STATS 3001 / STATS 4104 / STATS 7054
Statistical Modelling III
Workshop 4 - GAMs p2

John Maclean

Load packages

```
pacman::p_load(tidyverse, gglm, broom)
```

Preface

In this workshop there is a little theory and application for GAMs. After the workshop you'll be able to answer

- what types of smoother can be employed?
- how is a GAM tuned?
- how to check assumptions in a GAM?
- how to incorporate factors in (the smoothing terms of) a GAM?

In the finale you'll apply a GAM to two stations from the bioluminescence dataset from workshop 2. Do you think the GAM will be a good model?

Content

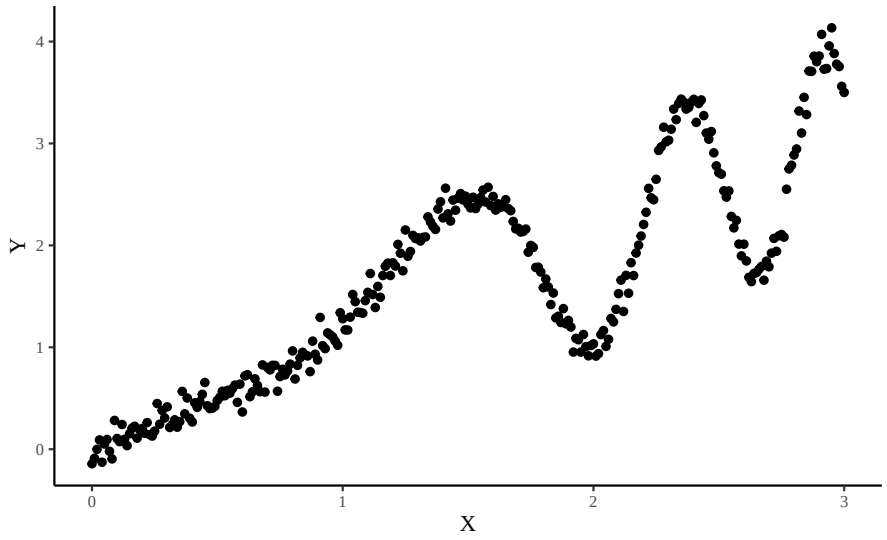
I alternate between questions - for you to do - and discussion points. I will usually take over to go over discussion points.

Do:

1. Make the synthetic dataset `df` using

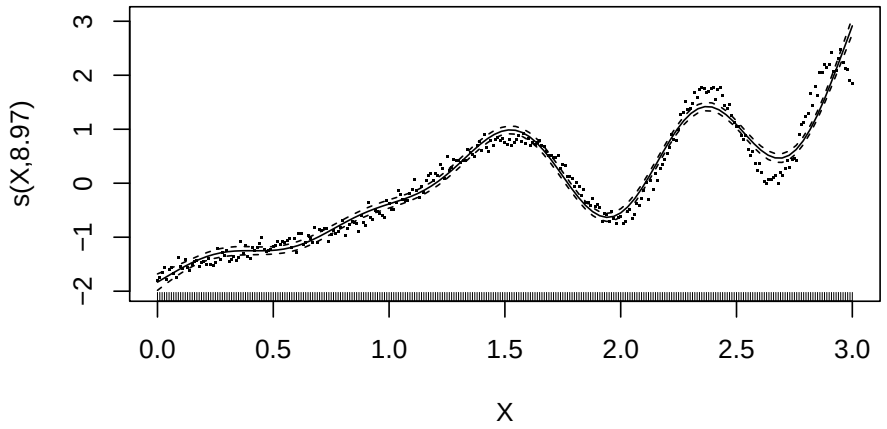
```
X <- seq(0,3,by=0.01) #predictor
bnd <-mean(X)/2 #boundary between linear and nonlinear response
a <- 3 #arbitrary
b <- -2*bnd*a #match first derivatives at X=7
c <- -bnd^2*a-bnd*b #match value at X=7
Y <- X + (X>bnd) * sin( a*X^2 + b*X + c) #combines linear and nonlinear response
s = 0.1
Y <- Y + rnorm(length(X),sd=s)#add noise
df <- tibble(X=X, Y=Y)

df %>% ggplot(aes(X,Y)) + geom_point()
```



2. Fit a GAM to predict Y given X, e.g. with

```
gam1 <- mgcv::gam(Y ~ s(X), data = df, method = "REML")
plot(gam1,residuals=TRUE)
```



The GAM probably fits the data well initially, but does poorly at thr right side of the plot.

