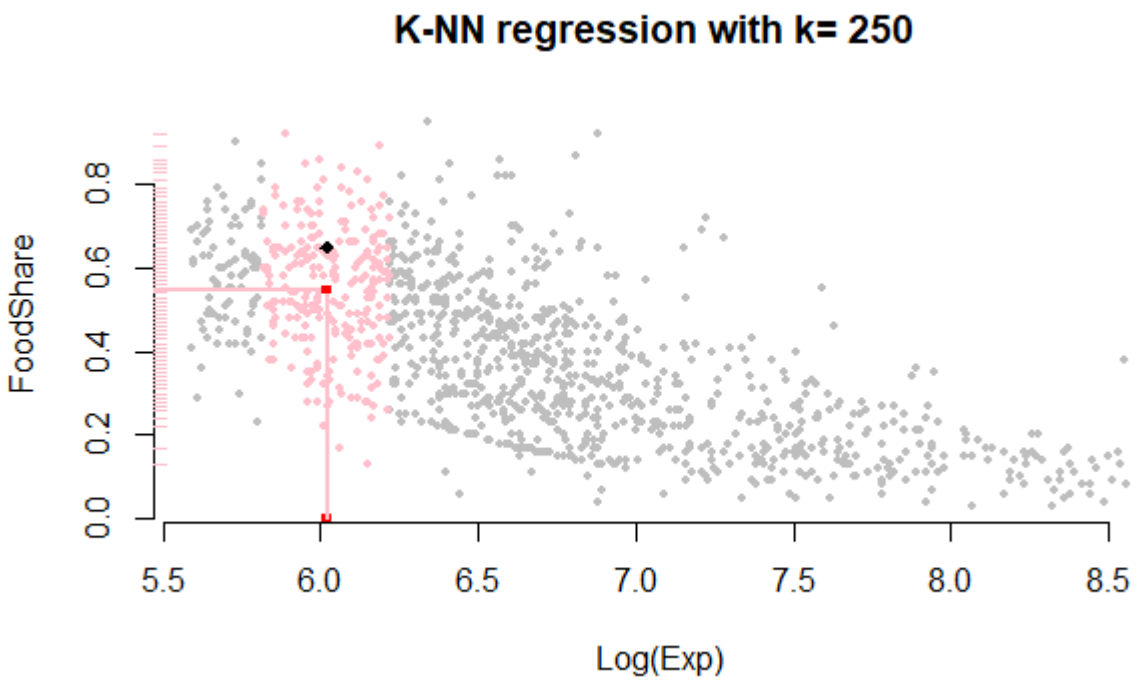
Hide

```
# Scatter plot
plot(ltexp,FoodShr,type="n",
     main= paste("K-NN regression with k=", k.choice,""),
     sub = "Points used for a specific x",
     xlab="Log(Exp)", ylab = "FoodShare",
    pch=19, cex = 0.5,col = "grey", frame.plot = FALSE )
points(ltexp,FoodShr,
    pch=19, cex = 0.5,col = "grey" )

# for a specific x, highlight the points included in computation
my.index <- 200  #  <-- value can be changed here
my.x <- ltexp[my.index]
my.y <- FoodShr[my.index]

library(tidyverse)
# computing x's nearest neighbors
df <- as.data.frame(cbind(ltexp, FoodShr))
df <- df %>%
  mutate( dist = abs(ltexp - my.x) ) %>%
  arrange(dist) %>%
  slice(1:k.choice)

points(df$ltexp,df$FoodShr,
    pch=19, cex = 0.6,col = "pink" )
# Original values
points(my.x,0,
    pch=15, cex = 0.6,col = "red" )

points(my.x,my.y,
    pch=18, cex = 0.9,col = "black" )
```



