GAM and Model selection Weeks 9-10

Shadan Khan

2024-09-22

1. Define and print a string including your name, unit name, and task name

```
name <- "Shadan Khan"
unit_name <- "Statistical Data Analysis"
task_name <- "T6 P6"
message <- paste("Name:", name, " Unit Name:", unit_name, " Task Name:", task_name)
print(message)
```

```
## [1] "Name: Shadan Khan  Unit Name: Statistical Data Analysis  Task Name: T6 P6"
```

##2. Load the gapminder dataset. For Afghanistan only, fit a generalised additive model ##using gam() that predicts life expectancy from the population (using the default ##smoothing function). Report on the model outputs.

```
# Load necessary libraries
library(gapminder)
library(mgcv)
```

```
## Loading required package: nlme
```

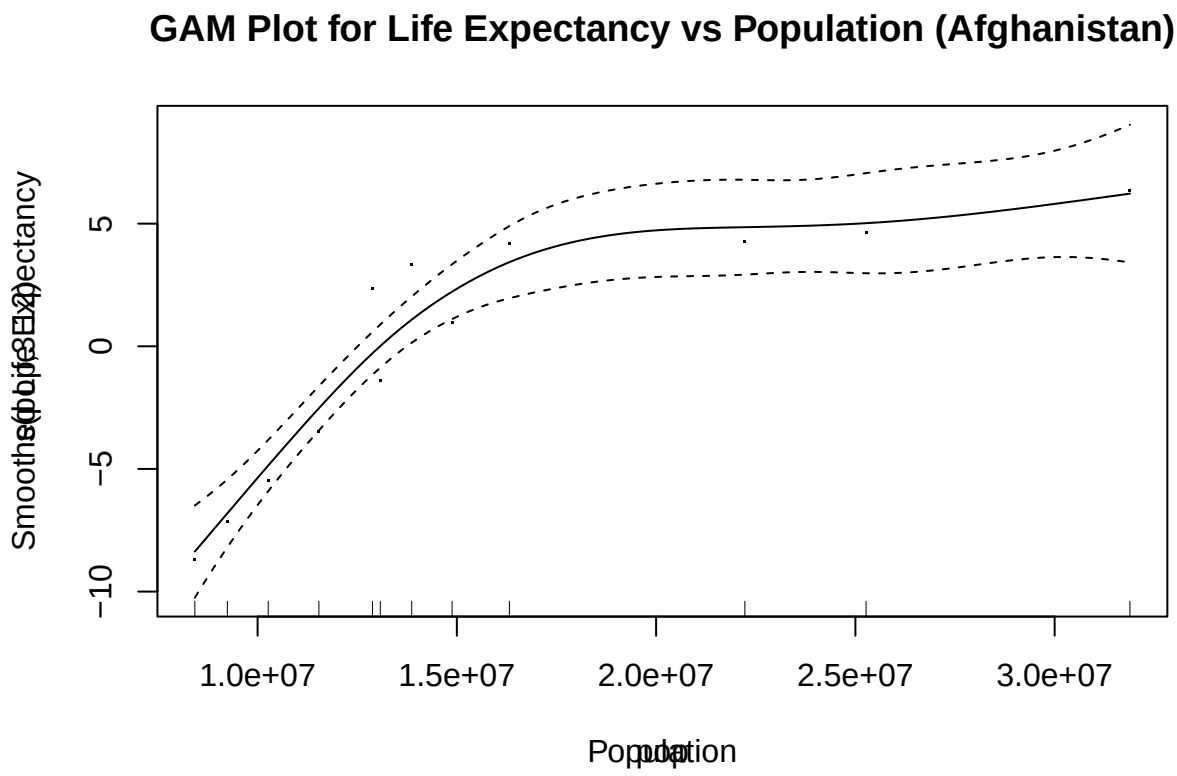
```
## This is mgcv 1.9-1. For overview type 'help("mgcv-package")'.
```

```
# Filter for Afghanistan data
afghanistan_data <- gapminder[gapminder$country == "Afghanistan",]

# Fit a generalized additive model predicting life expectancy from population
gam_model <- gam(lifeExp ~ s(pop), data = afghanistan_data)

# Summarize the model output
summary(gam_model)
```

```
##
## Family: gaussian
## Link function: identity
##
## Formula:
```

```
## lifeExp ~ s(pop)
##
## Parametric coefficients:
##             Estimate Std. Error t value Pr(>|t|)
## (Intercept)  37.4788     0.4396   85.25 5.77e-13 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Approximate significance of smooth terms:
##          edf Ref.df     F p-value
## s(pop) 3.123  3.754 29.85 7.7e-05 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## R-sq.(adj) =  0.911   Deviance explained = 93.6%
## GCV = 3.5332  Scale est. = 2.3191    n = 12
```

```r
# Plot the model to visualize the smooth term
plot(gam_model, residuals = TRUE, rug = TRUE, main = "GAM Plot for Life Expectancy vs Population (Afghan

# Add labels to the plot
title(xlab = "Population", ylab = "Smoothed Life Expectancy")
```

