A Beginner's Guide to Generalized Additive Models with R

Alain F Zuur

Highland Statistics Ltd.

6 Generalized Additive Models applied on northern gannets

Alain F Zuur and Kees CJ Camphuysen

6.1 Northern gannets in the North Sea

Camphuysen (2011) collected and analysed field observations on northern gannets (*Morus bassanus*) from the North Sea. Searching and feeding tactics, along with foraging association with other predators, were investigated. The behaviour of gannets at sea was described from visual observations during ship-based surveys and sessions of experimental discarding on board fishery research vessels, in June and July of 1991 to 2004.

The surveys covered most of the feeding range of gannets nesting on Bass Rock (Hamer et al. 2000; Nelson 2002). Counts were conducted between periods when the survey ship was engaged in fishing. The survey periods were spaced some time apart with normally no more than 3–4 per day, when the vessel travelled at full speed of 8–10 knots, and the number of gannets that followed the ship was low and could be controlled for.

We analyse the number of gannets observed in a 300 m wide strip transect. It is assumed that all individual gannets were detected. The underlying question is when and where gannets feed in the vicinity of the Bass Rock colonies.

6.2 The variables

The data are in the file `gannets2.txt`, and Table 6.1 gives the variables and their interpretation. The interpretation of most variables is self-explanatory. The response variable is the number of gannets (`Gannets_in_transect`).

Table 6.1. Variables in the gannets data set

Variable	Description	Type
Day	Day of the month of sampling	Continuous response
Month	Month of sampling	Categorical covariate
Year	Year of sampling	Continuous covariate
Hours	Hour of sampling	Continuous covariate
Minutes	Minute of sampling	Continuous covariate
Latitude, longitude, Y, X	Survey location	Continuous covariate
Area_surveyedkm_2	Surveyed area	Offset variable
Seastate	Sea state	Categorical covariate
Gannets_in_transect	Number of gannets in 300 metre wide strip transect	Response variable

Each row in the spreadsheet represents a transect. The following R code imports the data, and the `str` function is used to verify that the data have been imported correctly. The `names` function gives all variable names.

```
> Gannets <- read.table(file = "gannets2.txt",
                        header = TRUE)
> str(Gannets)          #Results not shown here
> names(Gannets)        #Show all variable names

 [1] "Poskey"                "Day"
 [3] "Month"                 "Year"
 [5] "Hours"                 "Minutes"
 [7] "Latitude"              "Longitude"
 [9] "Area_surveyedkm2"      "Seastate"
[11] "Gannets_in_transect"   "X"
[13] "Y "
```

Not all variables will be used in the analyses. We begin by loading the necessary packages. The `lattice` package is used for multi-panel plots, `mgcv` for smoothing, and `gstat` for creating variograms. At a later stage we will be using the package `gamlss` for smoothing

```
> library(lattice)   #For multi-panel graphs
> library(mgcv)      #For GAM
> library(gstat)     #For checking spatial patterns
> library(gamlss)    #For GAM
```

6.3 Brainstorming

We have the spatial position of each observation and we know when it was made. The response variable is the number of gannets sampled in a transect. The word 'transect' is slightly misleading in this context, as most scientists are likely to associate it with a series of observations along a line, with each of these observations coded as a unique value in a spreadsheet or database. In the gannet example, the transect is a series of observations covering a few hundred metres, and they are made over a short time span. The key point is that the total number of observations per unit surface area are recorded. Hence each transect is represented by one value in the spreadsheet.

For the data exploration we first plot the geographical position of the transects, and we need to visualize whether, in each year, roughly the same areas were sampled. We also need to investigate whether the transects are equivalent in size.

Figure 6.1 shows the locations of the transects per sampling year. Each transect is represented by a point, but, because of the large number of observations (i.e. transects) made during a boat survey, it appears as though we have a continuous line, although that is not the case. It seems that in each year roughly the same areas were sampled. R code to make Figure 6.1 follows. First we convert the spatial coordinates of the transects from metres to kilometres. The `xyplot` from the `lattice` package does the plotting. The `aspect = "iso"` is crucial, as it ensures that units along the *x*-axis are the same as those along the *y*-axis.


```
> Gannets$Xkm <- Gannets$X / 1000
> Gannets$Ykm <- Gannets$Y / 1000
> xyplot(Ykm ~ Xkm | factor(Year), aspect = "iso",
         strip = strip.custom(bg = 'white',
         par.strip.text = list(cex = 1.2)),
         data = Gannets, pch=16, cex = 0.4, col = 1,
         xlab = list(label = "Xkm", cex = 1.5),
         ylab = list(label = "Ykm", cex = 1.5))
```

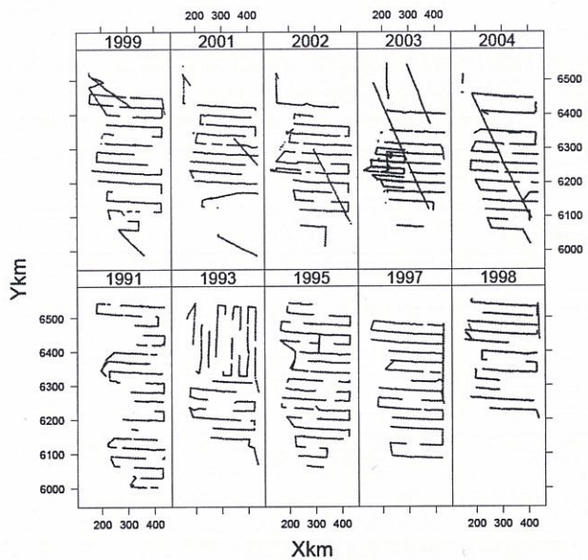

Figure 6.1. Spatial position of each transect per year.

The first challenge is that the sampling effort differs per transect. Sampling effort is quantified by the transect size. Figure 6.2 shows a Cleveland dotplot of the variable `Area_surveyedkm2`. Note that there are substantial size differences. It is crucial to take this into account in the analyses. For example, if we have a transect with a length of 100 m and another of 500 m, there is a high chance of recording greater abundance in the second transect. The question is whether we would expect 5 times as many birds. When analyzing density data (count divided by sampling effort), 10 birds in a 100 m transect is equivalent to 50 birds in a 500 meter transect; both have a density of 0.1/m. In a generalized linear (or additive) model in which the natural logarithm of `Area_surveyedkm2` is used as an offset variable, we are essentially doing the same (10 birds in a 100 m transect being equivalent to 50 birds in a 500 meter transect). However, if double effort does not imply double the number of birds, we can consider using sampling effort (`Area_surveyedkm2` in this case) as a covariate in the model. (See also Chapter 4). In this chapter we will use

Area_surveyedkm2 as an offset variable. A problem arises in that there are a few transects of size 0 (presumably the boat did not move). We will remove these observations from the analyses. R code to make the Cleveland dotplot in Figure 6.2 follows. We used the plot function in preference to the dotchart function, as the latter consumes a great deal of memory in the text editor (Word) used for this book.

