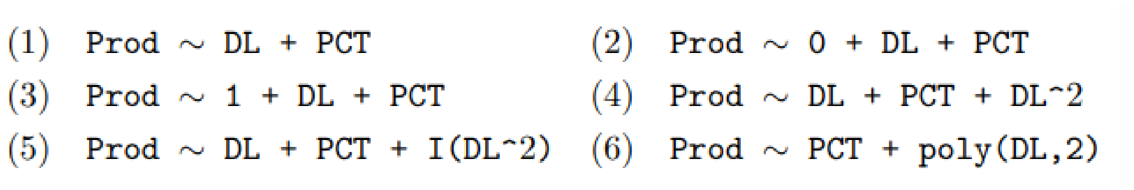
Regression Coursework 1

2025-01-17

Problem 2 Practical (20 marks)

1. **To begin, we are going to investigate the linear model:**

   - *By conducting your own research, read and describe the predictors included in each of the following lm formulas. Do any of the below models give the same parameter estimates? If so, which?* ***(2 marks)***

(1)	`Prod ~ DL + PCT`	(2)	`Prod ~ 0 + DL + PCT`
(3)	`Prod ~ 1 + DL + PCT`	(4)	`Prod ~ DL + PCT + DL^2`
(5)	`Prod ~ DL + PCT + I(DL^2)`	(6)	`Prod ~ PCT + poly(DL,2)`

     Model (1) , (3) are the same because the intercept is implicitly included in Model (1), whereas Model (3) is explicitly included. Model (2) explicitly removes the intercept because of 0, therefore parameter estimates for DL and PCT change because the model assumes the line passes through the origin. Model (4) shares same parameter estimates and intercept terms with Model (1) and (3), because DL is demeaned and rescaled, therefore having no effect, Model (4) contains the intercept term , DL and PCT as predictors. Model (5) is not the same because the I() function is applied and R explicitly includes the quadratic effect of DL, rather than begin treated as special formula operators. Model (6) differs from the rest because poly() creates orthogonal basis functions, meaning the polynomial terms for DL (linear and quadratic) are uncorrelated.

   - *Using only base R commands, write a function my_lm. Your function must:*
     - *take as input two data frames X and Y , representing a design matrix and response vector respectively,*
     - *return as output the β^{OLS} estimator as a matrix,*
     - *check if X includes an intercept column. If X does not include an intercept column, your code must add one to the model. You may assume for this part of the exercise that X does not contain categorical data.*

     You may assume for this part of the exercise that X does not contain categorical data. ***(4 marks)***

```r
load(file = "C:/Users/tanwe/OneDrive/Documents/Stats_Machine_Learning/.RData")

my_lm <- function(X,y){
  # Check if X has a column which contains entirely of ones
  if (!any(colSums(X == 1) == nrow(X))){
    X <- cbind(Intercept = 1, X)
  }

  #Compute OLS estimator
  beta_hat <- solve(t(X) %*% X) %*% t(X) %*% y
  return(beta_hat)
}
```

- *Using the continuous variables in the pets dataset, write code demonstrating that your function* my_lm *gives the same output as lm.* ***(1 mark)***

```r
#continuous random variables are all predictors except PT

X <- as.matrix(pets[, !colnames(pets) %in% c("PT","Prod")])
y <- as.matrix(pets[,"Prod"])

#Using my_lm function
beta_hat_my_lm <- my_lm(X,y)

#Use built in lm function
lm_model <- lm(Prod ~ NTT + PCT + EW + DL, data=pets)
beta_hat_lm <- coef(lm_model)

# Print results and compare
print("Manually generated my_lm coefficients:")
```

```
## [1] "Manually generated my_lm coefficients:"
```

```r
print(beta_hat_my_lm)
```

```
##                  [,1]
## Intercept  7.64311951
## NTT       -0.14157986
## PCT       -0.02737829
## EW         0.07367194
## DL         0.03646101
```

```r
print("Built-in R lm coefficients:")
```

```
## [1] "Built-in R lm coefficients:"
```

```r
print(beta_hat_lm)
```

```
## (Intercept)         NTT         PCT          EW          DL
##  7.64311951 -0.14157986 -0.02737829  0.07367194  0.03646101
```

```r
#All coefficients are the same for both models. Hence, proven.
```