Hide

```
plot(ltexp,FoodShr,type="n",
     main= paste("K-NN regression with k=", k.choice,""),
     xlab="Log(Exp)", ylab = "FoodShare",
    pch=19, cex = 0.5,col = "grey", frame.plot = FALSE )
points(ltexp,FoodShr,
    pch=19, cex = 0.5,col = "grey" )


# Plotting x's nearest neighbors
points(df$ltexp,df$FoodShr,
    pch=19, cex = 0.6,col = "pink" )

# Computing estimation of Y using x's nearest neighbors
my.y.hat <- mean(df$FoodShr)
segments(0, my.y.hat, my.x, my.y.hat,
         lw = 2,
         col= 'pink')

# Original values
points(my.x,0,
    pch=15, cex = 0.6,col = "red" )

points(my.x,my.y,
    pch=18, cex = 0.9,col = "black" )

# Some illustration on the graphic
segments( my.x, 0, my.x, my.y.hat,
         lw = 2,
         col= 'pink')
points(my.x,my.y.hat,
    pch=15, cex = 0.6,col = "red" )
rug(df$FoodShr, side=2, col = "pink")

points(0,my.y.hat,
    pch=15, cex = 0.6,col = "red" )
```

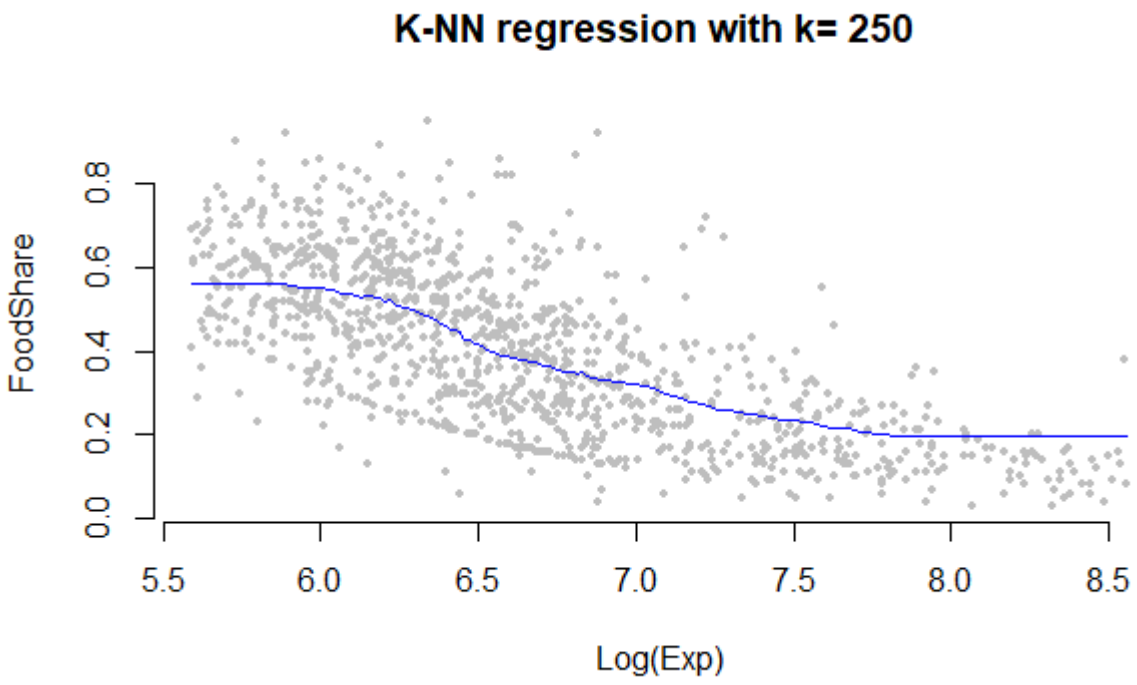


Hide

```
#Estimating Food Shares using k-NN (CARET package)
knn.est <- knnreg(FoodShr~ltexp, data = MyData, k = k.choice)
```

