STAT 8020 R Session 7: Logistic and Poisson Regression

Whitney Huang, Clemson University

Contents

Logistic Regression: Horseshoe Crab Mating

Data Source: Brockmann, H. J. (1996). Satellite male groups in horseshoe crabs, Limulus polyphemus. *Ethology*, 102(1), 1-21.

Load the data

This dataset is obtained from the website of *Alan Agresti*, Distinguished Professor Emeritus at the University of Florida.

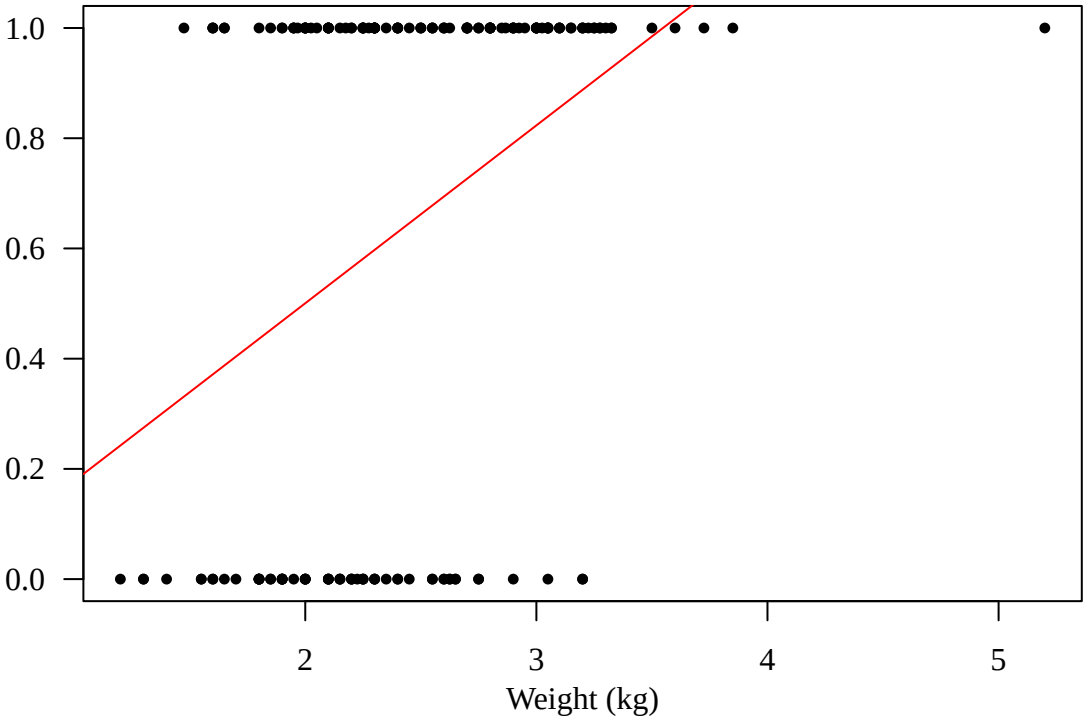
```
# Read Horseshoe Crab data from GitHub
crab <- read.table(
  "https://raw.githubusercontent.com/alanagresti/categorical-data/master/Crabs.dat",
  header = TRUE
)
```

Fit a Simple Linear Regression

As a starting point, we fit a simple linear regression model using `weight` as the predictor variable.

Although the response variable is binary, this provides a useful comparison to logistic regression and illustrates some limitations of linear regression for binary outcomes.

```
lmFit <- lm(y ~ weight, data = crab)
par(las = 1, mar = c(3.5, 3.5, 1, 0.5), mgp = c(2, 1, 0), family = "serif")
with(crab, plot(weight, y, pch = 16, cex = 0.75, xlab = "", ylab = ""))
mtext("Weight (kg)", side = 1, line = 2)
abline(lmFit, col = "red")
```



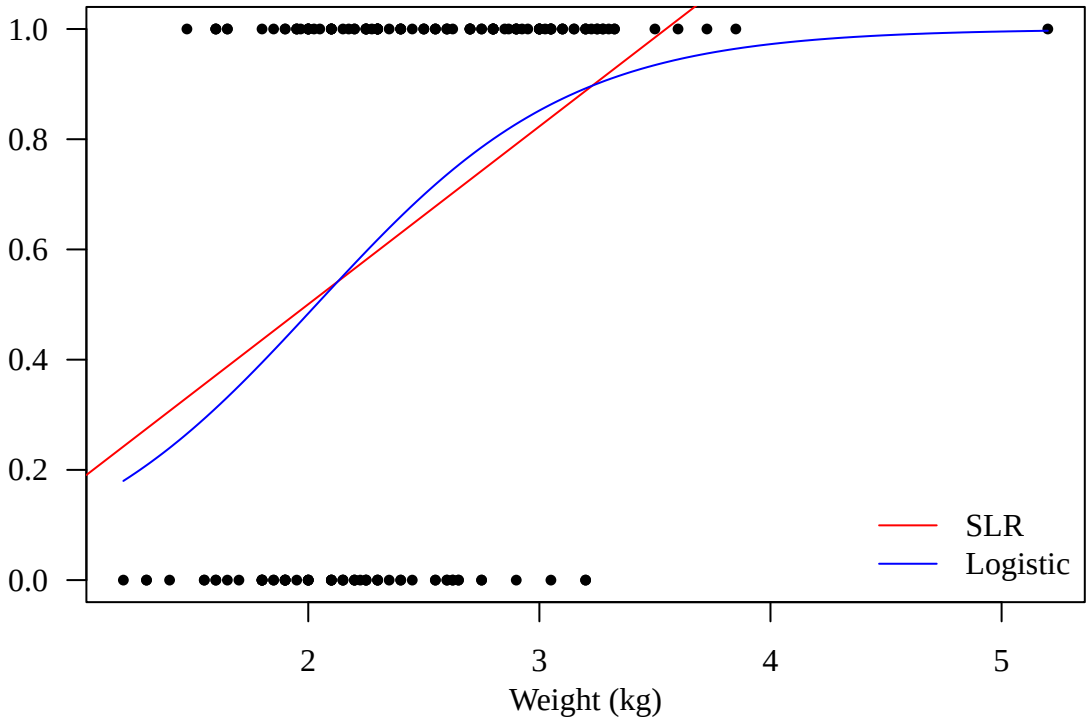
Fit a Logistic Regression

```
logitFit <- glm(y ~ weight, data = crab, family = "binomial")
summary(logitFit)
```

```
##
## Call:
## glm(formula = y ~ weight, family = "binomial", data = crab)
##
## Coefficients:
##             Estimate Std. Error z value Pr(>|z|)
## (Intercept)  -3.6947     0.8802  -4.198 2.70e-05 ***
## weight        1.8151     0.3767   4.819 1.45e-06 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

