BEE 552 Week 10 Lab

Maria Feiler

2022-04-07

Challenger Analysis

1. The night before the launch, there was a meeting to discuss the influence of temperature on o-ring failure. The focus was on Figure 1a showing the number of o-ring failures CONDITIONAL ON there being at least one o-ring failure.
2. There are 6 o-rings on each shuttle, what is the appropriate model for o-ring failure? $X \sim Binom(n = 6, p(t,s))$ where p(t,s) is the probability of o-ring failure as a function of temperature t and pressure s.
3. Therefore, the appropriate GLM for o-ring failure is $log\left(\frac{p(t,s)}{1-p(t,s)}\right) = \alpha + \beta t + \gamma s$
4. They then fit the model using maximum likelihood.
5. They calculate the "goodness of fit" statistic G^2. This is just another name for the Deviance.

$$
\begin{aligned}
G^2 &= 2 * log\left(\frac{\text{Likelihood saturated model}}{\text{Likelihood model being considered}}\right) \\
\text{Deviance} &= -2 * (\text{LL(model being considered)} - \text{LL(saturated model)}) \\
\text{Deviance} &= 2 * (\text{LL(saturated model)} - \text{LL(model being considered)}) \\
\text{Deviance} &= 2 * log\left(\frac{\text{Likelihood saturated model}}{\text{Likelihood model being considered}}\right) = G^2
\end{aligned}
$$

6. Recognizing that devaince is only really meaningful relative to another model, they fit the temperature-only model $log\left(\frac{p(t)}{1-p(t)}\right) = \alpha + \beta t$. The difference in deviances is given by $\text{Deviance difference} \sim \chi^2_{\text{additional parameters}}$. So there is now only one additional parameter, so $\text{Deviance difference} \sim \chi^2$. The difference in deviance is not significant, i.e. this model fits about as well as the more complex model, so we can say that pressure has little effect on the probability of o-ring failure and we drop it from the model.
7. They construct 90th percentile confidence intervals for the expected number of incidents. *Checkpoint #1: What exactly are they doing here? Do you understand what they have done and why?* They are sampling with replacement from the original data and are refitting the model each time.
8. Next they plot the contours of the log-likelihood function and note that the contours are elliptical and therefore the data were not leading to ill-conditioned computation.
9. They collapse the binomial data to make a Bernoulli dataset in which "0" means that no o-rings failed, and "1" means that at least one o-ring failed. *Checkpoint #2: Why did they do this?* Because the o-rings may not be independent.
10. They refit the data using the Bernoulli model and find that the fits are quite close.

11. They want to construct confidence intervals for the model parameters. They say “instead of using the aymptotic theory to construct confidence intervals, we use the parametric bootstrap procedure.” *Checkpoint #3: Why might they have used a bootstrap approach here?* Small size size probably… They take the best-fit model, sample with replacement from the logistic model (presumably drawing (x,predicted y) pairs at random with replacement), and refit the bootstrapped data to get new model parameter estimates.

12. Then they look at the sensitivity of the model to each of the data points, by pulling out each data point in turn and refitting the model. *Checkpoint #4: What is this called?* Jackknife!

13. They next consider a non-linear model of the form $log\left(\frac{p(t,s)}{1-p(t,s)}\right) = \alpha + \beta(t-t_0) + \gamma(t-t_0)^2$

14. They again consider the change in deviance in going from the simpler linear model to the more complex quadratic model, and they find the quadratic term is not significant.

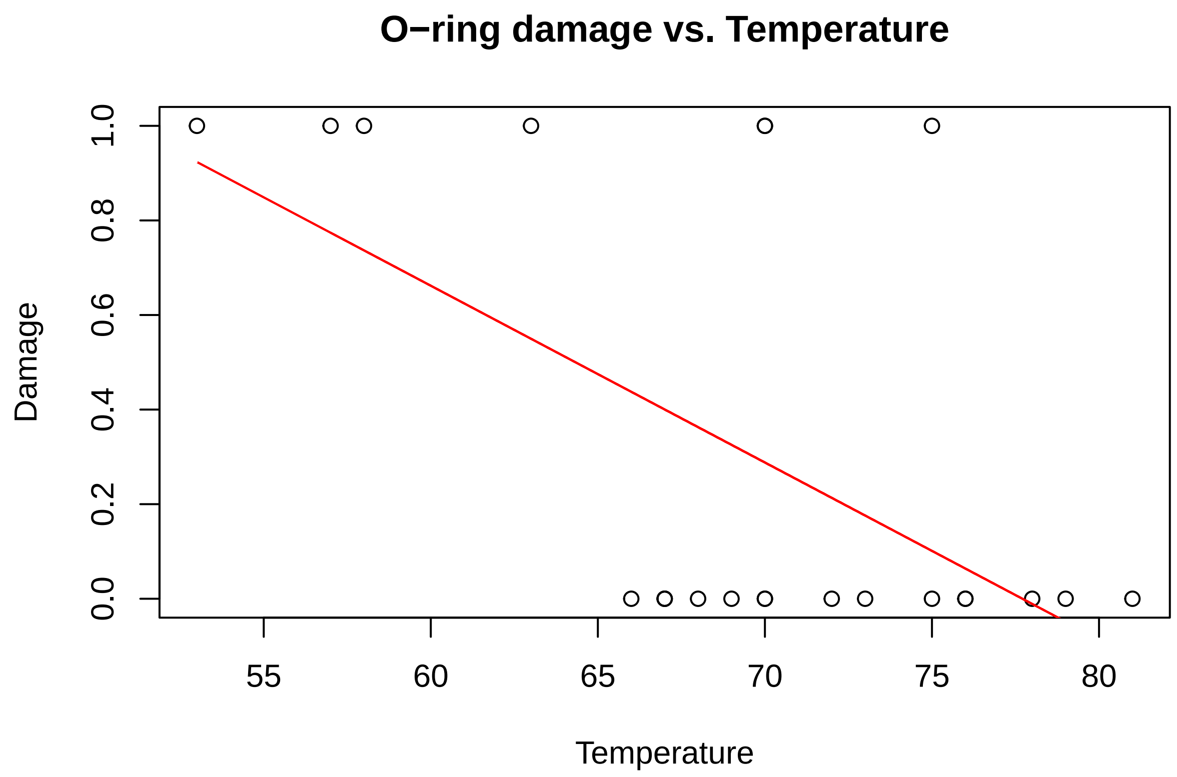
15. They then consider a model in which they use a non-parametric smoothing fit (using a moving window appraoch) in Figure 8.