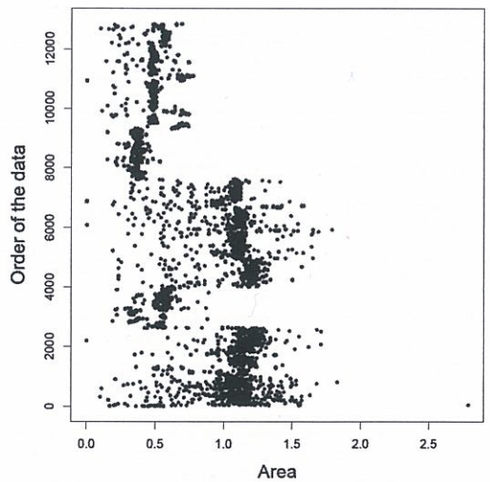

Figure 6.2. Cleveland dotplot illustrating the sizes of the transects.

```
> plot(x = Gannets$Area_surveyedkm2,
       y = 1:nrow(Gannets), pch = 16, cex = 0.8,
       xlab = "Area", ylab = "Order of the data")
```

To keep things orderly, we rename the variables. The object Gannets2 contains all data from the original object except those rows with a transect size of 0. The natural logarithm of Area_surveyedkm2 will be used as offset variable in the generalized additive model (GAM) that will be applied later, and TimeH is time in hours, on a continuous scale where 14.50 is 14.30 h.

```
> Gannets2 <- Gannets[Gannets$Area_surveyedkm2 > 0,]
> Gannets2$LArea <- log(Gannets2$Area_surveyedkm2)
> Gannets2$TimeH <- Gannets2$Hours +
                    Gannets2$Minutes/60
```

We also need to quantify the date using the variables Year, Month, and Day. The latter is the day of the month. R has some useful functions for manipulating time variables. The following code concatenates the time variables and calculates the Julian day within the year using the strptime function.

```
> Gannets2$Date <- paste(Gannets2$Day, Gannets2$Month,
                         Gannets2$Year, sep = "/")
> Gannets2$DayInYear <- strptime(Gannets2$Date,
                                 "%d/%m/%Y")$yday + 1
```


Note the capital Y after the %. This is essential; if we use a lowercase y, R will ignore the first two digits of the year. For some models and data sets it can be useful to have a variable that quantifies the number of days between the sampling date and a particular date (e.g. 1 January of the first sampling year). The following code calculates the number of days between sampling day and 1 January 1991:

```
> Gannets2$DaySince0 <- ceiling(julian(strptime(
                Gannets2$Date, format = "%d/%m/%Y"),
                origin = as.Date("1991-01-01")))
```

This variable can be used to make a time series graph of the number of gannets.

We now discuss potential models for the gannet abundances. We have data collected in 10 years from 1991 to 2004, all in June and July. Sampling took place at different times of the day between 04.00 and 20.00. Based on biological knowledge and common sense we expect the gannet abundance to differ among years, the days in a particular year, and over the course of a day. Obviously numbers will differ among transects. This means that we need a model of the form:

Gannet abundance = year effect + day in the year effect +
hour of the day effect + spatial effect +
sea state effect + size of transect effect.

In a more mathematical notation our model is as follows:

$$G_i \sim Poisson(\mu_i)$$
$$\log(\mu_i) = \alpha + f_1(Year_i) + f_2(Day_i) + f_3(Hour_i) + f_4(X_i, Y_i) + \beta \times SeaState_i + LSA_i$$

G_i is gannet abundance in the i^{th} observation (transect). The terms f_1 to f_4 are smoothing functions. Although sea state is a categorical variable with seven levels, we have preceded it with β. The software will give 6 estimated parameters. LSA (LArea in the R code) is the natural log-transformed area and will be used as an offset variable; hence we did not add a parameter. This model assumes that the day effect does not change over the year, the hour effect does not change over the months or years, and the spatial effect does not change over time.

Sampling took place only in June and July, so it is unlikely that the day effect will change during the survey due to differences in day length. This means that we will not use a 2-dimensional $f(\text{Day}_i, \text{Hour}_i)$ smoother. To allow for changes in the spatial patterning over time, we use a model of the form

$$G_i \sim Poisson(\mu_i)$$
$$\log(\mu_i) = \alpha + f_1(Year_i, X_i, Y_i) + f_2(Day_i) + f_3(Hour_i) + \beta \times SeaState_i + LSA_i$$

Due to the 3 dimensional smoother, this model will be extremely computing intensive, and perhaps using $f_1(Period_i, X_i, Y_i)$, where $Period_i$ separates the sampling years into 2 distinct groups, may be a more pragmatic approach. We

suggest starting with the simplest model and building it up gradually. As always, we first apply a data exploration, and in Section 6.5 we start the analysis with a model that contains only the year effect and add one covariate at a time, until computation time becomes an issue. The process of expanding term by term should give insight into the roles of the covariates. Note that this is not a forward model selection approach. It is a pragmatic 'let us try what is numerically possible' approach. Depending on the results, the outcome may indicate that it is time to buy a faster computer with more memory.

6.4 Data exploration

Figure 6.3 shows the gannet density versus spatial coordinates and year. The dot size is proportional to density. Figure 6.3 shows that the spatial patterns change over time, indicating that we may need a 3-dimensional smoother of the form $f(Period_i, X_i, Y_i)$ or $f(Year_i, X_i, Y_i)$. From a modelling standpoint, it would be best to base the decision to use a 3-dimensional smoother on the underlying biological questions, but software and hardware limitations introduce a data phishing element.

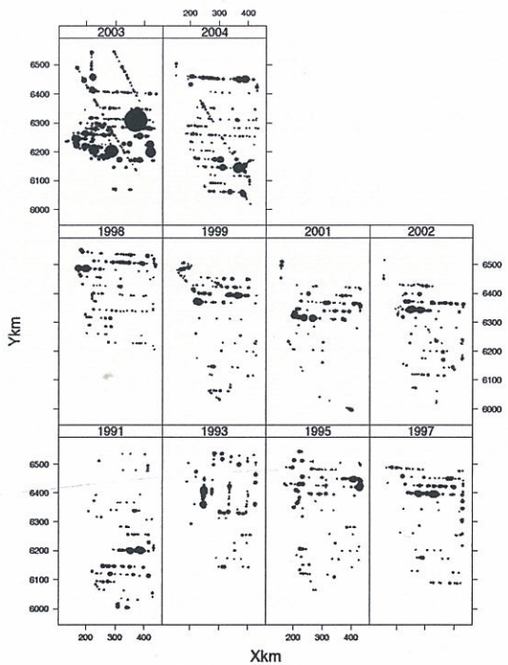

Figure 6.3. Gannet density plotted versus spatial coordinates and year. The size of a dot is proportional to the gannet density.