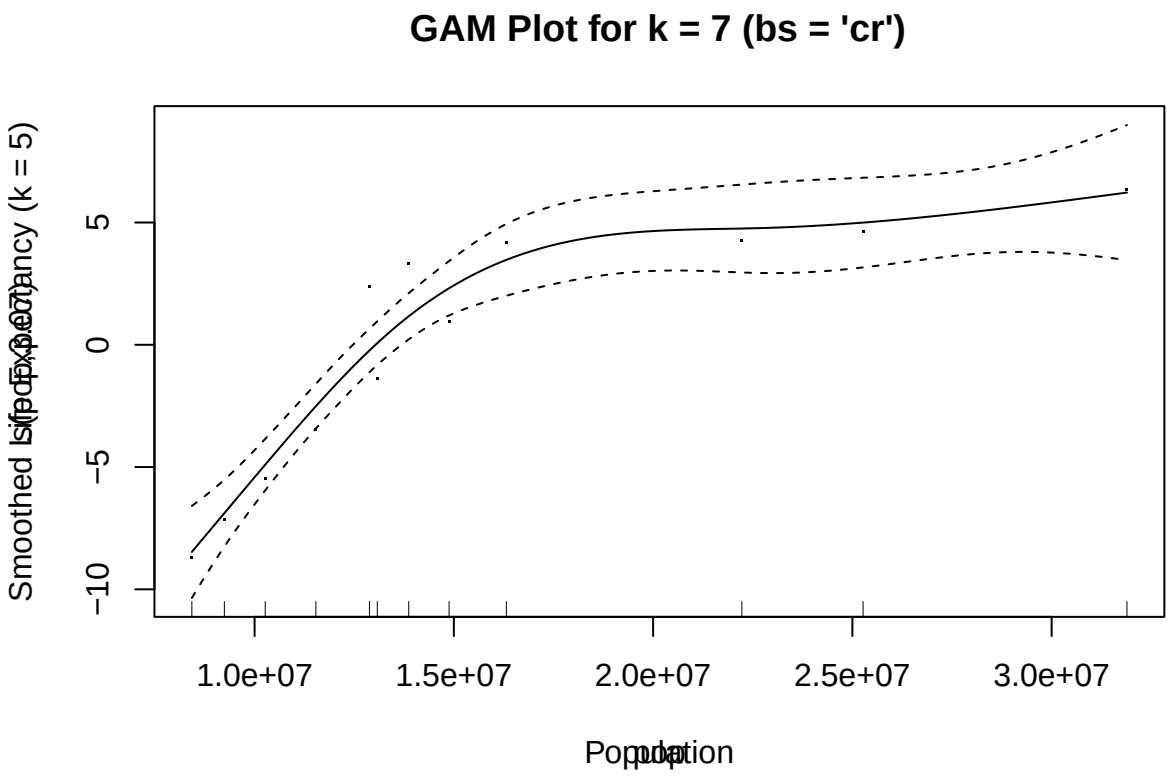


3.Repeat the fitting process, this time using one of the other smoothing functions (search "smooth.terms" in the R ###studio help window to see those available). Show how changing the value of k affects the fit and AIC value.