16. They identify possible outliers in the data (more on model criticism in three weeks).

```
challenger <- read.csv("https://raw.githubusercontent.com/hlynch/Biometry2022/master/_data/Challenger_da

attach(challenger)

challenger.fit1 <- lm(O.ring.failure ~ Temp)
summary(challenger.fit1)
```

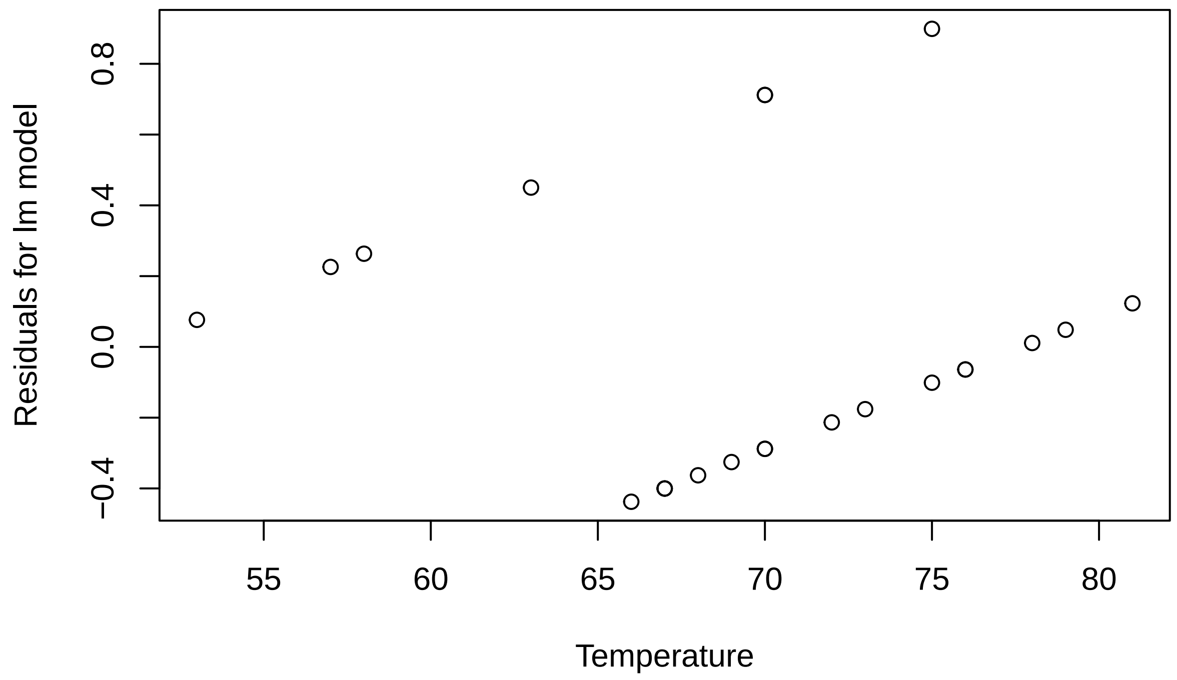
```
##
## Call:
## lm(formula = O.ring.failure ~ Temp)
##
## Residuals:
##      Min       1Q   Median       3Q      Max
## -0.43762 -0.30679 -0.06381  0.17452  0.89881
##
## Coefficients:
##             Estimate Std. Error t value Pr(>|t|)
## (Intercept)  2.90476    0.84208   3.450  0.00240 **
## Temp        -0.03738    0.01205  -3.103  0.00538 **
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 0.3987 on 21 degrees of freedom
## Multiple R-squared:  0.3144, Adjusted R-squared:  0.2818
## F-statistic:  9.63 on 1 and 21 DF,  p-value: 0.005383
```

```
plot(Temp,
     O.ring.failure,
     xlab = "Temperature",
     ylab = "Damage",
     main = "O-ring damage vs. Temperature")
lines(Temp,fitted(challenger.fit1),col="red")
```



This model is flawed because the predictions will escape the bounds (0,1) and the residuals are clearly not normal.

```
resid1<-residuals(challenger.fit1)
plot(Temp,
     resid1,
     xlab = "Temperature",
     ylab = "Residuals for lm model")
```



Weighted Linear Regression

How can we solve this? Weighted linear regression.

$$\text{weighted SS} = \sum_{i=1}^{n} w_i (Y_i - \hat{Y}_i)^2$$

$$\text{weighted SS} = \sum_{i=1}^{n} w_i (Y_i - \hat{Y}_i)^2 = \frac{1}{\pi_i(1-\pi_i)}$$

$$\hat{w}_i = \frac{1}{\hat{Y}_i(1-\hat{Y}_i)}$$

The procedure is then as follows:

1. Fit ordinary least squares
2. Obtain estimates $\hat{Y}_i$
3. If an estimate is less than 0 or greater than 1, set it to 0.001 (or something small) and 0.999 (or something close to but less than one), respectively
4. Compute the weights W_i
5. Fit weighted least squares

```
fitted(challenger.fit1)
```

```
##           1           2           3           4           5           6 
##  0.43761905  0.28809524  0.32547619  0.36285714  0.40023810  0.21333333 
##           7           8           9          10          11          12 
##  0.17595238  0.28809524  0.77404762  0.54976190  0.28809524 -0.01095238 
##          13          14          15          16          17          18 
##  0.40023810  0.92357143  0.40023810  0.10119048  0.28809524 -0.12309524 
##          19          20          21          22          23 
##  0.06380952 -0.04833333  0.10119048  0.06380952  0.73666667
```

Set $\hat{Y}_i < 0.001$ and $\hat{Y}_i = 0.999$.

```
pmin(c(1,2,3),
     c(0,3,5))
```

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

```
pmin(1,
     c(0,3,5))
```