R code for Figure 6.3 follows. We first convert the response variable `Gannet_in_transect` to a shorter term, and then we calculate the density. The size of a dot corresponds to the `cex` value in the `xyplot`.

```
> Gannets2$G <- Gannets2$Gannets_in_transect
```



```
> Gannets2$GProp <- Gannets2$G /
                    Gannets2$Area_surveyedkm2
> MyCex <- 2 *sqrt(20 *Gannets2$GProp /
          max(Gannets2$GProp))
> xyplot(Ykm ~ Xkm | factor(Year), aspect = "iso",
       strip = strip.custom(bg = 'white',
       par.strip.text = list(cex = 1.2)),
       data = Gannets2, pch = 16, col = 1,
       layout = c(4,3), cex = MyCex,
       xlab = list(label = "Xkm", cex = 1.5),
       ylab = list(label = "Ykm", cex = 1.5))
```

Figure 6.4 shows a similar graph. We have plotted `DayInYear` versus `HourC` for each year, and the size of the dot is proportional to gannet density. This graph illustrates how the time of day effect changes over the days, at least during the final two years of sampling. It also shows that in 2003 we have a group of observations from a period that was not sampled in other years; in 2003 the data collection began 20 days earlier. One could argue that the 20 extra days means that we are no longer comparing like with like, and that these observations should be removed. We will keep them.

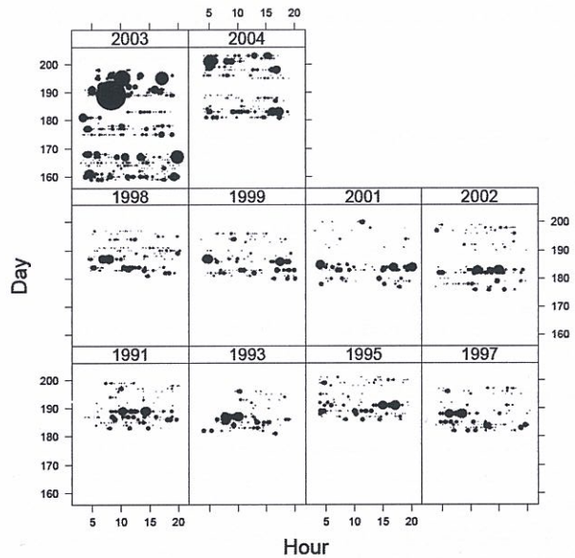

Figure 6.4. Gannet density plotted versus hour, day, and year. The size of a dot is proportional to the gannet density.

A further step in the data exploration is to look at zero-inflation of the response variable. In this case 81% of the observed values are 0. Such a high number is likely to cause problems, e.g. overdispersion. Zuur et al. (2012) argued to first apply a Poisson GLM (or GAM) and, depending on the results, apply zero-inflated models, add correlation, or change the distribution (e.g. use a negative binomial distribution).

Collinearity is not an issue with these covariates, unless the sea state differs among the sampling areas (e.g. rougher seas further from the coast) or if there are consistent changes in sea state within the season. Both these scenarios are plausible. Boxplots of `TimeH` conditional on `Seastate` and `DayInYear` conditional on `Seastate` do not show clear patterns, indicating that there are probably no major sources of collinearity.

The data set consist of 12,842 observations. In the next section we will begin with a Poisson distribution, and, to get an initial impression of its efficacy, we produce a Cleveland dotplot of the abundance (Figure 6.5). Note that some values are large. We have mentioned the zero-inflation aspect. Based on our experience, we predict that a zero-inflated GAM with a negative binomial distribution is needed. We mention the negative binomial distribution, because we expect the 6 larger values to cause overdispersion in a Poisson GAM. The zeros may also cause overdispersion. R code to make Figure 6.5 is analogous to that of Figure 6.2 and is not presented here.

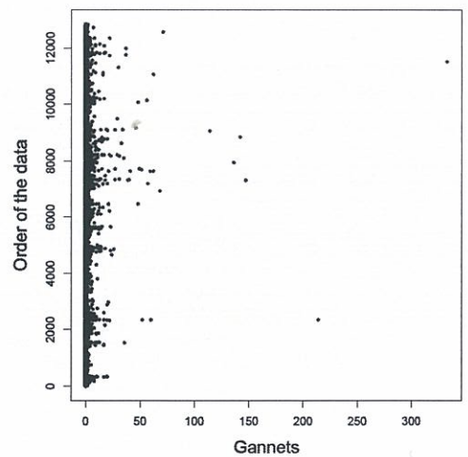

Figure 6.5. Cleveland dotplot of gannet abundance.

6.5 Building up the complexity of the GAMs

We stated that we expect to need a zero-inflated GAM with negative binomial distribution for the analysis of these data. Implementing such a model requires advanced coding and long computing time, so we prefer to start with a Poisson GAM and see how far it will take us. We start with a GAM rather than a GLM because, based on our experience, sea bird abundance will change non-linearly throughout the day, and also over months and years. We have sufficient unique time values during the day to use a smoother, `s(HourC)`, but with `s(Year)` we have to be more cautious as there are 'only' 10 years in the sampling period. Obviously, 10 years represents a huge effort in manpower, but, with respect to smoothing, 10 unique values for a covariate is minimal.

The first model we apply contains a year effect, and the natural logarithm of the sampled area is used as an offset variable. The underlying equation for this model is

$$\begin{aligned} G_i &\sim Poisson(\mu_i) \\ \log(\mu_i) &= \alpha + f_1(Year_i) + LSA_i \end{aligned} \tag{6.1}$$

R code to fit this model and assess the overdispersion is as follows:

```
> M1 <- gam(G ~ s(Year) + offset(LArea),
            family = poisson, data = Gannets2)
> E1 <- resid(M1, type = "pearson")
> Overdispersion <- sum(E1^2) / M1$df.res
> Overdispersion
```

```
36.94357
```

Overdispersion is calculated as the sum of squared Pearson residuals divided by sample size minus the number of parameters (Hilbe 2011). There is considerable overdispersion, but before concluding that we need to use a zero-inflated or negative binomial model, let us improve the model and add more covariates.

The next model contains a smoothing function for `Year`, and `Seastate` is used as a categorical variable.

$$\begin{aligned} G_i &\sim Poisson(\mu_i) \\ \log(\mu_i) &= \alpha + f_1(Year_i) + \beta \times Seastate_i + LSA_i + \varepsilon_i \end{aligned} \tag{6.2}$$

The overdispersion of this model is 29.8. R code to execute this model is

```
> M2 <- gam(G ~ s(Year) + factor(Seastate) +
            offset(LArea), family = poisson,
            data = Gannets2)
```

Next we add a smoothing function of `TimeH` to the model in Equation (6.2), but the overdispersion is only reduced to 26.7. The model is given by

$$\begin{aligned} G_i &\sim Poisson(\mu_i) \\ \log(\mu_i) &= \alpha + f_1(Year_i) + f_2(TimeH_i) + \beta \times Seastate_i + LSA_i \end{aligned} \tag{6.3}$$

R code to fit this model is

```
> M3 <- gam(G ~ s(Year) + s(TimeH) +
            factor(Seastate) + offset(LArea),
            family = poisson, data = Gannets2)
```

In the next model we add `DayInYear` as a smoother, resulting in overdispersion of 23.05. The model is given by

$$\begin{aligned} G_i &\sim Poisson(\mu_i) \\ \log(\mu_i) &= \alpha + f_1(Year_i) + f_2(TimeH_i) + f_3(DayInYear_i) + \beta \times Seastate_i + LSA_i \end{aligned} \tag{6.4}$$

