The GMRF implementation in GAMLSS

Fernanda De Bastiani, Mikis Stasinopoulos and Robert A. Rigby

August 8, 2017

Contents

1 Introduction

The package `gamlss.spatial` provides a set of functions to facilitate the fitting of spatial models within GAMLSS, Rigby and Stasinopoulos [2005]. At the moment it allows only Gaussian Markov random fields (GMRF) terms, Rue and Held [2005], other spatial data facilities will be added in the future. Chapter 9 of Stasinopoulos et al. [2017] provides more information of what other types of additive terms can be used within the **gamlss** packages. De Bastiani et al. [2016] describes the implementation of GMRF within GAMLSS and the material presented here is supplementary to this article.

Markov random fields (MRF) is a generic term to describe k random variables whose joint distribution is specified using local conditional independence assumptions. This package uses spatial Gaussian Markov random field (GMRF) models. More specifically it uses intrinsic autoregressive models (IAR) (which are a limiting case of the conditional autoregressive models (CAR) of Besag [1974]. Besag and Kooperberg [1995] is a good reference for the definition of those models. The IAR models are ideal for modelling a response variable measured in geographical areas. When we model a response variable measured in areas we expect neighbouring areas to have a more similar response variable distribution than areas which are far apart. This is what a IAR term in the model for a response variable distribution parameter aims to achieve by bringing the fitted parameter values of neighbouring areas closer to each other. The fitting of an IAR model requires the specification of the precision matrix. The precision matrix can be constructed by using the geographical information of the areas.

The package provides several functions. The functions `MRF()` and `MRFA()` are appropriate for fitting a simple IAR model. Those two functions are called by the function `gmrf()` in order to fit an additive IAR term within the model for a response variable distribution parameter in the `gamlss()` function.

Section 1.1 provides information about the `MRF()` and `MRFA()` functions and their arguments. Section 1.2 describes additional functions for converting the way graphical information is stored. Section 1.3 gives an example of using the `MRF()` and `MRFA()` functions. Section 2 describes the GAMLSS additive function `gmrf()` and Section 3 gives an example of its use. Conclusions are given in Section 4.

1.1 The functions `MRF()` and `MRFA()`

The model fitted by the two functions `MRF()` and `MRFA()` can be written as:

$$\mathbf{y} = \mathbf{Z}\boldsymbol{\gamma} + \boldsymbol{\epsilon}$$

where $\mathbf{Z}$ is an $n \times q$ incidence matrix ($Z_{ij} = 1$ if observation i belongs to area j and $Z_{ij} = 0$ otherwise), $\boldsymbol{\gamma}$ is a $q \times 1$ vector of random effects for the areas, and where $\boldsymbol{\gamma} \sim N_q(\mathbf{0}, \sigma_b^2 \mathbf{G}^{-1})$ for a specific scaled precision matrix $\mathbf{G}$ and $\boldsymbol{\epsilon} \sim N(\mathbf{0}, \sigma_e^2 \mathbf{W}^{-1})$ where $\mathbf{W}$ is a diagonal matrix of prior weights. If the number of observations equals the number of areas, then $\mathbf{Z} = \mathbf{I}$ the identity matrix.

To estimate the random effect $\boldsymbol{\gamma}$'s one can use the (weighted) penalised least square solution

$$\hat{\gamma} = \left(\mathbf{Z}\mathbf{W}\mathbf{Z}^\top + \lambda \mathbf{G}\right)^{-1} \mathbf{Z}^\top \mathbf{W}\mathbf{y}$$

where $\lambda = \sigma_e^2/\sigma_b^2$.

As described in De Bastiani et al. [2016], assume that a response variable and explanatory variables are recorded at observations which belong spatially to one of a set of areas (or regions). Zero, one or more than one observation may be recorded in each region. To incorporate IAR models within the GAMLSS model, set $\mathbf{Z}$ to be an incidence matrix defining which observation belongs to which area, and let $\boldsymbol{\gamma}$ be the vector of q spatial random effects and assume $\boldsymbol{\gamma} \sim N_q(0, \sigma_b^2 \mathbf{G}^{-1})$, where $\mathbf{G}^{-1}$ is the (generalized) inverse of a $q \times q$ matrix, $\mathbf{G}$. In the following IAR model, based on Besag and Higdon [1999], the matrix $\mathbf{G}$ contains the information about the neighbours (adjacent regions), with elements given by $G_{mm} = n_m$ where n_m is the total number of adjacent regions to region m and $G_{mt} = -1$ if region m and t are adjacent, and zero otherwise, for $m = 1, \ldots, q$ and $t = 1, \ldots, q$. This model has the attractive property that conditional on σ_b^2 and γ_t for all $t \neq m$, then $\gamma_m \sim N(\sum \gamma_t n_m^{-1}, \sigma_b^2 n_m^{-1})$ where the summation is over all regions which are neighbours of region m. For a graphical interpretation of the nonzero pattern of the matrix $\mathbf{G}$ see De Bastiani et al. [2016].

The functions `MRF()` and `MRFA()` differ in the way the estimates of σ_e^2 and σ_b^2 are calculated. Both functions use "local' maximum likelihood estimation as described in De Bastiani et al. [2016] and should give identical estimates. The function `MRF()` maximizes numerically the log likelihood of the marginal normal likelihood (called the Q function) in terms of parameters $\log \sigma_e^2$ and $\log \sigma_b^2$. Note that if the log-likelihood function is relatively flat implementing an informative prior for $\log(\sigma_e^2)$ can help convergence (using the argument `penalty`). The function `MRFA()` fits the same IAR model as `MRF()` but it uses an alternating algorithm described in

Chapter 3 of Stasinopoulos et al. [2017] and a special case of the one described in Section 2 of Rigby and Stasinopoulos [2013]. The estimates should be identical. The function `MRF()` needs starting values for the parameters σ_e^2 and σ_b^2, while `MRFA()` needs a starting value for λ. Note that while the function `MRF()` provides standard errors for the $\log \sigma_e^2$ ans $\log \sigma_b^2$, the function `MRFA()` does not.

The arguments of the function `MRF()` are:

`y` response variable

`x` a factor containing the areas

`precision` the (scaled) precision matrix $\mathbf{G}$ if known

`neighbour` an object containing the neighbour information for the areas

`polys` the polygon geographical information if known

`area` this argument is useful if we have more areas than levels of the factor `x`, (i.e. if there are some areas with no observations in the data set) . This specifies a factor containing all the areas.

`weights` vector of prior weights (the diagonal of $\mathbf{W}$)

`sig2e` starting value for the error variance σ_e^2

`sig2bs` starting value for the random effect variance σ_b^2

`sig2e.fix` whether σ_e^2 is fixed in the fitting, default equals `FALSE`

`sig2b.fix` whether σ_b^2 is fixed in the fitting, default equals `FALSE`

`penalty` whether an extra quadratic penalty is required to help convergence in case the likelihood function is flat. This is equivalent of putting a normal prior distribution for $\log(\sigma_e^2)$ given by $\log(\sigma_e^2) \sim N(\mu_s, 1/\delta)$

`delta` the precision of the prior i.e. δ

`shift` the mean of the prior i.e. μ_s

The function `MRFA()` has extra arguments:

`lambda` for fixing the smoothing parameter for `MRFA()` function

`start` starting value for the smoothing parameter λ for `MRFA()` function

`df` for fixing the degrees of freedom

Note that both `MRF()` and `MRFA()` create an `MRF` object in **R**. There are several method to operate on a `MRF` object i) `fitted()`, ii) `coef()` iii) `residuals()` iv) `AIC()` v) `deviance()` vi) `plot()` vii) `print()` viii) `summary()` ix) `logLik()` x) `predict()`.

1.2 Additional functions

First we explain 3 different ways in which the graphical information about the areas (or regions) can be stored:

i) a neighbour object is a `R` list comprising each region label followed by its neighbouring region labels.

ii) a polygon object is a `R` list comprising the region label followed by coordinates of points in two columns in matrix form defining the boundary for each area.

iii) a (scaled precision) matrix $\mathbf{G}$ is defined in Section 1.1 for the specific IAR model fitted by the **gamlss.spatial** package which determines the prior distribution $\boldsymbol{\gamma} \sim N_q(\mathbf{0}, \sigma_b^2 \mathbf{G}^{-1})$ where $\mathbf{G}^{-1}$ is a generalized inverse of $\mathbf{G}$.

There are several additional supporting functions in the package:

`nb2nb()` transforms an object with neighbour information in a shapefile format (geospatial vector data format for geographic information system, written in **R** as a `S4` object) to the neighbour required form for functions `MRF()` and `MRFA()`. The single argument takes a `S4` neighbour object.

`polys2nb()` creates the neighbour object from the geographical polygons. The single argument takes a polygon object.

`nb2prec()` creates the matrix $\mathbf{G}$ from the neighbour information.There are three arguments here:

`neighbour` is a neighbour object.

`x` is the area factor. This factor can have less levels than the number of areas defined in the neighbour object. In such cases the third argument `area` has to be specified.

`area` all possible areas involved, with the number of areas is equal to the number of neighbours in the **`neighbour`** object of the first argument.

`polys2polys()` transforms polygons in shapefile format (S4 object) to the polygons required form for the `MRF()` and `MRFA()`.

`draw.polys()` Plots the fitted values a fitted `MRF` object. This function has arguments:

`polys` An object containing the polygon information for the area.

`object` This can be either a fitted `MRF` object or a vector of values to plot. Note that in later case the vector should also have names corresponding to the names of the `polys` (see the example below for how this can be achieved.).

`scheme` The scheme of colours to use, it can be ”heat”, ”rainbow”, ”terrain”, ”topo”, ”cm” or any colour.

`swapcolor` To reverse the colours, it just works for ”heat”, ”rainbow”, ”terrain”, ”topo”, ”cm” options.

`n.col` A range for different colours.

1.3 Example using `MRF()` and `MRFA()`

R data file: `columb` in package **mgcv** of dimensions 49×8

var `area` : land area of district

home.value : housing value in 1000USD

income : household income in 1000USD.

crime : residential burglaries and auto thefts per 1000 households (the response variable)

open.space : measure of open space in district

district : code identifying district, and matching `names(columb.polys)`

x,y middle point area coordinates

purpose: to demonstrate spatial functions in GAMLSS

Note: in the above data set the variable `district` contains the names of the districts or regions or areas (and should not be confused with the variable area which is quantitative).