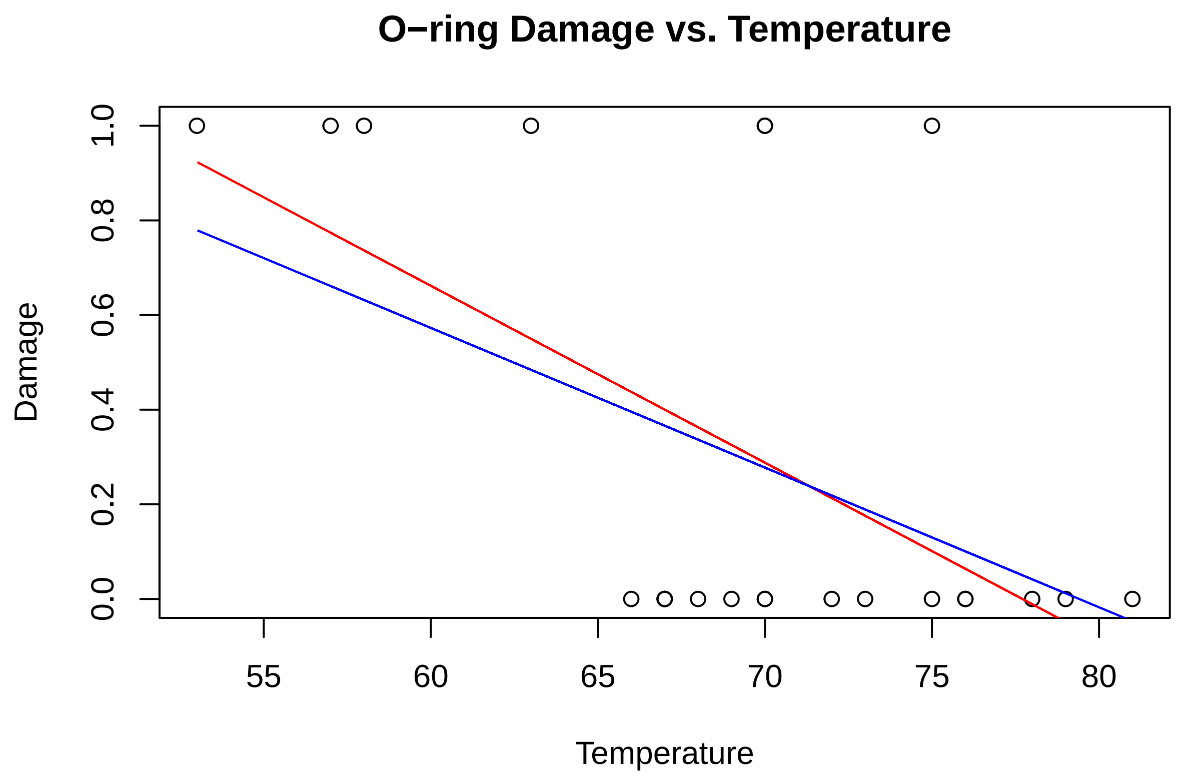
```
## [1] 0 1 1
```

```
new.predictions <- pmin(0.999,
                        pmax(0.001,
                             fitted(challenger.fit1)
                             )
                        )

vars <- new.predictions*(1-new.predictions)

challenger.fit2 <- lm(O.ring.failure ~ Temp,
                      weights = (1/vars)
                      )
summary(challenger.fit2)
```

```
## 
## Call:
## lm(formula = O.ring.failure ~ Temp, weights = (1/vars))
## 
## Weighted Residuals:
##     Min      1Q  Median      3Q     Max 
## -1.3136 -0.6779 -0.4313  0.8330  2.8846 
## 
## Coefficients:
##              Estimate Std. Error t value Pr(>|t|)    
## (Intercept)  2.343825   0.532408   4.402 0.000248 ***
## Temp        -0.029517   0.006746  -4.376 0.000265 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
## 
```

```
## Residual standard error: 1.105 on 21 degrees of freedom
## Multiple R-squared:  0.4769, Adjusted R-squared:  0.452
## F-statistic: 19.15 on 1 and 21 DF,  p-value: 0.0002647
```

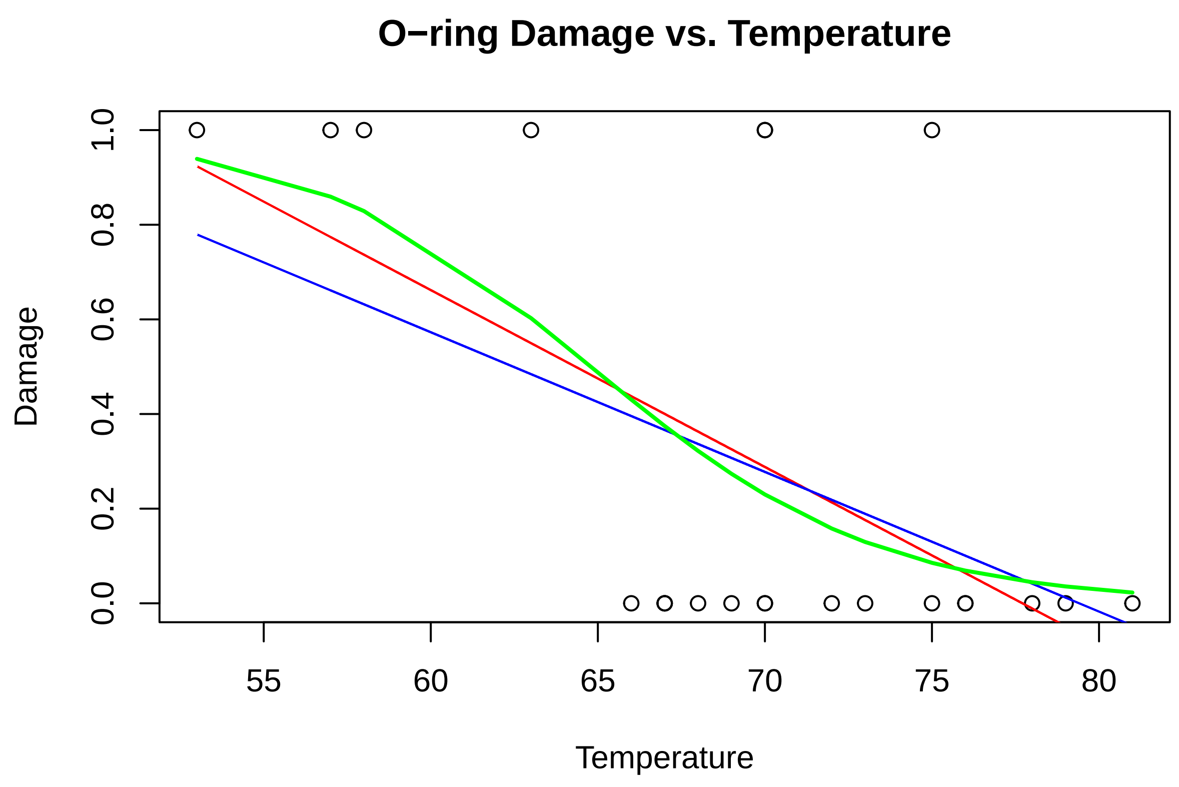
```
plot(Temp,
     O.ring.failure,
     xlab = "Temperature",
     ylab = "Damage",
     main = "O-ring Damage vs. Temperature")
lines(Temp,fitted(challenger.fit1),
      col="red")
lines(Temp,fitted(challenger.fit2),
      col="blue")
```



Still have the same problems as before.