Computing time on a reasonably fast computer is still in terms of seconds. R code to execute the model in Equation (6.4) is as follows:

```
> M4 <- gam(G ~ s(Year) + s(TimeH) +
            s(DayInYear) + factor(Seastate) +
            offset(LArea), family = poisson,
            data = Gannets2)
```

We now add a 2-dimensional smoother for the geographical coordinates and fit the following model:

$$\begin{aligned} G_i &\sim Poisson(\mu_i) \\ \log(\mu_i) &= \alpha + f_1(Year_i) + f_2(TimeH_i) + f_3(DayInYear_i) + \\ &\quad \beta \times Seastate_i + LSA_i + f_4(X_i, Y_i) \end{aligned} \tag{6.5}$$

R code to execute this model follows. There is an issue with respect to the smoother for $f_4(X_i,Y_i)$. Inside the GAM algorithm there is a roughness penalty term for each smoother. For the 2-dimensional smoother there is also a single roughness penalty. Obviously, each covariate in a 2-dimensional smoother should be equally important in determining the shape of the 2-dimensional smoother, and, therefore, a so-called non-isotropic smoother should be used, for example the tensor product smoother. The tensor product smoother (obtained using the `te` function) is useful if the covariates are expressed in different units or have a different scale. However, the help file of the `s` function says that it is inferior to the thin plate regression spline for well-scaled covariates. Our study area is more rectangular than square (Figure 6.1), and, although the spatial covariates use the same units (unlike latitude and longitude), we should perhaps use `te(Xi, Yi)` instead of `s(Xi, Yi)`, which uses the default thin plate regression spline. To recap, if covariates are in different units or scales, we use `te()` for a 2-dimensional smoother, otherwise we use `s()`. The following R code is used to fit the model in Equation (6.5):

```
> M5 <- gam(G ~ s(Year) + s(TimeH) + s(DayInYear) +
            te(Xkm, Ykm) + factor(Seastate) +
            offset(LArea),
            family = poisson, data = Gannets2)
```

Computing time for this model on a year-old laptop was 20 seconds, so we have room to add more terms. The overdispersion of the model in Equation (6.5) is 20.08.

In order to gain insight into whether the model in Equation (6.5) is stable, it is useful to determine if all the models produce similar smoothers. If a particular covariate has different smoothers for the various models, we suspect collinearity or other problems (e.g. dependence). The `Year`, `TimeH`, and `DayInYear` smoothers in Equation (6.5) show patterns similar to those from the previous models.

We also try a model of the form

$$\begin{aligned} G_i &\sim Poisson(\mu_i) \\ \log(\mu_i) &= \alpha + f_2(TimeH_i) + f_3(DayInYear_i) + \beta \times Seastate_i + LSA_i + \\ &\quad s_4(Year_i, X_i, Y_i) \end{aligned} \tag{6.6}$$

In this model, the 3-dimensional smoother for `Year`, `X`, and `Y` consumes 106 degrees of freedom, which is large. An option would be to specify an upper limit for the degrees of freedom by adding, for example, `k = 30` to the code below. Computing time for the model is considerable, and we will not explore it further. R code to execute it is

```
> M6 <- gam(G ~ s(TimeH) + s(DayInYear) +
            factor(Seastate) + offset(LArea) +
            te(Xkm, Ykm, Year, k = 30),
            family = poisson, data = Gannets2)
```

Extending the model in Equation (6.5) with a 2-dimensional smoother for `TimeH` and `DayInYear` produces overdispersion of 17.84, and, if we extend the model with a 2-dimensional `TimeH-Year` smoother, the overdispersion is 20.43. Since the overdispersion is calculated from Pearson residuals, this tells us that there is only marginal gain from using a 2-dimensional smoother `te(TimeH, Year)`, but it may be an option to add `te(TimeH, DayInYear)`. This model is given by

$$\begin{aligned} G_i &\sim Poisson(\mu_i) \\ \log(\mu_i) &= \alpha + f_1(Year_i) + f_2(TimeH_i, DayInYear_i) + \\ &\quad \beta \times Seastate_i + LSA_i + f_4(X_i, Y_i) \end{aligned} \tag{6.7}$$

and it is fitted in R with the code

```
> M7 <- gam(G ~ s(Year) + te(TimeH, DayInYear) +
            factor(Seastate) + offset(LArea) +
            te(Xkm, Ykm),
            family = poisson, data = Gannets2)
```

The problem is that this model contains two 2-dimensional smoothers, and for more complicated models computing time will become an issue. We will try to extend the model in Equation (6.7), but if computing time is extensive, we may have to simplify it.

To investigate the source of the overdispersion of 17.84 in the model in Equation (6.7), we extract the Pearson residuals and apply a detailed model validation. Figure 6.6 shows a scatterplot of the Pearson residuals versus fitted values. Note that there are various large residuals and, combined with the information we obtained from the Cleveland dotplot in Figure 6.5, we suspect that the large observed values are partly responsible for the overdispersion.

R code to produce Figure 6.6 follows. The `resid` function extracts the residuals and the `fitted` function the fitted values. The rest is elementary plotting code:

```
> E5 <- resid(M5, type = "pearson")
```

```
> F5 <- fitted(M5, type = "response")
> plot(x = F5, y = E5, xlab = "Fitted values",
       ylab = "Pearson residuals")
> abline(0, 0)
```

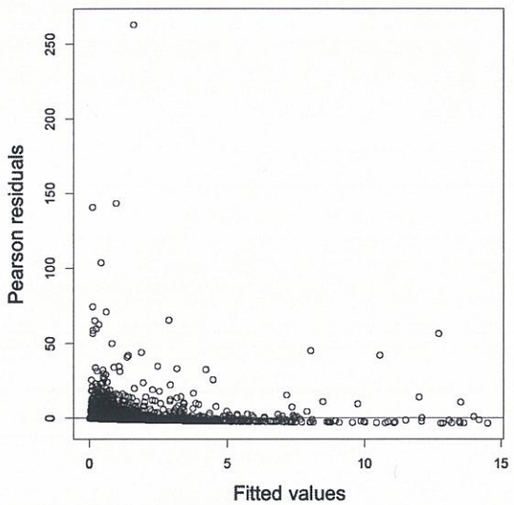

Figure 6.6. Pearson residuals plotted versus fitted values for the Poisson GAM in Equation (6.7).

The next logical step is to apply the negative binomial equivalent of the model in Equation (6.7). Before running the code and examining the numerical and graphic output, we write the mathematical form of the negative binomial GAM:

$$G_i \sim NB(\mu_i, k)$$

$$E(G_i) = \mu_i \quad \text{and} \quad \text{var}(G_i) = \mu_i + \frac{\mu_i^2}{k} = \mu_i + \gamma \times \mu_i^2 \qquad (6.8)$$

$$\log(\mu_i) = \alpha + f_1(Year_i) + f_2(TimeH_i, DayInYear_i) + \beta \times Seastate_i + LSA_i + f_4(X_i, Y_i)$$

The negative binomial GAM uses a quadratic variance. The extra quadratic term in the variance can also be written as $\gamma \times \mu_i^2$, which is more intuitive (Hilbe 2011), but this is not the way it is expressed in R. The code gives us k, which is called `theta` in the numerical output.