- *The term "one-hot encoding" refers to the act of converting a categorical predictor Z to a binary representation, as shown below:*

$$Z = \begin{bmatrix} A \\ C \\ B \\ A \\ B \\ B \\ C \end{bmatrix} \mapsto \begin{bmatrix} 1 & 0 & 0 \\ 0 & 0 & 1 \\ 0 & 1 & 0 \\ 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix}$$

- *In practice, whenever you enter a factor into lm, the factor is being one-hot-encoded in this way behind the scenes. Write your own function which takes in a dataframe X and checks for non-numeric columns of X. If your code encounters a column with non-numeric data it must do the following: (i) remove the column from X, (ii) one-hot encode the column, (iii) add the one-hot encoded data back into X.* ***(3 marks)***

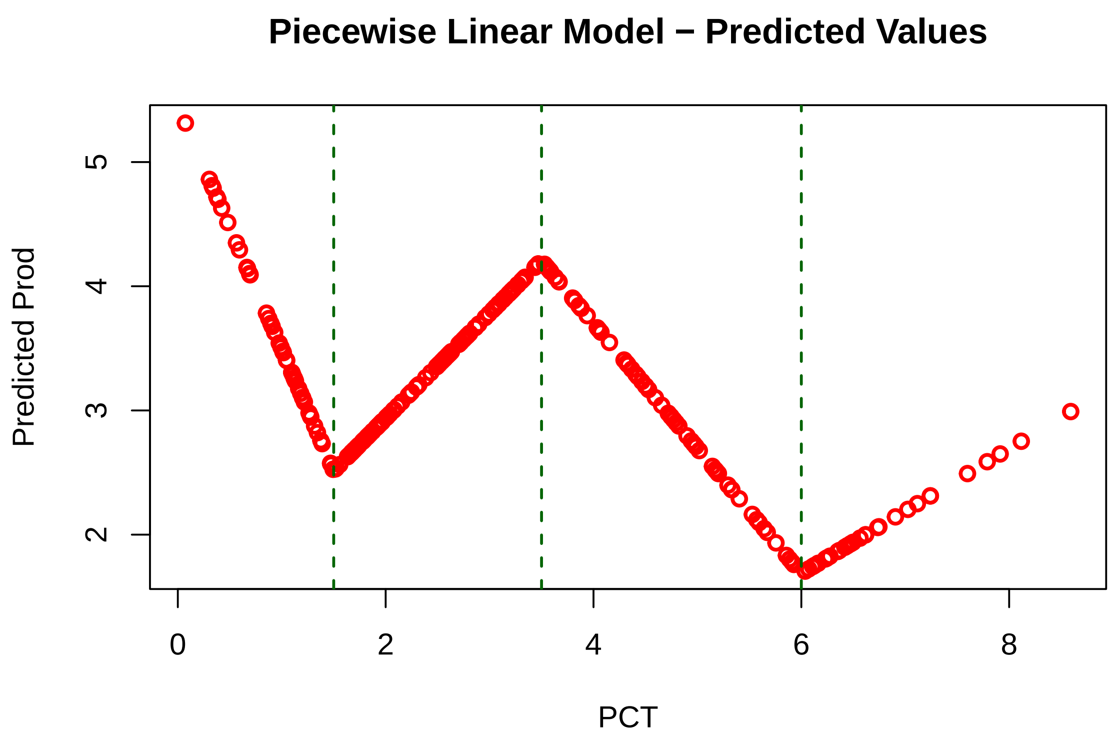
-
  ```
  one_hot_encoding <- function(X) {
    for (col_name in colnames(X)) {

      if (!is.numeric(X[[col_name]])) {    # Check if the column is non-numeric
        unique_values <- unique(X[[col_name]])
        for (val in unique_values) {
          new_col <- as.numeric(X[[col_name]] == val)
          col_name_encoded <- paste(col_name, val, sep = "_")
          X[[col_name_encoded]] <- new_col
        }

        X[[col_name]] <- NULL   # Remove the original non-numeric column
      }
    }
    return(X)
  }

  X <- head(data.frame(pets[, !colnames(pets) %in% c("Prod")]))
  one_hot_encoding(X)
  ```

```
##   NTT      PCT        EW        DL PT_Dog PT_Cat PT_Ranitomeya Amazonica
## 1  34 4.771030 0.5980490 1.9972050      1      0                       0
## 2  31 1.874099 0.6958123 2.7778355      0      1                       0
## 3  29 2.747655 0.5657481 1.6811024      1      0                       0
## 4  38 5.144583 0.2093155 0.3035256      0      0                       1
## 5  31 4.821496 0.5721773 1.7272703      0      1                       0
## 6  26 2.720366 0.7383718 1.9464782      0      1                       0
```