```
##
## (Dispersion parameter for binomial family taken to be 1)
##
##     Null deviance: 225.76  on 172  degrees of freedom
## Residual deviance: 195.74  on 171  degrees of freedom
## AIC: 199.74
##
## Number of Fisher Scoring iterations: 4
```

```r
# Generate predicted probabilities from the fitted logistic model
# over a fine grid of weight values for smooth visualization
rg <- range(crab$weight)
xg <- seq(rg[1], rg[2], 0.01)
# Compute predicted probabilities on the response scale
pred <- predict(logitFit, newdata = data.frame(weight = xg), type = "response")
par(las = 1, mar = c(3.5, 3.5, 1, 0.5), mgp = c(2, 1, 0), family = "serif")
plot(crab$weight, crab$y, pch = 16, cex = 0.75, xlab = "", ylab = "")
mtext("Weight (kg)", side = 1, line = 2)
abline(lmFit, col = "red"); lines(xg, pred, col = "blue")
legend("bottomright", legend = c("SLR", "Logistic"),
       col = c("red", "blue"), lty = 1, bty = "n")
```



Confidence Intervals

```r
# Construct confidence interval using the asymptotic normal approximation
est <- summary(logitFit)$coefficients
est
```

```
##             Estimate Std. Error   z value     Pr(>|z|)
```

```
## (Intercept) -3.694726  0.8801975 -4.197611 2.697457e-05
## weight       1.815145  0.3766959  4.818594 1.445736e-06
```

```
(CI_norm <- est[2, 1] + c(-1, 1) * qnorm(0.975) * est[2, 2])
```

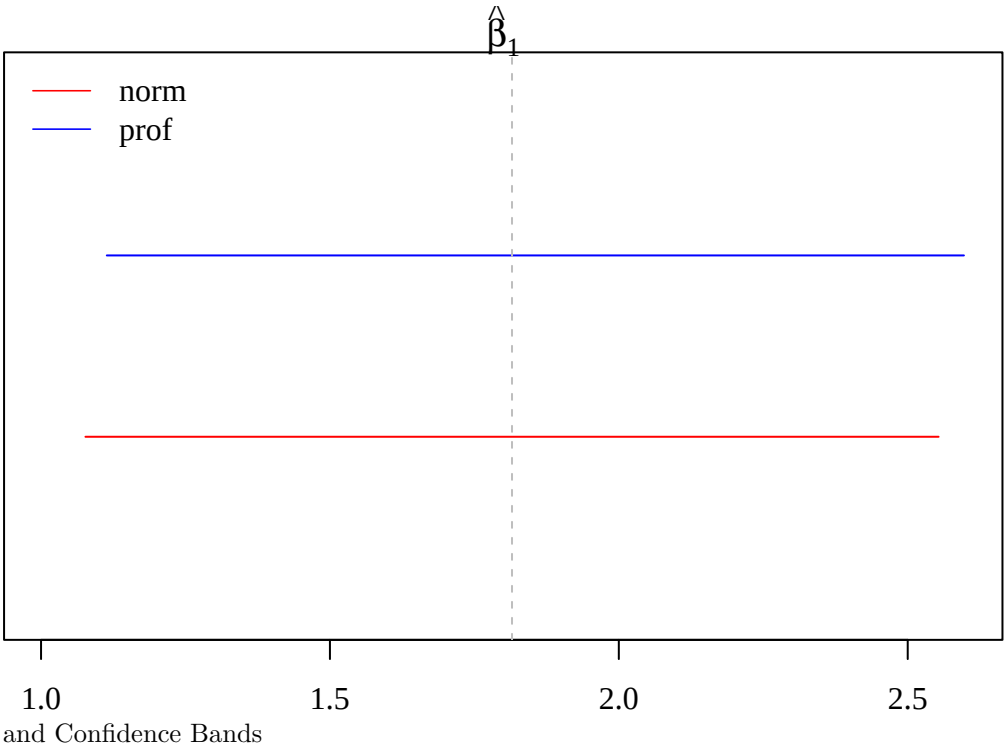
```
## [1] 1.076834 2.553455
```

```
# Construct profile likelihood confidence interval
# (often more accurate for logistic regression)
library(MASS)
(CI_prof <- confint(logitFit)[2,])
```