The GAM with the negative binomial distribution can be fitted with the following R code:

```
> M8 <- gam(G ~ s(Year) + te(TimeH, DayInYear) +
            factor(Seastate) + offset(LArea) +
            te(Xkm, Ykm),
            family = negbin(c(0.1, 1.5), link = log),
            data = Gannets2)
```


The argument `c(0.1,1.5)` in the `negbin` function specifies that the parameter k for the variance term is between 0.1 and 1.5. We chose this range to speed up the calculations. If the estimated value is on the boundary of this interval, we can extend the interval and rerun the code. Computing takes about an hour on a fast computer.

The overdispersion of the negative binomial GAM is 3.33, which is still too high. We will present all numerical and graphic output, as we can learn from it. The numerical output is presented first:

```
> print(summary(M8), digits = 3, signif.stars = FALSE)

Family: Negative Binomial(0.155)  Link function: log
Formula: G ~ s(Year) + factor(Seastate) +
             te(DayInYear, TimeH) + te(Xkm, Ykm) +
             offset(LArea)

Parametric coefficients:
                    Estimate Std. Error z value Pr(>|z|)
(Intercept)         -0.44123    0.10204  -4.324 1.53e-05
factor(Seastate)1    0.33377    0.12608   2.647  0.00812
factor(Seastate)2   -0.08727    0.11816  -0.739  0.46013
factor(Seastate)3    0.01131    0.11503   0.098  0.92166
factor(Seastate)4   -0.24687    0.12010  -2.056  0.03982
factor(Seastate)5   -0.15250    0.14281  -1.068  0.28558
factor(Seastate)6   -0.39298    0.18831  -2.087  0.03690

Approximate significance of smooth terms:
                       edf Ref.df Chi.sq p-value
s(Year)              8.434   8.90  106.5  <2e-16
te(DayInYear,TimeH) 23.654  23.98  207.8  <2e-16
te(Xkm,Ykm)         16.916  19.06  547.3  <2e-16

R-sq.(adj) =  0.0246   Deviance explained = 24.1%
UBRE score = -0.51618  Scale est. = 1   n = 12820
```

Condensed output is obtained using the `anova(M8)` command:

```
Family: Negative Binomial(0.155)
Parametric Terms:
                  df Chi.sq  p-value
factor(Seastate)   6   40.4 3.81e-07

Approximate significance of smooth terms:
                       edf Ref.df Chi.sq p-value
s(Year)              8.434   8.900  106.5  <2e-16
te(DayInYear,TimeH) 23.654 23.977  207.8  <2e-16
te(Xkm,Ykm)         16.916 19.062  547.3  <2e-16
```

All terms in the model are significant at the 5% level. The parameter k is equal to 0.155. For repeat runs of this model it saves computing time to set `theta` to this value and keep it fixed. UBRE was used to estimate the optimal amount of smoothing for each smoother.

A plot of the Pearson residuals versus fitted values is presented in Figure 6.7. The filled dark dots are Pearson residuals for which the observed gannet abundance is greater than 25. It seems that the model is not capable of fitting the observations with high abundance. Besides the large residuals, the excessive number of zeros may also have caused some of the overdispersion, and a zero-inflated negative binomial GAM is a potential follow-up analysis.

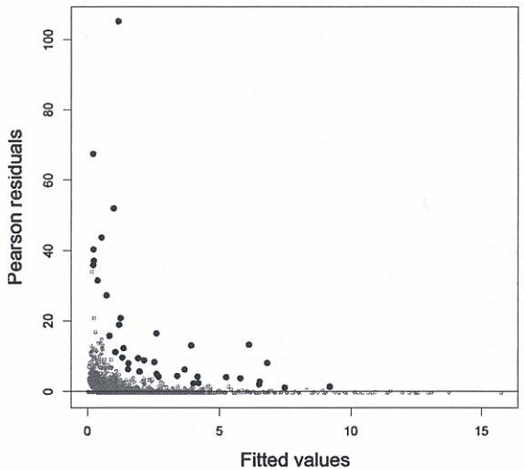

Figure 6.7. Pearson residuals plotted versus fitted values for the negative binomial GAM in Equation (6.8). The values plotted with a dark filled dot are observations for which the gannet abundance exceeds 25.

R code for Figure 6.7 follows. First we extract the Pearson residuals and the fitted values. A vector, `MyPch`, is created with the value 16 (filled dot) if the gannet abundance exceeds 25, and 1 (open circle) otherwise. The point size and colour are controlled in a similar way.

```
> E8 <- resid(M8, type = "pearson")
> F8 <- fitted(M8, type = "response")
> MyPch <- rep(1, length = nrow(Gannets2))
> MyPch[Gannets2$G > 25] <- 16
> MyCex <- rep(0.5, length = nrow(Gannets2))
> MyCex[Gannets2$G > 25] <- 1.2
> MyCol <- rep(grey(0.4), length = nrow(Gannets2))
> MyCol[Gannets2$G > 25] <- grey(0.2)
> plot(x = F8, y = E8, pch = MyPch, cex = MyCex,
       xlab = "Fitted values",
       ylab = "Pearson residuals")
> abline(0,0)
```

We also need to inspect the residuals for spatial correlation, and we use the variogram for this. Ideally the sample variogram should resemble a horizontal band of points, as this would indicate spatial independence. Spatial dependence is present if the variogram values increase at greater distances between sites. The

sample variogram for the Pearson residuals are presented in Figure 6.8. The pattern does not indicate any serious spatial correlation between sites that are separated by 5 km or less.

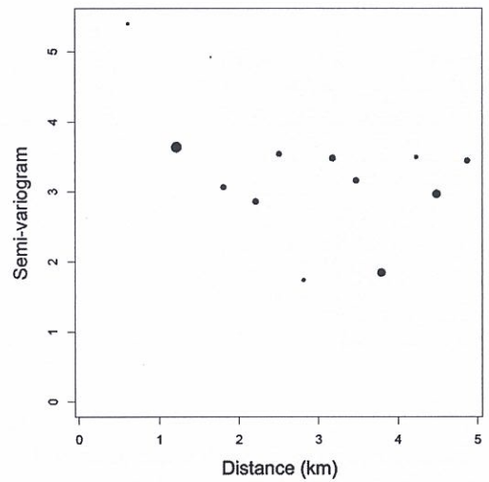

Figure 6.8. Sample variogram of Pearson residuals from the negative binomial GAM. Point size is proportional to the number of combinations of sites used for calculating the sample variogram at a particular distance.