First we bring the data frame `columb` and the polygons file `columb.polys` from the **mgcv** package. We also print the polygon information for the first district of the data called district `"0"`.

```
library(gamlss.spatial)
library(mgcv)
# bring the data
data(columb)
names(columb)
```

```
## [1] "area"       "home.value" "income"     "crime"      "open.space"
## [6] "district"   "x"          "y"
```

```
# getting the polygons file
data(columb.polys)
head(columb.polys,1)
```

```
## $`0`
##           [,1]     [,2]
##  [1,] 8.624129 14.23698
##  [2,] 8.559700 14.74245
##  [3,] 8.809452 14.73443
##  [4,] 8.808413 14.63652
##  [5,] 8.919305 14.63850
##  [6,] 9.087138 14.63049
##  [7,] 9.099965 14.24483
##  [8,] 9.015047 14.24184
##  [9,] 9.008951 13.99506
## [10,] 8.818140 14.00205
## [11,] 8.653305 14.00809
## [12,] 8.642902 14.08971
## [13,] 8.632592 14.17059
## [14,] 8.625826 14.22367
## [15,] 8.624129 14.23698
```

Above are the first district name `"0"` and the (horizontal and vertical) coordinates of the polygon defining the district. We now use the function `polys2nb()` to translate the information from polygons to neighbours:

```
# getting the neighbours object from the polygons object
vizinhos <- polys2nb(columb.polys)
vizinhos[[1]]["0"]

## $`0`
## [1] 2 3
```

For example the district "0" has as neighbours districts "2" and "3'.

The function `nb2prec()` is used to get the (scaled precision) matrix **G** from the neighbour information. The created (scaled precision) matrix `precisionC` is an 49×49 matrix, but here we plot only its first 10 rows and 20 columns.

```
# getting the precision matrix from the neighbours object
precisionC <- nb2prec(vizinhos,x=columb$district)
precisionC[1:10, 1:20]

##     0  1  2  3  4  5  6  7  8  9 10 11 12 13 14 15 16 17 18 19
## 0   2 -1 -1  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0
## 1  -1  3 -1 -1  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0
## 2  -1 -1  4 -1 -1  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0
## 3   0 -1 -1  4 -1  0  0 -1  0  0  0  0  0  0  0  0  0  0  0  0
## 4   0  0 -1 -1  8 -1  0 -1 -1  0 -1  0  0  0 -1 -1  0  0  0  0
## 5   0  0  0  0 -1  2  0  0 -1  0  0  0  0  0  0  0  0  0  0  0
## 6   0  0  0  0  0  0  4 -1  0  0  0 -1 -1 -1  0  0  0  0  0  0
## 7   0  0  0 -1 -1  0 -1  6  0  0 -1 -1 -1  0  0  0  0  0  0  0
## 8   0  0  0  0 -1 -1  0  0  8 -1  0  0  0  0 -1  0  0  0  0 -1
## 9   0  0  0  0  0  0  0  0 -1  4  0  0  0  0  0  0 -1  0  0 -1
```

The first row indicates that region "0" has 2 neighbours regions "1" and "2".

Now we will fit the IAR model using the two different functions `MRF()` and `MRFA()`, but also using different geographical information i) polygons ii) neighbours and iii) (scaled precision) matrix **G**. The **R** function `system.time()` checks the speed of the procedures. The model which used the (scaled precision) matrix **G** should be fastest, since all functions require matrix **G** to be obtained for fitting.

```
# fit using the  polygone information
# MRFA alternaing
system.time(m11<-MRFA(columb$crime, columb$district, polys=columb.polys))

##    user  system elapsed
##    0.56    0.00    0.56

# MRF Q-function
system.time(m21<-MRF(columb$crime, columb$district, polys=columb.polys))

##    user  system elapsed
##    0.30    0.00    0.29

# fit using the neighbour information
# MRFA alternaing
system.time(m12<-MRFA(columb$crime, columb$district,  neighbour=vizinhos))
```

```
##    user  system elapsed
##    0.25    0.00    0.25
```

```
# MRF Q-function
system.time(m22<-MRF(columb$crime, columb$district, neighbour=vizinhos))
```

```
##    user  system elapsed
##    0.23    0.00    0.23
```

```
# fit using the percision matrix
# MRFA alternaing
system.time(m13<-MRFA(columb$crime, columb$district, precision=precisionC))
```

```
##    user  system elapsed
##    0.11    0.01    0.12
```

```
# MRF Q-function
system.time(m23<-MRF(columb$crime, columb$district, precision=precisionC))
```

```
##    user  system elapsed
##    0.08    0.00    0.07
```

```
AIC(m11, m21, m12, m22, m13, m23, k=0)
```

```
##           df      AIC
## m21 24.46858 335.9114
## m22 24.46858 335.9114
## m23 24.46858 335.9114
## m11 24.46856 335.9115
## m12 24.46856 335.9115
## m13 24.46856 335.9115
```

All fitted models are identical, but note below the different information provided by object `MRF` when it is fitted using the function `MRF()` and when fitted using `MRFA()`. The algorithm used in `MRFA()`, while generally faster to converge, does not provides standard errors for the parameter estimates. The function `MRF()` also provides in addition the marginal deviance of the fit.

```
summary(m11)
```

```
##
## Markov Random Fields fit
## Fitting method: "altenating"
##
## Call:  "MRFA(columb$crime, columb$district, polys = columb.polys)"
##
##
## Coefficient(s):
##              Estimate  Std. Error  t value Pr(>|t|)
## log(sige^2)   4.51682          NA       NA       NA
## log(sigb^2)   5.83251          NA       NA       NA
##
##  Degrees of Freedom for the fit: 24.46856 Residual Deg. of Freedom   24.53144
## Global Deviance:     335.911
##             AIC:     386.849
##             SBC:     435.031
## Marginal Devia.:     0
```

```
summary(m21)
```

```
##
## Markov Random Fields fit
## Fitting method: "Q-function"
##
## Call:  "MRF(columb$crime, columb$district, polys = columb.polys)"
##
##
## Coefficient(s):
##              Estimate  Std. Error  t value   Pr(>|t|)
## log(sige^2)  4.516816    0.533713   8.4630 < 2.22e-16 ***
## log(sigb^2)  5.832515    0.534887  10.9042 < 2.22e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
##  Degrees of Freedom for the fit: 24.46858 Residual Deg. of Freedom   24.53142
## Global Deviance:     335.911
##             AIC:     386.849
##             SBC:     435.031
## Marginal Devia.:     462.338
```

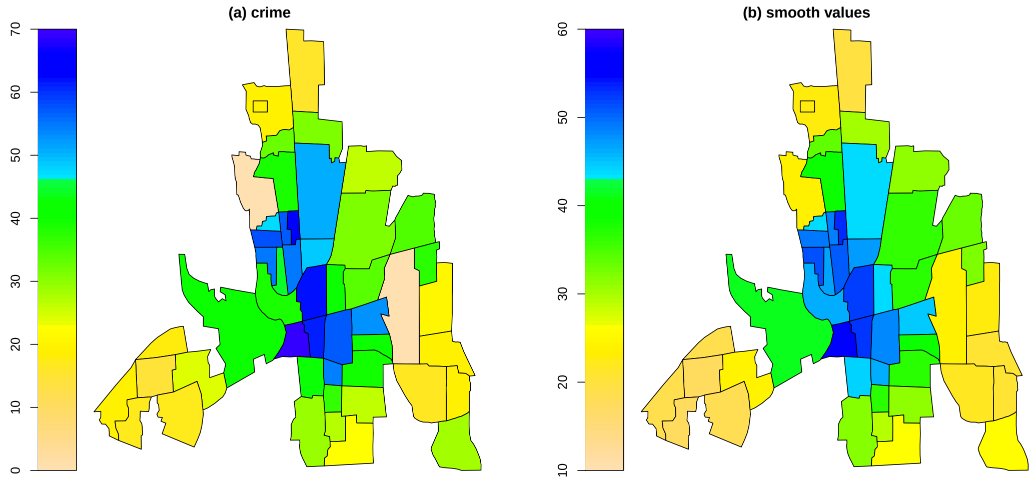
Next we plot both the observed and the fitted response variable values in maps by using the function `draw.polys()`. Note that the first argument of the function is the polygon information, while the second is either a vector (to plot the observed response variable value) or a fitted `MRF` model (to plot fitted response variable values). In the former case the vector should contain the observed response variable values together with their district labels stored as `names`. In commands below we create the vector `cr` for the row crime figures and then we assign the names of the districts as names for `cr`. [Note that if there were more than one observation in the same district, the mean `cr` for each district has to be computed (with names as the district labels) and replace `cr` with the computed mean `cr`.]

```
cr <- columb$crime
names(cr) <- as.character(columb$district)
draw.polys(columb.polys, cr,   scheme="topo",swapcolors=TRUE)
title("(a) crime")
draw.polys(columb.polys, m11, scheme="topo",swapcolors=TRUE)
title("(b) smooth values")
```

Figure 1

Figure 1(a) shows that the range of the observed crime values is from 0 to 70, while from Figure 1(b) the range for the fitted (mean) crime values is between 10 and 60. The 'shrinking' effect, where fitted (mean) values for different areas shrink towards their neighbours, is a typical behaviour in the IAR model.

It would be of interest to see what will happen if data from one of the areas defined in the polygons is missing. In this case we will have less areas in the data than the number of polygons



R code on page 8

Figure 1: Showing (a) the actual crime figures and (b) the fitted values from the IAR model.

defined in the polygon file. We will remove district "4" from the `columb` data set (but also level "4" for the factor `district`). In order to fit the model to the reduced data set we will need to define a factor which has as many levels as the number of areas in the polygon information file. Our original `columb$district` has this information and it will be used below, but in general we can get this information from the polygon file, i.e. `as.factor(names(columb.polys))`.

```
# the dimension of the original data
dim(columb)

## [1] 49  8

# removing one area (district ''4'') from  the data
columb2 <- columb[-5,]
# drop unused level from a factor
columb2$district <-droplevels(columb2$district)
dim(columb2)

## [1] 48  8

nlevels(columb2$district)

## [1] 48

# fitting the reduced data
# using  polys
r1<-MRF(columb2$crime, columb2$district, polys=columb.polys,
        area=columb$district)
# using neighbours
r2<-MRF(columb2$crime, columb2$district,  neighbour=vizinhos,
        area=columb$district)
# using the old precision
```

```
r3<-MRF(columb2$crime, columb2$district, precision=precisionC,
        area=columb$district)
# creating new precision matrix
precisionC2 <- nb2prec(vizinhos, x=columb2$district,
                       area=columb$district)
dim(precisionC2)

## [1] 49 49

# fitting  using the new 49 x 49 precision
r4<-MRF(columb2$crime, columb2$district, precision=precisionC2,
         area=columb$district)
# checking the results
AIC(r1,r2,r3,r4, k=0)

##         df      AIC
## r1 23.5671 330.9477
## r2 23.5671 330.9477
## r3 23.5671 330.9477
## r4 23.5671 330.9477
```

All fitted models produce identical results. Note that one consequence of having less areas (i.e. districts in the above example) in the data than the actual areas in the polygons is that the γ has length equal to the areas of the polygons while the fitted values has less district values. For instance in our example we have 49 areas defined by `area='columb$district'`, but in the data only 48 areas and with no repetition in areas. Therefore the estimated $\hat{\gamma}$ is of length 49, while the fitted values of the model are of length 48. Next using the function `plot.polys()` we plot the fitted (mean crime) values of model `r1` and the estimated $\hat{\gamma}$ from the same model.

```
draw.polys(columb.polys, fitted(r1), scheme="heat", swapcolors=TRUE )
title("(a)")
draw.polys(columb.polys, r1, scheme="heat",swapcolors=TRUE  )
title("(b)")
```

Figure 2

Note the white region in Figure 2(a) indicating the missing fitted value for area `"4"`. The colour of the same area in Figure 2(b) is filled with a colour (similar to its neighbours) representing the estimated $\hat{\gamma}$ for area `"4"`.

