Advanced Statistical Modelling.

Part 2. Nonparametric Modelling

Practice with R

Pedro Delicado

Departament d'Estadística i Investigació Operativa

Universitat Politècnica de Catalunya

October 16, 2016

Chapter 1

Local polynomial regression

1.1 Write your own local linear regression function

Use the following R script to generate artificial data.

```
sigma.e <- .5
x <- seq(0,1,by=.01)
mx <- sin(2*pi*x)
dmx <- cos(2*pi*x)*2*pi
y <- mx + rnorm(length(x),0,sigma.e)
plot(x,y)
lines(x,mx)

tg <- seq(0,1,length=51)
```

Now write a script in R that fits a local linear regression of each element of vector `tg`. The pseudo-code should be like that:

1. Decide the bandwidth value: $h = 0.1$, for instance.
2. For each t in `tg` do: (it can be useful to define `x.t <- x - tg[j]`)
   (a) Compute the weight of each x_i in `x` as $K((x_i - t)/h)$, where K is a density function symmetric around 0 (use, for instance, `dnorm`).
   (b) Fit the weighted linear model $y \sim (x - t)$ with the weights computed before.
   (c) Take the intercept of the preceding fitted linear model as estimation $\hat{m}(t)$ of $m(t)$.
3. Add to the last graphic the estimated function $\hat{m}(t)$, for $t \in$`tg`.

1.2 Aircraft Data

We are using *Aircraft data*, from the R library `sm`. These data record six characteristics of aircraft designs which appeared during the twentieth century.

```
Yr:     year of first manufacture
Period: a code to indicate one of three broad time periods
Power:  total engine power (kW)
Span:   wing span (m)
Length: length (m)
Weight: maximum take-off weight (kg)
Speed:  maximum speed (km/h)
Range:  range (km)
```

We transform data taken logs (except `Yr` and `Period`): `lgPower`, . . . , `lgRange`.

Go to `R` and charge the library `sm`:

```
library(sm)
```

Now upload the data:

```
data(aircraft)
help(aircraft)
attach(aircraft)
lgPower <- log(Power)
lgSpan <- log(Span)
lgLength <- log(Length)
lgWeight <- log(Weight)
lgSpeed <- log(Speed)
lgRange <- log(Range)
```

1. Draw the scatter plots of `lgPower`, `lgSpan`, `lgLength`, `lgWeigth`, `lgSpeed`, `lgRange` against variable `Yr`.

2. Use your own local linear regression function to estimate the regression function corresponding to each case. Draw the estimated regression functions over each of the previous scatterplots. Special attention must be given to bandwidth h choices (h can be different for different pairs of variables). Explain how did you choose the h values.

3. Do you think that the relationships between these variables are all linear?

4. Do you think that they are homoscedastic or heteroscedastic?

1.3 Local polynomial regression for Aircraft data

1. **Going into local polynomial regression.**
   Function `lpr_visual` (see file `lpr_visual.R`) performs local polynomial fitting and provides a visual inside of the method.
   Fit the nonparametric regression of `lgSpeed` against `Yr`. Use different values of parameters `h` and `p`. Try also to change vector `tg`.

2. Function `locpolreg` (see file `locpolreg.R`) performs local polynomial fitting. It also can be used to estimate derivatives.
   Fit the nonparametric regression of `lgSpeed` against `Yr`. Use different values of parameters `h`, `p` and `r`. Do not specify parameter `tg`.
   When was faster the increasing of speed of manufactured aircrafts?
   (You can try with $h = 15$, $p = 1$, $r = 1$).

1.4 Local polynomial estimation in R

These are some of the functions and packages that R provides for doing nonparametric regression, most of them by local polynomial regression. The functions listed below are available in package **stats** (one of the R default packages).

`loess`: It fits simple and multiple nonparametric-regression models by local polynomial regression. Fits local polynomials of any degree. It uses variable bandwidth (similar to `supsmu`). It is possible to specify the degree of smoothing using the *equivalent number of parameters*. It is not possible to estimate derivatives of the regression function.

`scatter.smooth`: It adds a `loess` regression smoother to a scatter plot.

`supsmu`: It performs local linear fitting with variable bandwidth: for each t, a bandwidth h_t is chosen in order to have a proportion (*span*) s of the data are in $[t \pm h_t]$. The value of s can be chosen by cross-validation.

`smooth`: Computes moving medians with refinements defined by Tukey.

`smooth.spline`: Scatterplot smoothing based on splines.

Package KernSmooth: It is the package of the book by Wand and Jones (1995). Look at the functions `locpoly` (local polynomial fitting). It is possible to choose the polynomial degree and the order of the derivative

to be estimated. It is possible to choose automatically the bandwidth by the plug-in method (function `dpill`). See also package **np**, which implements kernel smoothing methods for mixed data types, with emphasis in Econometric models.

Package sm: It is the accompanying package of the book *Smoothing methods* (Bowman and Azzalini, 1997). The function `sm.regression` estimates nonparametric regression by Nadaraya-Watson and local linear methods. It does not compute derivatives. The bandwidth can be chosen by several methods using the function `h.select`. It also covers nonparametric density estimation.