Logistic Regression Practice

```
challenger.fit3 <- glm(O.ring.failure~Temp,
                       family = "binomial")

plot(Temp,O.ring.failure,
     xlab = "Temperature",
     ylab = "Damage",
     main = "O-ring Damage vs. Temperature")

# Above line only needed because RMarkdown doesn't keep previous plot
lines(Temp,
      fitted(challenger.fit1),
      col = "red")

# Above line only needed because RMarkdown doesn't keep previous plot
lines(Temp,
      fitted(challenger.fit2),
      col = "blue")

# Above line only needed because RMarkdown doesn't keep previous plot
lines(sort(Temp),
      fitted(challenger.fit3)[order(Temp)],
      col = "green",
      lwd = 2)
```

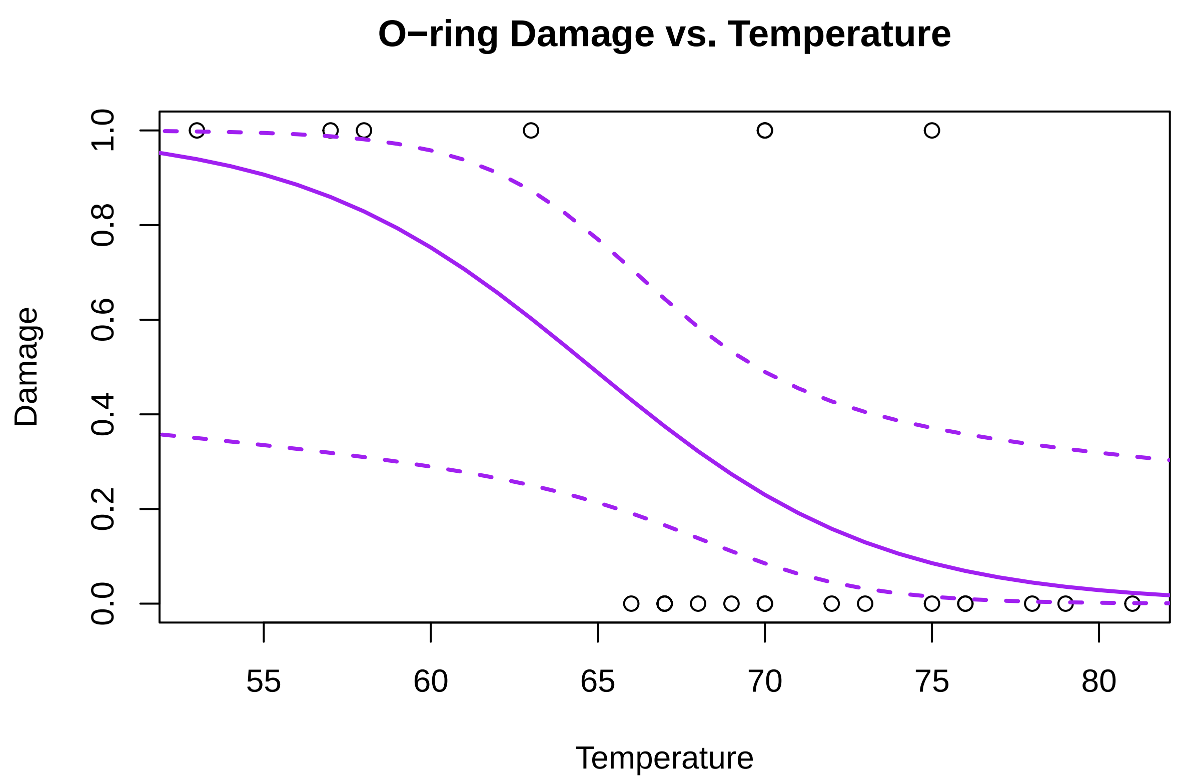


```
summary(challenger.fit3)
```

```
##
## Call:
## glm(formula = O.ring.failure ~ Temp, family = "binomial")
##
```

```
## Deviance Residuals:
##     Min       1Q   Median       3Q      Max
## -1.0611  -0.7613  -0.3783   0.4524   2.2175
##
## Coefficients:
##             Estimate Std. Error z value Pr(>|z|)
## (Intercept)  15.0429     7.3786   2.039   0.0415 *
## Temp         -0.2322     0.1082  -2.145   0.0320 *
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## (Dispersion parameter for binomial family taken to be 1)
##
##     Null deviance: 28.267  on 22  degrees of freedom
## Residual deviance: 20.315  on 21  degrees of freedom
## AIC: 24.315
##
## Number of Fisher Scoring iterations: 5
```

```
newdata <- data.frame(Temp = seq(30,85))

confidence.bands <- predict.glm(challenger.fit3,
                                newdata,
                                se.fit = TRUE)

challenger.fit3<-glm(cbind(O.ring.failure, 6-O.ring.failure) ~ Temp,
                     family="binomial")

library(boot)

plot(Temp,
     O.ring.failure,
     xlab ="Temperature",
     ylab = "Damage",
     main = "O-ring Damage vs. Temperature")
# Above line only needed because RMarkdown doesn't keep previous plot
lines(newdata[,1],
      inv.logit(confidence.bands$fit),
      col = "purple",
      lwd = 2)
lines(newdata[,1],
      inv.logit(confidence.bands$fit+1.96*confidence.bands$se.fit),
      col = "purple",
      lwd = 2,
      lty = 2)
lines(newdata[,1],
      inv.logit(confidence.bands$fit-1.96*confidence.bands$se.fit),
      col = "purple",
      lwd = 2,
      lty = 2)
```



Checkpoint #5: How would you construct a prediction interval for a binary value? You cannot, it doesn't make sense, the value is either 0 or 1.

```
-2*logLik(challenger.fit3)
```

```
## 'log Lik.' 25.51522 (df=2)
```

```
challenger.fit4<-glm(O.ring.failure~1,family=binomial)
-2*logLik(challenger.fit4)
```

```
## 'log Lik.' 28.26715 (df=1)
```