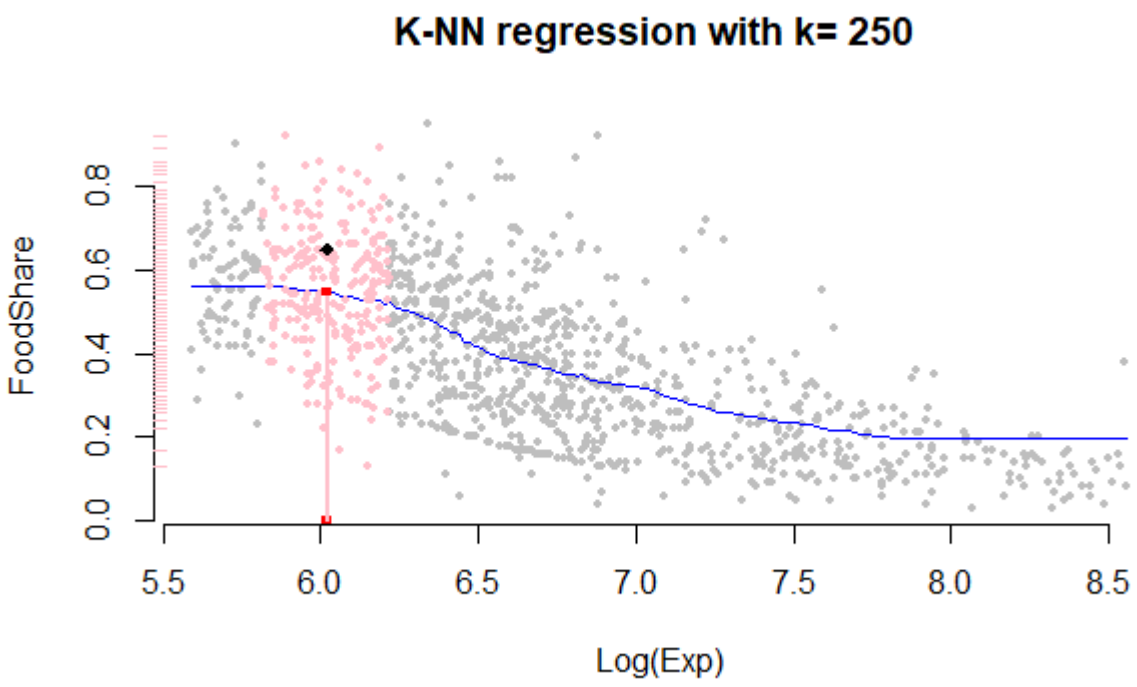
Hide

```
plot(ltexp,FoodShr,type="n",
     main= paste("K-NN regression with k=", k.choice,""),
     xlab="Log(Exp)", ylab = "FoodShare",
    pch=19, cex = 0.5,col = "grey", frame.plot = FALSE )
points(ltexp,FoodShr,
    pch=19, cex = 0.5,col = "grey" )

# Defining the sequence of 200 points where we will estimate the k-NN line
newx <- seq(from=min(ltexp),to=max(ltexp),
            length.out = 200)
# Estimating the k-NN regression line
newy <-predict(knn.est, data.frame(ltexp = newx))
# plotting the k-NN regression line
lines(newx, newy, col= "blue")
```



