GAM case study: Portugese larks

STA303/1002 Winter 2022

This example based on a talk by Gavin Simpson.

There is an optional video from last year talking through some of these functions here.

Get the data

The `gamair` package is the package for the GAM book by Wood (2016), see the syllabus for full reference.

```
# install.packages("gamair")
library(gamair)
# install.packages("gratia")
library(gratia)
library(tidyverse)
library(ggthemes)
library(mgcv)
data(bird)
glimpse(bird)
```

```
## Rows: 25,100
## Columns: 6
## $ QUADRICULA <fct> NG56, NG56, NG56, NG56, NG56, NG66, NG66, NG66, NG66, NG66,~
## $ TET        <fct> E, J, P, U, Z, E, J, P, U, Z, D, I, N, T, Y, D, I, N, T, Y,~
## $ crestlark  <dbl> NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA,~
## $ linnet     <dbl> NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA,~
## $ x          <dbl> 551000, 553000, 555000, 557000, 559000, 561000, 563000, 565~
## $ y          <dbl> 4669000, 4669000, 4669000, 4669000, 4669000, 4669000, 46690~
```

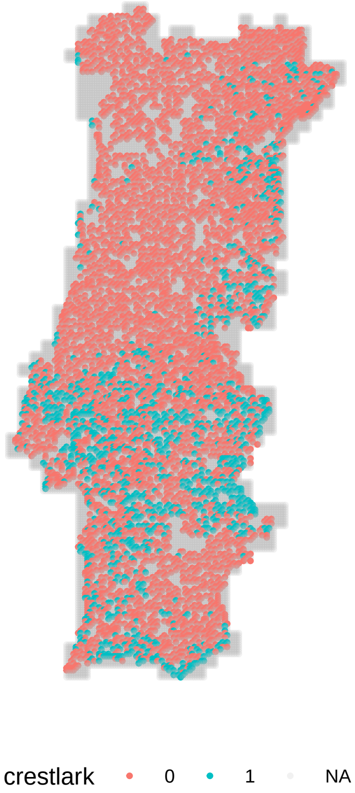
We're going to transform things a little to make them easier to use. You can find the description of each variable by running `?bird` in your console. It will open the help file. We want to scale the location to be in 1000s of kilometres.

```
bird <- transform(bird,
            crestlark = factor(crestlark),
            linnet = factor(linnet),
            e = x / 1000,
            n = y / 1000)
head(bird)
```

```
##       QUADRICULA TET crestlark linnet      x       y   e    n
## 13705       NG56   E      <NA>   <NA> 551000 4669000 551 4669
## 13710       NG56   J      <NA>   <NA> 553000 4669000 553 4669
## 13715       NG56   P      <NA>   <NA> 555000 4669000 555 4669
## 13720       NG56   U      <NA>   <NA> 557000 4669000 557 4669
## 13725       NG56   Z      <NA>   <NA> 559000 4669000 559 4669
## 13880       NG66   E      <NA>   <NA> 561000 4669000 561 4669
```

Plot it!

```
ggplot(bird, aes(x = e, y = n, colour = crestlark)) +
  geom_point(size = 0.5) +
  coord_fixed() +
  scale_colour_discrete(na.value = '#bbbbbb33') +
  labs(x = NULL, y = NULL) +
  theme_map() +
  theme(legend.position = "bottom")
```



Binomial GAM

```
crest <- gam(crestlark ~ s(e, n, k = 100),
             data = bird,
             family = binomial,
             method = 'REML')
```

$s(e, n)$ indicated by `s(e, n)` in the formula. Our default is thin plate splines, which is a pretty good default.

Recall that `k` sets size of basis dimension; upper limit on EDF.

Smoothness parameters estimated via REML.

```
summary(crest)
```

```
## 
## Family: binomial 
## Link function: logit 
## 
## Formula:
## crestlark ~ s(e, n, k = 100)
## 
## Parametric coefficients:
##             Estimate Std. Error z value Pr(>|z|)    
## (Intercept) -2.24184    0.07785   -28.8   <2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
## 
## Approximate significance of smooth terms:
##          edf Ref.df Chi.sq p-value    
## s(e,n) 74.04  86.46  857.3  <2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
## 
## R-sq.(adj) =  0.234   Deviance explained = 25.9%
## -REML = 2499.8  Scale est. = 1         n = 6457
```

Model checking with binary data is a pain with binomial models because our residuals look weird!