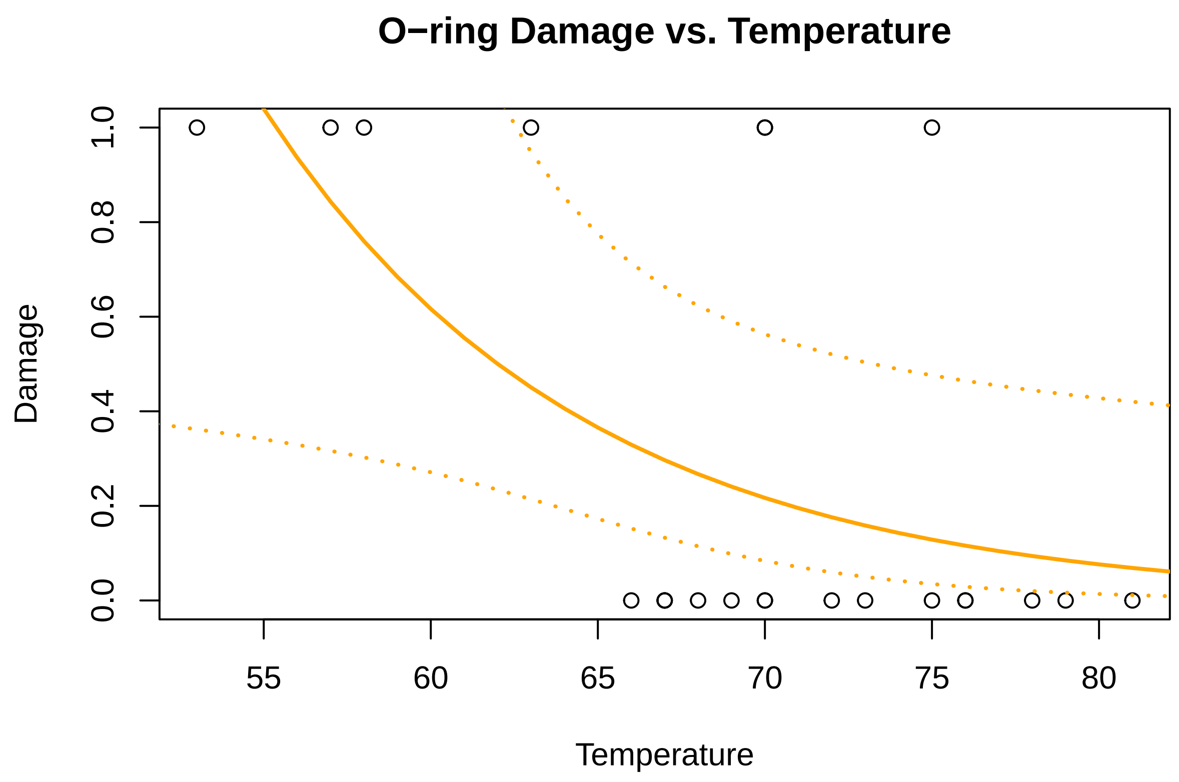
Poisson Regression Practice

```
challenger.fit4<-glm(O.ring.failure~Temp,family="poisson")
summary(challenger.fit4)
```

```
##
## Call:
## glm(formula = O.ring.failure ~ Temp, family = "poisson")
##
## Deviance Residuals:
```

```
##      Min        1Q    Median        3Q       Max
## -0.81159  -0.67620  -0.48134  -0.04632   1.53598
##
## Coefficients:
##             Estimate Std. Error z value Pr(>|z|)
## (Intercept)  5.78518    3.13068   1.848   0.0646 .
## Temp        -0.10448    0.04878  -2.142   0.0322 *
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## (Dispersion parameter for poisson family taken to be 1)
##
##     Null deviance: 16.654  on 22  degrees of freedom
## Residual deviance: 12.206  on 21  degrees of freedom
## AIC: 30.206
##
## Number of Fisher Scoring iterations: 6
```

```
confidence.bands<-predict.glm(challenger.fit4,newdata,se.fit=TRUE)
plot(Temp,O.ring.failure,xlab="Temperature",ylab="Damage",main="O-ring Damage vs. Temperature")
# Above line only needed because RMarkdown doesn't keep previous plot
lines(newdata[,1],exp(confidence.bands$fit),col="orange",lwd=2)
lines(newdata[,1],exp(confidence.bands$fit+1.96*confidence.bands$se.fit),col="orange",lwd=2,lty=3)
lines(newdata[,1],exp(confidence.bands$fit-1.96*confidence.bands$se.fit),col="orange",lwd=2,lty=3)
```



Getting a Feel for Deviance

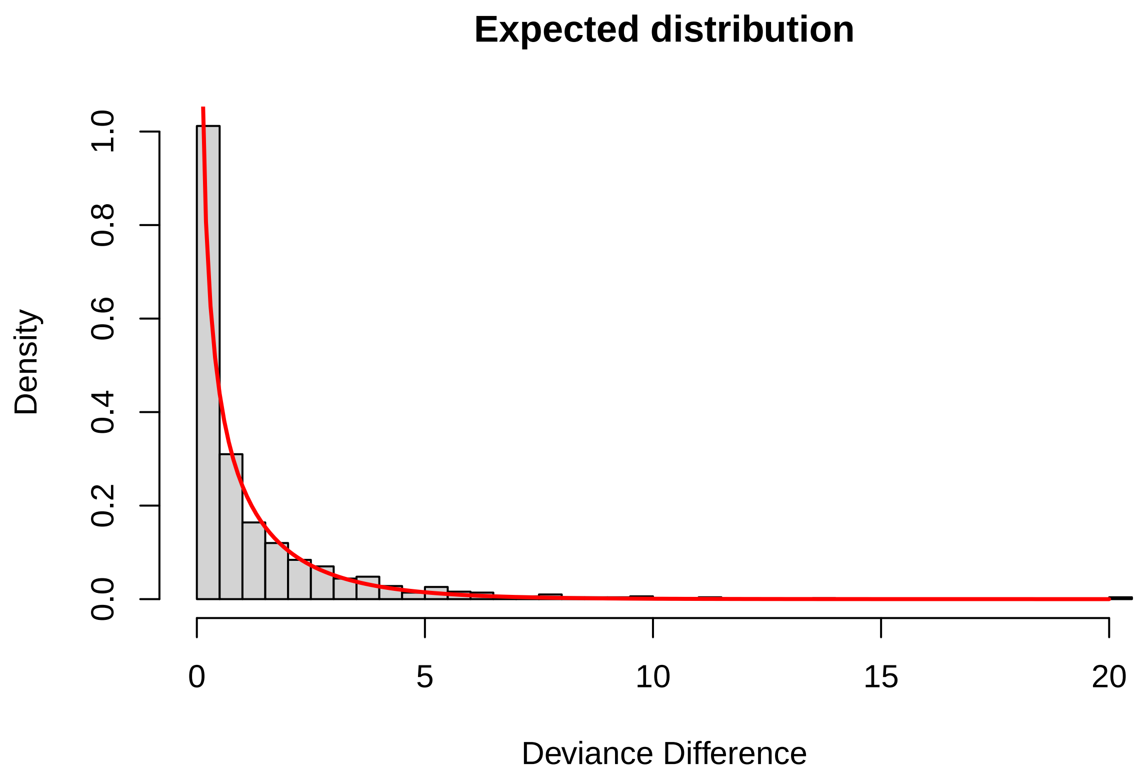
```r
# Fit a logistic regression with Temp as the only covariate
challenger.smaller.model <- glm(O.ring.failure ~ Temp, data=challenger, family="binomial")
# Generate a random covariate with same mean and sd as Temp
randvar <- rnorm(n=length(challenger$Temp), mean=mean(challenger$Temp), sd=sd(challenger$Temp))
# Add the random covariate to a data frame for model-fitting
newdata <- cbind(challenger, randvar)
# Fit the logistic regression with Temp and the random covariate
challenger.larger.model <- glm(O.ring.failure ~ Temp + randvar, data=newdata, family="binomial")
# Calculate the deviance difference of the two models
dev_diff <- deviance(challenger.smaller.model) - deviance(challenger.larger.model)
dev_diff
```

```
## [1] 0.1691084
```

```r
# and... repeat!
dev_diff <- c()

for (i in 1:1000){
  # Generate a random covariate with same mean and sd as Temp
  randvar <- rnorm(n=length(challenger$Temp), mean=mean(challenger$Temp), sd=sd(challenger$Temp))

  # Add the random covariate to a data frame for model-fitting
  newdata <- cbind(challenger, randvar)

  # Fit the model
  challenger.fit.larger.model <- glm(O.ring.failure ~ Temp + randvar, data=newdata, family="binomial")

  # Calculate the deviance difference
  dev_diff_rand <- deviance(challenger.smaller.model) - deviance(challenger.fit.larger.model)

  dev_diff <- c(dev_diff, dev_diff_rand)
}
```

```
## Warning: glm.fit: fitted probabilities numerically 0 or 1 occurred

## Warning: glm.fit: algorithm did not converge

## Warning: glm.fit: fitted probabilities numerically 0 or 1 occurred

## Warning: glm.fit: algorithm did not converge

## Warning: glm.fit: fitted probabilities numerically 0 or 1 occurred
```

```r
# plot the distribution and add a line for a chi-square with df=1
hist(dev_diff, xlab="Deviance Difference", main="Expected distribution", freq=FALSE,breaks=30)
lines(seq(0,20,0.1), dchisq(seq(0,20,0.1),df=1), col="red",lwd=2)
```