Hide

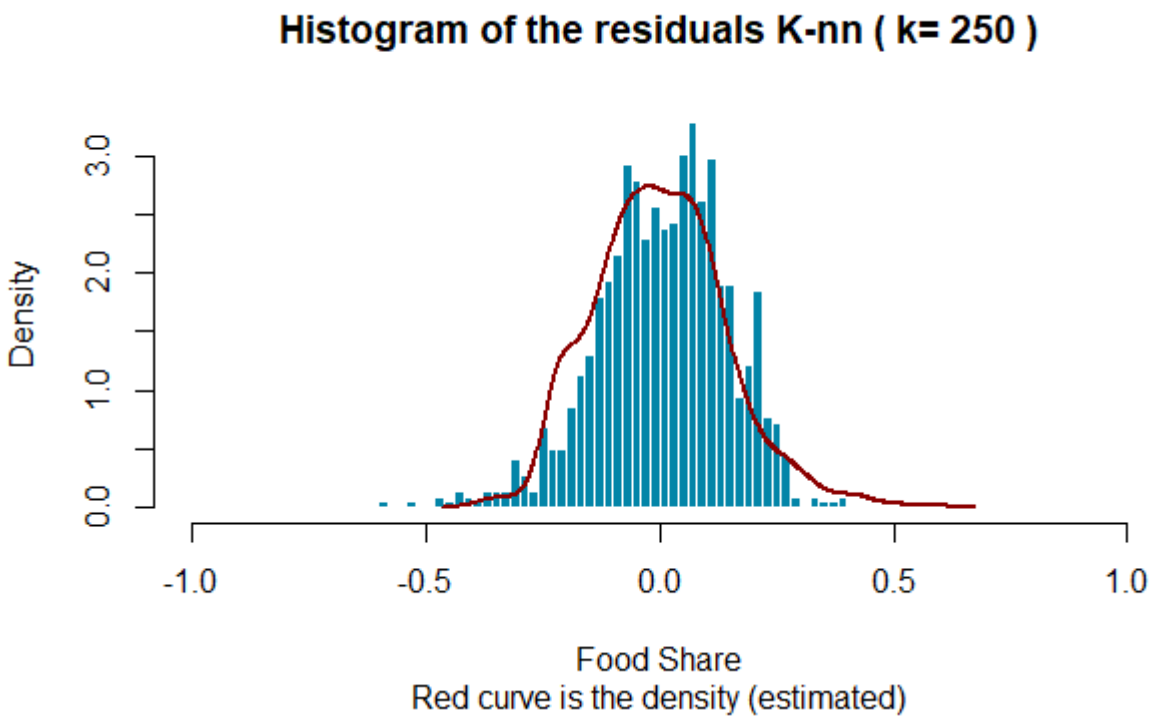
```
plot(ltexp,FoodShr,type="n",
     main= paste("K-NN regression with k=", k.choice,""),
     xlab="Log(Exp)", ylab = "FoodShare",
    pch=19, cex = 0.5,col = "grey", frame.plot = FALSE )
points(ltexp,FoodShr,
    pch=19, cex = 0.5,col = "grey" )

# Defining the sequence of 200 points where we will estimate the k-NN line
newx <- seq(from=min(ltexp),to=max(ltexp),
            length.out = 200)
# Estimating the k-NN regression line
newy <-predict(knn.est, data.frame(ltexp = newx))
# plotting the k-NN regression line
lines(newx, newy, col= "blue")


# Plotting x's nearest neighbors
points(df$ltexp,df$FoodShr,
    pch=19, cex = 0.6,col = "pink" )

# Original values
points(my.x,0,
    pch=15, cex = 0.6,col = "red" )

points(my.x,my.y,
    pch=18, cex = 0.9,col = "black" )

# Some illustration on the graphic
segments( my.x, 0, my.x, my.y.hat,
         lw = 2,
         col= 'pink')
points(my.x,my.y.hat,
    pch=15, cex = 0.6,col = "red" )
rug(df$FoodShr, side=2, col = "pink")

points(0,my.y.hat,
    pch=15, cex = 0.6,col = "red" )
```



Residuals distribution

Hide

```
# Computing the predictions for the observed x_i
yhat <-predict(knn.est, data.frame(ltexp))
knn.250.res <- yhat - FoodShr

# We now plot the residual against the observed values of the variable FoodShr.
hist(knn.250.res,prob=T,breaks=50,
     main = paste("Histogram of the residuals K-nn ( k=",k.choice,")") ,
     sub = "Red curve is the density (estimated)",
     xlab = "Food Share" , xlim=c(-1,1),
     col = SIAP.color, border = "white")
lines(density(lmFood2.res,na.rm = TRUE),lwd=2,col = "darkred")
```