Alternatively we can aggregate data at the `QUADRICULA` level & fit a binomial *count* model.

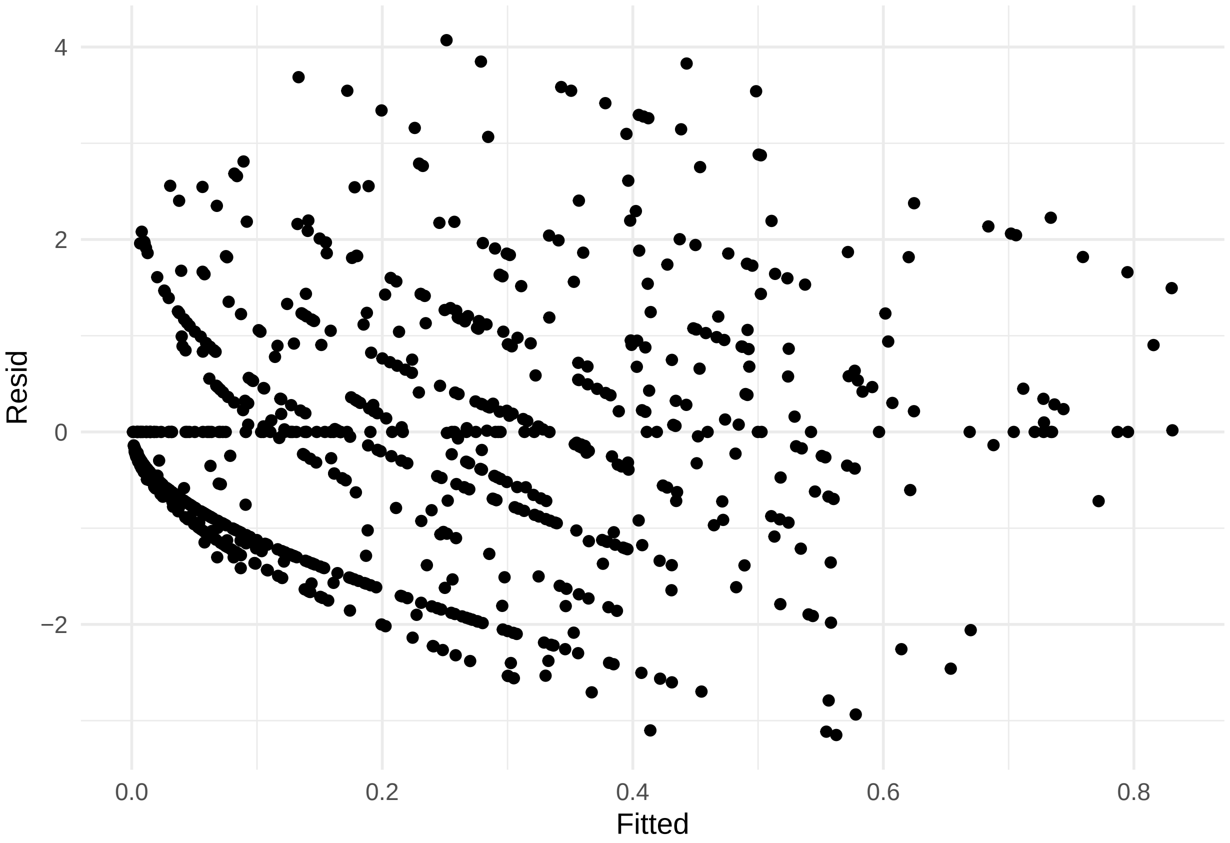
```
## convert back to numeric
bird <- transform(bird,
                  crestlark = as.numeric(as.character(crestlark)),
                  linnet = as.numeric(as.character(linnet)))
## some variables to help aggregation
bird <- transform(bird, tet.n = rep(1, nrow(bird)),
                  N = rep(1, nrow(bird)), stringsAsFactors = FALSE)
## set to NA if not surveyed
bird$N[is.na(as.vector(bird$crestlark))] <- NA
## aggregate
bird2 <- aggregate(data.matrix(bird), by = list(bird$QUADRICULA),
                   FUN = sum, na.rm = TRUE)
## scale by Quads aggregated
bird2 <- transform(bird2, e = e / tet.n, n = n / tet.n)
## fit binomial GAM
crest2 <- gam(cbind(crestlark, N - crestlark) ~ s(e, n, k = 100),
              data = bird2, family = binomial, method = 'REML')
```