2) *For each of the following, write R code which runs the specified model and plots the model's predicted values. Each model must use the knots 1.5, 3.5 and 6:*

   - *Using lm , fit a piecewise linear model with PCT as a predictor and Prod* as a response. **(1 mark)**

```r
# Define the knots
knots <- c(1.5, 3.5, 6)

# Create the piecewise terms
pets$piecewise_1 <- pmax(0, pets$PCT - knots[1])
pets$piecewise_2 <- pmax(0, pets$PCT - knots[2])
pets$piecewise_3 <- pmax(0, pets$PCT - knots[3])

# Fit the piecewise linear model
model <- lm(Prod ~ PCT + piecewise_1 + piecewise_2 + piecewise_3, data = pets)

# Generate predicted values
pets$Predicted <- predict(model)

# Sort the data by PCT for proper line plotting
pets <- pets[order(pets$PCT), ]

# Plot observed and predicted values as points
plot(pets$PCT, pets$Predicted, lwd=2, col = "red",
     main = "Piecewise Linear Model - Predicted Values",
     xlab = "PCT", ylab = "Predicted Prod",
     xlim = c(min(pets$PCT), max(pets$PCT)), ylim = c(min(pets$Predicted), max(pets$Predicted)))

# Add vertical dashed lines for knots
abline(v = knots, col = "darkgreen", lty = 2, lwd = 1.5)
```

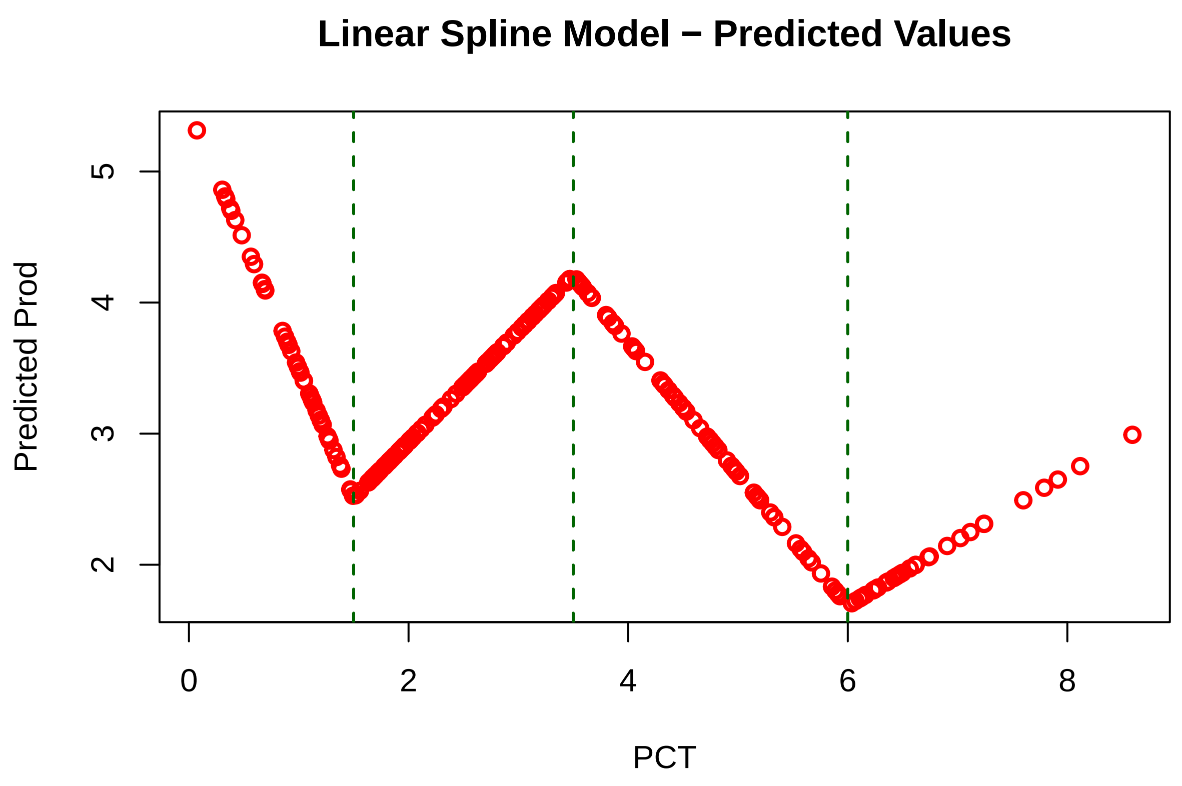


```r
print(model)
```

```
##
## Call:
## lm(formula = Prod ~ PCT + piecewise_1 + piecewise_2 + piecewise_3,
##     data = pets)
##
## Coefficients:
## (Intercept)          PCT  piecewise_1  piecewise_2  piecewise_3
##       5.456       -1.961        2.807       -1.853        1.510
```