Package locfit: It performs local-polynomial regression (and also density estimation) as explained in the book of Loader (1999) *Local Regression and Likelihood*, Springer.

Package mgcv: It covers Generalized Additive Models (GAMs), a nonparametric version of GLMs, using splines as basic fitting tools. Other packages with similar goals are **gam** and **gss**.

Do the following task:

1. **Local polynomial regression with** `locpoly`.
   Fit the regression function of `lgSpeed` against `Yr` using `locpoly`. Use several values for parameters `bandwidth`, `degree` and `drv` (derivative to be estimated).
   When was faster the increasing of speed of manufactured aircrafts?
   (You can try with *bandwidth* $= 7$, *degree* $= 1$ and *drv* $= 1$, or with *bandwidth* $= 10$, *degree* $= 2$ and *drv* $= 1$).

1.5 Estimating the conditional variance

Consider the heteroscedastic regression model

$$Y = m(x) + \sigma(x)\varepsilon = m(x) + \epsilon,$$

where $E(\varepsilon) = 0$, $V(\varepsilon) = 1$ and $\sigma^2(x)$ is an unknown function that gives the conditional variance of Y given that the explanatory variable is equal to x.

Let us define $Z = \log((Y - m(x))^2) = \log \epsilon^2$ and $\delta = \log(\varepsilon^2)$. Then

$$Z = \log \sigma^2(x) + \delta,$$

and $\delta = \log \varepsilon^2$ is a random variable with expected value close to 0 (observe that $E(\log \varepsilon^2) \approx \log E(\varepsilon^2) = \log V(\varepsilon) = \log 1 = 0$) taking the role of *noise* in

the regression of Z against x (that is, Z is the response variable and x is the predicting variable).

Given that the values of ϵ_i^2 are not observable, a way to estimate the function $\sigma^2(x)$ is as follows:

1. Fit a nonparametric regression to data (x_i, y_i) and save the estimated values $\hat{m}(x_i)$.

2. Transform the estimated residuals $\hat{\epsilon}_i = y_i - \hat{m}(x_i)$:

$$z_i = \log \epsilon_i^2 = \log((y_i - \hat{m}(x_i))^2).$$

3. Fit a nonparametric regression to data (x_i, z_i) and call the estimated function $\hat{q}(x)$. Observe that $\hat{q}(x)$ is an estimate of $\log \sigma^2(x)$.

4. Estimate $\sigma^2(x)$ by

$$\hat{\sigma}^2(x) = e^{\hat{q}(x)}.$$

Apply this procedure to estimate the conditional variance of `lgWeigth` (variable Y) given `Yr` (variable x). Draw a graphic of $\hat{\epsilon}_i^2$ against x_i and superimpose the estimated function $\hat{\sigma}^2(x)$. Lastly draw the function $\hat{m}(x)$ and superimpose the bands $\hat{m}(x) \pm 1.96\hat{\sigma}(x)$.

Chapter 2

Kernels. Bias-variance trade-off. Variance estimation

2.1 Kernels in R

The R function `density` performs nonparametric kernel density estimation. We can use this function to plot different kernel functions.

```
(kernels <- eval(formals(density.default)$kernel))

## show the kernels in the R parametrization:
## re-scaled versions to have variance equal to 1
plot (density(0, bw = 1), xlab = "",
      main = "R's density() kernels with bw = 1")
for(i in 2:length(kernels))
   lines(density(0, bw = 1, kernel =  kernels[i]), col = i)
legend(1.5,.4, legend = kernels, col = seq(kernels),
       lty = 1, cex = .8, y.intersp = 1)
```

The function `kernel` has been defined in the file `locpolreg.R`. It admits 3 types of kernels: `"normal"` (Gaussian, default), `"epan"` (Epanechnikov) or `"rs.epan"` (re-scaled Epanechnikov).

```
plot(kernel,xlim=c(-3,3),ylim=c(0,.8))
x<-seq(-3,3,length=201)
lines(x,kernel(x,type="epan"),col=4)
lines(x,kernel(x,type="rs.epan"),col=2)
legend("topleft",c("normal","epan","rs.epan"),lwd=1, col=c(1,2,4))
```

2.2 Bias-variance trade-off

The R script `Bias_Var_h.r` illustrates the trade-off between bias and variance of non-parametric regression estimators as a function of the smoothing parameter.

2.3 Linear smoothers

The R script `illustr_eq_num_param.R` illustrates several characteristics of linear smoothers, using the case of local polynomial regression: equivalent number of parameters; choice of the degree of the local polynomial; estimation of the error variance σ^2 when using linear smoothers.

The R script `S_matrix.R` shows how the rows of a smoothing matrix S look like.

2.4 Alternative residal variance estimation

We have seen several estimators of $\sigma^2 = V(\varepsilon_i)$, all of them require a previous estimation of the regression function. Here you have two alternative estimators of σ^2 not requiring a pilot regression estimation.