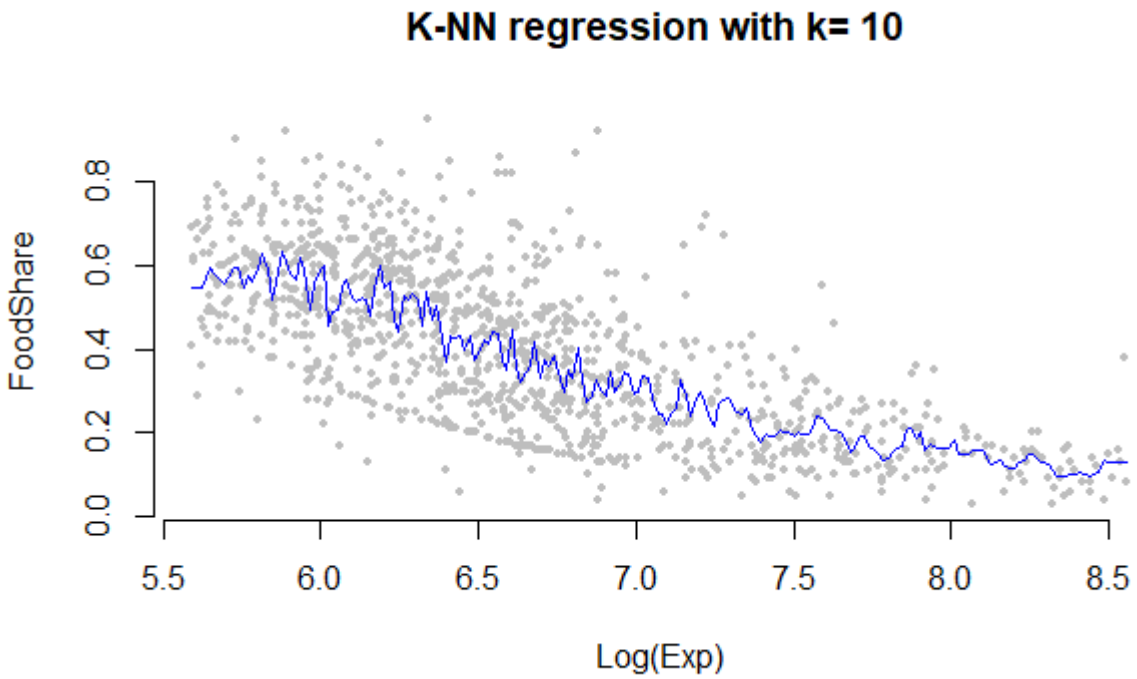
k = 10

Hide

```
k.choice = 10
knn.est <- knnreg(FoodShr~ltexp, data = MyData, k = k.choice)
```

Hide

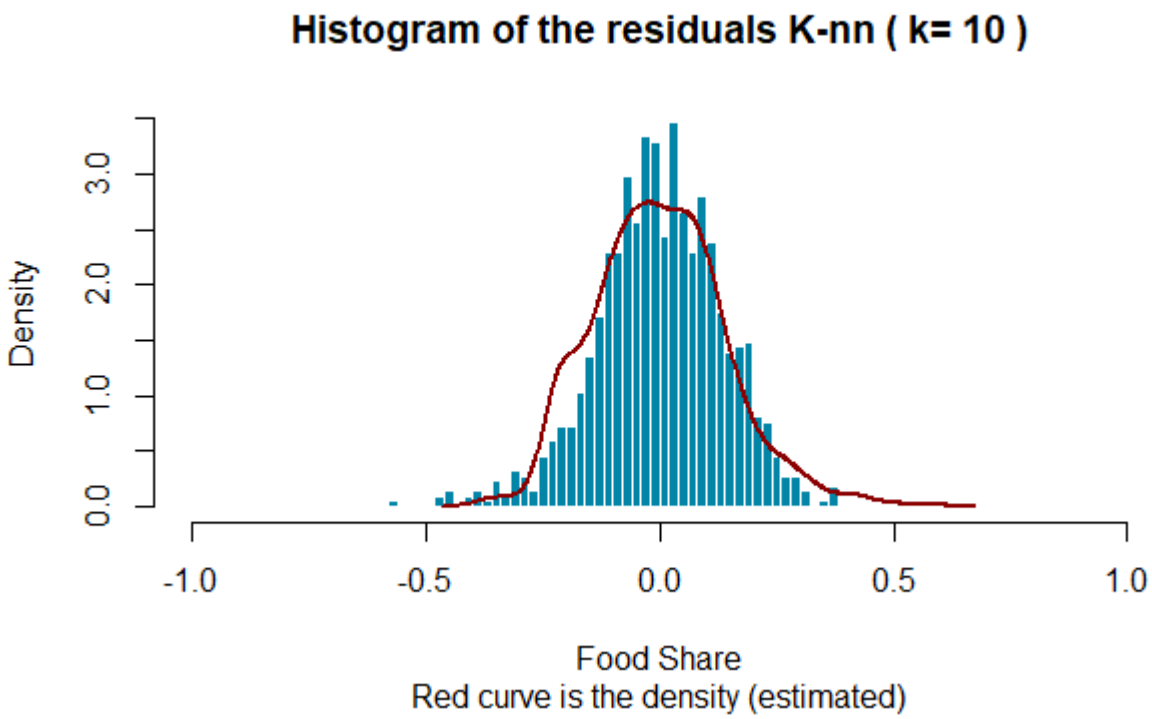
```
plot(ltexp,FoodShr,type="n",
     main= paste("K-NN regression with k=", k.choice,""),
     xlab="Log(Exp)", ylab = "FoodShare",
    pch=19, cex = 0.5,col = "grey", frame.plot = FALSE )
points(ltexp,FoodShr,
    pch=19, cex = 0.5,col = "grey" )

# Defining the sequence of 200 points where we will estimate the k-NN line
newx <- seq(from=min(ltexp),to=max(ltexp),
            length.out = 200)
# Estimating the k-NN regression line
newy <-predict(knn.est, data.frame(ltexp = newx))
# plotting the k-NN regression line
lines(newx, newy, col= "blue")
```