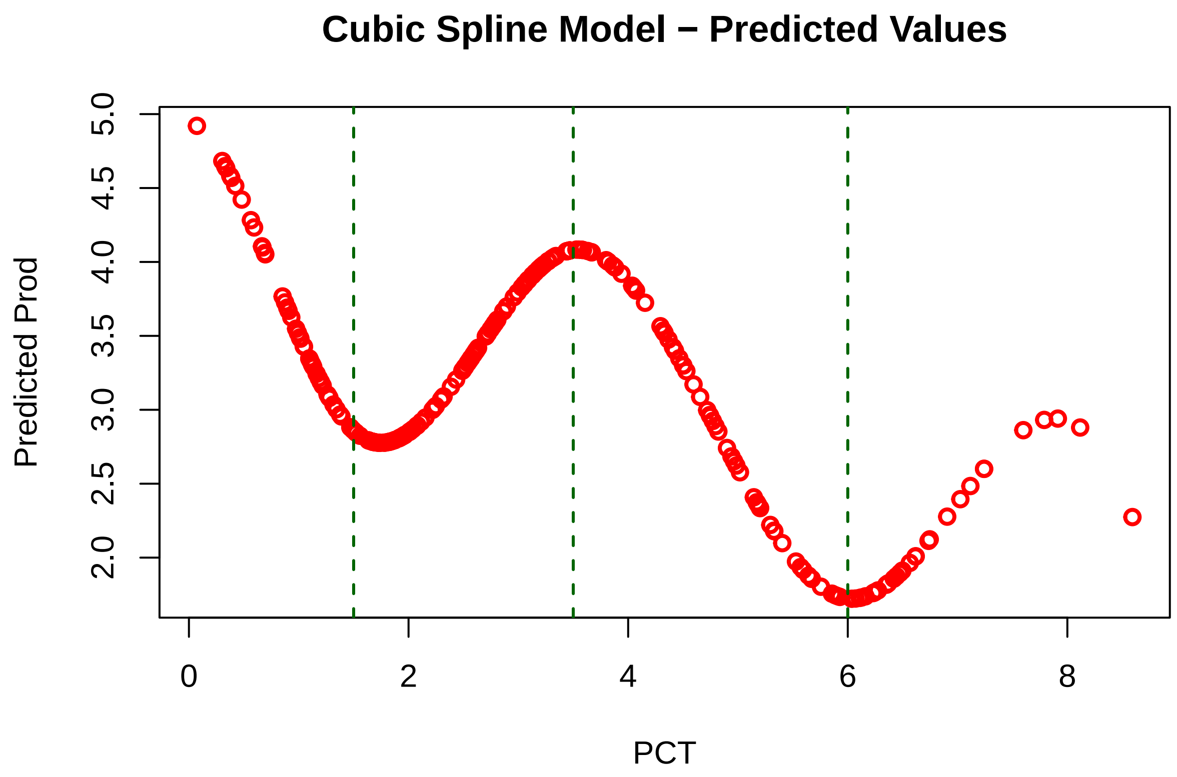
- *Using lm and functions from the splines package, fit a linear spline model with PCT as a predictor and Prod as a response.* ***(1 mark)***

```r
library(splines)

# Define knots for the spline
knots <- c(1.5, 3.5, 6)

# Fit the linear spline model using bs() for B-splines
spline_model <- lm(Prod ~ bs(PCT, knots = knots, degree = 1), data = pets)

# Predicted values
pets$Predicted_linear_spline <- predict(spline_model)

# Add the spline fit (predicted values)
plot(pets$PCT, pets$Predicted_linear_spline, col = "red", lwd = 2,main = "Linear Spline Model - Predicte

# Add vertical dashed lines for knots
abline(v = knots, col = "darkgreen", lty = 2, lwd = 1.5)
```