2 The GAMLSS additive function `gmrf()`

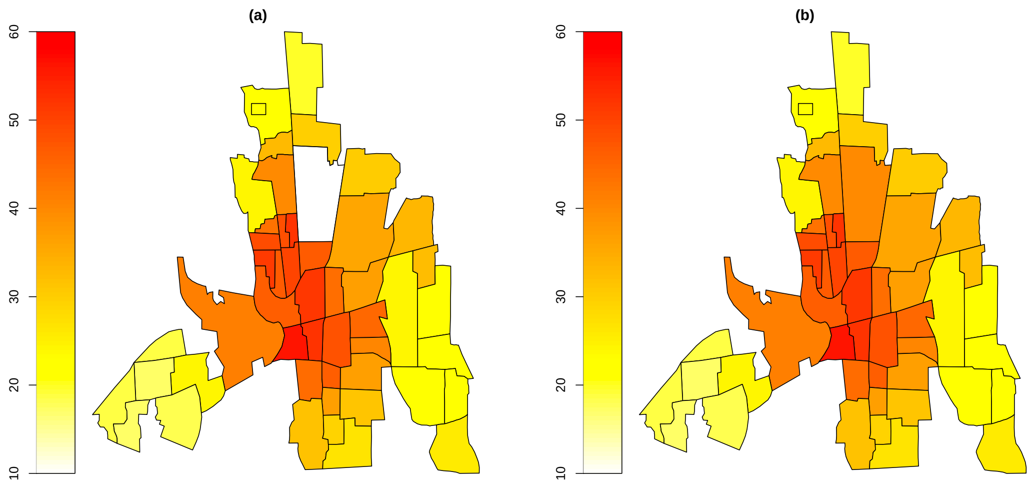
The function which can be used within GAMLSS to fit an IAR model is `gmrf()`. It has the following arguments:

x a factor containing the areas

precision the (scaled) precision matrix **G** if set (the quickest way to fit the model),

polys the polygon information if set,

area this argument is here to allow more areas than the levels of the factor `x`, as was described in Section 1.1,



R code on page 10

Figure 2: Showing (a) the fitted values of model `r1` and (b) the fitted $\hat{\gamma}$ from the same model.

start starting value for the smoothing parameter λ, only for `method="A"`

df degrees of freedom for fitting if required, only for `method="A"`

method `"Q"` for Q-function using `MRF()`, or `"A"` for alternating method using `MRFA()`,

adj.weight a value to adjust the iterative weight if necessary (to achieve convergence of the algorithm).

[Note that some of the arguments of the functions `MRF()` and `MRFA()` can be used here (according to the method selected).]

First we fit model `g1`, the IAR model fitted using `gamlss()`, and we compare the results with the model `m21` fitted by `MRF()`. Notice that in the output below the fitted values for the μ parameter are identical, and so are the estimates (18.47) for the parameter σ_b. The estimates for the parameter σ_e are different since the estimate (6.77) from the `gamlss()` model [see Rigby and Stasinopoulos [2005] and Nelder [2005]] is a maximum likelihood a posteriori (MAP) or penalized likelihood estimator, while the estimate (9.57) from `MRF()` is a REML estimator. Note that the `gamlss()` MAP estimator of σ_e can be substantially negatively biased [i.e. underestimates σ_e when, as here, the total effective degrees of freedom used in the model for μ (23.47, including the spatial smoother) is high relative to the sample size (49)]. This causes the deviances to be different in the two fitted models. Note that `gamlss()` uses a normal distribution for the response variable `crime` by default.

```
# fit the model
g1 <- gamlss(crime~gmrf(district,precision=precisionC), data=columb)

## GAMLSS-RS iteration 1: Global Deviance = 326.4786
## GAMLSS-RS iteration 2: Global Deviance = 326.4786

# comparing the fitted values
head(cbind(fitted(m21),fitted(g1)))
```

```
##       [,1]     [,2]
## 0 19.47122 19.47122
## 1 22.73844 22.73845
## 2 30.16383 30.16383
## 3 33.25372 33.25372
## 4 43.46635 43.46635
## 5 30.86439 30.86439

tail(cbind(fitted(m21),fitted(g1)))

##        [,1]     [,2]
## 43 31.07408 31.07408
## 44 31.57654 31.57654
## 45 16.95122 16.95122
## 46 25.43692 25.43692
## 47 29.06619 29.06619
## 48 26.12274 26.12274

#  the log-sigma coefficients
coef(m21)

## log(sigeˆ2) log(sigbˆ2)
##    4.516816    5.832515
## attr(,"se")
## log(sigeˆ2) log(sigbˆ2)
##   0.5337131   0.5348875

coef(getSmo(g1))

## log(sigeˆ2) log(sigbˆ2)
##    7.599620    5.832515
## attr(,"se")
## log(sigeˆ2) log(sigbˆ2)
##   0.5337129   0.5348875

# get sigma_b
m21$sigb

## [1] 18.47203

getSmo(g1)$sigb

## [1] 18.47202

# get sigma_e
m21$sige

## [1] 9.567847

fitted(g1,"sigma")[1]

##        0
## 6.769828

# comparing the deviances
```

```
deviance(g1)

## [1] 326.4786

deviance(m21)

## [1] 335.9114

# get degrees of freedom for mu
g1$mu.df

## [1] 23.46858
```

The nice thing about GAMLSS is its flexibility and the fact that you can check different models. Up to now we have used only the geographical information to model the crime figures. From Section 1.3, the dataset `columb` contains also other information like the available income, `income`, the value of homes in the area, `home.value`, the open space of the area, `open.space`, the size of the area, `area` and the coordinates of the middle points of the area. The latest two variables can be used to fit a geostatistics type of model. Here we will use it to fit a two dimensional smooth surface to the crime figures. To fit the model we are using the function `ga()` which is an interface to the function `gam()` of Simon Wood's package **mgcv**

```
library(gamlss.add)
g2 <- gamlss(crime~ga(~s(x,y)), data=columb)

## GAMLSS-RS iteration 1: Global Deviance = 307.3535
## GAMLSS-RS iteration 2: Global Deviance = 307.3535

AIC(g1,g2)

##          df      AIC
## g2 23.66369 354.6809
## g1 24.46858 375.4158
```

```
names(g1$mu.fv) <- names(g2$mu.fv) <- as.character(columb$district)
draw.polys(columb.polys, fitted(g2), scheme="terrain",swapcolors=TRUE  )
title("(a) 2-d smoothing")
draw.polys(columb.polys, fitted(g1), scheme="terrain", swapcolors=TRUE )
title("(b) IAR")
```

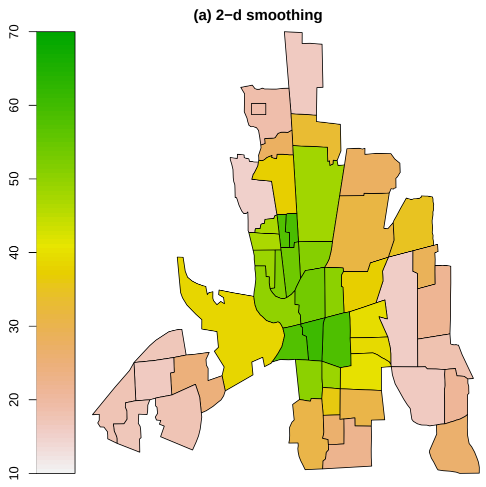
Figure 3

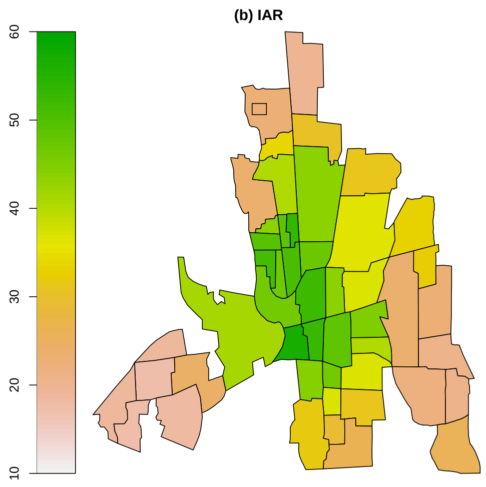
One can combine the geographical information with the other available variables to built a suitable model for modelling the crime figures. Unfortunately the number of observations in the data, 49, is very small to do justice to such an analysis. Next we will build suitable models using explanatory variables but no-geographical information and then we will compare those models with the ones using only geographical information.

We start by selecting a suitable model using only linear terms. We are using the model selection function `stepGAIC()`. We first transform the `open.space` variable by taking its log.

```
columb <- transform(columb, logos=log(open.space+1))
e0 <- gamlss(crime~1, data=columb)

## GAMLSS-RS iteration 1: Global Deviance = 414.1438
## GAMLSS-RS iteration 2: Global Deviance = 414.1438
```





R code on page 13

Figure 3: Showing (a) the fitted values of the 2-dimensional smoothing model `g2` and (b) the fitted IAR model `g1`.

```
# linear
e1 <- stepGAIC(e0, scope=list(lower=~1, upper=~area+income+home.value+logos))

## Distribution parameter:  mu
## Start:  AIC= 418.14
##  crime ~ 1
##
##              Df    AIC
## + income      1 387.74
## + home.value  1 400.52
## + area        1 412.27
## + logos       1 417.02
## <none>          418.14
##
## Step:  AIC= 387.74
##  crime ~ income
##
##              Df    AIC
## + home.value  1 382.75
## <none>          387.74
## + area        1 388.31
## + logos       1 389.72
## - income      1 418.14
##
## Step:  AIC= 382.75
##  crime ~ income + home.value
##
```

```
##              Df    AIC
## <none>          382.75
## + area        1 383.57
## + logos       1 384.70
## - home.value  1 387.74
## - income      1 400.52
```

```r
formula(e1)
```

```
## crime ~ income + home.value
```

Linear terms for `income` and `home.value` were selected. Figure 4 shows a visual representation of the model and how they effect the mean, μ, of the response.

```r
term.plot(e1)
```

Figure 4

R code on page 15

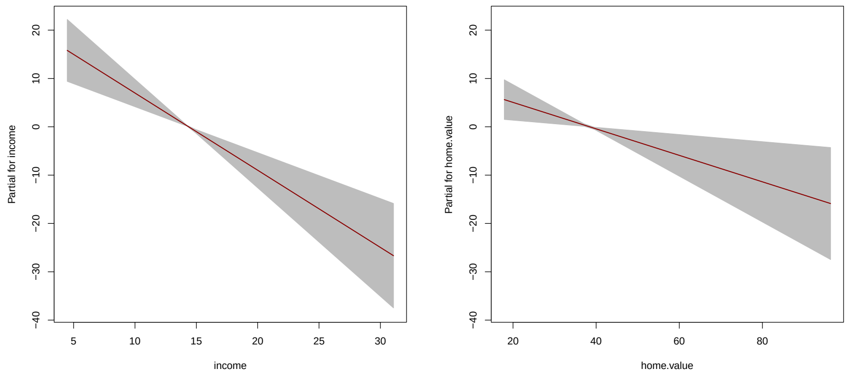


Figure 4: Showing term plots for the linear model `e1`.