Proposal of Rice (1984). Consider the nonparametric regression model $y_i = m(x_i) + \varepsilon_i$, $i = 1, \ldots, n$, where the data are sorted according to x_i in increasing order. Then

$$y_i - y_{i-1} = m(x_i) - m(x_{i-1}) + (\varepsilon_i - \varepsilon_{i-1}).$$

Therefore,

$$V(y_i - y_{i-1}) = E\left[(y_i - y_{i-1})^2\right] - (m(x_i) - m(x_{i-1}))^2 = V(\varepsilon_i - \varepsilon_{i-1}) = 2\sigma^2.$$

If function m is smooth enough and x_i and x_{i-1} are close, then $(m(x_i) - m(x_{i-1}))^2$ is negligible compared with $E\left[(y_i - y_{i-1})^2\right]$, and it follows that

$$E\left[(y_i - y_{i-1})^2\right] \approx 2\sigma^2,$$

implying that

$$\hat{\sigma}^2 = \frac{1}{2(n-1)} \sum_{i=2}^{n} (y_i - y_{i-1})^2$$

is an approximately unbiased estimator of σ^2.

Proposal of Gasser, Sroka, and Jennen-Steinmetz (1986). It is based on linear interpolation between the previous and following observation, (x_{i-1}, y_{i-1}) and (x_{i+1}, y_{i+1}), of any data (x_i, y_i), when they are ordered according to the increasing order of x_i. Let

$$\hat{y}_i = \frac{x_{i+1} - x_i}{x_{i+1} - x_{i-1}} y_{i-1} + \frac{x_i - x_{i-1}}{x_{i+1} - x_{i-1}} y_{i+1} = a_i y_{i-1} + b_i y_{i+1}$$

be the linear interpolation of (x_{i-1}, y_{i-1}) and (x_{i+1}, y_{i+1}) evaluated at $x = x_i$.

Define

$$\tilde{\varepsilon}_i = \hat{y}_i - y_i = a_i y_{i-1} + b_i y_{i+1} - y_i.$$

Its expected value is

$$E(\tilde{\varepsilon}_i) = a_i m(x_{i-1}) + b_i m(x_{i+1}) - m(x_i) = \hat{m}_l(x_i) - m(x_i) \approx 0,$$

where $\hat{m}_l(x_i)$ is the linear interpolation of $(x_{i-1}, m(x_{i-1}))$ and $(x_{i+1}, m(x_{i+1}))$ evaluated at $x = x_i$, that is approximately equal to $m(x_i)$ if function m is smooth enough and x_i and x_{i-1} are close enough. Thus,

$$E(\tilde{\varepsilon}_i^2) \approx V(\tilde{\varepsilon}_i) = (a_i^2 + b_i^2 + 1)\sigma^2,$$

implying that

$$\hat{\sigma}^2 = \frac{1}{n-2} \sum_{i=2}^{n-1} \frac{1}{a_i^2 + b_i^2 + 1} \tilde{\varepsilon}_i^2$$

is an approximately unbiased estimator of σ^2.

1. Write an R function implementing these two residual variance estimators:

   Input: Vectors x and y.

   Output: The estimation of σ^2 using both methods.

2. For the Boston Housing dataset, in the nonparametric regression of `ROOM` against `LSTAT`, estimate the residual variance with this R function and compare the squared root of your estimation with the estimation of σ provided by the functions `loess` and `sm.regression` (package **sm**).

Remark 1: Note that the first task this function must do is to check whether vector x is sorted or not in ascending order. If not, then sort x inside the function.

Remark 2: When working with real datasets (as the Boston Housing dataset, for instance) it is not unusual to find repeated values at the explanatory variable x. You must prevent this case in your function when implementing the variance estimator proposed by Gasser, Sroka, and Jennen-Steinmetz (1986). In case that 3 or more observed data (x_i, y_i) have the same explanatory variable value the denominator in the definitions of a_i and b_i is equal to 0. Then I propose you to change the value $x_{i+1} - x_{i-1}$ by

$$x_{u(i)} - x_{l(i)}$$

where $x_{u(i)}$ is the lowest value of x_j, among those being greater than x_i, and $x_{l(i)}$ is the largest value of x_j, among those being lower than x_i.

Chapter 3

Bandwidth choice

3.1 Country development data

We will be working with the file `countries.txt` containing information on development indicators measured in 132 countries (Source: World Bank, 1992). We will focus on the following variables:

`life.exp`	Life expectancy at birth.
`agricul`	% of agriculture contribution to the GDP (Gross Domestic Product).
`inf.mort`	Infant mortality rate: The annual number of deaths of infants under one year of age per 1,000 live births in the same year.
`life.exp.f`	Life expectancy at birth for females.
`life.exp.m`	Life expectancy at birth for males.
`le.fm`	Difference `life.exp.f` minus `life.exp.m`.