```
## Waiting for profiling to be done...
```

```
##    2.5 %   97.5 %
## 1.113790 2.597305
```

```
par(las = 1, mar = c(3.5, 3.5, 1.2, 0.5), mgp = c(2, 1, 0), family = "serif")
plot(1, type = "n", xlab = "", ylab = "", xlim = c(1, 2.6), ylim = c(-0.05, 0.1),
     yaxt = "n", main = expression(hat(beta)[1]))
segments(CI_norm[1], 0, CI_norm[2], col = "red")
segments(CI_prof[1], 0.05, CI_prof[2], col = "blue")
abline(v = est[2, 1], lty = 2, col = "gray")
legend("topleft", legend = c("norm", "prof"),
       col = c("red", "blue"), lty = 1, lwd = 0.8, bty = "n")
```

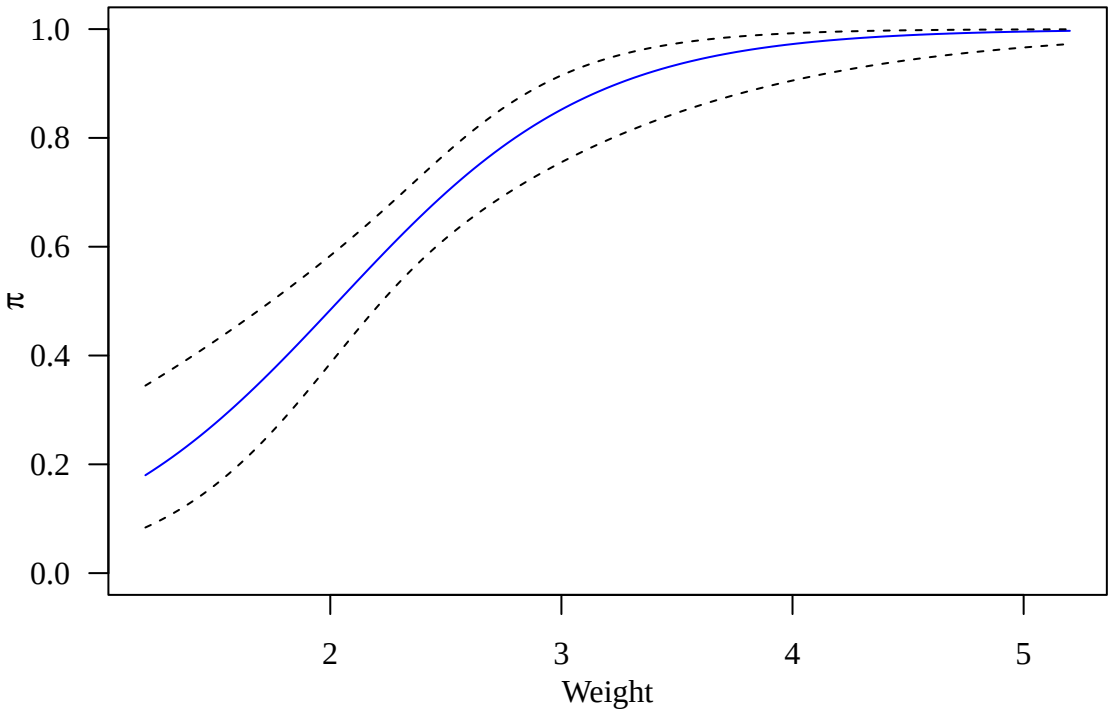


Prediction and Confidence Bands

We now visualize the estimated probability curve from the logistic regression model together with approximate 95% confidence bands.

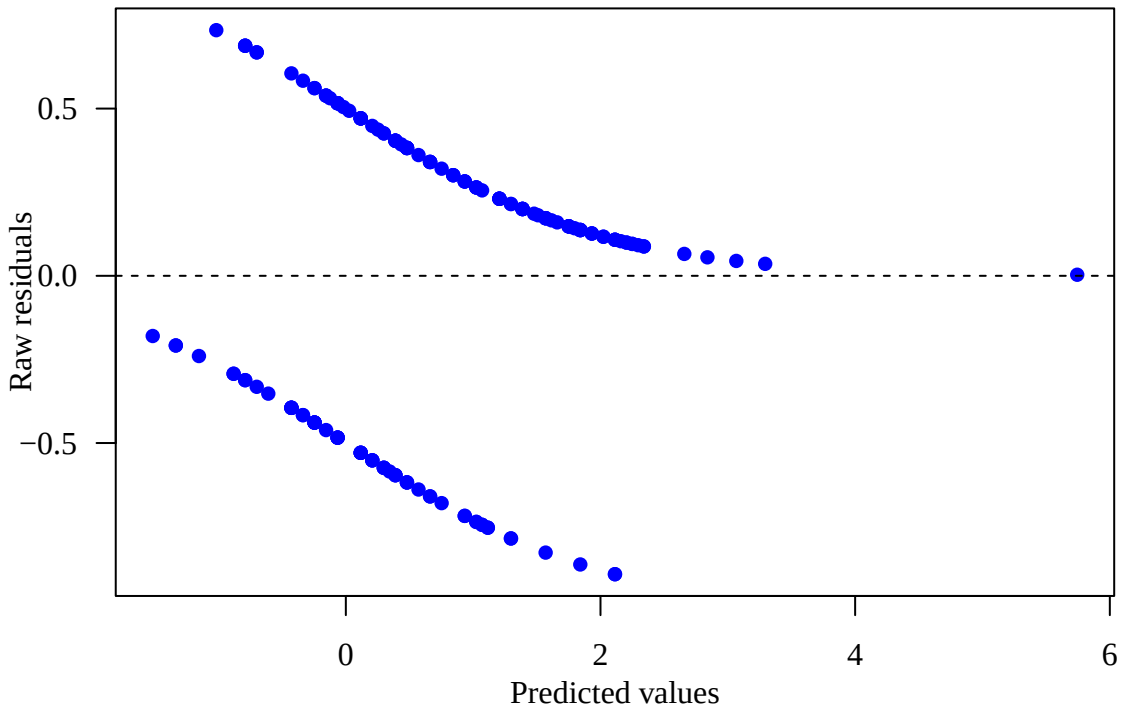
The fitted curve represents the estimated probability of mating success as a function of crab weight.

```r
pred <- predict(logitFit, newdata = data.frame(weight = xg), se.fit = TRUE)
fit <- pred$fit; se <- pred$se.fit

par(las = 1, mar = c(3.5, 3.5, 1.2, 0.5), mgp = c(2, 1, 0), family = "serif")
plot(xg, exp(fit) / (1 + exp(fit)), type = "l", col = "blue", ylim = c(0, 1),
     las = 1, xlab = "Weight", ylab = expression(pi))
lines(xg, exp(fit + 1.96 * se) / (1 + exp(fit + 1.96 * se)), lty = 2)
lines(xg, exp(fit - 1.96 * se) / (1 + exp(fit - 1.96 * se)), lty = 2)
```



Raw Residual Plot