Now edit your gam to try to improve it:

- try altering the smoothing function using the optional `bs` input to `s()`. Try `bs = cr` for cubic splines, `bs = tp` for thin plate regression, and `bs = cs` & `bs = ts` for *shrinkage* implementations of both of the prior types. (Shrinkage tries to use as few basis elements as possible - always good.)
- try altering the `method` argument to `gam`: later in the prac we will explain, and see that `method = REML` is often recommended by experts for having better properties than the gam default behaviour.
- try altering the smoothing parameter `sp` as we did last workshop. Try `sp=1e-11`. Any difference?
- (the reason for this example) try altering the number of basis functions to `k=20`. (Hopefully this step gives you a good GAM!)
- clean up your `gam` model. Either remove the smoothing function argument or leave it as a shrinkage option. Leave `method=REML`. *Remove* your manually set `sp` parameter (as you remember, `sp` governs the degree to which your GAM tries to match the data. It can have huge effects on your output and you should generally let `gam` optimise it).

In this example you've learned that `k` *is not cleverly selected/optimised by the* `gam` *function* (Read details by calling `?mgcv::s`). If your data is particularly wiggly, may need to experiment. We'll see a diagnostic later on.

3. (Discussion: smoothing parameter estimation)

4. Are the assumptions of your model satisfied?

`plot` isn't useful here - by default it plots predictors and response variables. Try `mgcv::gam.check()` - should produce 4 plots. You may need to `knit` in an rmarkdown to see all four, or just call `qq.gam()` to get the missing Q-Q plot. (You can also call `summary()` and `anova()` as usual.)

5. Learn how to diagnose low `k`.

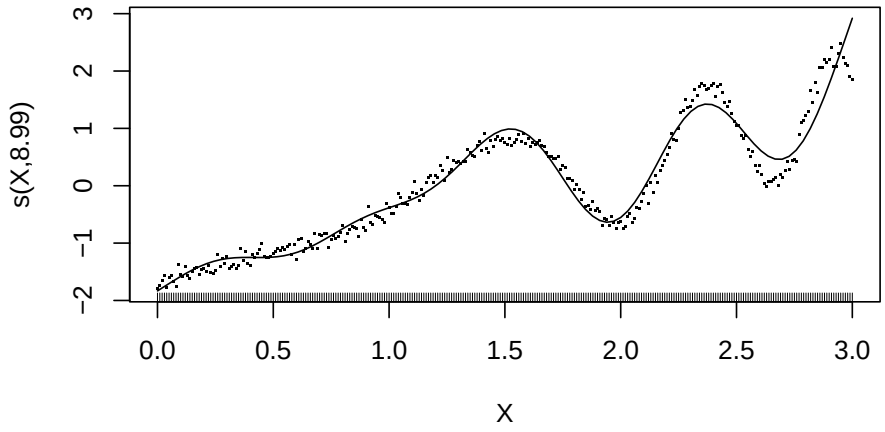
Remove the `k=` argument from your GAM and re-run it. Call `mgcv::gam.check()` again and check the printed output. The author says: "The test of whether the basis dimension for a smooth is adequate (Wood, 2017, section 5.9) is based on computing an estimate of the residual variance ... The further below 1 this is, the more likely it is that there is missed pattern left in the residuals. ... Low p-values may indicate that the basis dimension, k, has been set too low, especially if the reported edf is close to k', the maximum possible EDF for the term. ... *Doubling a suspect k and re-fitting is sensible*: if the reported edf increases substantially then you may have been missing something in the first fit."

(Emphasis mine). So given a low k-index, the advice is to count the number of basis elements chosen (e.g. if your model is called `gam1`, get them by looking at the number of terms in `gam1$coefficients`) and then try doubling them.

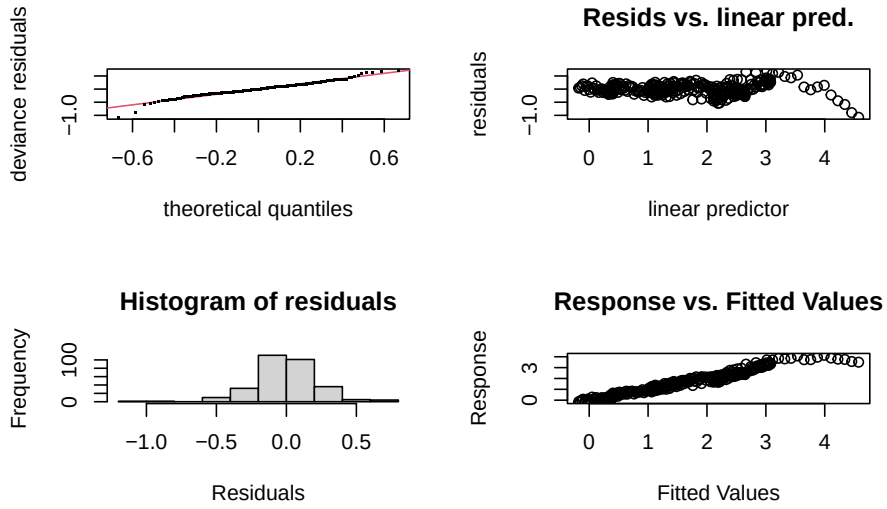
```
gam_model <- mgcv::gam(Y ~ s(X,k=-1), data = df)
summary(gam_model)
```

```
##
## Family: gaussian
## Link function: identity
##
## Formula:
## Y ~ s(X, k = -1)
##
## Parametric coefficients:
##             Estimate Std. Error t value Pr(>|t|)
## (Intercept)   1.6546     0.0131   126.3   <2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Approximate significance of smooth terms:
##       edf Ref.df     F p-value
## s(X) 8.99      9 678.3  <2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## R-sq.(adj) =  0.953   Deviance explained = 95.4%
## GCV = 0.053408  Scale est. = 0.051635  n = 301
```

```
plot(gam_model,residuals = TRUE, se = FALSE)
```




```
mgcv::gam.check(gam_model)
```




```
##
## Method: GCV   Optimizer: magic
## Smoothing parameter selection converged after 11 iterations.
## The RMS GCV score gradient at convergence was 3.491747e-07 .
## The Hessian was positive definite.
## Model rank =  10 / 10
##
## 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(X) 9.00 8.99    0.21  <2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

6. Try on real data

Get the `isit` dataset again. Filter to the 8th and 13th station, e.g. with