R code to make Figure 6.8 follows. First a data frame `mydata` is created that contains the residuals and the spatial coordinates. The function `coordinates` from the `gstat` package converts the `Xkm` and `Ykm` variables to coordinates, and the sample variogram for distances up to 5 km is calculated. In the `plot` function, we make the dot size proportional to the number of transect combinations used for the calculations of the sample variogram at each distance.

```
> mydata <- data.frame(E8, Gannets2$Xkm, Gannets2$Ykm)
> coordinates(mydata) <- c("Gannets2.Xkm",
                           "Gannets2.Ykm")
> V12 <- variogram(E8 ~ 1, mydata, cutoff = 5,
                   robust = TRUE)
> plot(x = V12$dist, y = V12$gamma,
       xlab = "Distance (km)",
       ylab = "Semi-variogram", pch = 16,
       cex = 2 *V12$np / max(V12$np))
```

Figure 6.9, Figure 6.10 and Figure 6.11 show the estimated smoothers for `Year`, `te(DayInYear, TimeH)`, and `te(Xkm, Ykm)` obtained by the negative binomial GAM. The `Year` smoother shows a decrease to 1993 and an increase from 1993 to 1997 followed by a sharp decrease and another increase. The `te(DayInYear, TimeH)` smoother (daily and seasonal patterns) is difficult to interpret, but it clearly shows a change in the daily pattern during the season. The spatial smoother shows that higher abundances are obtained at transects in the south-western portion of the study area.

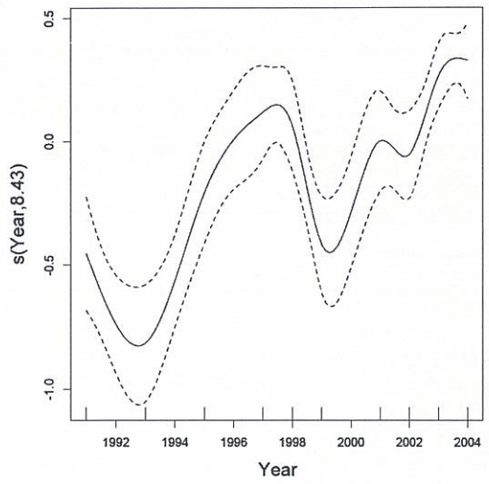

Figure 6.9. Estimated smoothers s(Year) for the negative binomial GAM.

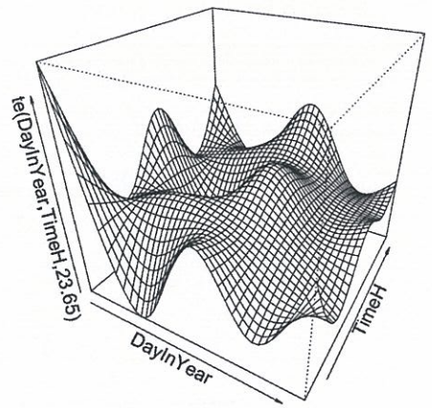

Figure 6.10. Estimated smoothers te(DayInYear, TimeH) for the negative binomial GAM.

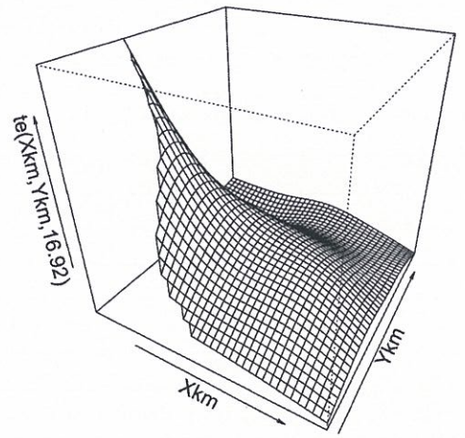

Figure 6.11. Estimated smoothers te(Xkm, Ykm) for the negative binomial GAM.

R code to make these three figures consists of elementary plotting commands. The `pers = TRUE` ensures that a 3-dimensional graph is produced.

```
> plot(M8, select=c(1), cex.lab = 1.5)
> plot(M8, select=c(2), cex.lab = 1.5, pers = TRUE)
> plot(M8, select=c(3), cex.lab = 1.5, pers = TRUE)
```

We also inspect the smoothers produced by the Poisson equivalent (model `M7`). The shapes of all smoothers are similar, with the only difference being the wider confidence bands in the negative binomial GAM.

6.6 Zero-inflated GAM

The computations in the previous section resulted in a negative binomial GAM that was still overdispersed. Hilbe (2011) and Zuur et al. (2012) discuss a range of possible causes for overdispersion that might apply here: outliers, missing covariates, missing interactions, wrong link function, high abundances with large variation, correlation that is not being modelled, and zero-inflation are the most likely causes. We could potentially have all these problems. However, as discussed in Zuur et al. (2012) spatial correlation may be confounded with zero-inflation; neighbouring transects may have zero gannet abundance. To take this into account we can either apply a GAM model with a spatial correlation structure (e.g. a so-called CAR correlation) or apply a zero-inflated GAM. A model that contains both components is likely to cause numerical problems, for instance non-mixing chains in Bayesian estimation techniques. This means that we have to choose between the two approaches, and, in our opinion, this choice should be based on biological arguments.

With respect to the statistical implementation, neither of the two options is appealing, as software is limited. Zero-inflated GAM can be covered in a *Beginner's Guide to GAM*; explaining a CAR correlation and showing how to fit it requires Bayesian techniques and is beyond the scope of this book. The interested reader is referred to Zuur et al. (2012). We therefore briefly define a zero-inflated model and implement a zero-inflated GAM.

6.6.1 A zero-inflated model for the gannet data

There are different types of zero-inflated models, and here we discuss the so-called mixture models. In a mixture model we make a distinction between false zeros and true zeros. False zeros are zeros recorded because sampling took place at the wrong time (e.g. in the winter when birds are absent), observer error (bad weather or poor visibility), or animal error (gannets should have been present but were not). True zeros indicate absence of animals because the habitat is not suitable. This distinction between true and false zeros is primarily for the purpose of providing a convenient ecological justification for applying a mixture model. Without accepting the true and false zero proposition, we can view the zero-inflated models as a means of dealing with overdispersion caused by an excessive number of zeros.

In a mixture model, two GLMs (or GAMs) are glued together; a Poisson (or negative binomial) GAM to model the non-zero counts and the true zeros and a (binary) binomial GLM or GAM to model the zeros that cannot be explained by the Poisson (count). If all zeros can be explained by the covariates in the count, the binary part automatically disappears. The zeros that cannot be explained by the covariates in the count component have a high probability of being false zeros in the binary component of the model.

Viewed simplistically, the zero-inflated Poisson (ZIP) GAM models the observed abundances as a function of covariates. A zero that does not comply with this model is labelled 'highly likely a false zero.' A diagram of a zero-inflated model is given in Figure 6.12.