Read data from `countries.txt`:

```
countries<-read.table(file="countries.txt",head=T,row.names=2,dec=",")
head(countries)
summary(countries)
attach(countries)
# Matrix of scatterplots
plot(countries[,2:7])

# Using sm.regression, from library sm. An example
library(sm)
plot(x=log(agricul+1),y=life.exp)
sm.regression(x=log(agricul+1),y=life.exp,h=.3)
```

```
# Choosing the bandwidth by different criteria
h.cv.1 <- hcv(x=log(agricul+1),y=life.exp,hstart=.05)
h.cv.2 <- h.select(x=log(agricul+1),y=life.exp,method="cv")
h.df <- h.select(x=log(agricul+1),y=life.exp,method="df")
h.aicc <- h.select(x=log(agricul+1),y=life.exp,method="aicc")

# Using hdpill from library KernSmooth
library(KernSmooth)
h.dpill <- pill(x=log(agricul+1),y=life.exp,
               gridsize=101,range.x=range(log(agricul+1)))

print(c(h.cv.1,h.cv.2,h.df,h.aicc,h.dpill))
sm.regression(x=log(agricul+1),y=life.exp,h=h.cv.2,panel=TRUE)
```

3.2 Write your own bandwidth choice function

1. Write an R function implementing the following bandwidth choice criteria: leave-one-out ($\text{PMSE}_{\text{CV}}(h)$), 5-fold cross-validation ($\text{PMSE}_{5-\text{CV}}(h)$), 10-fold cross-validation ($\text{PMSE}_{10-\text{CV}}(h)$), generalized cross-validation ($\text{PMSE}_{\text{GCV}}(h)$). You can use function `locpolreg` to obtain the smoothing matrix S.

   Input: Vectors x and y; a vector `h.v` of candidate values for h.

   Output: For each element h in `h.v`, the value of $\text{PMSE}_{\text{CV}}(h)$, $\text{PMSE}_{5-\text{CV}}(h)$, $\text{PMSE}_{10-\text{CV}}(h)$, $\text{PMSE}_{\text{GCV}}(h)$. Additionally you can plot these values against $\log(h)$ in the same graphic.

2. For the Boston Housing dataset, in the nonparametric regression of `ROOM` against `LSTAT`, use your preceding function to choose the bandwidth h by the four criteria. As vector of candidates values for h take

   ```
   exp( seq(from=log(.5), to = log(15), length=12)
   ```

   Compare your results with those provided by function `h.select` (package **sm**) and `dpill` (package **KernSmooth**).

Chapter 4

Generalized nonparametric regression model

4.1 Write your own local logistic regression function

Use the following R script to generate artificial data.

```
beta0=0; beta1=1; k=4; s=.5; n<-400
x<-sort(rnorm(n))
l.x <- beta0 + beta1*x
theta.x <- l.x + k*x*dnorm(x,m=0,sd=s)
m.x <- 1/(1+exp(-theta.x))
logit.x <- 1/(1+exp(-l.x))
y <- rbinom(n,1,m.x)

# grid
nt<-101; tg<-seq(-3.5,3.5,length=nt)
l.t <- beta0 + beta1*tg
theta.t <- l.t + k*tg*dnorm(tg,m=0,sd=s)
m.t <- 1/(1+exp(-theta.t))
logit.t <- 1/(1+exp(-l.t))

par(mfrow=c(1,2))
plot(tg,m.t,xlim=c(-3.5,3.5),ylim=c(0,1),type="l")
lines(tg,logit.t,col=2); points(x,y,col=4,pch=3)

plot(tg,theta.t,xlim=c(-3.5,3.5),type="l")
lines(tg,l.t,col=2)
```

Now write a script in R that fits a local logistic regression of each element of vector `tg`. The pseudo-code should be like that:

1. Decide the bandwidth value: $h = 0.35$, for instance.
2. For each t in `tg` do: (it can be useful to define `x.t <- x - tg[j]`)
   (a) Compute the weight of each x_i in `x` as $K((x_i - t)/h)$, where K is a density function symmetric around 0 (use, for instance, `dnorm`).
   (b) Fit the weighted generalized linear model $y \sim (x - t)$ with family "binomial" and the weights computed before.
   (c) Take the intercept coefficient b_0 of the preceding fitted generalized linear model and define the estimation $\hat{m}(t)$ of $m(t)$ as $\hat{m}(t) = 1/(1 + \exp(-b_0))$.
3. Add the estimated function $\hat{m}(t)$, for $t \in$`tg`, to the graphic $(t, m(t))$, $t \in$`tg`.

4.2 Illustrating the local logistic regression

See the R script `local.logistic.R` for an illustration of the local logistic regression.

4.3 Generalized nonparametric regression model

We are using package `sm`:
`library(sm); help(package=sm)`

Conisder the country development dataset described in Section 3.1.

1. **Scatterplot matrix of all variables.** `plot(countries[,2:7])`
2. **Classify countries according to** `agricul`**.**
   The median value of variable `agricul` is 16. Create a new binary variable `ind.agr` indicating for any country if variable `agricul` is lower than 16 (1) or not (0). This new variable is a development indicator.
3. **Classify countries according to** `life.exp`**.**
   The median value of variable `life.exp` is 68. Create a new binary variable `ind.le` indicating for any country if variable `life.exp` is greater than 68 (1) or not (0). This new variable is a development indicator.