Note that by fitting the linear terms in `income` and `home.value` the geographical GMRF terms becomes redundant. As a result of this if you try to fit the following model it will fail.

```r
ge1<- gamlss(crime~income+home.value+gmrf(district,precision=precisionC),
             data=columb)
```

We are trying now to select a model using smooth additive terms. We are using P-splines and the function `pb()` as a smoother.

```r
e2 <- stepGAIC(e0, scope=list(lower=~1, upper=~pb(area)+pb(income)+
                          pb(home.value)+pb(logos)))
```

```
## Distribution parameter:  mu
## Start:  AIC= 418.14
##  crime ~ 1
##
##                      Df    AIC
## + pb(income)     1.0000 387.74
## + pb(home.value) 3.2053 395.15
## + pb(area)       2.7543 399.20
## + pb(logos)      2.1472 414.86
## <none>                  418.14
##
## Step:  AIC= 387.74
##  crime ~ pb(income)
##
##                       Df    AIC
## + pb(home.value) 1.0000 382.75
## + pb(area)       2.2756 384.95
## <none>                  387.74
## + pb(logos)      1.0000 389.72
## - pb(income)     1.0000 418.14
##
## Step:  AIC= 382.75
##  crime ~ pb(income) + pb(home.value)
##
##                        Df    AIC
## + pb(area)         2.6400 380.74
## <none>                    382.75
## + pb(logos)        2.1663 383.96
## - pb(home.value)   1.0000 387.74
## - pb(income)      -1.2053 395.15
##
## Step:  AIC= 380.74
##  crime ~ pb(income) + pb(home.value) + pb(area)
##
##                        Df    AIC
## <none>                    380.74
## - pb(income)      -1.8038 382.50
## + pb(logos)        1.0117 382.71
## - pb(area)         2.6400 382.75
## - pb(home.value)   1.3644 384.95
```

```
formula(e2)
```

```
## crime ~ pb(income) + pb(home.value) + pb(area)
```

Additive smoothing terms for`income`, `home.value` and `area` were selected this time. Figure 4 shows their effect on the mean, μ, of the response variable `crime`.

```
term.plot(e2)
```

Figure 5

The GAMLSS framework allows the fitting of a neural network model for one or more of the distribution parameters of the model. This is done by the interface function `nn()` which calls the `nnet()` function of Brian Ripley's package **nnet**. We fit a neural network model for the mean of the response (i.e. parameter μ).

```
e3 <- gamlss(crime~nn(~income+home.value+logos+area, decay=0.1), data=columb)
```

Figure 6

```
## GAMLSS-RS iteration 1: Global Deviance = 194.0747
```

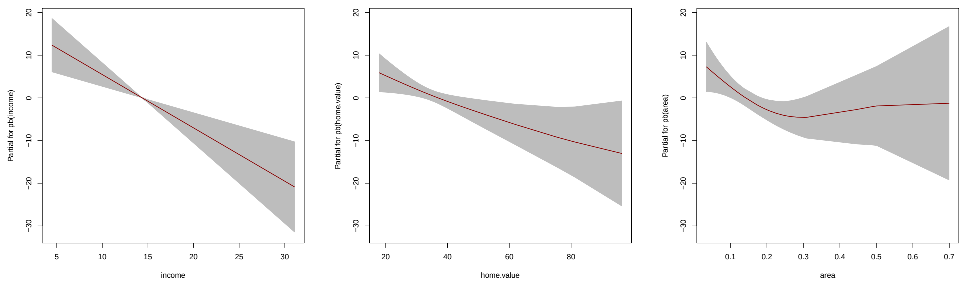


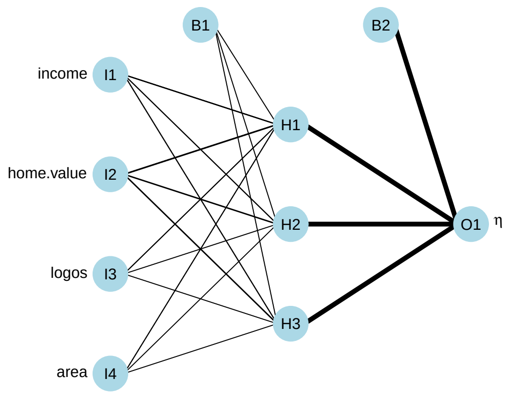
Figure 5: Showing term plots for additive model `e2`.

R code on page 16

```
## GAMLSS-RS iteration 2: Global Deviance = 121.519
## GAMLSS-RS iteration 3: Global Deviance = 121.4659
## GAMLSS-RS iteration 4: Global Deviance = 121.4659

term.plot(e3)
```

[Note in the above neural network fitting it may be better to rescale the explanatory variables to interval $[0, 1]$, as recommended by Ripley [1996][page 157].] While the neural network model



R code on page 16

Figure 6: Showing a graphical interpretation of the neural network model `e3`.

can take into account interactions between the explanatory variables, unfortunately it is very difficult to interpret and usually over-fits the data. Figure 6 shows how the explanatory variables are connected with three hidden variables (`H1, H2, H3`), where the thickness of the lines reflects how large the coefficients are.

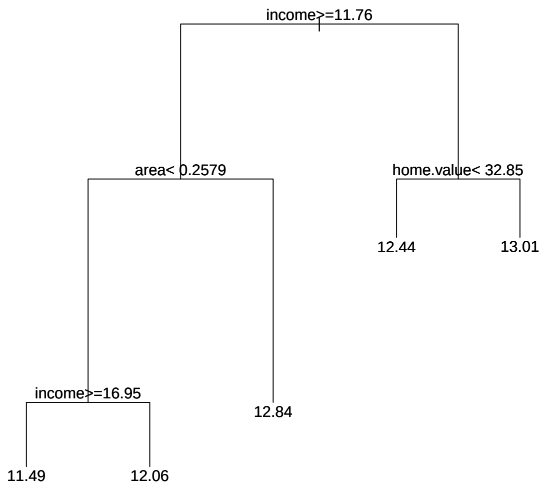
Finally we fit a decision tree model to the parameter μ of the model.

Figure 7

```
e4 <- gamlss(crime~tr(~income+home.value+logos+area), data=columb)

## GAMLSS-RS iteration 1: Global Deviance = 97.9616
## GAMLSS-RS iteration 2: Global Deviance = 97.9616

term.plot(e4)
```



R code on page 18

Figure 7: Showing a term plot for the mean μ of the decision tree model `e4`.

Decision trees are easy to interpret as Figure 7 shows, where the parameter μ is a function of `income`, `area` and `home.value`. In each split of the decision tree (e.g. `income>=11.76`) the left branch is YES (i.e. `income>=11.76`) and the right branch is NO (i.e. `income<11.76`).

We can compare the different models fitted here using AIC and SBC/BIC.

```
AIC(g1,g2, e1, e2, e3, e4)

##           df      AIC
## e4  7.000000 111.9616
## e3 21.000000 163.4659
## g2 23.663691 354.6809
## g1 24.468577 375.4158
## e2  6.639968 380.7350
## e1  4.000000 382.7545

AIC(g1,g2, e1, e2, e3, e4, k=log(49))
```

```
##           df      AIC
## e4  7.000000 125.2043
## e3 21.000000 203.1941
## e1  4.000000 390.3218
## e2  6.639968 393.2966
## g2 23.663691 399.4484
## g1 24.468577 421.7059
```

The decision tree and neural network models using explanatory variables, i.e. models `e4` and `e3`, respectively, do better that the models using only geographical information, `g1` and `g2`, in this case. Of course in a larger data set both may be needed as we show in the next section where we analyse the Munich rent data.

We finish our analysis by showing in Figure 8 the fitted values for each of our four models on a map.

```
names(e4$mu.fv) <- names(e3$mu.fv) <- names(e2$mu.fv) <-
  as.character(columb$district)
draw.polys(columb.polys, fitted(g1), scheme="terrain", swapcolors=TRUE )
title("(a) IAR")
draw.polys(columb.polys, fitted(e2), scheme="terrain",swapcolors=TRUE  )
title("(b) Additive Smooth")
draw.polys(columb.polys, fitted(e3), scheme="terrain",swapcolors=TRUE  )
title("(c) neural network")
draw.polys(columb.polys, fitted(e2), scheme="terrain",swapcolors=TRUE  )
title("(d) decision tree")
```

Figure 8

3 The **`rent99`** data analysis

3.1 The 1999s Munich rent data

The `rent` data come from a survey conducted in 1999, a random sample of accommodation with new tenancy agreements or increases of rents. The data are in the package **gamlss.data** (which is automatically loaded when **gamlss** is loaded). There are 3081 observations on nine variables. The data were analyzed by De Bastiani et al. [2016] but here we reproduce the results to demonstrate how the package **gamlss.spatial** is working in **R**.