Model checking

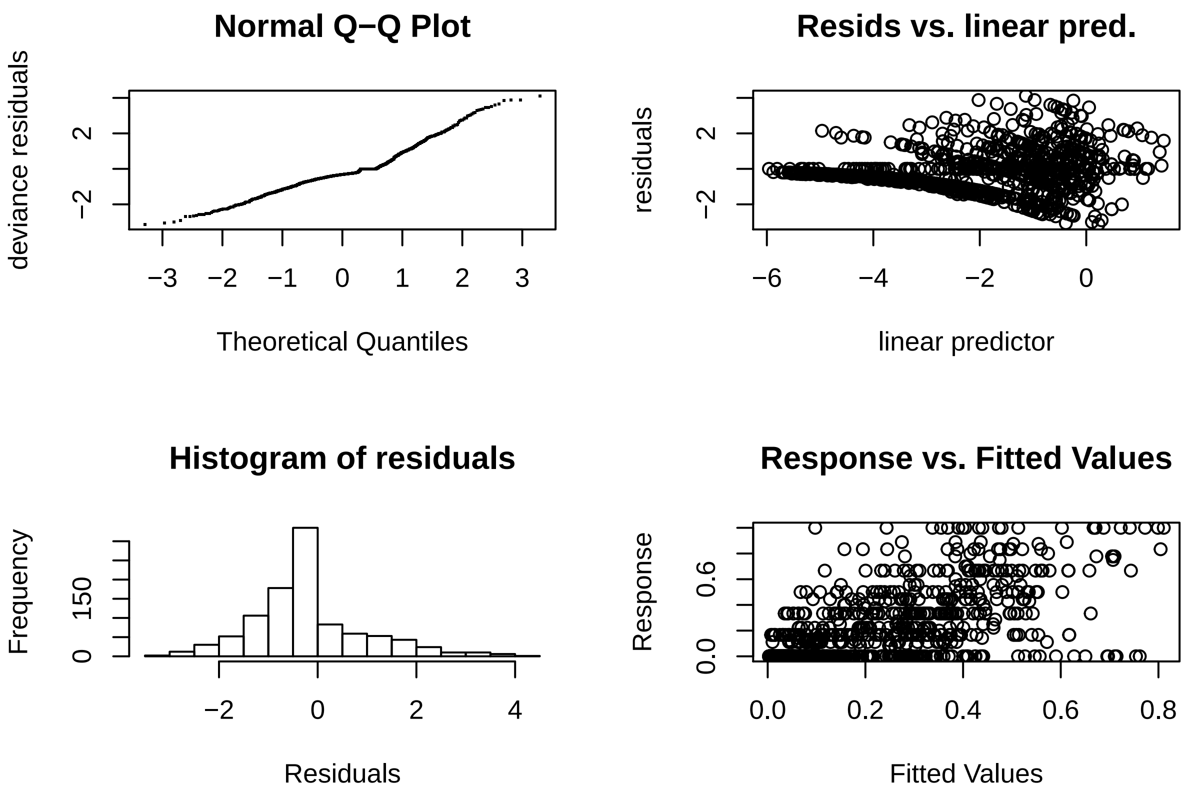
```
crest3 <- gam(cbind(crestlark, N - crestlark) ~
                  s(e, n, k = 100),
              data = bird2, family = quasibinomial,
              method = 'REML')
```

Model residuals don't look too bad. The bands of points we see are due to working with integers. Some overdispersion, $\phi = 2.32$

```
ggplot(data.frame(Fitted = fitted(crest2),
                  Resid = resid(crest2)),
       aes(Fitted, Resid)) +
  geom_point() +
  theme_minimal()
```