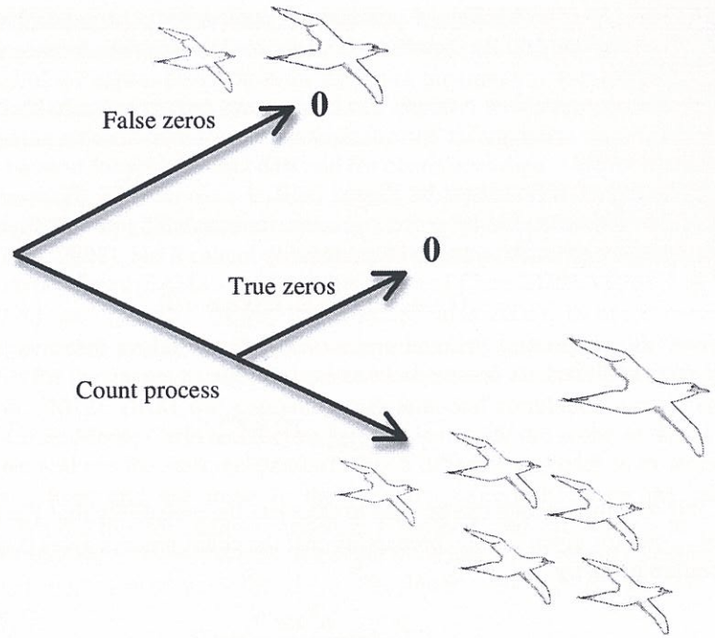

Figure 6.12. Sketch of a zero-inflated model.

We briefly review the underlying mathematics of a zero-inflated Poisson (ZIP) GAM. Recall from Chapter 4 that a Poisson GAM is defined by the following set of equations:

$$
\begin{aligned}
&G_i \sim Poisson(\mu_i)\\
&\log(\mu_i) = \alpha + f_1(Year_i) + f_2(TimeH_i) + f_3(DayInYear_i) + \\
&\qquad \beta \times Seastate_i + LSA_i + f_4(X_i, Y_i)\\
&E(G_i) = \mu_i \quad \text{and} \quad \text{var}(G_i) = \mu_i
\end{aligned}
$$

Instead of this model, we can use a model with two 2-dimensional smoothers. The observed counts, G_i, are assumed to follow a Poisson distribution with mean μ_i, and the log of μ_i is modelled as a function of the covariates. Because we use a Poisson distribution, the mean and the variance are equal to μ_i. This model was applied in previous sections. This modelling approach assumes that G_i is a non-negative count. If G_i is a binary variable representing presence or absence of gannets, then a binomial GAM (or GLM) should be used, which is defined by

$$
\begin{aligned}
&G_i \sim Bin(\pi_i)\\
&\text{logit}(\pi_i) = \alpha + f_1(Year_i) + f_2(TimeH_i) + f_3(DayInYear_i) + \\
&\qquad \beta \times Seastate_i + LSA_i + f_4(X_i, Y_i)\\
&E(G_i) = \pi_i \quad \text{and} \quad \text{var}(G_i) = \pi_i \times (1 - \pi_i)
\end{aligned}
$$

Here π_i is the probability of presence of gannets for observation (transect) i. The expressions for the mean and variance are derived from a binomial distribution.

In our example, G_i is a count, but it has many zeros; hence neither of these distributions is suitable. We now show how we can combine them to form the ZIP distribution.

Following the flowchart in Figure 6.12, a zero can be obtained by two processes. Either the binary process generates a zero with a probability π_i, or the count process generates a zero with probability

$$(1 - \pi_i) \times P(\text{Count process} = 0).$$

How do we quantify $P(\text{Count process} = 0)$? If we assume that a variable Y_i is Poisson distributed, its density function is given by

$$P(Y_i; \mu_i) = \frac{\mu_i^{Y_i} \times e^{-\mu_i}}{Y_i!}$$

This density function can be used to calculate the probability that $Y_i = 0$, $Y_i = 1$, $Y_i = 2$, etc. for given μ_i. The probability that the count process gives 0 gannets is therefore given by

$$P(0; \mu_i) = \frac{\mu_i^0 \times e^{-\mu_i}}{0!} = e^{-\mu_i}$$

Hence the probability that $G_i = 0$ is modelled by the ZIP as

$$\pi_i + (1 - \pi_i) \times e^{-\mu_i}$$

The positive counts are modelled using the Poisson distribution. Using standard mathematical tools, expressions for the mean and the variance of a ZIP can be derived (see Zuur et al. 2012). The full ZIP GAM is then specified by

$$\begin{aligned}
&G_i \sim ZIP(\mu_i, \pi) \\
&\log(\mu_i) = \alpha + f_1(Year_i) + f_2(TimeH_i) + f_3(DayInYear_i) + \\
&\qquad \beta \times Seastate_i + LSA_i + f_4(X_i, Y_i) \\
&\text{logit}(\pi) = \gamma \\
&E(G_i) = (1 - \pi_i) \times \mu_i \quad \text{and} \quad \text{var}(G_i) = (1 - \pi_i) \times (\mu_i + \pi_i \times \mu_i^2)
\end{aligned}$$

If the probability of a false zero is 0, $\pi_i = 0$ and we obtain an ordinary Poisson GAM. It is the task of the scientist to choose which covariates are used for the log-link function and which for the logistic link function. An option is to model the mean of the count process as a function of all covariates, as we do above, and also model the probability of a false zero as a function of all covariates, which we do not do above. Sea state is the most obvious candidate to be used in the binary part; poor weather conditions may cause false zeros. But such a model is more complex, and it may be better to start with an intercept-only in the logistic link function. Zuur et al. (2009a; 2012) state that the choice of the covariates to include in each component of the ZIP should be based on biological arguments.

Instead of a Poisson distribution in the count process, one can use a negative binomial distribution leading to zero-inflated negative binomial GLM or GAM. The expression for the mean and the variance will change. The mixture models can also be used for proportional data and for continuous data.

Ordinary ZIP GLM or zero-inflated negative binomial (ZINB) GLM can be executed in R with the function `zeroinfl` available in the package `pscl` (Zeileis et al. 2008), but it cannot cope directly with smoothers. Packages that can execute zero-inflated GAMs are `COZIGAM` (Liu and Chan 2010), `VGAM` (Yee and Wald 1996), and `gamlss` (Rigby and Stasinopoulos 2005). In our experience `VGAM` is a useful package, but at times is confusing to work with. We tried `COZIGAM` for the gannet data but encountered numerical estimation problems. In Zuur et al. (2012), GAM was combined with temporal correlation models using Markov Chain Monte Carlo techniques, but that is outside the scope of this book. Instead we will use the package `gamlss`. Yet a different approach is to program our own splines and use these in the function `zeroinfl` from the `pscl` package. We did this for additive models in Chapter 3, and extending it to GLM and GAM models is simple. We are not eager to program and explain a spline for a 2-dimensional tensor smoother in a 'Beginner's Guide,' so we will try `gamlss`.

6.6.2 ZIP GAM using gamlss

The package `gamlss` is not part of the R base installation, so it needs to be installed. The website http://www.gamlss.org/ contains a wealth of information, help files, and sample code.

One of the major advantages of `gamlss` is that it can deal with a much wider range of distributions than the `glm` function in R or the `gam` function in `mgcv`. Also, `gamlss` allows modelling the variance as a function of covariates, but we will not make use of this for the gannet example.