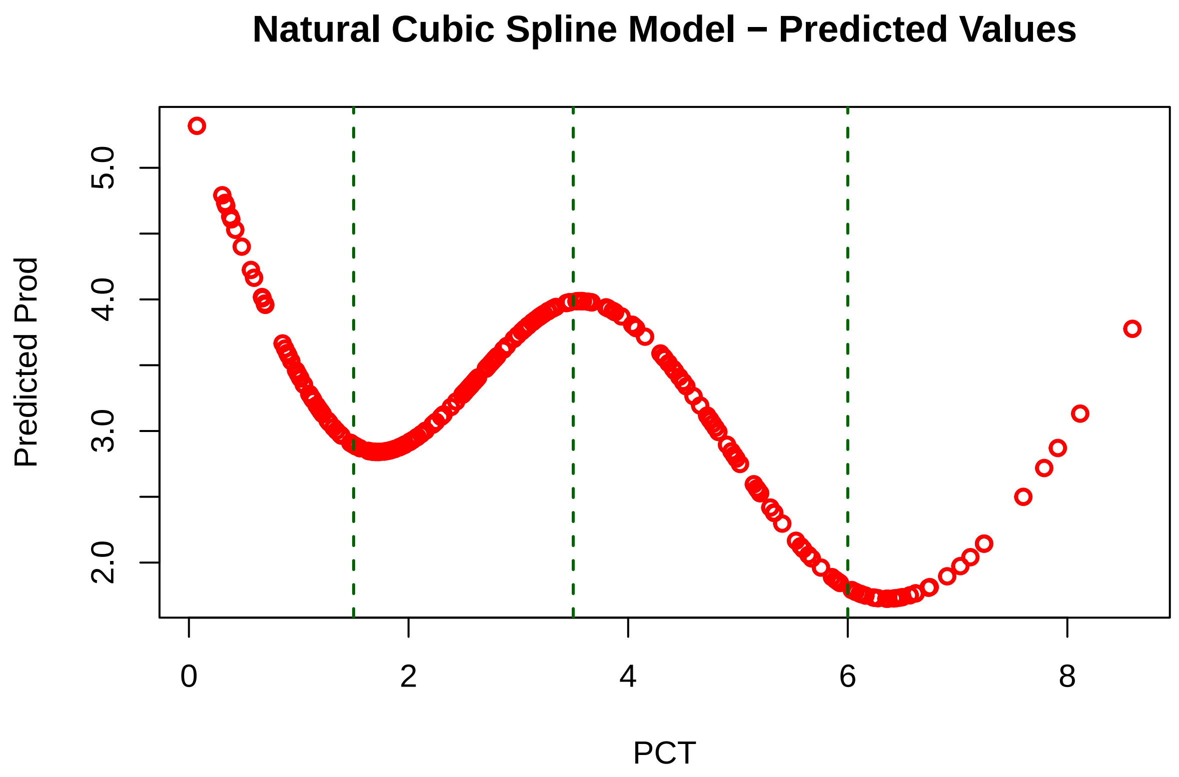
- *Using lm and functions from the splines package, fit a cubic spline model with PCT as a predictor and Prod as a response.* **(1 mark)**

```
# Define knots for the spline
knots <- c(1.5, 3.5, 6)

# Fit the linear spline model using bs() for B-splines
spline_model <- lm(Prod ~ bs(PCT, knots = knots, degree = 3), data = pets)

# Predicted values
pets$Predicted_cubic_spline <- predict(spline_model)

# Add the spline fit (predicted values)
plot(pets$PCT, pets$Predicted_cubic_spline, col = "red", lwd = 2, main = "Cubic Spline Model - Predicte

# Add vertical dashed lines for knots
abline(v = knots, col = "darkgreen", lty = 2, lwd = 1.5)
```



- *Using lm and functions from the splines package, fit a natural cubic spline model with PCT as a predictor and Prod as a response.* ***(1 mark)***

```
# Define knots for the spline
knots <- c(1.5, 3.5, 6)

# Fit the linear spline model using bs() for B-splines
spline_model <- lm(Prod ~ ns(PCT, knots = knots), data = pets)

# Predicted values
pets$Predicted_natural_cubic_spline <- predict(spline_model)

# Add the spline fit (predicted values)
plot(pets$PCT, pets$Predicted_natural_cubic_spline, col = "red", lwd = 2,main = "Natural Cubic Spline Mo

# Add vertical dashed lines for knots
abline(v = knots, col = "darkgreen", lty = 2, lwd = 1.5)
```