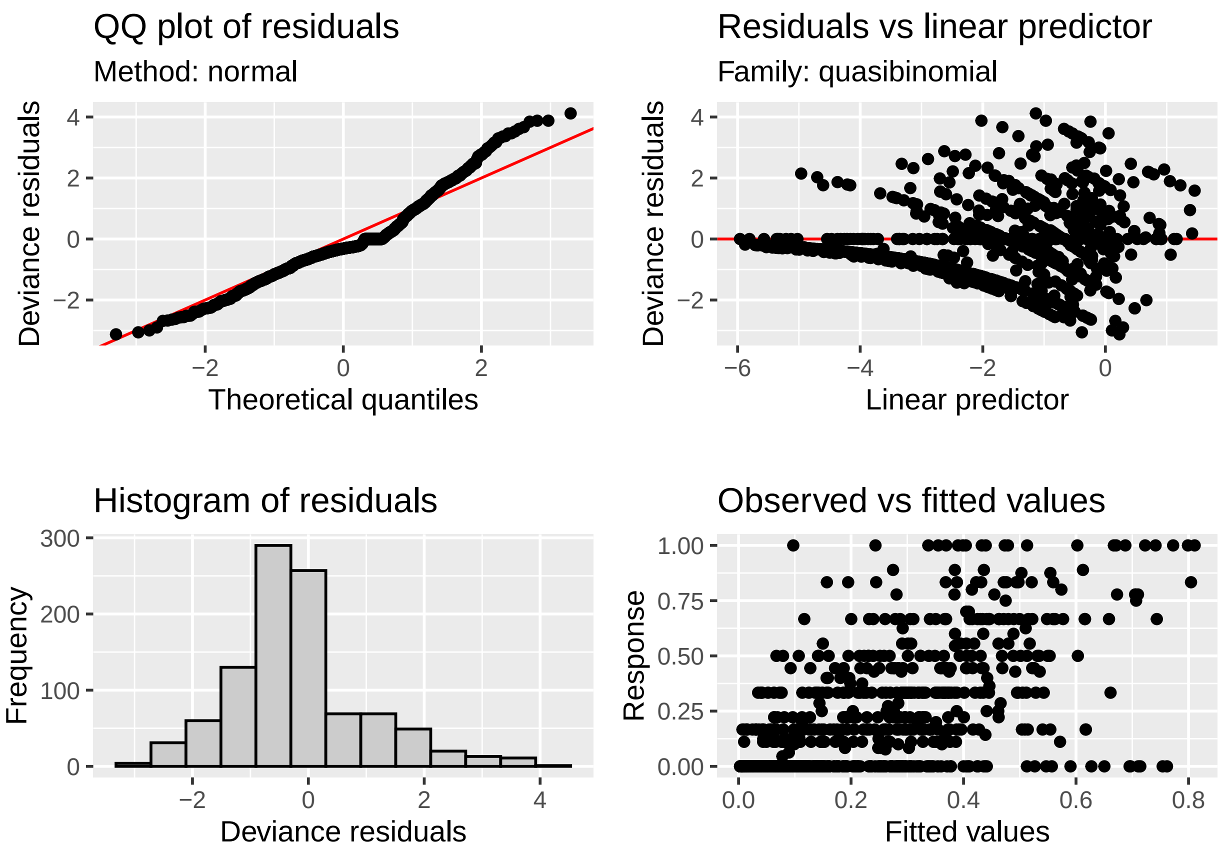


```
# gam.check
gam.check(crest3)
```



```
##
## Method: REML   Optimizer: outer newton
## full convergence after 6 iterations.
## Gradient range [-4.198304e-05,1.018993e-06]
## (score -125.1791 & scale 2.320408).
## Hessian positive definite, eigenvalue range [14.75829,456.8747].
## Model rank =  100 / 100
##
## Basis dimension (k) checking results. Low p-value (k-index<1) may
## indicate that k is too low, especially if edf is close to k'.
##
##          k'  edf k-index p-value
## s(e,n) 99.0 62.9    1.02    0.95
```

```
# ggplot style
appraise(crest3)
```


QQ plot of residuals
Method: normal
Deviance residuals
4
2
0
−2
−2
0
2
Theoretical quantiles
Residuals vs linear predictor
Family: quasibinomial
Deviance residuals
4
2
0
−2
−6
−4
−2
0
Linear predictor
Histogram of residuals
Frequency
300
200
100
0
−2
0
2
4
Deviance residuals
Observed vs fitted values
Response
1.00
0.75
0.50
0.25
0.00
0.0
0.2
0.4
0.6
0.8
Fitted values