Residuals distribution

Hide

```
# Computing the predictions for the observed x_i
yhat <-predict(knn.est, data.frame(ltexp))
knn.10.res <- yhat - FoodShr

# We now plot the residual against the observed values of the variable FoodShr.
hist(knn.10.res,prob=T,breaks=50,
     main = paste("Histogram of the residuals K-nn ( k=",k.choice,")") ,
     sub = "Red curve is the density (estimated)",
     xlab = "Food Share" , xlim=c(-1,1),
     col = SIAP.color, border = "white")
lines(density(lmFood2.res,na.rm = TRUE),lwd=2,col = "darkred")
```



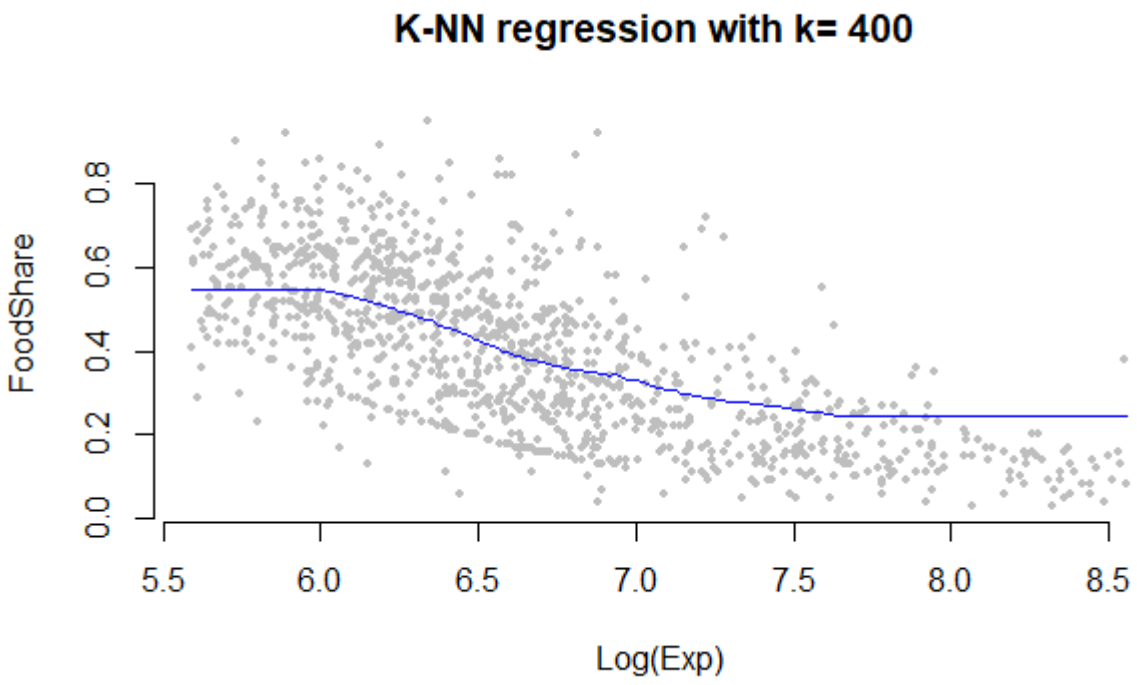
k = 400

Hide

```
k.choice = 400
knn.est <- knnreg(FoodShr~ltexp, data = MyData, k = k.choice)
```