4. **Local binary regression.**
   Use `sm.binomial` to fit the following local binary regression models:

   `ind.agr` as a function of `life.exp`,

   `ind.agr` as a function of `inf.mort`,

   `ind.le` as a function of `inf.mort`,

   `ind.le` as a function of `agricul`.

   In the same graphics, compare your results with those obtained using (i) a standard nonparametric regression fit (using `sm.regression`, for instance), and (ii) a parametric fitting of a logistic regression model (using `glm`).

5. **Local Poisson regression.**
   Variable `le.fm` always takes non-negative values, except for one country. Define the variable
   `le.fm.0 <- pmax(0,le.fm)`
   and plot its histogram:
   `hist(le.fm.0,br=40)`
   Use `sm.poisson` to fit the following local Poisson regression models:

   `le.fm.0` as a function of `inf.mort`

   `le.fm.0` as a function of `life.exp`

   In the same graphics, compare your results with those obtained using (i) a standard nonparametric regression fit (using `sm.regression`, for instance), and (ii) a parametric fitting of a Poisson Generalized Linear Model with (using `glm`).

4.4 Bandwidth choice when fitting local logistic regression

The R script `h.cv.sm.binomial.R` implements two leave-one-out cross-validation (CV) bandwidth choice methods in the local logistic regression. The first one is based on the minimization of the CV classification error. The second one maximize a CV version of the likelihood function. Function `sm.binomial`, from library `sm`, is used.

1. Modify the script `h.cv.sm.binomial.R` to obtain a bandwidth choice methods for the local Poisson regression, using function `sm.poisson`.

Chapter 5

Inference in the nonparametric regression model

5.1 Inference for country development data

Conisder the country development dataset described in Section 3.1.

1. **Variability bands.**
   Define the variable
   `lgagr <- log(agricul+1)`

   (a) Use `sm.regression` to fit the nonparametric regression of variable
   `life.exp` as a function of `lgagr`.
   You can choose the bandwidth using, for instance, cross-validation:
   `h=h.select(lgagr,life.exp,method=''cv'')`

   (b) Draw variability bands around the fitted function using `sm.regression` with parameter `display="se"`

2. **Testing the linear model.**
   Test the null hypothesis that the following regression models:

   `life.exp` as a function of `lgagr`

   `life.exp` as a function of `inf.mort`

   `le.fm` as a function of `life.exp`

   are linear, using `sm.regression` with parameter `model="linear"`.

   In order to know how the testing decision depends on bandwidth h, use the function `sig.trace`.

3. **Testing the logistic model.**
   In the binary regression model with response variable `ind.le` and explanatory variable `inf.mort` we want to test the null hypothesis stating that the logistic model is appropriate. Use the function `sm.binomial.bootstrap`.

```
aux <- sort(inf.mort,index.return = T)
I <- aux$ix

aux.sm <- sm.binomial(inf.mort[I],ind.le[I],h=10)
aux.glm <- glm(ind.le[I]~inf.mort[I],family="binomial")
lines(inf.mort[I],aux.glm$fitted.values,col=2)

sm.binomial.bootstrap(inf.mort[I],ind.le[I],h=10,disp="none")
lines(inf.mort[I],aux.glm$fitted.values,col=2,lwd=2)
lines(aux.sm$eval.points,aux.sm$estimate,col=1,lwd=2)
```

4. **Testing the Poisson GLM**
   In the Poisson regression model with response variable `le.fm.0` and explanatory variable `life.exp` we want to test the null hypothesis stating that the Poisson GLM is appropriate. Use the function `sm.poisson.bootstrap`.

5. **Comparing regression function in several subpopulations.**
   We are using `sm.ancova` with different values of parameter `model`: `"none", "equal", "parallel"`.

   (a) Compare the regression functions
   `le.fm` as a function of `life.exp`
   fitted in both subpopulations defined by the indicator variable `ind.agr`.

   (b) Compare the regression functions
   `le.fm` as a function of `inf.mort`
   fitted in both subpopulations defined by the indicator variable `ind.agr`.

   (c) Compare the regression functions
   `le.fm` as a function of `inf.mort`
   fitted in both subpopulations defined by the indicator variable `ind.le`.

5.2 Hirsutism dataset

Hirsutism is the excessive hairiness on women in those parts of the body where terminal hair does not normally occur or is minimal -for example, a beard or chest hair. It refers to a male pattern of body hair (androgenic hair) and it is therefore primarily of cosmetic and psychological concern. Hirsutism is a symptom rather than a disease and may be a sign of a more serious medical condition, especially if it develops well after puberty.

The amount and location of the hair is measured by a Ferriman-Gallwey score. The original method used 11 body areas to assess hair growth, but was decreased to 9 body areas in the modified method: Upper lip, Chin, Chest, Upper back, Lower back, Upper abdomen, Lower abdomen, Upper arms, Thighs, Forearms (deleted in the modified method) and Legs (deleted in the modified method). In the modified method, hair growth is rated from 0 (no growth of terminal hair) to 4 (extensive hair growth) in each of the nine locations. A patient's score may therefore range from a minimum score of 0 to a maximum score of 36.