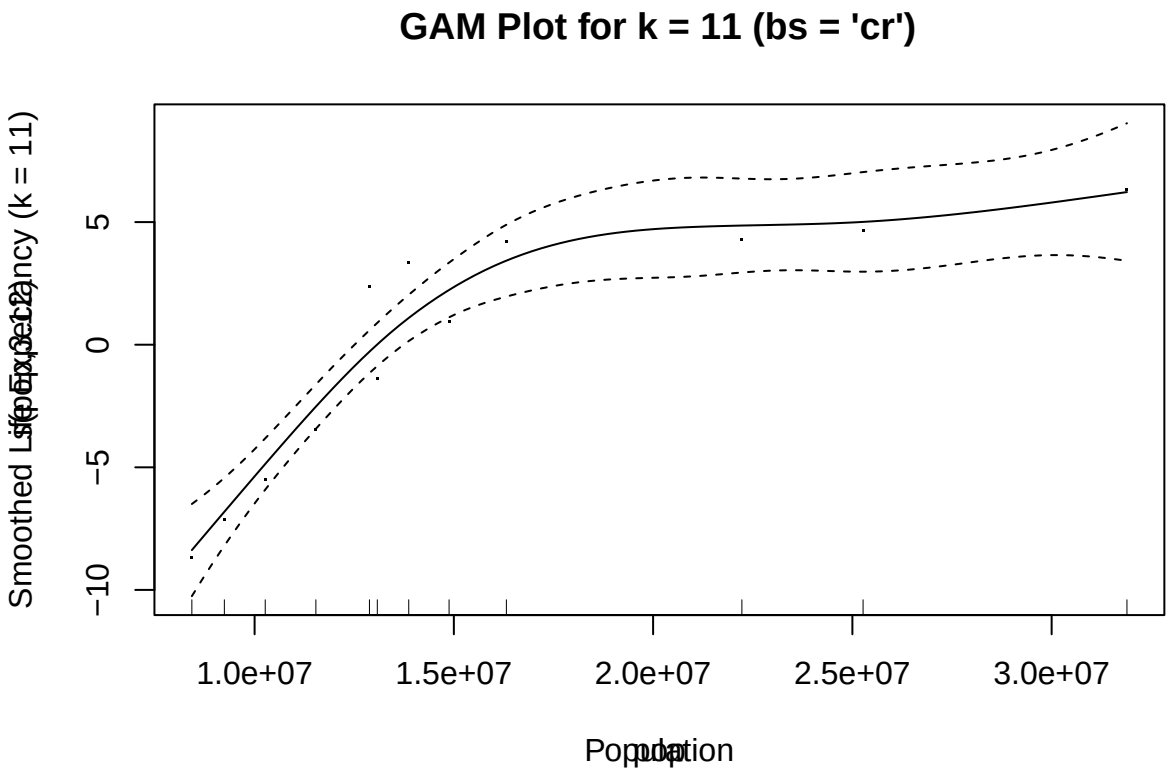
```r
# Load necessary libraries
library(gapminder)
library(mgcv)

# Filter for Afghanistan data
afghanistan_data <- gapminder[gapminder$country == "Afghanistan",]

# Fit two generalized additive models with bs = "cr" and different values of k
model_cr_k7 <- gam(lifeExp ~ s(pop, bs = "cr", k = 7), data = afghanistan_data)
model_cr_k11 <- gam(lifeExp ~ s(pop, bs = "cr", k = 11), data = afghanistan_data)

# Compare models using AIC
AIC_values <- AIC(model_cr_k7, model_cr_k11)
print(AIC_values)
```

```
##                    df      AIC
## model_cr_k7  5.068026 48.87126
## model_cr_k11 5.115024 49.31583
```

```r
# Plot the GAM model with k = 7
plot(model_cr_k7, residuals = TRUE, rug = TRUE, main = "GAM Plot for k = 7 (bs = 'cr')", se = TRUE)
title(xlab = "Population", ylab = "Smoothed Life Expectancy (k = 5)")
```



```r
# Plot the GAM model with k = 11
plot(model_cr_k11, residuals = TRUE, rug = TRUE, main = "GAM Plot for k = 11 (bs = 'cr')", se = TRUE)
title(xlab = "Population", ylab = "Smoothed Life Expectancy (k = 11)")
```




```r
###3. Repeat the fitting process, this time using one of the other smoothing functions
###(search "smooth.terms" in the R studio help window to see those available). Show how
###changing the value of k affects the fit and AIC value.

model_cr_k7 <- gam(lifeExp ~ s(pop, bs = "cr", k = 7), data = afghanistan_data)
model_cr_k11 <- gam(lifeExp ~ s(pop, bs = "cr", k = 11), data = afghanistan_data)
# Compare models using AIC
AIC_values <- AIC(model_cr_k7, model_cr_k11)
print(AIC_values)
```

```
##                    df      AIC
## model_cr_k7  5.068026 48.87126
## model_cr_k11 5.115024 49.31583
```