3) *In part (ii), we used splines to model the nonlinear relationship between Prod and PCT. In this question, we shall continue to explore this relationship, but using different methods.*

   - *Based on what we have seen in class, suggest a different approach which could be used for modeling the relationship between Prod and PCT. Use an appropriate R package to fit your proposed model and make a plot of the fitted values.* ***(2 marks)***

```
# We can use Generalised Additive Models(GAM)
library(mgcv)
```

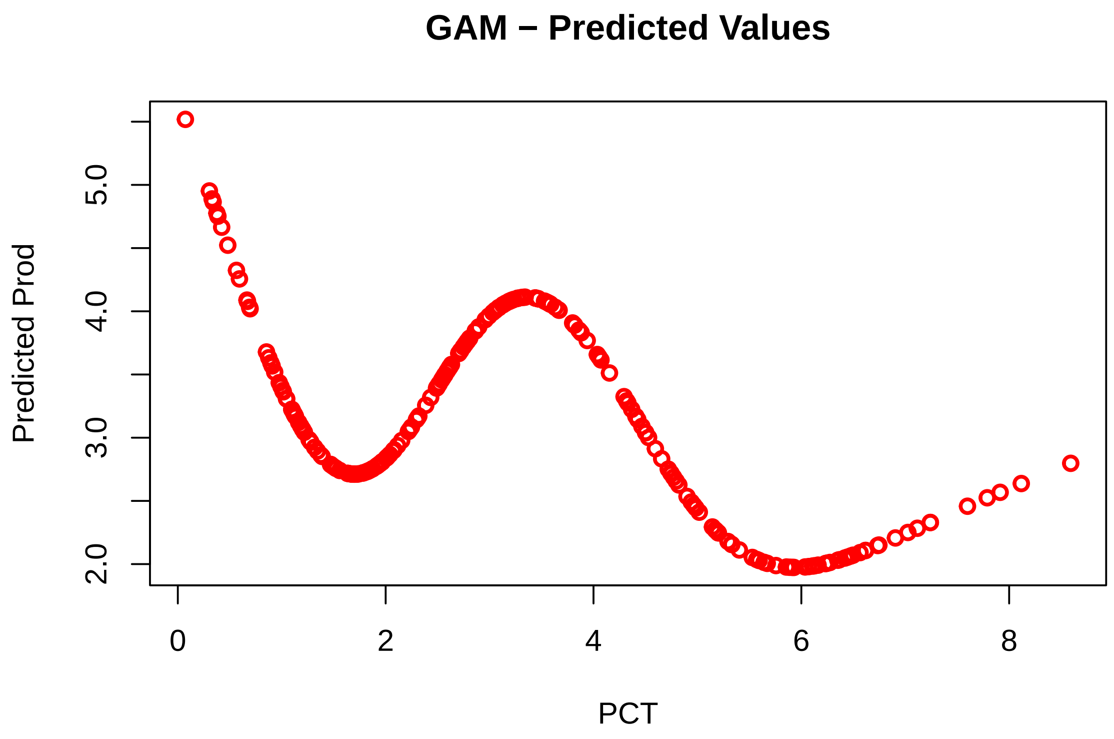
```
## Loading required package: nlme

## This is mgcv 1.8-41. For overview type 'help("mgcv-package")'.
```

```
# Fit a GAM with a smooth term for PCT
gam_model <- gam(Prod ~ s(PCT, k=6), data = pets)

summary(gam_model)
```

```
##
## Family: gaussian
## Link function: identity
##
## Formula:
## Prod ~ s(PCT, k = 6)
```

```
##
## Parametric coefficients:
##             Estimate Std. Error t value Pr(>|t|)
## (Intercept)  3.15630    0.02536   124.4   <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(PCT) 4.987      5 169.3  <2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## R-sq.(adj) =  0.773   Deviance explained = 77.7%
## GCV = 0.16478  Scale est. = 0.16083   n = 250
```

```
pets$Fitted_gam <- predict(gam_model)

# Add the spline fit (predicted values)
  plot(pets$PCT, pets$Fitted_gam, col = "red", lwd = 2,main = "GAM - Predicted Values", xlab="PCT",
```