R data file: `rent99` in package **gamlss.data** of dimensions 3081 × 9

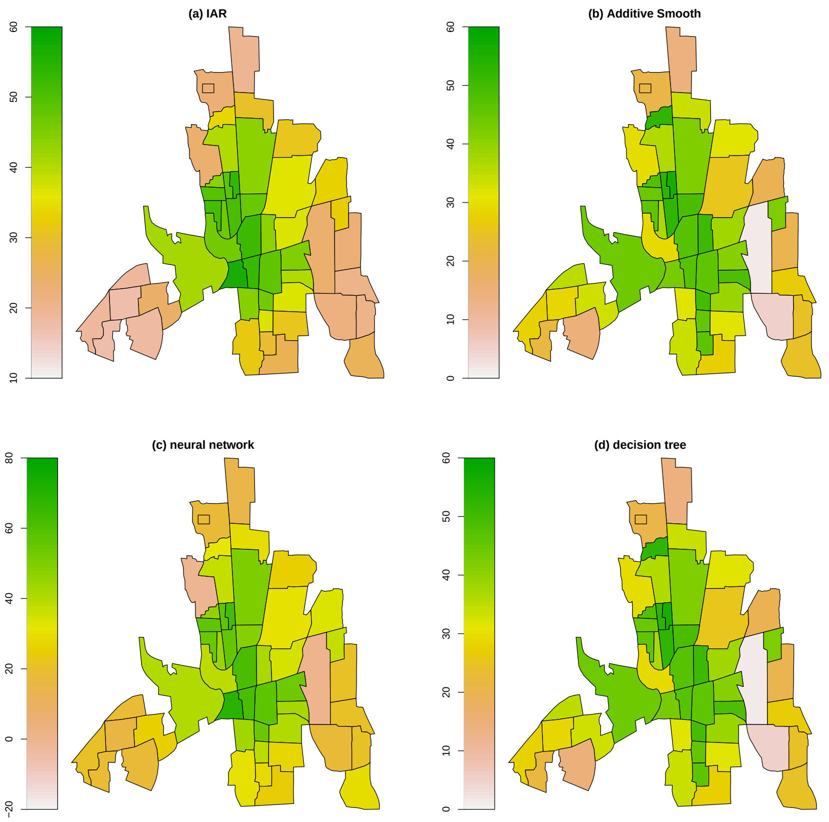
var **`rent`** : the response variable which is the monthly net rent per month (in Euro)

`rentsqm` : the net rent per month per square meter (in Euro)

`area` : living area in square meters.

`yearc` : the year of construction

`location` : the quality of location as a factor indicating whether the location is



R code on page 19

Figure 8: Showing the fitted values for μ in four of our fitted models (a) the IAR/GMRF model `g2` and (b) the additive smoothing terms model `e2`, (c) neural network model `e3` and (d) decision tree model `e4`.

average, 1, good, 2, and top location, 3.

bath : the quality of bathroom: as a factor indicating whether the bath facilities are standard, 0, or premium, 1

kitchen : the quality of the kitchen: 0 standard, 1 premium

cheating : central heating as a factor with two levels, 0, without central heating, 1 with central heating

district : the district in Munich.

purpose: to demonstrate fitting IAR models using `gamlss.spatial` **R** package

We input the data and create a few new variables to take into account suitable interactions later.

```
library(gamlss.spatial)
data(rent99)
data(rent99.polys)
rent99$cheating<-relevel(rent99$cheating,"1")
# creating new variables for interactions
# heating and years interaction
cy<-(as.numeric(rent99$cheating)-1)*rent99$yearc
# kitchen and years interation
ky<-(as.numeric(rent99$kitchen)-1)*rent99$yearc
# kitchen and area interation
ka<-(as.numeric(rent99$kitchen)-1)*rent99$area
# heating has its relevant level changed from 0 to 1
heating<-relevel(rent99$cheating,"1")
rent99 <- transform(rent99,heating=heating, cy=cy, ky=ky, ka=ka)
```

Figure 9 shows a histogram and a box-plot of the response variable `rent` which shows asymmetry and positive skewness.

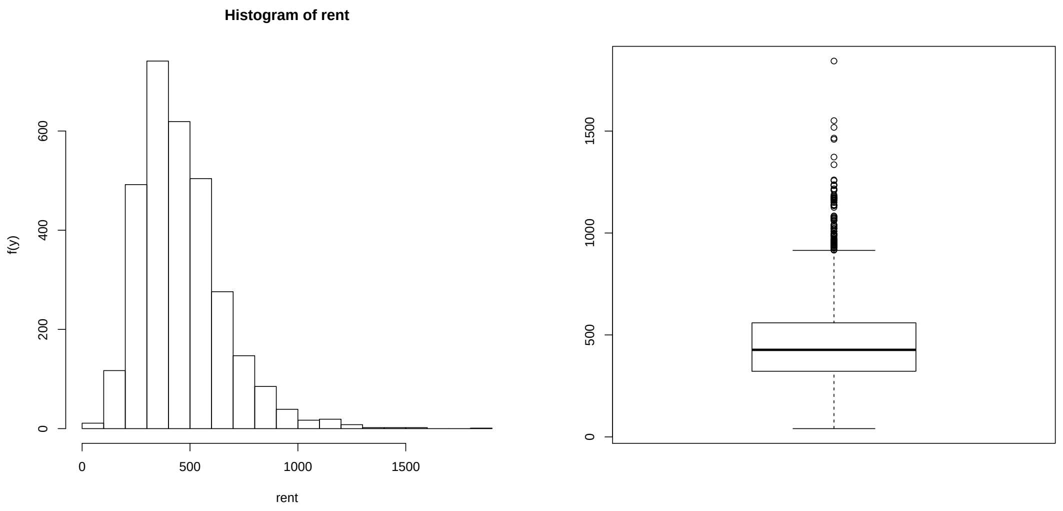
```
hist(rent99$rent,ylab="f(y)",main="Histogram of rent", xlab="rent")
boxplot(rent99$rent)
```

Figure 9

The complexity of the relationship between the response and the explanatory variables is shown in Figure 10. Note that those plots are bivariate exploratory plots and take no account of the interactions between the explanatory variables.

```
plot(rent99$rent~rent99$area, data=rent, col=gray(0.7),
     pch = 15, cex = 0.5, xlab="area", ylab="rent")
plot(rent99$rent~rent99$yearc, data=rent, col=gray(0.7),
     pch = 15, cex = 0.5, xlab="yearc", ylab="rent")
plot(rent99$rent~rent99$location, data=rent, col=gray(0.7),
     pch = 15, cex = 0.5, xlab="location", ylab="rent")
plot(rent99$rent~rent99$bath, data=rent, col=gray(0.7),
     pch = 15, cex = 0.5, xlab="bath", ylab="rent")
plot(rent99$rent~rent99$kitchen, data=rent, col=gray(0.7),
     pch = 15, cex = 0.5, xlab="kitchen", ylab="rent")
plot(rent99$rent~rent99$cheating, data=rent, col=gray(0.7),
```

Figure 10



R code on page 21

Figure 9: A marginal histogram and box plot for response variable `rent`.

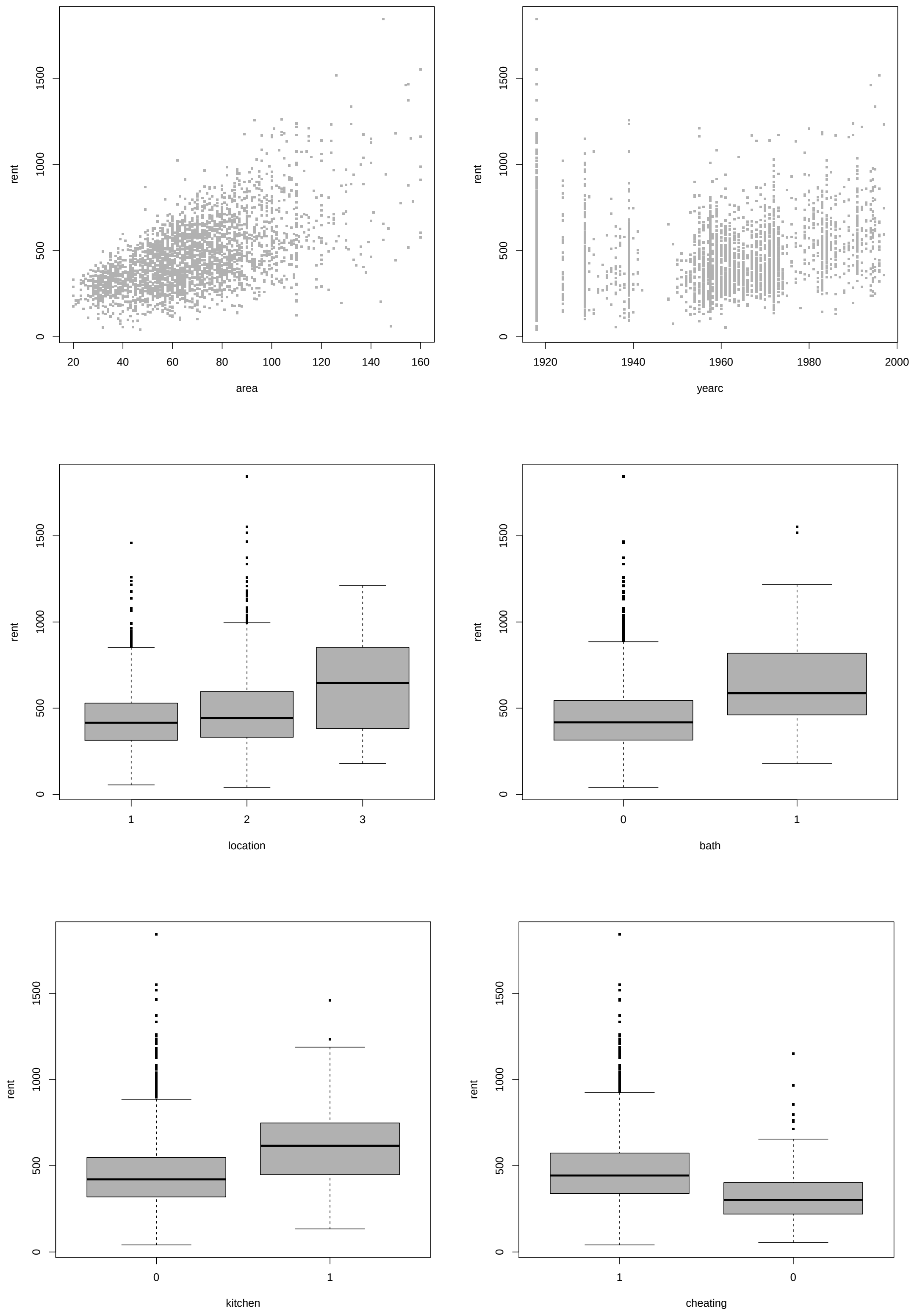
```
     pch = 15, cex = 0.5, xlab="cheating", ylab="rent")
```

Here we use the strategy described in De Bastiani et al. [2016] where appropriate models for μ, σ and ν are selected first without taking into account the spatial structure of the data. We first fit a basic model `m0` and then we use the `stepGAICALL.A()` function to select models for all the distribution parameters μ, σ and ν. The selection takes several minutes and the output (which is omitted below) gives the steps to selecting the final model `m1`.

```
m0<-gamlss(rent~location+bath+kitchen+cheating+area+yearc+pb(area)+pb(yearc),
          sigma.fo=~area + yearc + pb(area) + pb(yearc),
          nu.fo=~area + yearc + pb(area) + pb(yearc), family=BCCGo,
          data=rent99)
m1 <- stepGAICAll.A(m0, scope=list(lower=~location+bath+kitchen+cheating+area+
        yearc+pb(area)+pb(yearc), upper=~(location+bath+kitchen+cheating+
        area+yearc)^2+pb(area)+pb(yearc)), sigma.scope=list(lower=~area+yearc,
        upper=~location+bath+kitchen+cheating+area+yearc+pb(area)+pb(yearc)),
        nu.scope=list(lower=~area+yearc, upper=~location+bath+kitchen+
        cheating+area+yearc+pb(area)+pb(yearc)),k=4)
```

Preparing to fit the IAR model for μ [using the interaction variables `cy`, `ky` and `ka`, so that the linear and smoothing (`pb`) effects for each continuous variable are combined together in the term plot for μ].

```
fd<-as.factor(rent99$district)
farea<-as.factor(names(rent99.polys))
vizinhos <- polys2nb(rent99.polys)
#creating the precision matrix
precision <- nb2prec(vizinhos,fd,area=farea)
#adding spatial effect for mu
```



R code on page 21

Figure 10: A response `rent` against the explanatory variables.