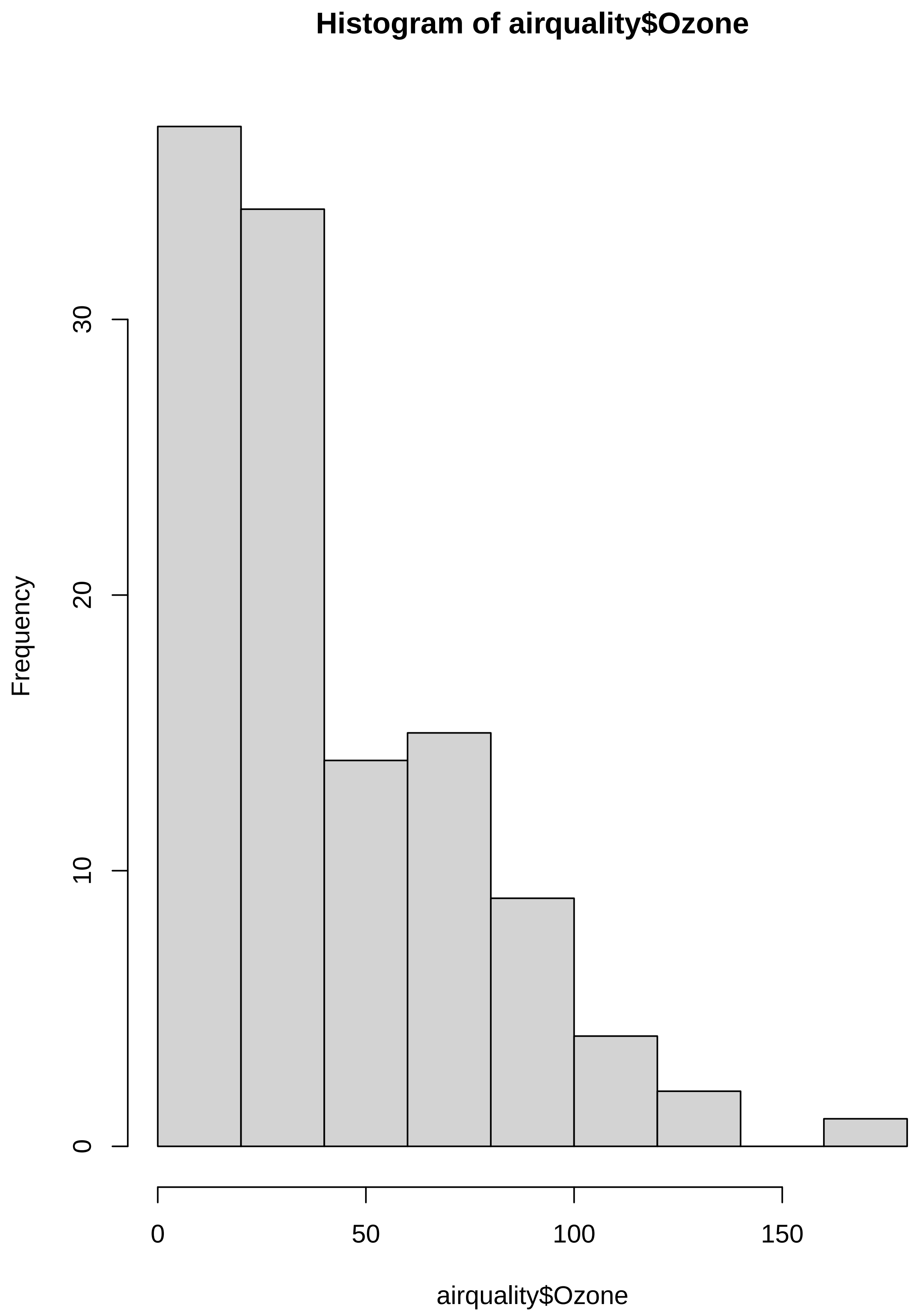


Generalized Additive Models

```
library(gam)
```

```
## Loading required package: splines

## Loading required package: foreach

## Loaded gam 1.20.1
```

```
hist(airquality$Ozone)
```


Histogram of airquality$Ozone
Frequency
0
10
20
30
0
50
100
150
airquality$Ozone

```r
air.lm<-lm(log(Ozone)~Solar.R+Wind+Temp,data=airquality)
summary(air.lm)
```