Hide

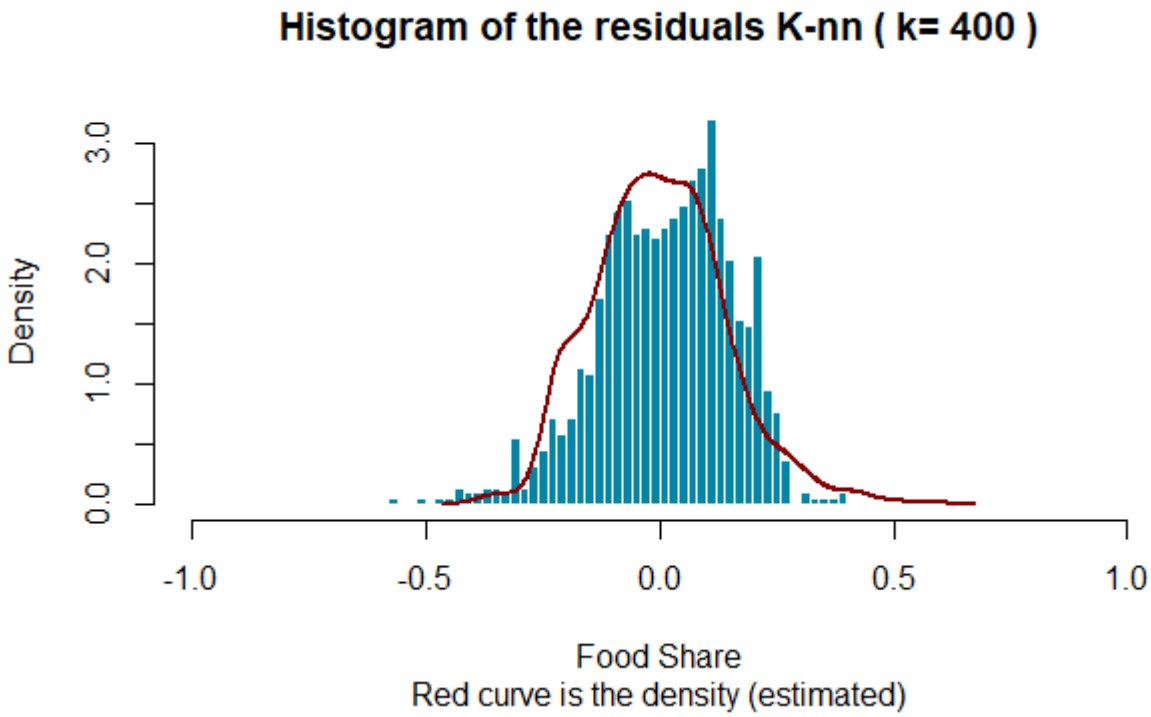
```
plot(ltexp,FoodShr,type="n",
     main= paste("K-NN regression with k=", k.choice,""),
     xlab="Log(Exp)", ylab = "FoodShare",
    pch=19, cex = 0.5,col = "grey", frame.plot = FALSE )
points(ltexp,FoodShr,
    pch=19, cex = 0.5,col = "grey" )

# Defining the sequence of 200 points where we will estimate the k-NN line
newx <- seq(from=min(ltexp),to=max(ltexp),
            length.out = 200)
# Estimating the k-NN regression line
newy <-predict(knn.est, data.frame(ltexp = newx))
# plotting the k-NN regression line
lines(newx, newy, col= "blue")
```



Residuals distribution

Hide

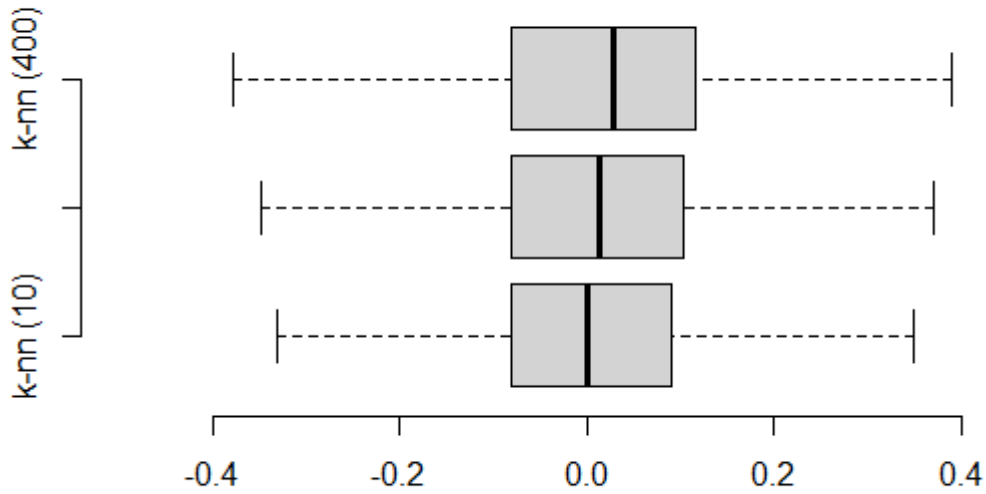
```
# Computing the predictions for the observed x_i
yhat <-predict(knn.est, data.frame(ltexp))
knn.400.res <- yhat - FoodShr

# We now plot the residual against the observed values of the variable FoodShr.
hist(knn.400.res,prob=T,breaks=50,
     main = paste("Histogram of the residuals K-nn ( k=",k.choice,")") ,
     sub = "Red curve is the density (estimated)",
     xlab = "Food Share" , xlim=c(-1,1),
     col = SIAP.color, border = "white")
lines(density(lmFood2.res,na.rm = TRUE),lwd=2,col = "darkred")
```