Fitting the reparametrized model `m1` with the additional IAR spatial model for μ

```
m2<- gamlss(formula = rent ~ location + bath + kitchen + cheating +
            pb(area) + pb(yearc) + cy + ky + ka +
            gmrf(fd, area = farea, precision = precision, method="A"),
            sigma.formula = ~area +  pb(yearc) + cheating,
            nu.formula = ~pb(area) + pb(yearc) +
            kitchen, family = BCCGo, data = rent99, start.from=m1)
```

Using AIC:

```
AIC(m1,m2, k=2)
```

```
##          df      AIC
## m2 86.31527 38083.46
## m1 32.96566 38128.37
```

Refitting the final model with mean centred variables (`nyearc` and `narea`) in the interactions, so that the term plots for the factors are easier to interpret.

```
rent99$nyearc<-rent99$yearc-mean(rent99$yearc)
rent99$narea<-rent99$area-mean(rent99$area)
m2final<- gamlss(formula = rent ~ location + bath +
          cheating*nyearc + kitchen*nyearc +
          kitchen*narea + pb(area) + pb(yearc) +
          gmrf(fd, area = farea, precision = precision),
          sigma.formula = ~area  + cheating + pb(yearc),
          nu.formula = ~  kitchen + pb(area) + pb(yearc),
          family = BCCGo, data = rent99, start.from=m2)
```

Note that Figure 11 combines the term plots for μ for the explanatory factors obtained from `m2final`, with the term plots for μ for the explanatory continuous variables obtained from `m2`.

To plot the term plots for μ. (Interactions are automatically omitted from the plots and also no spatial effect is plotted with this function).

```
#to get the termplot for the factors (without interaction)
term.plot(m2final, what="mu", ylim="free")
```

Figure 11

```
#to get the termplot for the continuous variables
term.plot(m2, what="mu", ylim="free")
```

[Note that the term plots for μ gives the contribution from the explanatory variables to the predictor of μ, i.e. $\log \mu$, for the *BCCGo* distribution.]

To plot the term plot for the interactions we used an extra function called `int.term`, available at the website `www.gamlss.org`.

```
source("int-term-plot.R")
#to find out the position of the interaction terms
head(lpred(m2final, type="terms"))
```

Figure 12

```
##     location         bath    cheating     nyearc      kitchen      narea
## 1  0.04283594 -0.004178144 -0.22838175 -0.2163778 -0.006215623 -0.4315904
```

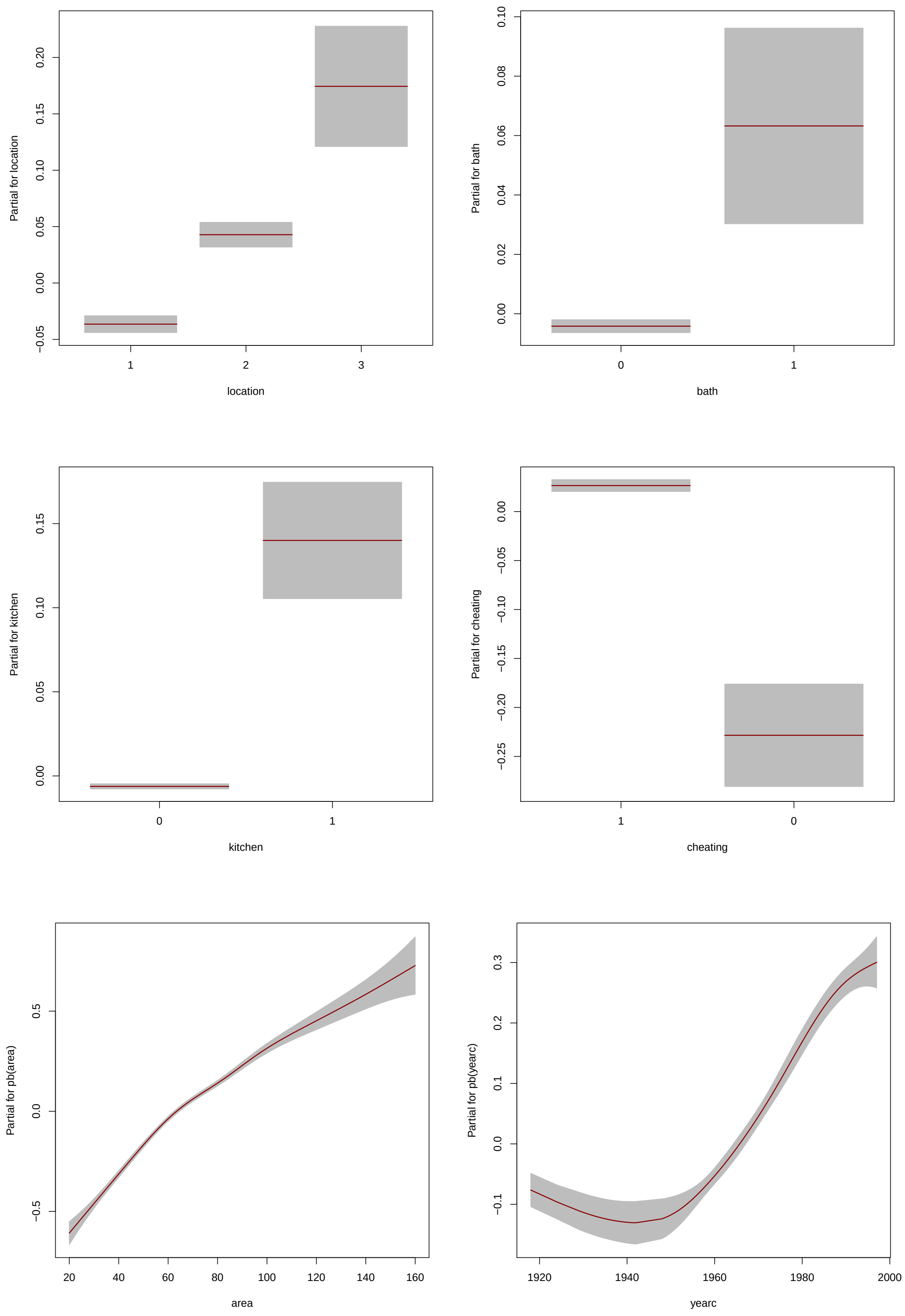


Figure 11: Term plots for μ.

R code on page 24

```
## 2  0.04283594 -0.004178144  0.02655217 -0.2163778 -0.006215623 -0.4107279
## 3 -0.03647352 -0.004178144  0.02655217 -0.2163778 -0.006215623 -0.3898654
## 4  0.04283594 -0.004178144 -0.22838175 -0.2163778 -0.006215623 -0.3898654
## 5  0.04283594 -0.004178144  0.02655217 -0.2163778 -0.006215623 -0.3898654
## 6  0.04283594 -0.004178144  0.02655217 -0.2163778 -0.006215623 -0.3898654
##      pb(area) pb(yearc) gmrf(fd, area = farea, precision = precision)
## 1 -0.08792058 0.1402521                                    -0.022470371
## 2 -0.07926578 0.1402521                                     0.006771048
## 3 -0.07065139 0.1402521                                     0.001981685
## 4 -0.07065139 0.1402521                                    -0.005860875
## 5 -0.07065139 0.1402521                                     0.027865816
## 6 -0.07065139 0.1402521                                     0.007908263
##   cheating:nyearc nyearc:kitchen kitchen:narea
## 1    -0.136217579    0.001698891 -0.0004707379
## 2     0.009470226    0.001698891 -0.0004707379
## 3     0.009470226    0.001698891 -0.0004707379
## 4    -0.136217579    0.001698891 -0.0004707379
## 5     0.009470226    0.001698891 -0.0004707379
## 6     0.009470226    0.001698891 -0.0004707379
```

```
int.term(object=m2final, xvar=rent99$yearc, position=10,
         fac=rent99$cheating, factor.plots=TRUE, xlabel="yearc",
         ylabel="cheating", which.lev="0")
int.term(object=m2final, xvar=rent99$yearc, position=11,
         fac=rent99$kitchen, factor.plots=TRUE,
         xlabel="yearc", ylabel="kitchen", which.lev="1")
int.term(object=m2final, xvar=rent99$area, position=12,
         fac=rent99$kitchen, factor.plots=TRUE,
         xlabel="area", ylabel="kitchen", which.lev="1")
```

To plot the term plot for the spatial effect for μ.

```
draw.polys(rent99.polys,getSmo(m2final, what="mu", which=3),
           scheme="heat")
```

Figure 13

[Note that `which=3` is used in order to choose the third smoothing term for μ, which corresponds to the spatial term in the `m2final` model.]