4.Load the abalone dataset. Use the rsample library and vfold_cv() function to create training and test sets with ###10-fold cross-validation.

```r
#Loaded necessary libraries
library(rsample)
```

```r
library(mgcv)
abalone_data<-read.csv("~/Downloads/abalone.csv")
# Assuming abalone_data is already loaded
# Create 10-fold cross-validation
summary(abalone_data)
```

```
##      Sex                Length         Diameter          Height
##  Length:4177        Min.   :0.075   Min.   :0.0550   Min.   :0.0000
##  Class :character   1st Qu.:0.450   1st Qu.:0.3500   1st Qu.:0.1150
##  Mode  :character   Median :0.545   Median :0.4250   Median :0.1400
##                     Mean   :0.524   Mean   :0.4079   Mean   :0.1395
##                     3rd Qu.:0.615   3rd Qu.:0.4800   3rd Qu.:0.1650
##                     Max.   :0.815   Max.   :0.6500   Max.   :1.1300
##   Whole.weight    Shucked.weight   Viscera.weight    Shell.weight
##  Min.   :0.0020   Min.   :0.0010   Min.   :0.0005   Min.   :0.0015
##  1st Qu.:0.4415   1st Qu.:0.1860   1st Qu.:0.0935   1st Qu.:0.1300
##  Median :0.7995   Median :0.3360   Median :0.1710   Median :0.2340
##  Mean   :0.8287   Mean   :0.3594   Mean   :0.1806   Mean   :0.2388
##  3rd Qu.:1.1530   3rd Qu.:0.5020   3rd Qu.:0.2530   3rd Qu.:0.3290
##  Max.   :2.8255   Max.   :1.4880   Max.   :0.7600   Max.   :1.0050
##      Rings
##  Min.   : 1.000
##  1st Qu.: 8.000
##  Median : 9.000
##  Mean   : 9.934
##  3rd Qu.:11.000
##  Max.   :29.000
```

```r
vfold <- vfold_cv(abalone_data, v = 10)
```

5.Use the fit_gam() and predict_gam() functions defined in the online R activity, then generate fitted models and

###predititctions for each fold.

```r
# Define fit_gam() function
fit_gam <- function(train_data) {
gam(Rings ~ s(Length), data = train_data) # Using Length to predict Rings
}
# Define predict_gam() function
predict_gam <- function(model, test_data) {
predict(model, newdata = test_data)
}
# Fit GAM models and generate predictions for each fold
results <- lapply(vfold$splits, function(split) {
train_data <- analysis(split)
test_data <- assessment(split)
model <- fit_gam(train_data)
predictions <- predict_gam(model, test_data)
list(model = model, predictions = predictions, actual = test_data$Rings)
})
```

6.Report on the errors and their variation for each fold.

```
# Calculate Mean Squared Errors for each fold
errors <- lapply(results, function(res) {
  mean((res$predictions - res$actual)^2) # Mean Squared Error
})

# Calculate mean and standard deviation of errors
mean_error <- mean(unlist(errors))
sd_error <- sd(unlist(errors))

# Print mean error and standard deviation
print(paste("Mean Error:", mean_error))
```

```
## [1] "Mean Error: 7.10239881902816"
```

```
print(paste("Error Standard Deviation:", sd_error))
```

```
## [1] "Error Standard Deviation: 0.786759284285156"
```

```
print("The errors for each of the folds are: ")
```