To fit the Poisson GAM as in Equation (6.5) we use the following code:

```
> library(gamlss)
> library(gamlss.add)
> M9 <- gamlss(G ~ cs(Year, df = 8) +
                   cs(TimeH, df = 8) +
                   cs(DayInYear, df = 8) +
                   ga(~s(Xkm,Ykm, fx = TRUE, k=28)) +
                   factor(Seastate) + offset(LArea),
                   family = PO(), data = Gannets2)
```

The `cs(Year, df =8)` implements a cubic spline with 8 degrees of freedom. The package does not do cross-validation as in `mgcv`, but it does allow the user to specify multiple models (e.g. with different degrees of freedom) and compare them via the AIC. Other types of smoothers are available. The function `gamlss` cannot do 2-dimensional smoothers, but an advantage is the function `ga()` that allows for an interface to the function `gam` from `mgcv`. Hence

`ga(~s(Xkm, Ykm, fx = TRUE, k = 28))` applies a 2-dimensional smoother using the `s()` function from `mgcv`. We also tried the `te` function, but this gave a list of warning and error messages. The problem is that `s(Xkm, Ykm)` assumes that both variables have the same scale and variation. Rescaling them to between 0 and 1 is not an option, as it will distort the spatial position of the sites, so we are faced with a challenge.

We choose the degrees of freedom based on the `gam` results presented earlier. For alternatives to such an approach (first `mgcv` then `gamlss`), see the help files of `addterm`, `dropterm`, and `stepGAIC`. We could follow an old-fashioned approach and start with 4 degrees of freedom, extract residuals, and assess them for patterns. If there are patterns, we need to increase the degrees of freedom. Computing time for the ZIP GAM discussed below is about an hour, so the number of models to compare may depend on one's level of patience.

In `gamlss`, the functions to obtain the estimated smoothers are different from those of `mgcv`; `gamlss` does not use deviance or Pearson residuals but randomized quantile residuals. It is not difficult, however, to obtain Pearson residuals in `gamlss`.

The function `plot(M9)` gives model validation graphs, and `termplot(M9)` shows the estimated smoothers. Instead of demonstrating the various `gamlss` functions, we refer the reader to the manual and continue with the ZIP GAM. It is executed with the following code:

```
> con <- gamlss.control(n.cyc = 200)
> M10 <- gamlss(G ~ cs(Year, df = 8) +
                    cs(TimeH, df = 8) +
                    cs(DayInYear, df = 8) +
                    ga(~s(Xkm,Ykm, fx = TRUE, k=28)) +
                    factor(Seastate) + offset(LArea),
                    family = ZIP(), data = Gannets2,
                    control = con)
```

The model converges after about 50 iterations. The default setting stops at 20 iterations; hence the `gamlss.control` code.

We first investigate whether we really need the ZIP model. We have mentioned that if π equals 0, the ZIP GAM becomes a Poisson GAM. In the current model this will occur if the intercept in the logistic link function is highly negative. The following code extracts the estimated π:

```
> Pi <- fitted(M10, "sigma")[1]
> Pi
```

```
0.7508386
```

This means the probability that a 0 is a false zero is 0.75, a high value. We next assess whether the ZIP GAM is overdispersed. A rapid method is to take the Pearson residuals, square them, add them together, and divide by the residual degrees of freedom. In order to calculate the Pearson residuals, we need to calculate the mean and the variance of the ZIP. The required equations for the mean and variance were given in the previous subsection.


```
> Pi <- M10$sigma.fv[1]
> mu <- M10$mu.fv
> ExpY <- (1 - Pi) * mu
> varY <-  mu * (1 - Pi) * (1 + mu * Pi)
> PearsonRes <- (s2$G - ExpY) / sqrt(varY)
> N <- nrow(Gannets2)
> p <- M10$df.fit
> Overdispersion <- sum(PearsonRes^2) / (N - p)
> Overdispersion
```

```
4.691985
```

The result shows that the ZIP GAM is still overdispersed, which is surprising, as we expect that we have high bird abundances (which require a negative binomial distribution) and zero-inflation.

At the time of writing this book the following model gave errors messages, but perhaps it will work by the time you read it. We attempted to implement two 2-dimensional tensor smoothers.

```
> M11 <- gamlss(G ~ cs(Year, df = 8) +
                    ga(~te(TimeH, DayInYear, fx = TRUE, k=28))
                    ga(~s(Xkm,Ykm, fx = TRUE, k=28)) +
                    factor(Seastate) + offset(LArea),
                    family = ZIP(), data = Gannets2)
```

The next step is to use a zero-inflated negative binomial GAM to reduce the overdispersion of 3.33 obtained by the negative binomial GAM. The function `gamlss` is not currently capable of executing a zero-inflated negative binomial distribution. This is unfortunate, as we are quite sure that this model would have been our end point.

6.7 Discussion

We began this chapter with a simple Poisson GAM and extended the model step by step. The shape of the smoothers changed only minimally each time a new covariate was added. We switched mid-analysis from a Poisson distribution to a negative binomial distribution, but the resulting GAM was still overdispersed. The most likely reasons for overdispersion were missing covariates or interactions, dependency among the observations, large variation, and/or zero-inflation.

We do not think that we are dealing with inherent correlation that would result from gannets following, or circling around, the boat and being counted multiple times. These are birds that fly like a rocket in a straight line. In a forthcoming book on generalized additive mixed effects models (GAMM) we will analyse data of species that do follow a boat. We will show how a short-range correlation structure can be added to the zero-inflated GLM or GAM.

With respect to the overdispersion, it is likely that we have missed an interaction term. Perhaps we should allow for a change in the spatial smoother over time, but computing time becomes a real issue. It would be helpful to have

formulated an *a priori* question: Did the spatial patterns differ between the first 5 and the final 5 years? In that case, the `by` argument can be used inside the `s(Xi,Yi)` smoother, and we can allow for two spatial smoothers and test whether such a model is better than one with a single spatial smoother. Without such an *a priori* question, any attempt would be data phishing.

The remaining two plausible causes for overdispersion are large variation and zero-inflation. We were able to fit a ZIP GAM, but we need software for a zero-inflated negative binomial GAM. In previous chapters we programmed simple quadratic splines and fitted the resulting additive model with the `lm` function. The same can be done with GAM and the `glm` function by programming a 2-dimensional smoother. The resulting covariates for the splines can be used in the `glm` function or in the `zeroinfl` function for ZIP GLM from the `pscl` package. In our follow-up book on GAMM we will show how to program a 2-dimensional tensor smoother.

6.8 What to present in a paper

The analyses are not complete, so there is no scope for presenting results at this point. What would we do next? Given the fact that the complete data set contains an additional 10–15 seabird species, some of which will show dependency (e.g. follow the boat) we would do the following:

1. Adopt a quadratic regression spline or low rank thin plate regression spline for each smoother and obtain the resulting covariates for each smoother using functions presented in Chapter 3.
2. Use these covariates in the function `glm` or `zeroinfl` to fit the GAM and use the results for baseline comparison.
3. Implement the same model in WinBUGS or OpenBUGS and compare results.
4. Extend the model with an error term and add a CAR residual correlation to the GAM to allow for short-range correlation. We can also allow for zero-inflation.

All components of the approach outlined in step 4 are given in Zuur et al. (2012) and will be included in our forthcoming book on GAMM.