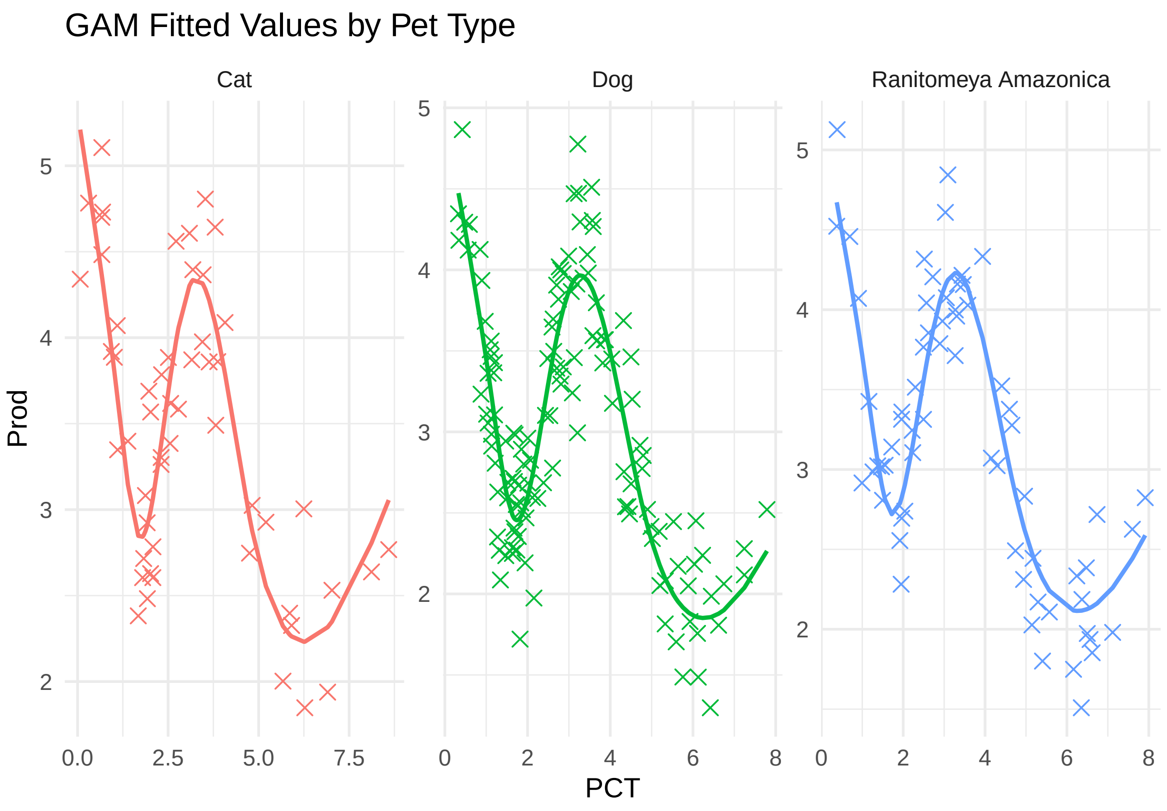
- *Modify your solution to (a) to incorporate the PetType variable in your model. Then, create separate plots of the fitted values for each type of pet.* ***(2 marks)***

```
library(ggplot2)
```

```
## Warning: package 'ggplot2' was built under R version 4.2.3
```

```r
# Fit a GAM with an interaction between PCT and PT
gam_model <- gam(Prod ~ s(PCT, k=10 , bs="cs") + PT, data = pets)

# Generate fitted values
pets$Fitted <- predict(gam_model)

# Create separate plots for each pet type
ggplot(pets, aes(x = PCT, y = Fitted, color = PT)) +
geom_line(size = 0.8) +
geom_point(aes(y = Prod), size = 2.5, shape = 4) +
facet_wrap(~PT, scales = "free") +
theme_minimal() +
labs(
title = "GAM Fitted Values by Pet Type",
x = "PCT",
y = "Prod",
color = "Pet Type"
) +
theme(legend.position = "none")
```