```
isit <- read_delim("workshops/ZuurData/Isit.txt") #read_delim will infer the delimitor
isit813 <- isit %>%
  filter(Station %in% c(8,13)) %>%
  mutate(Station = as_factor(Station))
```

```
isit813 <- isit813 %>% arrange(SampleDepth) #organise by increasing SampleDepth
```

We want to use `SampleDepth` (numeric) and `Station` (a factor) to predict. These commands are unwieldy in GAM and I have printed them for you. Try out these models:

```
m1 <- mgcv::gam(Sources ~ s(SampleDepth) +
                  Station, data = isit813)

m2 <- mgcv::gam(Sources ~ s(SampleDepth) +
                  s(SampleDepth,by = as.numeric(Station == 13)) +
                  Station, data = isit813)
```

6a. Use the plotting, summary, and diagnostic tools to investigate the two models. What model do they fit (try referencing lectures on factors)?

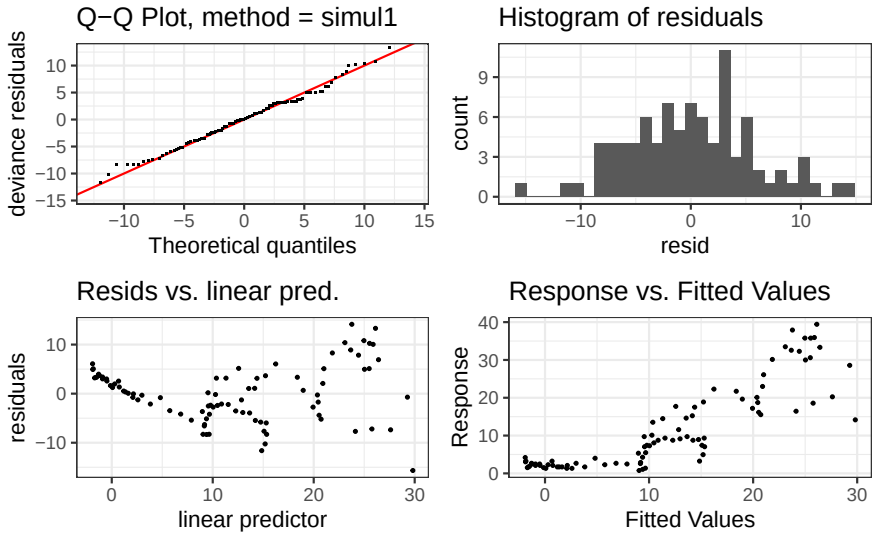
6b. Use the optional inputs to `gam` and `s()` to customise and try to improve the models. How good (in terms of good model assumptions) can you get them?

P.S. If you don't like standard `mgcv::gam.check()` for knitting. I googled 'ggplot mgcv::gam.check' and tried the first result:

```
pacman::p_load(mgcViz)
viz2 <- getViz(m1)
check.gamViz(viz2) #ggplot version of assumption checking plots - useful for customisation
```

```
##
## Method: GCV   Optimizer: magic
## Smoothing parameter selection converged after 6 iterations.
## The RMS GCV score gradient at convergence was 0.0004832222 .
## The Hessian was positive definite.
## Model rank =  11 / 11
##
## 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(SampleDepth) 9.00 6.37    1.37       1
```

```
## `stat_bin()` using `bins = 30`. Pick better value with `binwidth`.
```



Outlook

p-values obtained for GAMs are approximate. You can safely make inferences from smoothers with a p-value of 0.001 or less, or with a p-value of 0.1 or more, but p-values close to 0.05 are not to be trusted. Zuur et al (2009) recommend bootstrapping to analyse smoothers in this case (for bootstrapping, see Data Science course in second semester).