```r
res <- resid(logitFit, type = "response")
pred <- predict(logitFit)

par(las = 1, mar = c(3.5, 3.5, 1.2, 0.5), mgp = c(2, 1, 0), family = "serif")
plot(pred, res, col = "blue", pch = 16, xlab = "Predicted values", ylab = "Raw residuals")
abline(h = 0, lty = 2)
```

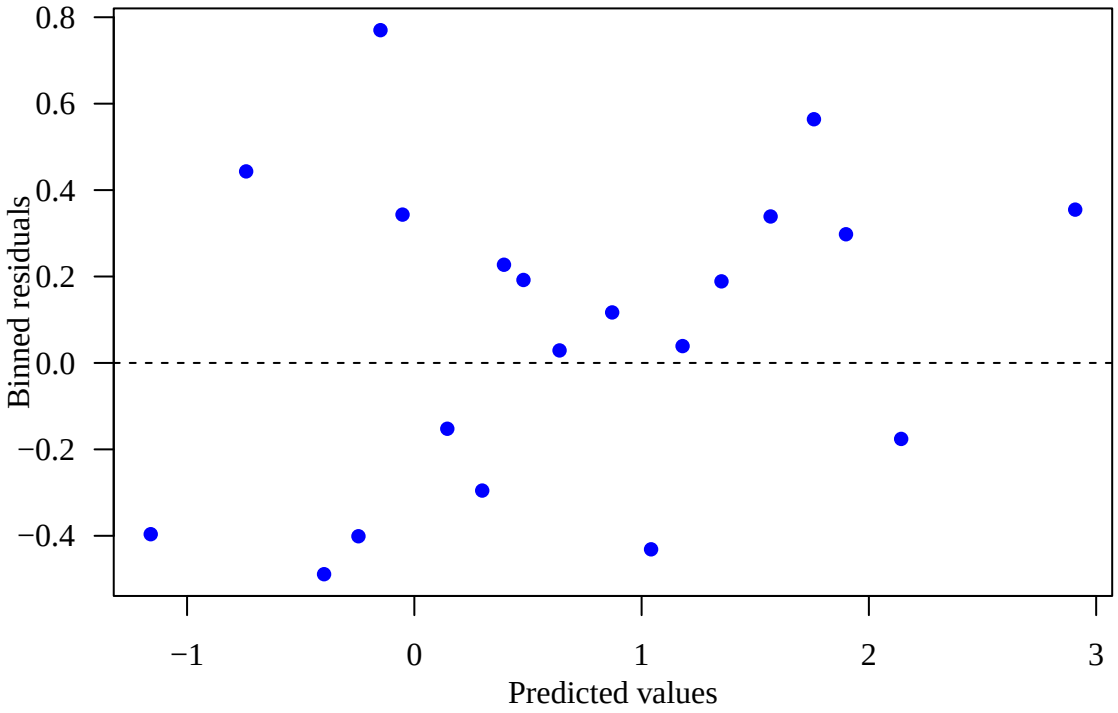


Binned Residuals

Because binary responses produce highly discrete residuals, raw residual plots can be difficult to interpret.

A binned residual plot averages residuals within groups of similar predictor values, making systematic patterns easier to detect.

```
breaks <- quantile(crab$weight, seq(0, 1, length.out = 20 + 1))
wt_bin <- findInterval(crab$weight, breaks, rightmost.closed = TRUE)
library(dplyr)
crab.res <- mutate(crab, res = residuals(logitFit), Lpred = predict(logitFit), bin = wt_bin)

res_bin <- tapply(crab.res$res, crab.res$bin, mean)
Lpred_bin <- tapply(crab.res$Lpred, crab.res$bin, mean)

par(las = 1, mar = c(3.5, 3.5, 1.2, 0.5), mgp = c(2, 1, 0), family = "serif")
plot(res_bin ~ Lpred_bin, xlab = "Predicted values",
     ylab = "Binned residuals", col = "blue", pch = 16, las = 1)
abline(h = 0, lty = 2)
```



Model Selection

We now consider whether adding additional predictors improves model fit.

Stepwise selection procedures can be used to compare competing models and identify a more parsimonious logistic regression model.

```
logitFit2 <- glm(y ~ weight + width, data = crab, family = "binomial")
summary(logitFit2)
```

```
##
## Call:
## glm(formula = y ~ weight + width, family = "binomial", data = crab)
##
## Coefficients:
##             Estimate Std. Error z value Pr(>|z|)
## (Intercept)  -9.3547     3.5280  -2.652  0.00801 **
## weight        0.8338     0.6716   1.241  0.21445
## width         0.3068     0.1819   1.686  0.09177 .
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## (Dispersion parameter for binomial family taken to be 1)
##
##     Null deviance: 225.76  on 172  degrees of freedom
## Residual deviance: 192.89  on 170  degrees of freedom
## AIC: 198.89
##
## Number of Fisher Scoring iterations: 4
```

```r
step(logitFit2)
```

```
## Start:  AIC=198.89
## y ~ weight + width
##
##          Df Deviance    AIC
## - weight  1   194.45 198.45
## <none>        192.89 198.89
## - width   1   195.74 199.74
##
## Step:  AIC=198.45
## y ~ width
##
##         Df Deviance    AIC
## <none>       194.45 198.45
## - width  1   225.76 227.76
```