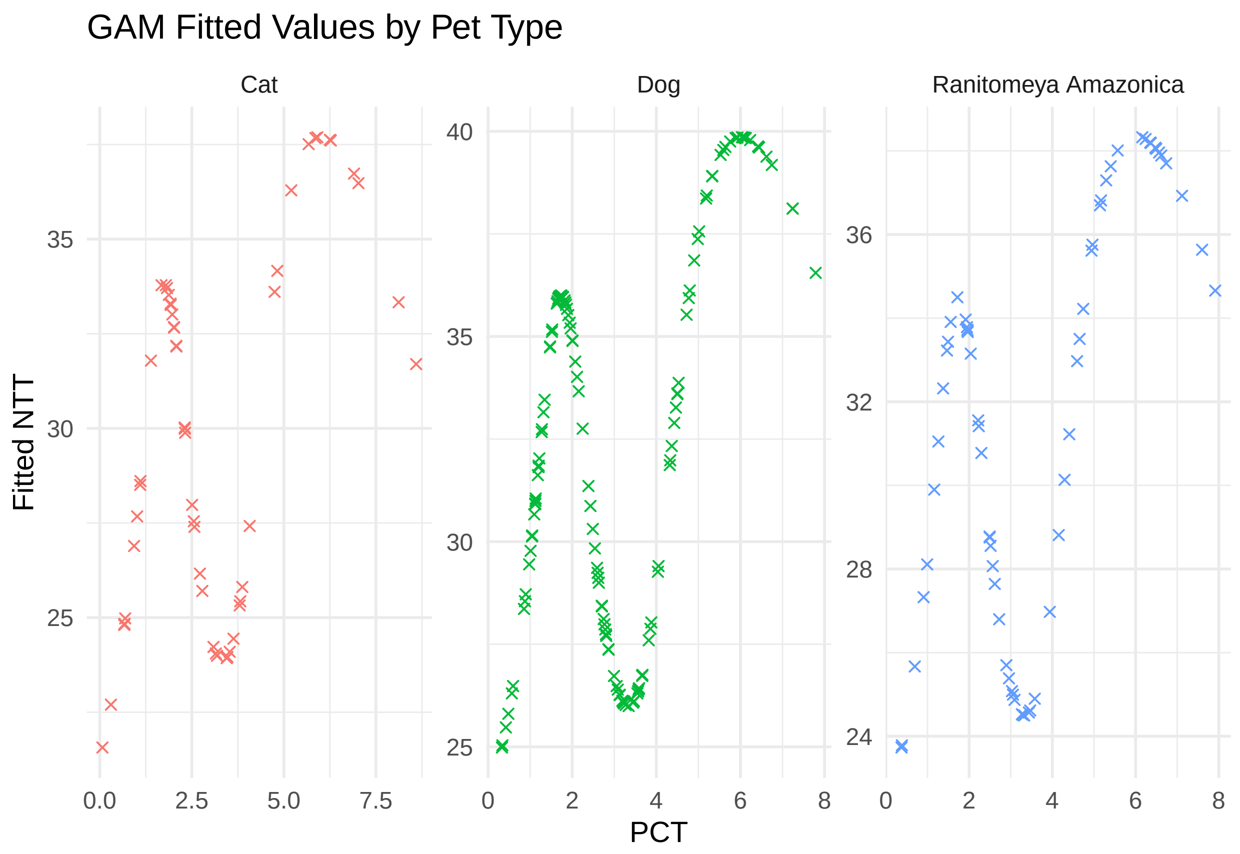
```
## Warning: Using `size` aesthetic for lines was deprecated in ggplot2 3.4.0.
## i Please use `linewidth` instead.
## This warning is displayed once every 8 hours.
## Call `lifecycle::last_lifecycle_warnings()` to see where this warning was
## generated.
```



- *Suppose that instead of Prod, we wished to model NTT as the response variable. In one sentence,* *describe how you would modify your solution to part (b) to reflect this change. Provide the updated code (including plots) reflecting this substitution.* ***(2 marks)***

I changed all Prod terms to NTT and adjusted all axis labels.

```r
# Fit a GAM with an interaction between PCT and PT
gam_model <- gam(NTT ~ s(PCT, k = 10, bs = "cs") + PT, data = pets)

# Generate fitted values
pets$Fitted_New <- predict(gam_model)

# Plot fitted values only as points
ggplot(pets, aes(x = PCT, y = Fitted_New, color = PT)) +
  geom_point(size = 1.5, shape = 4) +
  facet_wrap(~PT, scales = "free") +
  theme_minimal() +
  labs(
    title = "GAM Fitted Values by Pet Type",
    x = "PCT",
    y = "Fitted NTT",
    color = "Pet Type"
  ) +
  theme(legend.position = "none")
```


GAM Fitted Values by Pet Type
Cat
Dog
Ranitomeya Amazonica
Fitted NTT
PCT