Comparing the residuals distribution

Hide

```
res.all <- as.data.frame(cbind(knn.10.res, knn.250.res, knn.400.res))
boxplot(res.all, ylim = c(-0.5, 0.5),outline=FALSE,
        frame.plot = FALSE, horizontal = TRUE,
        names = c("k-nn (10)", "k-nn (250)", "k-nn (400)") )
```



An important Criterion: Mean Squared Error

The MSE is a **theoretical**[3] measure of the precision for any estimator, it is defined as the Expectation of the distance between the estimated $\widehat{f}(x_i)$ and true (unknown) value $f(x_i)$ for a particular observation i:

$$E\left[\widehat{f}(x_i) - f(x_i)\right]^2$$

It can be shown that the MSE can be decomposed into 2 terms:

$$MSE = E\left[\left(\widehat{f}(x_i) - f(x_i)\right)^2\right] = \left\{E\left[\widehat{f}(x_i) - f(x_i)\right]\right\}^2 + Var\left[\widehat{f}(x_i)\right]$$

The MSE is a measure of the precision for any estimator, and we always have

$$MSE = Bias^2 + Variance$$

Since the *true* function $f(\cdot)$ is unknown, we cannot estimate the first term that depends on the *true* function $f(\cdot)$ (more precisely on its second derivative) and since we cannot assume we obtain an unbiased estimator, the MSE for our nonparametric k-NN estimator is:

$$MSE_{K-NN} = E\Big[\widehat{f}(x_i) - f(x_i)\Big]^2 \approx \left\{f''(x_i)\frac{1}{24}\left(\frac{k}{n}\right)^2\right\}^2 + \frac{1}{k}\sigma_\varepsilon^2 \,.$$

Bias-Variance Trade-Off

To minimize the MSE, we should balance squared bias and variance. from this expression we can learn that:

- bias increases when k increases
- variance decreases when k increases

and also that:

- bias decreases when n increases

Then , we should choose k such that the squared bias is of the same order than the variance. One can show that the **optimal** k is: $k^* \propto n^{4/5}$

Under/Over Smoothing

- *Undersmoothing* occurs is when we use too small a k.

Think about $k = 1$: we have *interpolation*. More generally undersmoothing occurs when we obtain a very **wiggly** curve: bias is small (we are near each observation), but variance is large (curve is wiggly).

- *Oversmoothing* is when we use too large a k.

Think about $k = n$: the estimator $\widehat{f}(x) = \bar{y}$ for any x! More generally oversmothing occurs when we obtain too flat a curve: variance is small, but bias is large.

> In practice, it may be tricky to determine the right number of neighbors that is the right amount of smoothing!

Wrap-up

- ML is about estimating an unknown function $f(\cdot)$
- To estimate regression models, we have to solve an **optimization problem**
- With “big data”, we can go over a simple linear model: e.g. polynomial models or nonparametric mode such as *k-NN* regression.
- There is a *bias-variance trade-off*: a more complex model allows to estimate the regression more accurately, but introduces more variability in estimation.
- Theory tells us exactly how to balance squared bias and variance but it does not tell us how to choose the model in practice!

---

1. *This document uses teaching materials developped by Pascal Lavergne* (*Toulouse School of Economics*)↩

2. Data from A. Yatchew *Semiparametric Regression for the Applied Econometrician* https://www.economics.utoronto.ca/yatchew/ (https://www.economics.utoronto.ca/yatchew/) ↩

3. One can estimate the MSE *on the sample* when y_i is observed and then:

$$MSE = \frac{1}{n} \sum_{i=1}^{n} \left(y_i - \widehat{f}(x_i)\right)^2$$

↩