```
##
## Call:  glm(formula = y ~ width, family = "binomial", data = crab)
##
## Coefficients:
## (Intercept)        width
##    -12.3508       0.4972
##
## Degrees of Freedom: 172 Total (i.e. Null);  171 Residual
## Null Deviance:       225.8
## Residual Deviance: 194.5     AIC: 198.5
```

Generalized Additive Logistic Regression

Generalized additive models (GAMs) extend logistic regression by allowing flexible nonlinear relationships between predictors and the response.

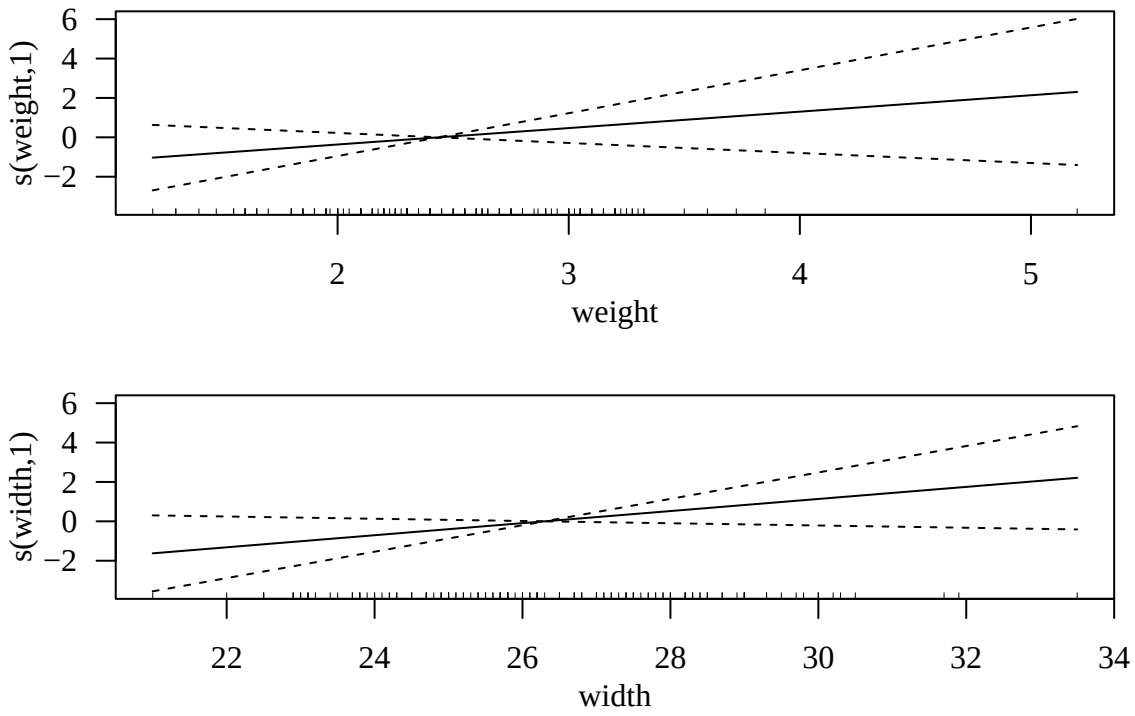
Instead of assuming a strictly linear effect, smooth functions are estimated directly from the data.

```r
library(mgcv)
logit_gam <- gam(y ~ s(weight) + s(width), family = "binomial", data = crab)
summary(logit_gam)
```

```
##
## Family: binomial
## Link function: logit
##
## Formula:
## y ~ s(weight) + s(width)
##
## Parametric coefficients:
##             Estimate Std. Error z value Pr(>|z|)
## (Intercept)   0.7456     0.1847   4.036 5.43e-05 ***
## ---
## 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(weight)   1      1  1.541  0.2145
## s(width)    1      1  2.843  0.0918 .
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## R-sq.(adj) =  0.162   Deviance explained = 14.6%
## UBRE = 0.14966  Scale est. = 1         n = 173
```

```
par(las = 1, mar = c(3.5, 3.5, 1.2, 0.5), mgp = c(2, 1, 0), family = "serif",
    mfrow = c(2, 1))
plot(logit_gam)
```



Poisson Regression

Flying-Bomb Hits on London During World War II [Clarke, 1946; Feller, 1950]

This classic example is often used to illustrate the Poisson distribution.

The data record the number of bomb hits across equally sized regions of London during World War II. We compare the observed frequencies with frequencies expected under a Poisson model.

```
count <- c(229, 211, 93, 35, 7, 1)
grids <- 576
hits <- 537
lambda <- hits / grids
count_expected <- c(grids * dpois(0:4, lambda = lambda),
                    grids * ppois(4, lambda = lambda, lower.tail = F))
round(count_expected, 1)
```

```
## [1] 226.7 211.4  98.5  30.6   7.1   1.6
```

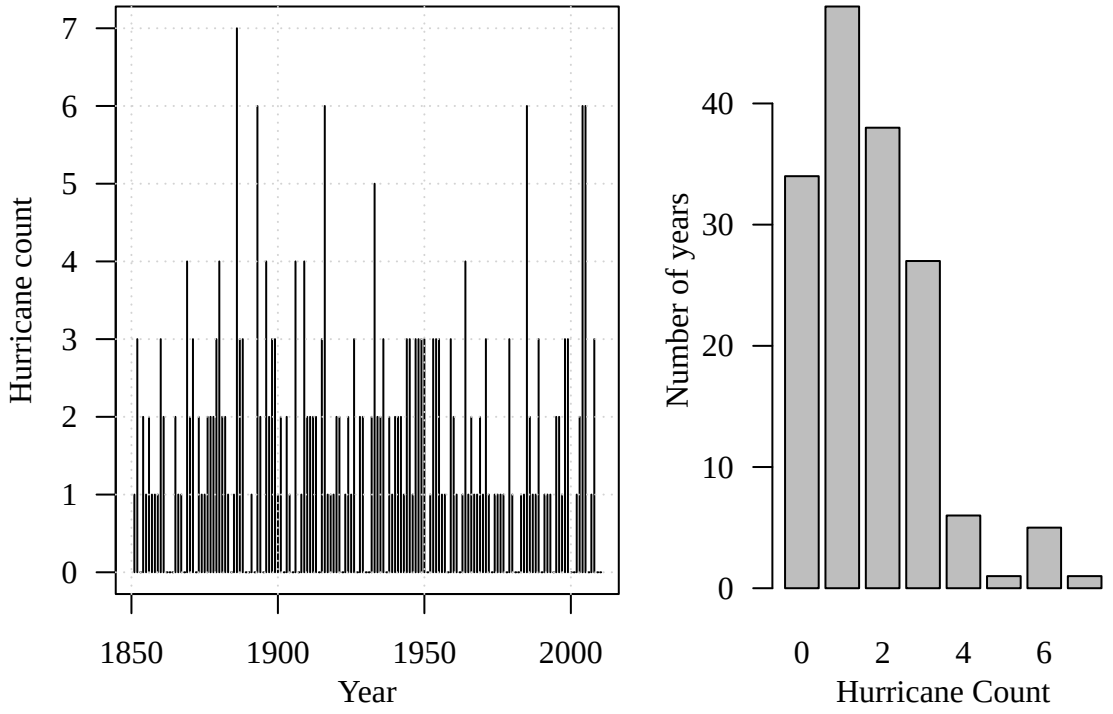
US Landfalling Hurricane

This dataset is courtesy of *James Elsner*, Earl B. and Sophia H. Shaw Professor in the Department of Geography at Florida State University.