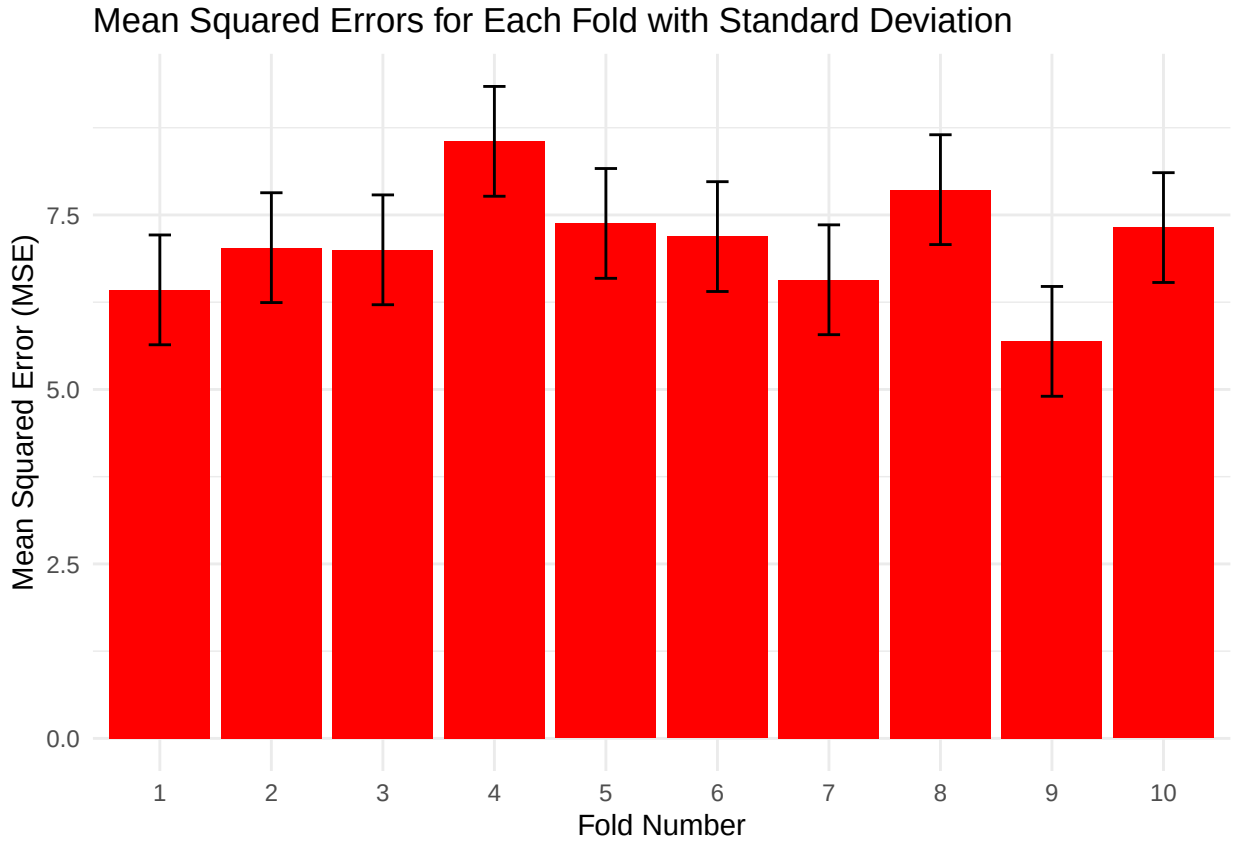
```
## [1] "The errors for each of the folds are: "
```

```
print(errors)
```

```
## [[1]]
## [1] 6.426703
##
## [[2]]
## [1] 7.031455
##
## [[3]]
## [1] 7.000483
##
## [[4]]
## [1] 8.554937
##
## [[5]]
## [1] 7.378646
##
## [[6]]
## [1] 7.189976
##
## [[7]]
## [1] 6.57143
##
## [[8]]
## [1] 7.862378
##
```

```
## [[9]]
## [1] 5.6887
##
## [[10]]
## [1] 7.31928
```

```r
# Plotting the errors and their variation for each fold
library(ggplot2)

# Convert the errors list to a data frame for plotting
errors_df <- data.frame(Fold = seq_along(errors), MSE = unlist(errors))

# Create a bar plot for errors with error bars
ggplot(errors_df, aes(x = factor(Fold), y = MSE)) +
  geom_bar(stat = "identity", fill = "red") +
  geom_errorbar(aes(ymin = MSE - sd_error, ymax = MSE + sd_error), width = 0.2) +
  labs(title = "Mean Squared Errors for Each Fold with Standard Deviation",
       x = "Fold Number", y = "Mean Squared Error (MSE)") +
  theme_minimal()
```