A clinical trial was conducted to evaluate the effectiveness of an antiandrogen combined with an oral contraceptive in reducing hirsutism for 12 consecutive months. It is known that contraceptives have positive effects on reduction of hirsutism. The degree of hirsutism is measured by the modified Ferriman-Gallwey scale. Patients were randomized into 4 treatment levels: levels 0 (only contraceptive), 1, 2, and 3 of the antiandrogen in the study (always in combination with the contraceptive). The clinical trial was double-blind.

The data set `hirsutism.dat` contains artificial values of measures corresponding to some patients in this study. The variables are the following:

- `Treatment`, with values 0, 1, 2 or 3.
- `FGm0`, it indicates the baseline hirsutism level at the randomization moment (the beginning of the clinical trial). Only women with baseline FG values grater than 15 where recruited.
- `FGm3`, FG value at 3 months.
- `FGm6`, FG value at 6 months.
- `FGm12`, FG value at 12 months, the end of the trial.
- `SysPres`, baseline systolic blood pressure.
- `DiaPres`, baseline diastolic blood pressure.

- `weight`, baseline weight.
- `height`, baseline height.

1. In the following regression models test the *no effect* null hypothesis (`model="no effect"` in `sm.regression`):

   `weight` as a function of `height`

   `SysPres` as a function of `height`

   `SysPres` as a function of a `weight`

   `FGm0` as a function of `weight`

   `SysPres` as a function of `DiaPres`

2. For this point use only patients with treatments 0 or 2. We want to know if the final result `FGm12` can be used to guess if a patient was assigned to treatment 0 or 2.

   (a) Create a binary variable (`Tr02`) with value 0 for patients with treatment 0, and 1 for patients with treatment 2.

   (b) Test the null hypothesis stating that the logistic model is appropriate in the regression:

   `Tr02` as a function of `FGm12`.

3. Repeat the last point using now treatments 2 and 3.

4. For this point use only patients with treatments 0 or 2. In the following regressions test whether the regression functions are equal in both groups.

   `weight` as a function of `height`

   `SysPres` as a function of `height`

5. Repeat the last point using now treatments 2 and 3.

6. In the following regressions test whether the regression functions are equal in the 4 groups defined by variable `Treatment`.

   `weight` as a function of `height`

   `SysPres` as a function of `height`

7. Comparing the regression function

   `FGm12` as a function of `FGm0`
   in the 4 groups defined by `Treatment`.
   In `sm.ancova`, use both `h1` and `h2` obtained by cross-validation and AICc criteria, respectively (use argument `method="cv"` or `method="aicc"` in `h.select`).
   In function `sig.trace` use `hvec = seq(min(h1,h2)/3,3*max(h1,h2), length=20)`.

8. Test whether the regression function

   `FGm12` as a function of `FGm0`
   can be considered equal or parallel in the two subpopulations defined according to `Treatment==0` or not.
   Use the indications given in the last point for choosing the bandwidth.

9. For this point use only patients with treatments 1 or 3. Test whether the regression function

   `FGm12` as a function of `FGm0`
   can be considered equal or parallel in the two subpopulations defined by `Treatment`.
   Use the indications given above for choosing the bandwidth.

10. For this point use only patients with treatments 1, 2 or 3. Test the linearity of the regression function

    `FGm12` as a function of `FGm0`.
    Use the indications given above for choosing the bandwidth.

Chapter 6

Smoothing splines

6.1 Splines in R: Libraries and functions

The following R libraries and functions implement the techniques related with splines.

Library stats. Installed by default in R and charged when starting R.

spline. Return the cubic interpolating spline for a data set.

```
# Example of interpolating natural cubic spline
# set.seed(3333)
n <- 9
x <- runif(n)
y <- rnorm(n)
spl.xy <- spline(x, y, method ="natural",
                 xmin=0, xmax=1, n = 201)
plot(x, y, pch=19, xlim=c(0,1), ylim=range(spl.xy$y),
     main = paste("spline[fun](.) through", n, "points"))
lines(spl.xy, col = 2)
```

smooth.spline. Fit a cubic spline to data set solving the penalized least squares problem.

predict.smooth.spline. Predict the value of a spline function fitted by `smooth.spline` for new values of the explanatory variable.

Library splines. See `help(package=splines)` for a complete list of functions in the library. Among them we remark the following:

bs, ns. Build bases of B-splines and natural B-splines. (The example of the combined use of `bs` and `lm` to fit a smoothing spline to a data set has special interest.)

interpSpline. Return the cubic interpolating spline for a data set.