```
# load the hurriance count
con = "http://myweb.fsu.edu/jelsner/Book/Chap07/US.txt"
hurricanes = read.table(con, header = T)
head(hurricanes)
```

```
##   Year All MUS G FL E
## 1 1851   1   1 0  1 0
## 2 1852   3   1 1  2 0
## 3 1853   0   0 0  0 0
## 4 1854   2   1 1  0 1
## 5 1855   1   1 1  0 0
## 6 1856   2   1 1  1 0
```

```
par(las = 1, mar = c(3.5, 3.5, 1.2, 0.5), mgp = c(2, 1, 0), family = "serif")
layout(matrix(c(1, 2), 1, 2, byrow = TRUE), widths = c(0.57, 0.43))
plot(hurricanes$Year, hurricanes$All, type = "h", xlab = "Year", ylab = "Hurricane count")
grid()
barplot(table(hurricanes$All), xlab = "Hurricane Count", ylab = "Number of years", main = "")
```

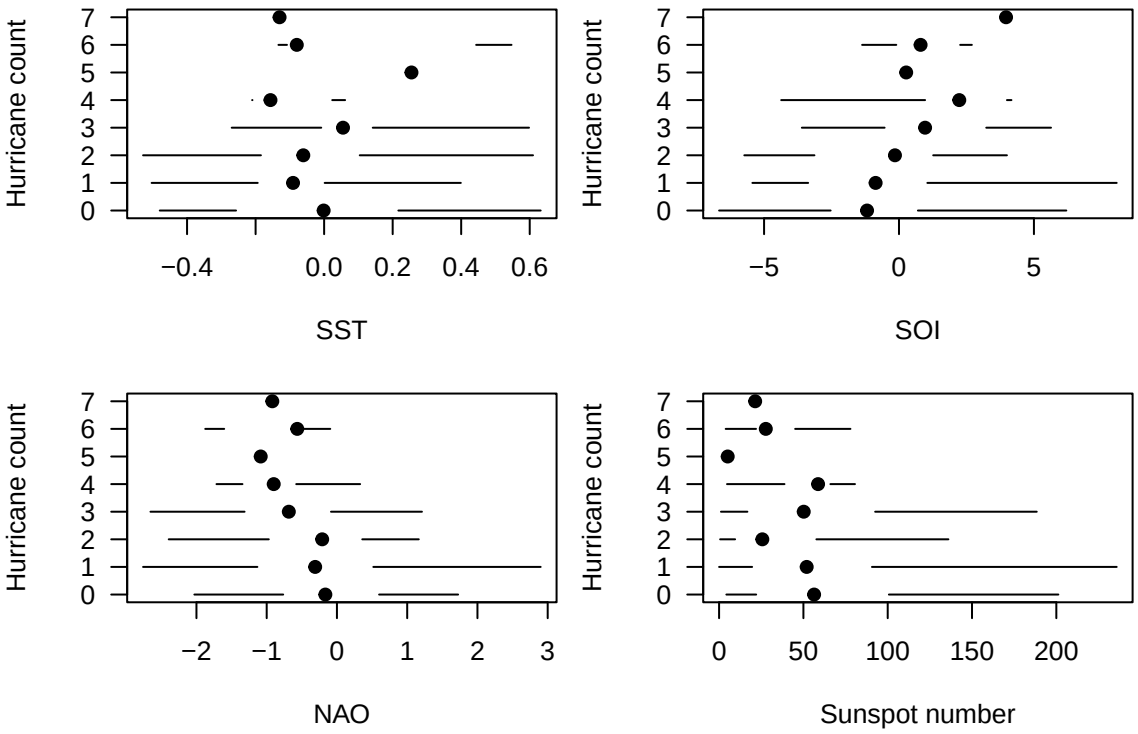


Load Environmental Variables

```
load("annual.RData")
data <- data.frame(All = hurricanes$All, SOI = annual$soi, NAO = annual$nao, SST = annual$sst,
                   SSN = annual$ssn)
data <- data[-(1:15),]
```

Explore Relationships Between Hurricane Counts and Climate Variables

```
H <- hurricanes

par(mfrow = c(2, 2), mar = c(4.5, 4, 1, 0.6))
plot(range(annual$sst, na.rm = TRUE), c(0, 7), type = "n", ylab = "Hurricane count", xlab = "SST",
     las = 1)
for(i in 0:7){
  points(fivenum(annual$sst[H$All == i])[3], i, pch = 19)
  lines(c(fivenum(annual$sst[H$All == i])[1], fivenum(annual$sst[H$All == i])[2]), c(i, i))
  lines(c(fivenum(annual$sst[H$All == i])[4], fivenum(annual$sst[H$All == i])[5]), c(i, i))
}
plot(range(annual$soi, na.rm = TRUE), c(0, 7), type = "n", ylab = "Hurricane count", xlab = "SOI",
     las = 1)

for(i in 0:7){
  points(fivenum(annual$soi[H$All == i])[3], i, pch=19)
  lines(c(fivenum(annual$soi[H$All == i])[1], fivenum(annual$soi[H$All == i])[2]), c(i, i))
  lines(c(fivenum(annual$soi[H$All == i])[4], fivenum(annual$soi[H$All == i])[5]), c(i, i))
}
plot(range(annual$nao, na.rm = TRUE), c(0, 7), type = "n", ylab = "Hurricane count", xlab = "NAO",
     las = 1)

for(i in 0:7){
  points(fivenum(annual$nao[H$All == i])[3], i, pch=19)
  lines(c(fivenum(annual$nao[H$All == i])[1], fivenum(annual$nao[H$All == i])[2]), c(i, i))
  lines(c(fivenum(annual$nao[H$All == i])[4], fivenum(annual$nao[H$All == i])[5]), c(i, i))
}
plot(range(annual$ssn, na.rm = TRUE), c(0, 7), type = "n", ylab = "Hurricane count",
     xlab = "Sunspot number", las = 1)

for(i in 0:7){
  points(fivenum(annual$ssn[H$All == i])[3], i, pch = 19)
  lines(c(fivenum(annual$ssn[H$All == i])[1], fivenum(annual$ssn[H$All == i])[2]), c(i, i))
  lines(c(fivenum(annual$ssn[H$All == i])[4], fivenum(annual$ssn[H$All == i])[5]), c(i, i))
}
```



Linear Regression