To plot the term plots for σ and ν (which give the contributions from the explanatory variables to the predictor of σ and ν, i.e. $\log \sigma$ and ν, respectively).

```
term.plot(m2final, what="sigma", ylim="free")
```

```
term.plot(m2final, what="nu", ylim="free")
```

Figure 14

Figure 15

The fitted model is given by:

```
summary(m2final)
```

```
## *******************************************************************
## Family:  c("BCCGo", "Box-Cox-Cole-Green-orig.")
##
```

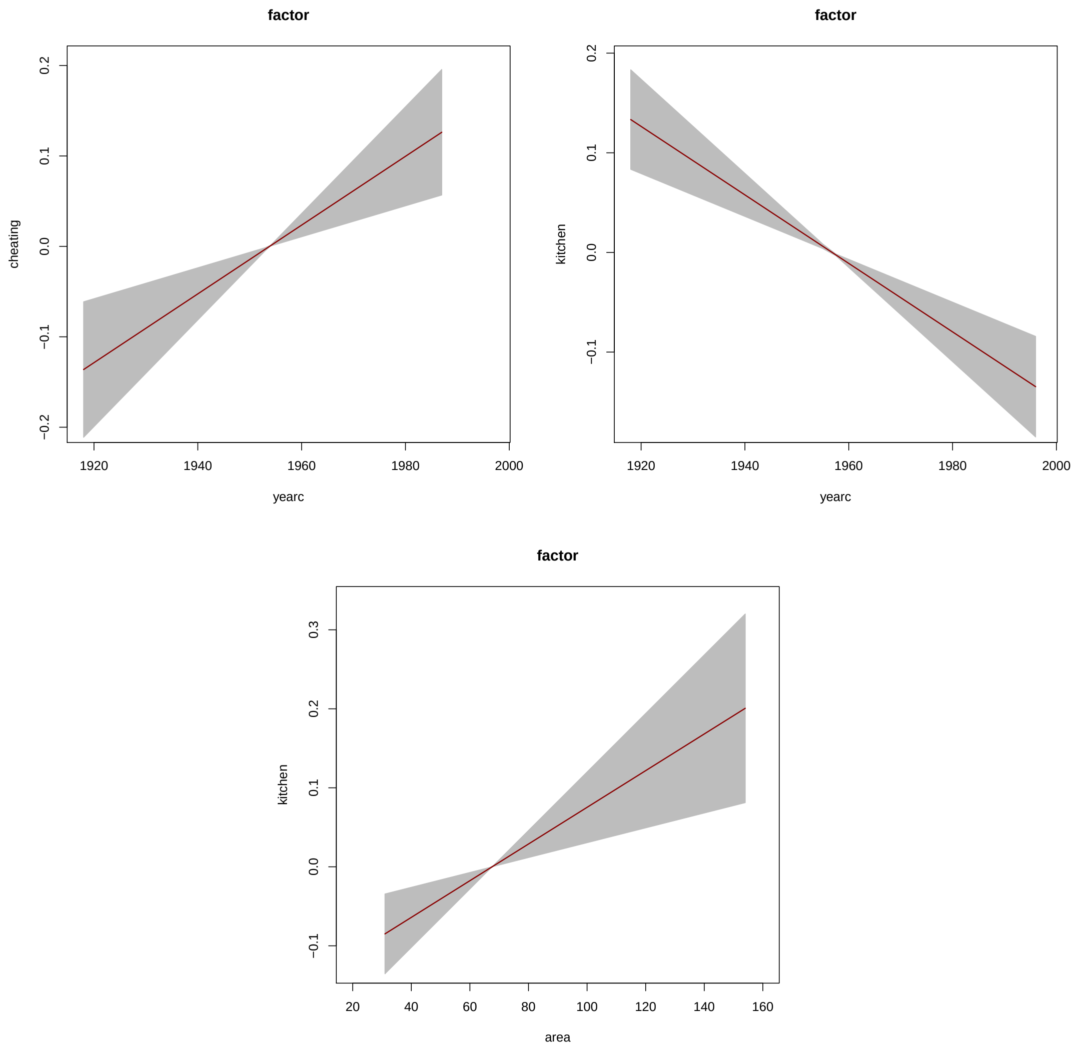
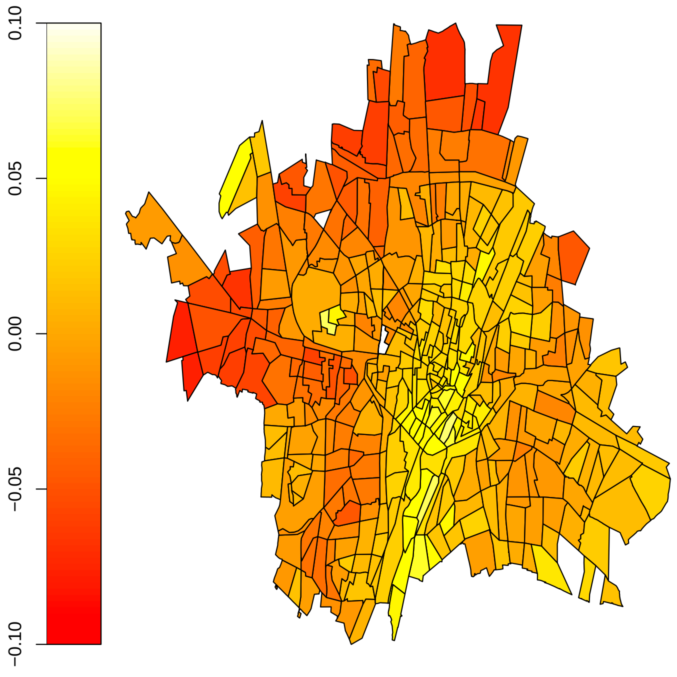


Figure 12: Term plots of the interactions for μ.

R code on page 24



R code on page 26

Figure 13: The fitted spatial effect for μ for the chosen model with spatial effect.

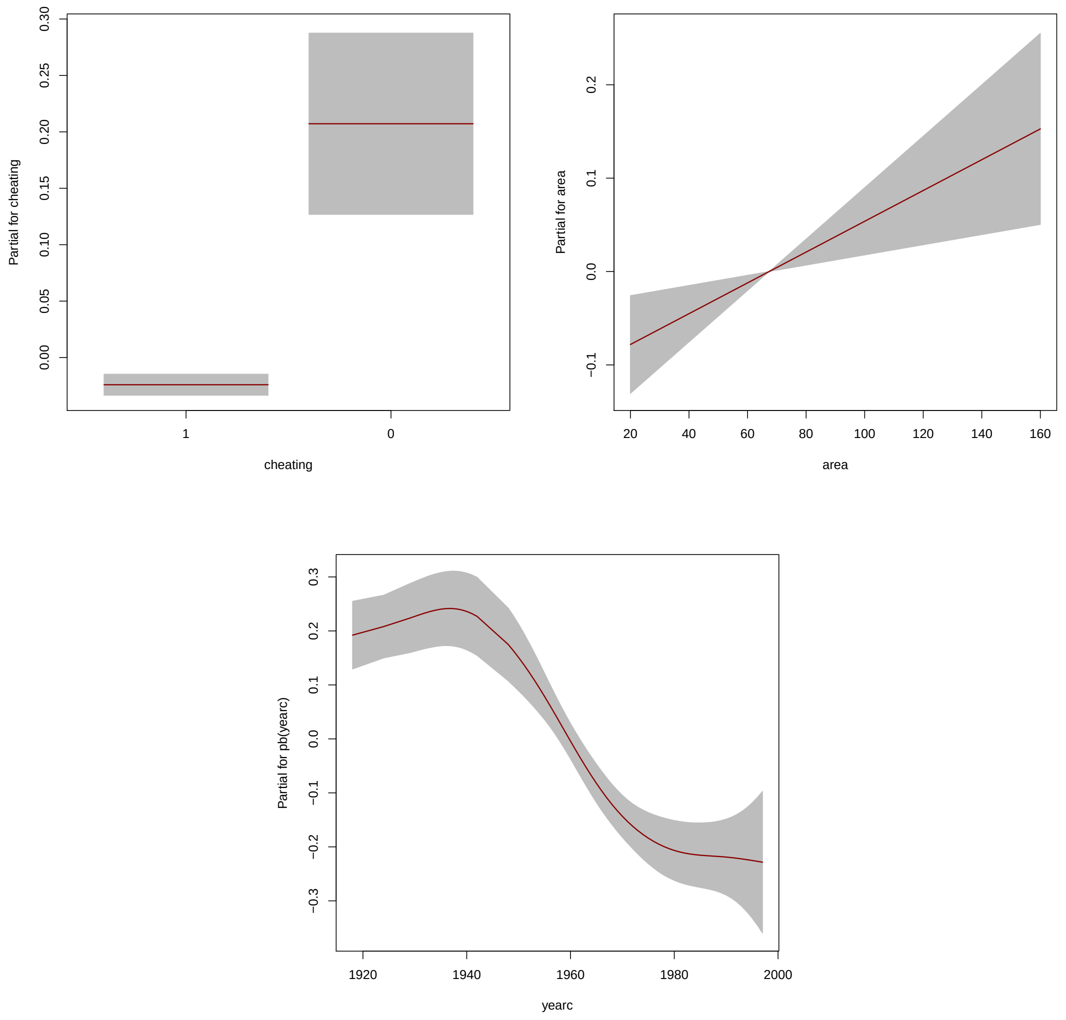


Figure 14: Term plots for σ.

R code on page 26

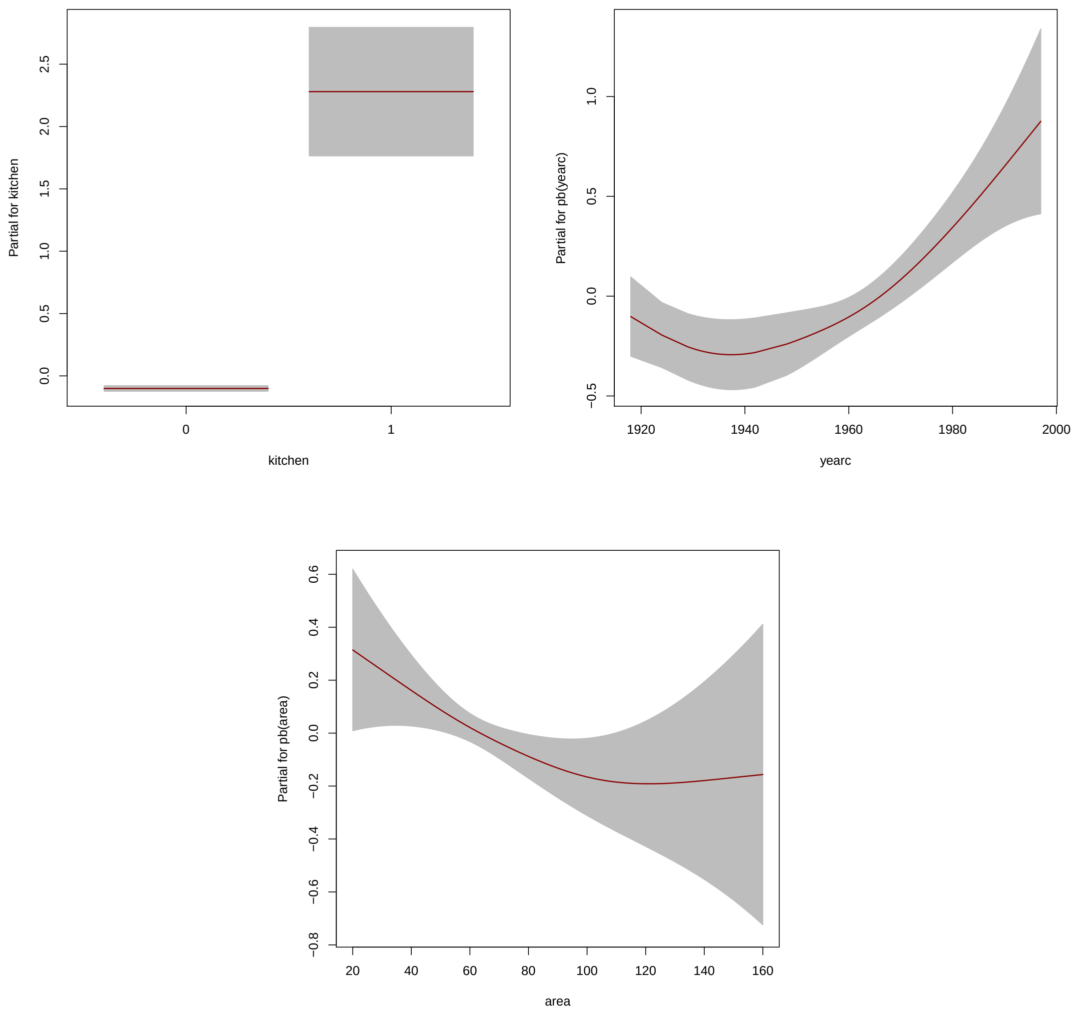


Figure 15: Term plots for ν.