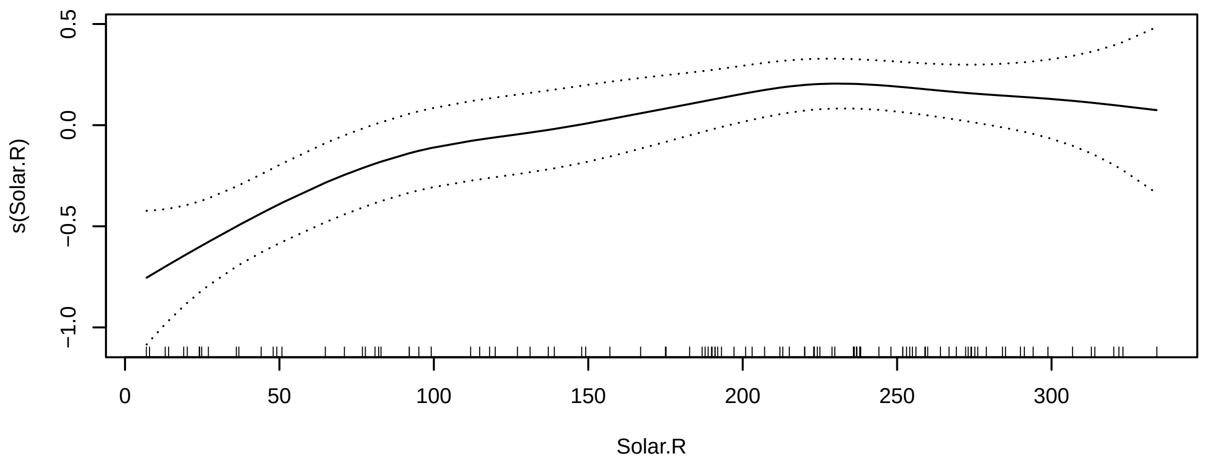
```
##
## Call:
## lm(formula = log(Ozone) ~ Solar.R + Wind + Temp, data = airquality)
##
## Residuals:
##      Min       1Q   Median       3Q      Max
## -2.06193 -0.29970 -0.00231  0.30756  1.23578
##
## Coefficients:
##               Estimate Std. Error t value Pr(>|t|)
## (Intercept) -0.2621323  0.5535669  -0.474 0.636798
## Solar.R      0.0025152  0.0005567   4.518 1.62e-05 ***
## Wind        -0.0615625  0.0157130  -3.918 0.000158 ***
## Temp         0.0491711  0.0060875   8.077 1.07e-12 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 0.5086 on 107 degrees of freedom
##   (42 observations deleted due to missingness)
## Multiple R-squared:  0.6644, Adjusted R-squared:  0.655
## F-statistic: 70.62 on 3 and 107 DF,  p-value: < 2.2e-16
```

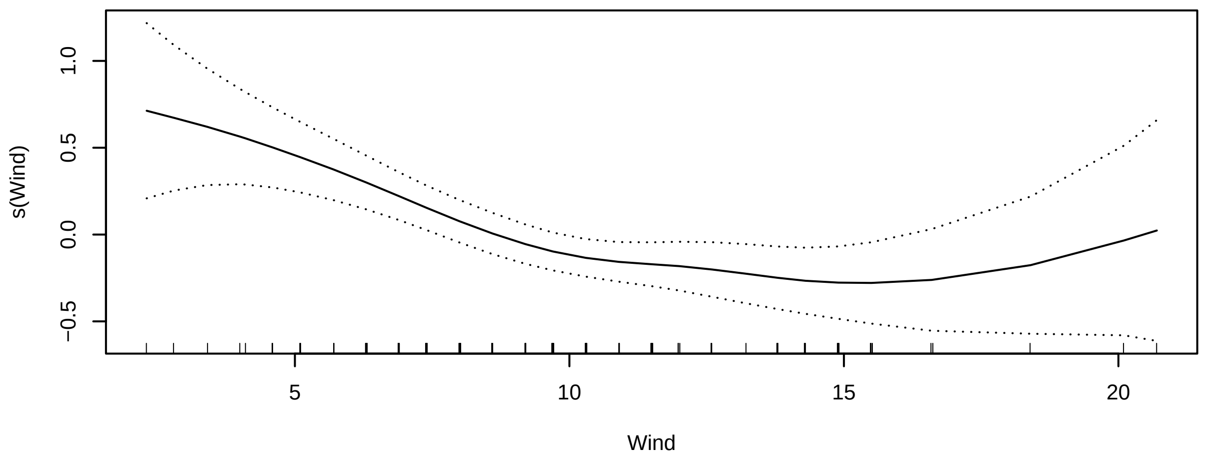
```r
air.gam<-gam(log(Ozone)~s(Solar.R)+s(Wind)+s(Temp),data=airquality)
summary(air.gam)
```