```
lmFull <- lm(All ~ ., data = data)
predict(lmFull, newdata = data.frame(SOI = -3, NAO = 3, SST = 0, SSN = 250))
```

```
##         1
## -0.318065
```

Poisson Regression

Poisson regression is specifically designed for count data.

The model assumes the response follows a Poisson distribution, with the logarithm of the mean modeled as a linear combination of predictors.

```
PoiFull <- glm(All ~ ., data = data, family = "poisson")
summary(PoiFull)
```

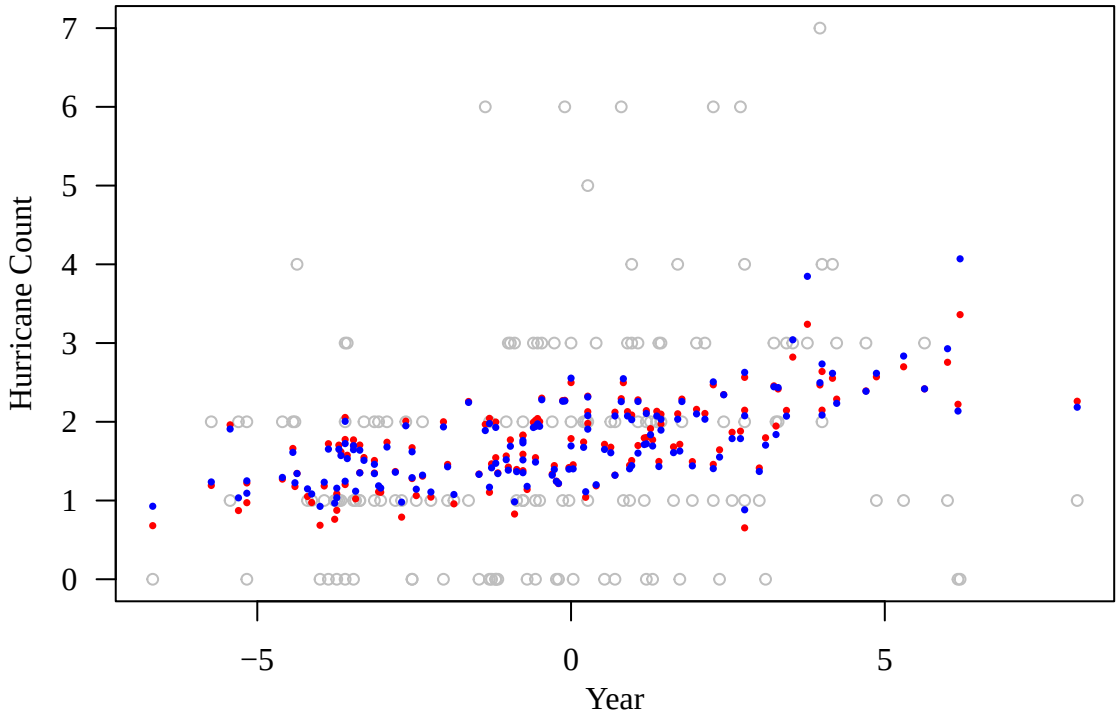
```
##
## Call:
## glm(formula = All ~ ., family = "poisson", data = data)
##
## Coefficients:
##              Estimate Std. Error z value Pr(>|z|)
## (Intercept)  0.595288   0.103342   5.760 8.39e-09 ***
## SOI          0.061863   0.021319   2.902  0.00371 **
## NAO         -0.166595   0.064427  -2.586  0.00972 **
```

```
## SST          0.228972   0.255289   0.897  0.36977
## SSN         -0.002306   0.001372  -1.681  0.09284 .
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## (Dispersion parameter for poisson family taken to be 1)
##
##     Null deviance: 197.89  on 144  degrees of freedom
## Residual deviance: 174.81  on 140  degrees of freedom
## AIC: 479.64
##
## Number of Fisher Scoring iterations: 5
```

```
step(PoiFull)
```

```
## Start:  AIC=479.64
## All ~ SOI + NAO + SST + SSN
##
##        Df Deviance    AIC
## - SST   1   175.61 478.44
## <none>      174.81 479.64
## - SSN   1   177.75 480.59
## - NAO   1   181.58 484.41
## - SOI   1   183.19 486.02
##
## Step:  AIC=478.44
## All ~ SOI + NAO + SSN
##
##        Df Deviance    AIC
## <none>      175.61 478.44
## - SSN   1   178.29 479.12
## - NAO   1   183.57 484.41
## - SOI   1   183.91 484.74

##
## Call:  glm(formula = All ~ SOI + NAO + SSN, family = "poisson", data = data)
##
## Coefficients:
## (Intercept)          SOI          NAO          SSN
##    0.584957     0.061533    -0.177439    -0.002201
##
## Degrees of Freedom: 144 Total (i.e. Null);  141 Residual
## Null Deviance:       197.9
## Residual Deviance: 175.6     AIC: 478.4
```