R code on page 26

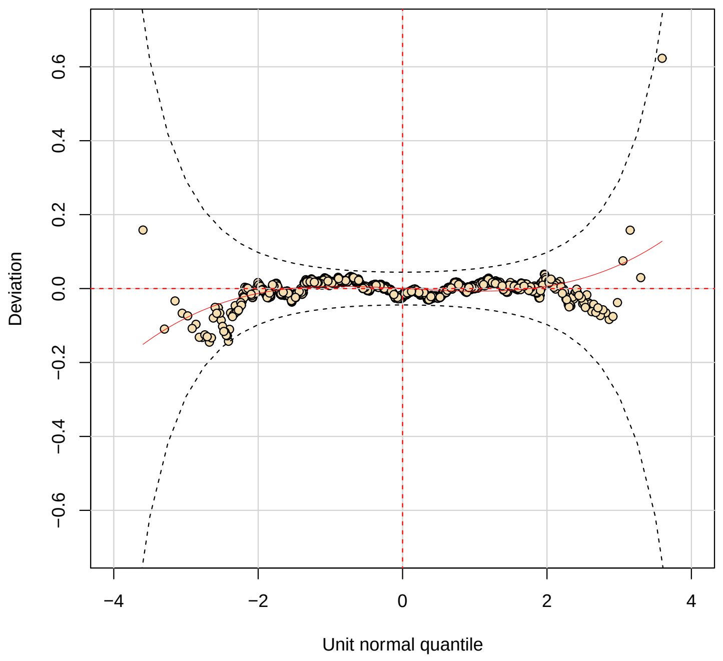
```
## Call:  gamlss(formula = rent ~ location + bath + cheating *
##     nyearc + kitchen * nyearc + kitchen * narea + pb(area) +
##     pb(yearc) + gmrf(fd, area = farea, precision = precision),
##     sigma.formula = ~area + cheating + pb(yearc), nu.formula = ~kitchen +
##         pb(area) + pb(yearc), family = BCCGo, data = rent99,
##     start.from = m2)
##
## Fitting method: RS()
##
## -------------------------------------------------------------------
## Mu link function:  log
## Mu Coefficients:
##                    Estimate Std. Error t value Pr(>|t|)
## (Intercept)       6.0607879  0.0067821 893.644  < 2e-16 ***
## location2         0.0793095  0.0090845   8.730  < 2e-16 ***
## location3         0.2108582  0.0288719   7.303 3.58e-13 ***
## bath1             0.0674191  0.0173826   3.879 0.000107 ***
## cheating0        -0.2549339  0.0282578  -9.022  < 2e-16 ***
## nyearc            0.0056486  0.0002598  21.745  < 2e-16 ***
## kitchen1          0.1462332  0.0190100   7.692 1.95e-14 ***
## narea             0.0104312  0.0002254  46.269  < 2e-16 ***
## cheating0:nyearc  0.0038032  0.0009652   3.941 8.32e-05 ***
## nyearc:kitchen1  -0.0034371  0.0006572  -5.230 1.81e-07 ***
## kitchen1:narea    0.0023216  0.0007850   2.958 0.003124 **
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## -------------------------------------------------------------------
## Sigma link function:  log
## Sigma Coefficients:
##               Estimate Std. Error  t value Pr(>|t|)
## (Intercept)  1.182e+01  2.994e-01   39.465  < 2e-16 ***
## area         1.649e-03  5.947e-04    2.773   0.0056 **
## cheating0    2.313e-01  4.699e-02    4.922 9.04e-07 ***
## pb(yearc)   -6.788e-03  2.276e-05 -298.203  < 2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## -------------------------------------------------------------------
## Nu link function:  identity
## Nu Coefficients:
##               Estimate Std. Error  t value Pr(>|t|)
## (Intercept) -1.225e+01  1.373e-01  -89.210  < 2e-16 ***
## kitchen1     2.381e+00  5.413e-01    4.399 1.12e-05 ***
## pb(area)    -4.411e-03  1.815e-03   -2.430   0.0152 *
## pb(yearc)    6.813e-03  2.560e-06 2661.869  < 2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

```
##
## ------------------------------------------------------------------
## NOTE: Additive smoothing terms exist in the formulas:
##  i) Std. Error for smoothers are for the linear effect only.
## ii) Std. Error for the linear terms maybe are not accurate.
## ------------------------------------------------------------------
## No. of observations in the fit:  3082
## Degrees of Freedom for the fit:  86.3153
##       Residual Deg. of Freedom:  2995.685
##                       at cycle:  1
##
## Global Deviance:     37910.83
##             AIC:     38083.46
##             SBC:     38604.23
## ******************************************************************
```

The residual diagnostics plot, the worm plot, for the final model.

```r
wp(m2final, ylim.all=0.7)
```

Figure 16



R code on page 32

Figure 16: Worm plot of the residuals for the chosen final model `m2final`.

Worm plots for cases in each of the 16 joint intervals for different combinations of the two continuous explanatory variables `yearc` and `area`.

```
wp(m2final, xvar=~yearc+area, n.inter=4, ylim.worm=1)
```

```
## number of missing points from plot= 0  out of  130
## number of missing points from plot= 0  out of  236
## number of missing points from plot= 0  out of  305
## number of missing points from plot= 1  out of  243
## number of missing points from plot= 0  out of  219
## number of missing points from plot= 0  out of  245
## number of missing points from plot= 0  out of  251
## number of missing points from plot= 0  out of  211
## number of missing points from plot= 0  out of  219
## number of missing points from plot= 0  out of  236
## number of missing points from plot= 0  out of  300
## number of missing points from plot= 0  out of  200
## number of missing points from plot= 0  out of  326
## number of missing points from plot= 0  out of  169
## number of missing points from plot= 0  out of  188
## number of missing points from plot= 0  out of  188
```

Figure 17

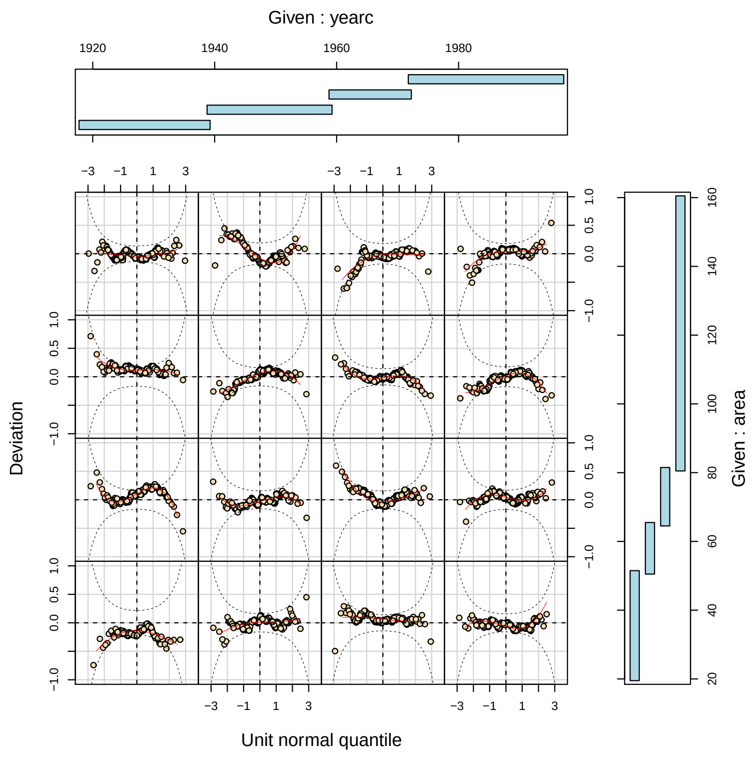


Figure 17: Worm plot of the residuals split by the `yearc` and `area` variables for the final model.

R code on page 33

4 Conclusions

We have shown that the GAMLSS framework provides a platform to fit, compare and check spatial models for the parameters of the distribution of a response variable which may be non exponential family. For more details about GMRF (and in particular IAR models) in GAMLSS see De Bastiani et al. [2016].

References

J. Besag. Spacial interaction and the statistical analysis of lattice systems. *Journal of the Royal Statistical Society, Series B*, 36:192–236, 1974.

J. Besag and P. Higdon. Bayesian analysis of agriculture field experiments (with discussion). *J. R. Statist. Soc., B*, 61:691–746, 1999.

J. Besag and C. Kooperberg. On conditional and intrinsic autoregressions. *Biometrika*, 82(4): 733–746, 1995.

F. De Bastiani, R. A. Rigby, D. M. Stasinopoulos, A. H. M. A. Cysneiros, and M. A. Uribe-Opazo. Gaussian Markov random field spatial models in GAMLSS. *Journal of Applied Statistics*, online:1–19, 2016.

J. A. Nelder. Contribution to the discussion of Rigby and Stasinopoulos, Generalized additve models for location, scale and shape. *Appl. Statist.*, 54:547, 2005.

R. A. Rigby and D. M. Stasinopoulos. Generalized additive models for location, scale and shape, (with discussion). *Appl. Statist.*, 54:507–554, 2005.

R. A. Rigby and D. M. Stasinopoulos. Automatic smoothing parameter selection in GAMLSS with an application to centile estimation. *Statistical Methods in Medical Research*, 23(4): 318–332, 2013. doi: 10.1177/0962280212473302.

B. D. Ripley. *Pattern Recognition and Neural Networks.* Cambridge University Press, Cambridge, 1996.

H. Rue and L. Held. *Gaussian Markov Random Fields: Theory and Applications.* Chapman and Hall, New York, 2005.

D. M. Stasinopoulos, R. A. Rigby, G. Z. Heller, V. Voudouris, and F. De Bastiani. *Flexible Regression and Smoothing: Using GAMLSS in R.* Chapman and Hall, Boca Raton, 2017.