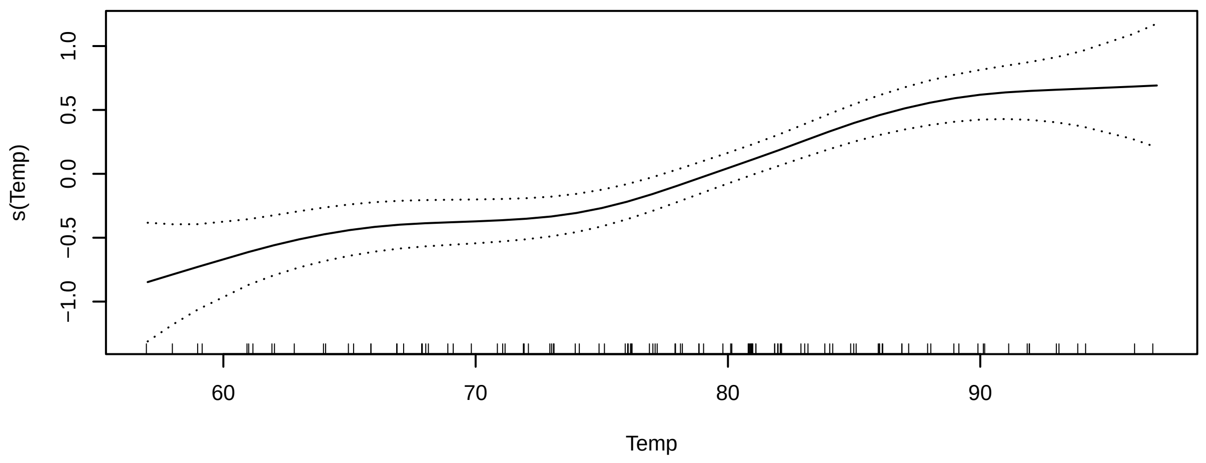
```
##
## Call: gam(formula = log(Ozone) ~ s(Solar.R) + s(Wind) + s(Temp), data = airquality)
## Deviance Residuals:
##      Min       1Q   Median       3Q      Max
## -1.84583 -0.24538 -0.04403  0.31419  0.99890
##
## (Dispersion Parameter for gaussian family taken to be 0.2235)
##
##     Null Deviance: 82.47 on 110 degrees of freedom
## Residual Deviance: 21.9077 on 98.0001 degrees of freedom
## AIC: 162.8854
## 42 observations deleted due to missingness
##
## Number of Local Scoring Iterations: NA
##
## Anova for Parametric Effects
##            Df Sum Sq Mean Sq F value    Pr(>F)
## s(Solar.R)  1 16.041 16.0408  71.756 2.484e-13 ***
## s(Wind)     1 17.208 17.2083  76.978 5.521e-14 ***
## s(Temp)     1 12.723 12.7227  56.913 2.351e-11 ***
## Residuals  98 21.908  0.2235
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
```

```
## Anova for Nonparametric Effects
##             Npar Df Npar F   Pr(F)
## (Intercept)
## s(Solar.R)        3 2.8465 0.04151 *
## s(Wind)           3 3.4736 0.01897 *
## s(Temp)           3 2.9358 0.03713 *
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

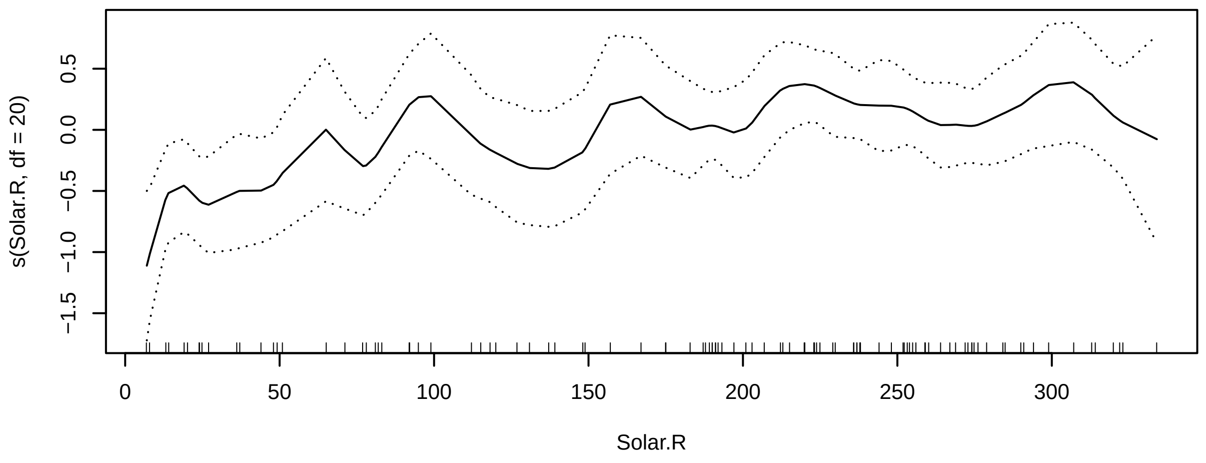
```
par(mfrow=c(3,1))
plot(air.gam,se=T)
```


s(Solar.R)
Solar.R
−1.0
−0.5
0.0
0.5
0
50
100
150
200
250
300

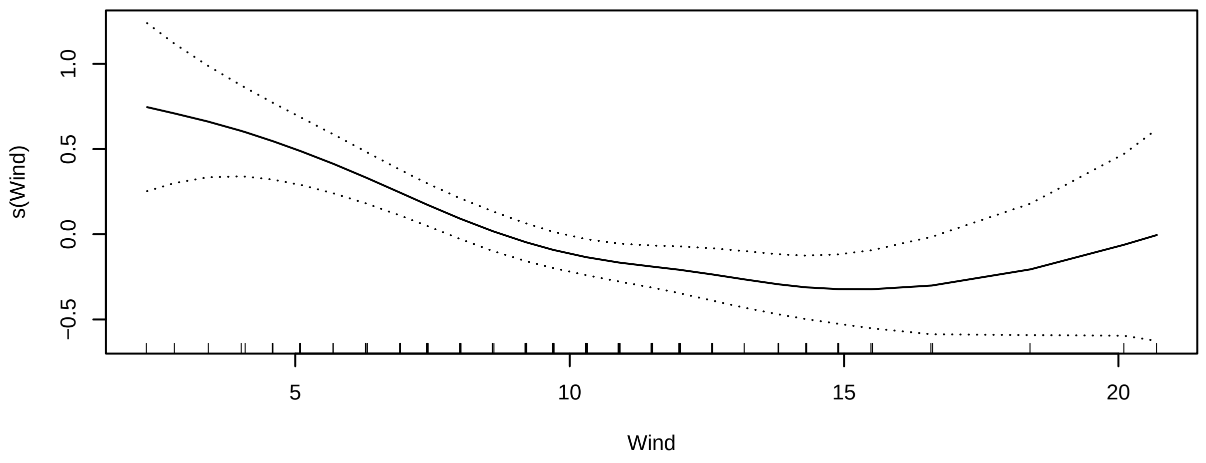

s(Wind)
Wind
−0.5
0.0
0.5
1.0
5
10
15
20


s(Temp)
Temp
−1.0
−0.5
0.0
0.5
1.0
60
70
80
90

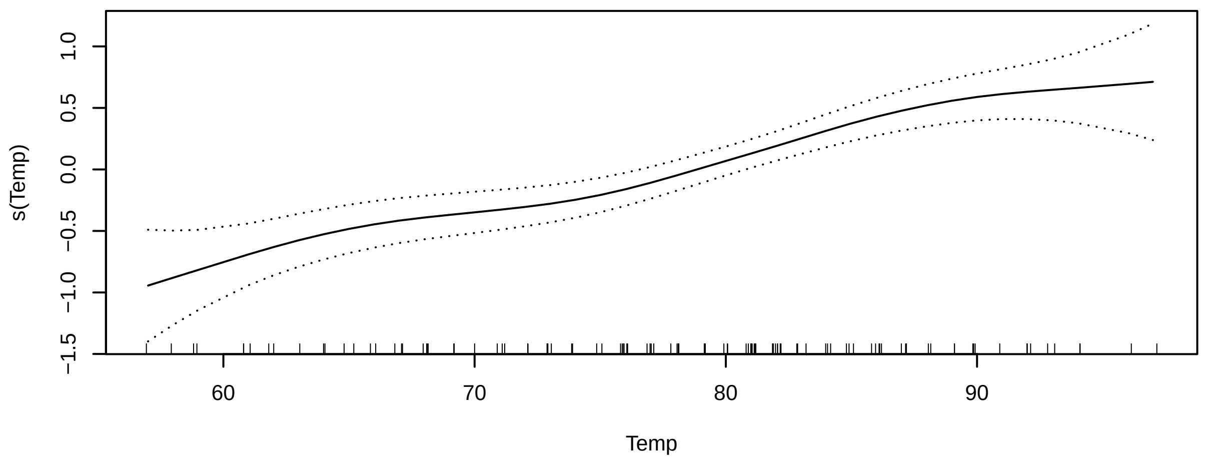
```
# Change d.o.f. to 20 from default 4
air.gam<-gam(log(Ozone)~s(Solar.R,df=20)+s(Wind)+s(Temp),data=airquality)
plot(air.gam,se=T)
```



s(Solar.R, df = 20)
−1.5
−1.0
−0.5
0.0
0.5
0
50
100
150
200
250
300
Solar.R



s(Wind)
−0.5
0.0
0.5
1.0
5
10
15
20
Wind



s(Temp)
−1.5
−1.0
−0.5
0.0
0.5
1.0
60
70
80
90
Temp