```
par(mar = c(3.5, 3.5, 1, 0.5), mgp = c(2, 1, 0), family = "serif")
plot(data$SOI, hurricanes$All[-(1:15)], cex = 0.75, col = "gray",
     xlab = "", ylab = "", las = 1)
mtext("Hurricane Count", side = 2, line = 2)
mtext("Year", side = 1, line = 2)
points(data$SOI, predict(lmFull), col = "red", cex = 0.5, pch = 16)
points(data$SOI, predict(PoiFull, type = "response"), col = "blue", cex = 0.5, pch = 16)
```



Generalized additive Poisson regression

We next extend the Poisson regression model by allowing the effects of the environmental variables to be nonlinear.

First, we fit an **additive GAM**, where each predictor has its own smooth function:

$$\log\{\mathbb{E}(Y)\} = \beta_0 + f_1(\text{SOI}) + f_2(\text{NAO}) + f_3(\text{SST}) + f_4(\text{SSN}).$$

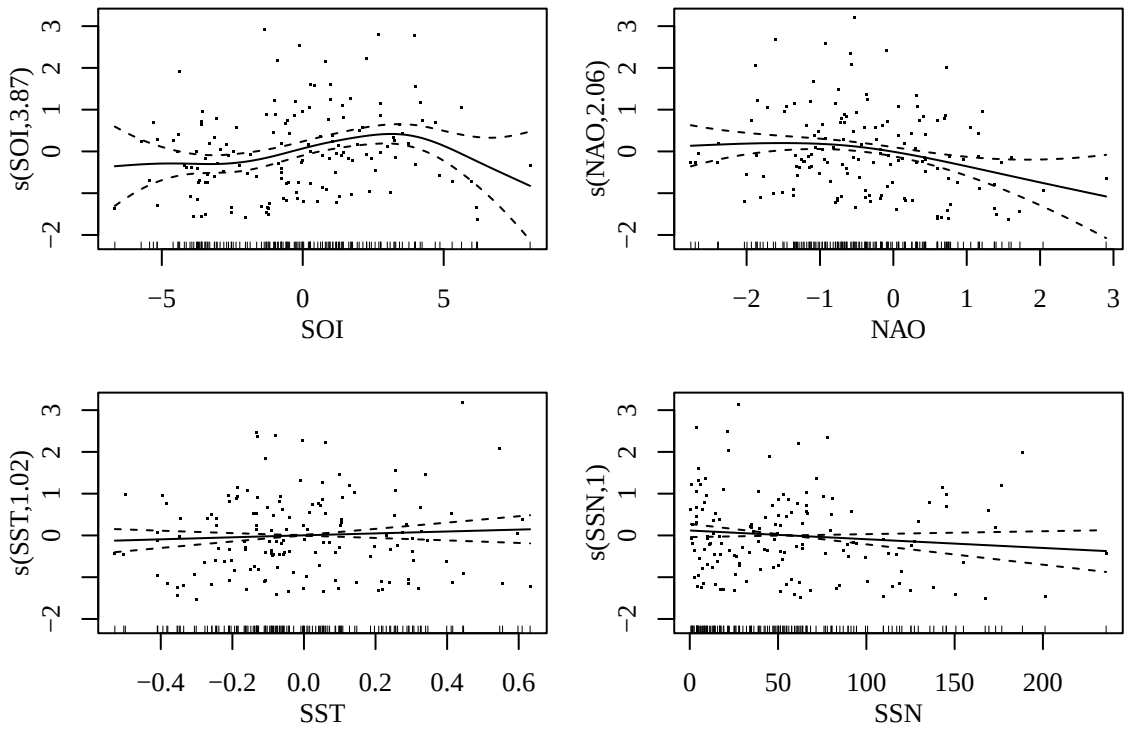
This model allows each climate variable to have a flexible nonlinear effect on hurricane counts while still assuming that the effects are additive.

```
poi_gam1 <- gam(All ~ s(SOI) + s(NAO) + s(SST) + s(SSN), family = "poisson", data = data)
summary(poi_gam1)
```

```
##
## Family: poisson
## Link function: log
##
## Formula:
## All ~ s(SOI) + s(NAO) + s(SST) + s(SSN)
##
## Parametric coefficients:
##             Estimate Std. Error z value Pr(>|z|)
## (Intercept)  0.48298    0.06721   7.187 6.64e-13 ***
## ---
## 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(SOI) 3.875  4.855 18.975 0.00177 **
## s(NAO) 2.060  2.631 10.994 0.01019 *
## s(SST) 1.017  1.034  0.773 0.38045
## s(SSN) 1.000  1.000  2.245 0.13404
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## R-sq.(adj) =  0.199   Deviance explained = 20.6%
## UBRE = 0.20739  Scale est. = 1         n = 145
```

```
par(mfrow = c(2, 2), mar = c(3.5, 3.5, 1, 0.5), mgp = c(2, 1, 0), family = "serif")
plot(poi_gam1, residuals = T)
```



The plots show the estimated smooth effect of each environmental variable on the log expected hurricane count, while holding the other variables fixed. The shaded bands represent approximate confidence intervals.

We can also fit a **non-additive GAM** using a multivariate smooth term:4

$$\log\{\mathbb{E}(Y)\} = \beta_0 + f(\text{SOI}, \text{NAO}, \text{SST}, \text{SSN}).$$

This model allows the climate variables to interact in a flexible nonlinear way, rather than assuming their effects are purely additive.

```
# Fit a non-additive generalized additive Poisson model
# The joint smooth term allows nonlinear interactions among predictors
poi_gam2 <- gam(All ~ s(SOI, NAO, SST, SSN), family = "poisson", data = data)
summary(poi_gam2)
```

```
##
```

```
## Family: poisson
## Link function: log
##
## Formula:
## All ~ s(SOI, NAO, SST, SSN)
##
## Parametric coefficients:
##             Estimate Std. Error z value Pr(>|z|)
## (Intercept)  0.47320    0.06849   6.909 4.89e-12 ***
## ---
## 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(SOI,NAO,SST,SSN) 14.31   14.6  32.49 0.00464 **
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## R-sq.(adj) =  0.125   Deviance explained = 18.6%
## UBRE = 0.32248  Scale est. = 1         n = 145
```

```
par(mfrow = c(1, 1), mar = c(3.5, 3.5, 2, 2.5), mgp = c(2, 1, 0), family = "serif")
plot(poi_gam2)
```