6.2 Bases of B-splines

Use the following sentences to build and plot a matrix of cubic B-splines with 9 knots. These bases are evaluated at 101 points.

```
x <- seq(0,1,by=.01)
knots <- seq(.1,.9,by=.1)
k <- 3
basis3 <- bs(x=x,knots=knots,intercept=T,degree=k)
matplot(x,basis3,type="l",ylim=c(0,1),lty=1)
abline(v=knots,lty=2)
```

Now create and plot the bases corresponding to degrees 1 and 2.
Compare the cubic B-splines basis with the Natural cubic B-splines basis.

```
inner.knots <- seq(.2,.8,by=.1)
basis3n <- ns(x=x,knots=inner.knots,intercept=T,Boundary.knots=c(.1,.9))
matplot(x,basis3n,type="l",ylim=c(-1,1),lty=1)
abline(v=inner.knots,lty=2)
abline(v=c(.1,.9),lty=2)
```

6.3 Country development data

We will be working with the country development dataset described in Section 3.1. It contains information on development indicators measured in 132 countries (Source: World Bank, 1992). We will focus on the following variables:

`life.exp`	Life expectancy at birth.
`agricul`	% of agriculture contribution to the GDP (Gross Domestic Product).
`inf.mort`	Infant mortality rate: The annual number of deaths of infants under one year of age per 1,000 live births in the same year.
`life.exp.f`	Life expectancy at birth for females.
`life.exp.m`	Life expectancy at birth for males.
`le.fm`	Difference `life.exp.f` minus `life.exp.m`.

1. Select randomly 15 rows in the data.frame `countries`. Plot the simple regression line (use lm). Then interpolate the data

$$(x_i = \texttt{log(life.exp)},\ y_i = \texttt{log(inf.mort)}), i = 1, \ldots, 15,$$

using the function spline. Fit a cubic spline using smooth.spline. Use different values for the smoothing parameter to move from the interpolating spline to the regression line.

2. From now on use all the data. Use function smooth.spline to fit a cubic spline to the following data:

   `log(inf.mort)` as a function of `log(life.exp)`

   `inf.mort` as a function of `agricul`

   `log(inf.mort)` as a function of `log(agricul+1)`

   `inf.mort` as a function of `le.fm`

3. Combine functions bs and `lm` to fit a degree 1 smoothing spline to the data `log(inf.mort)` as a function of `log(life.exp)`. Use different values for the number of knots.

4. Repeat the last point to fit splines with degree 2 and 3. Compare the last fits (cubic splines) with the results obtained with function smooth.spline.

5. **Classify countries according to `agricul`.**
   The median value of variable `agricul` is 16. Create a new binary variable `ind.agr` indicating for any country if variable `agricul` is lower than 16 (1) or not (0). This new variable is a development indicator.

6. **Classify countries according to `life.exp`.**
   The median value of variable `life.exp` is 68. Create a new binary variable `ind.le` indicating for any country if variable `life.exp` is greater than 68 (1) or not (0). This new variable is a development indicator.

7. **Local binary regression using splines.**
   Combine functions bs and `glm` to fit the following local binary regression models:

   `ind.agr` as a function of `life.exp`,

   `ind.le` as a function of `agricul`.

6.4 NIR data analyzed with splines

The data file `meat.Rdata` contains information on 240 samples of chopped meat that have been analyzed by both a chemical procedure and a near infrared (NIR) absorbance spectroscopy. There are three main variables in the file for each sample of meat:

Fat: Fat content (in percentage) as it was determined buy the chemical analysis.

abs.850: Near-infrared absorbance spectra for 850 nm wavelength.

abs.957: Near-infrared absorbance spectra for 975 nm wavelength.

The objective is to predict the fat content from the spectral data, because the chemical analysis is expensive and time consuming but spectroscopy is faster and cheaper.

1. Combine functions `bs` (from library **splines**) and `lm` to estimate the nonparametric regression function of `log(Fat)` as a function of `abs.850` by a cubic spline. Use a 10-fold cross-validation procedure to determine the number of inner knots.

2. Use the function `smooth.spline` to fit the same regression function using the same number of effective parameters (`df`) as before. Compare both fits in the same graphic.

Chapter 7

Generalized Additive Models. Semiparametric models

We are using the following R functions:

sm.regression from package `sm`, only useful for bivariate regression.

gam. Function in package `mgcv` that fits Generalized Additive Models. It also fits Additive Models as a particular case.

```
library(mgcv)
help(package=mgcv)
help(gam)
```

Pay attention to parameters `formula` and `family`.

ppr. This function fits the Projection Pursuit Regression Model. It cames with package `stats`, that is charged by default when R starts.

spm. This function fits Semiparametric models. It is included in package `SemiPar` accompanying the book
Ruppert, D., Wand, M.P. and Carroll, R.J. (2003) *Semiparametric Regression*. Cambridge University Press.
The use and utility of `spm` is very similar to these of function `gam`.