R Markdown

This is an R Markdown document. Markdown is a simple formatting syntax for authoring HTML, PDF, and MS Word documents. For more details on using R Markdown see http://rmarkdown.rstudio.com.

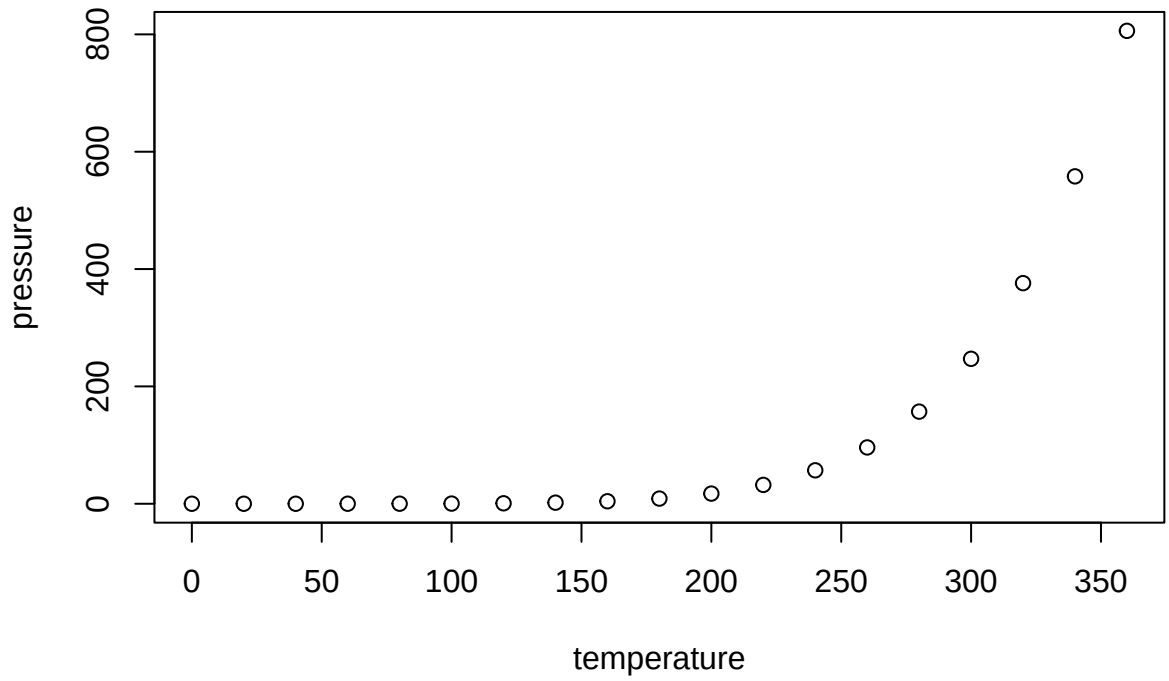
When you click the **Knit** button a document will be generated that includes both content as well as the output of any embedded R code chunks within the document. You can embed an R code chunk like this:

```
summary(cars)
```

```
##      speed           dist       
##  Min.   : 4.0   Min.   :  2.00  
##  1st Qu.:12.0   1st Qu.: 26.00  
##  Median :15.0   Median : 36.00  
##  Mean   :15.4   Mean   : 42.98  
##  3rd Qu.:19.0   3rd Qu.: 56.00  
##  Max.   :25.0   Max.   :120.00
```

Including Plots

You can also embed plots, for example:



Note that the `echo = FALSE` parameter was added to the code chunk to prevent printing of the R code that generated the plot.