7.1 Bivariate non-parametric regression with library `sm`

Use the Boston Housing dataset to fit the following bivariate non-parametric regression model:

`RM` as a function of `LSTAT` and `AGE`.

```
library(sm)
library(rgl)#R graphic library

load("boston.Rdata")
attach(boston.c)

a<-sm.regression(x=cbind(LSTAT,AGE), y=RM, h=c(2.85,10))
z<-a$estimate
y<-a$eval.points[,2]
x<-a$eval.points[,1]
persp(x,y,z, theta =40, phi = 10, d=4,xlab='LSTAT',ylab='AGE',zlab='RM',
      box = TRUE, axes = TRUE,  nticks = 5,ticktype = "detailed")

persp3d(x,y,z, shininess= 5, col="green",xlab='LSTAT',ylab='AGE',zlab='RM')
points3d(x=LSTAT,y=AGE, z=RM,size=8)
```

7.2 Additive Models

Use the Boston Housing dataset to fit the following bivariate additive model:
`RM` as a function of `LSTAT` and `AGE`.

```
library(mgcv)
gam.RM<-gam(RM ~ s(LSTAT)+ s(AGE))
# perspective
vis.gam(gam.RM,se=0,theta =40, phi = 10, d=4,nticks=3)
text(-.61,-.1,'RM',srt=90)
# contour
vis.gam(gam.RM,se=0,plot.type="contour",contour.col=1)
points(LSTAT,AGE,col="blue")

# GAM: Gnrealized Additive Model. Functions g_k
op<-par(mfrow=c(1,2))
par(pty='s')
gam.RM<-gam(RM ~ s(LSTAT)+ s(AGE))#,sp=c(40,400))
plot.gam(gam.RM,se=FALSE,rug=FALSE)
par(op)
```

7.3 Projection pursuit regression

Use the Boston Housing dataset to fit the following bivariate additive model:
`RM` as a function of `LSTAT` and `AGE`.

```
ppr3 <- ppr(RM ~ LSTAT+ AGE, nterms = 3, max.terms = 5,
            sm.method="spline", df=c(6,6,6))
summary(ppr3)
op <- par(mfrow = c(2,2))
plot(ppr3, main = "RM ~ LSTAT + AGE, nterms = 3, max.terms = 5)")
plot(RM, predict(ppr3))
abline(a=0,b=1,col=2); abline(lm(RM~predict(ppr3)),col=4)
par(op)
```

7.4 Generalized additive models and semiparametric models

We are using country development dataset (see Section 3.1) to demonstrate the semiparametric model fitting with `gam` function and other related functions from library `mgcv`.

1. Use function `gam` from package `mgcv` to fit the following additive models:
   `inf.mort ~ agricul + life.exp`
   `inf.mort ~ s(agricul) + s(life.exp)`
   `inf.mort ~ s(agricul, life.exp)`
   `inf.mort` as a function of `agricul, life.exp.m, life.exp.f`. Combine linear fits, univariate and bivariate nonparametric components. Use function `anova` to compare the models you fit two by two. Then propose a final model.

2. Repeat the last point using `agricul`, `life.exp.m` and `life.exp.f` as explanatory variables and `inf.mort` as response.

3. Classify countries according to `agricul`. The median value of variable `agricul` is 16. Create a new binary variable `ind.agr` indicating for any country if variable `agricul` is lower than 16 (1) or not (0). This new variable is a development indicator.

4. Classify countries according to `life.exp`. The median value of variable `life.exp` is 68. Create a new binary variable `ind.le` indicating for any

country if variable `life.exp` is greater than 68 (1) or not (0). This new variable is a development indicator.

5. Variable `le.fm` always takes non-negative values, except for one country. Define the variable
   `le.fm.0 <- pmax(0,le.fm)`

6. **Binary nonparametric regression.** Use function `gam` from package `mgcv`, with `family=binomial`, to fit the following additive models:
   ```
   ind.agr ~ life.exp + inf.mort
   ind.agr ~ s(life.exp)
   ind.agr ~ s(inf.mort)
   ind.agr ~ s(life.exp) + s(inf.mort)
   ind.agr ~ s(life.exp,inf.mort)

   ind.le ~ inf.mort + le.fm
   ind.le ~ s(inf.mort)
   ind.le ~ s(le.fm)
   ind.le ~ s(inf.mort) + s(le.fm)
   ind.le ~ s(inf.mort,le.fm)
   ```
   Use function `anova` to compare the models you fit two by two. Then propose a final model.

7. **Poisson nonparametric regression.** Use function `gam` from package `mgcv`, with `family=poisson`, to fit the following local Poisson regression models:
   ```
   le.fm.0 ~ inf.mort + life.exp
   le.fm.0 ~ s(inf.mort) + life.exp
   le.fm.0 ~ inf.mort + s(life.exp)
   le.fm.0 ~ s(inf.mort) + s(life.exp)
   le.fm.0 ~ s(inf.mort,life.exp)
   ```

   Use function `anova` to compare the models you fit two by two. Then propose a final model.

7.5 Hirsutism dataset

Consider the Hirsutism dataset described in Section 5.2. Fit several semiparametric models explaining `FGm12` as a function of the variables that has been measured at the begining of the clinical trial (including `FGm0`) and

`Treatment`. Then select the model (or models) that you think is most appropriate.

Bibliography

Gasser, T., L. Sroka, and C. Jennen-Steinmetz (1986). Residual variance and residual pattern in nonlinear regression. *Biometrika* (3), 625–633.

Rice, J. (1984). Bandwidth choice for nonparametric regression. *Ann. Statist.* (4), 1215–1230.