

```

# Load necessary libraries
library(mgcv)
library(data.table)

# Read the initial data from data.csv
initialData <- read.table("data.csv", header = TRUE, sep = ";")

# Remove rows with missing age values
initialData <- initialData[!is.na(initialData$age),]

# Initialise agespan
x <- seq(15, 105, length = 901)
Ndata <- data.frame(Age = x)

# Initialise columns of interest.
initialColumns <- c(6:74)
initialDistance <- rep(0, 68)

# Initialise binary dataset
newData <- data.frame(initialData[c(1:1638), names(initialData)]) # 1638 is the number of
data rows

# Function for calculating logProb data for a given column from the binary data
csvOutput <- function(column, data2){
  age <- x
  if (dim(data2)[1] >= 10) { # at least 10 data points needed for proper analysis
    # Initialising data for models
    data.fit <- data.frame(Age = data2$age, y = data2[,column])

    # Generating generalised additive model (GAM)
    gam1 <- gam(y ~ s(Age) , family = binomial, data = data.fit)

    # Smoothing the GAM and updating the model
    gam1 <- gam(y ~ s(Age) , family = binomial, data = data.fit, sp = gam1$sp)

    # Generating generalised linear model (GLM) for both original and log-transformed
agespan
    glm1 <- glm(y ~ Age , family = binomial, data = data.fit)
    glmLog <- glm(y ~ log(Age) , family = binomial, data = data.fit)

    # Predicting transition probabilities using the calculated models
    Nprob.glm <- predict(glm1, newdata = Ndata, type = "response", se.fit = TRUE)
    Nprob.glmLog <- predict(glmLog, newdata = Ndata, type = "response", se.fit = TRUE)
    Nprob.gam <- predict(gam1, newdata = Ndata, type = "response", se.fit = TRUE)
  }
}

```

```

# Finding the best model according to the AIC values
chosen <- "GAM"
if (AIC(glm1) > AIC(glmLog)) {
  if (AIC(gam1) > AIC(glmLog)) {
    chosen <- "GLM log"
  }
} else if (AIC(gam1) > AIC(glm1)) {
  chosen <- "GLM"
}

# Log-transforming probabilities
glmProbs <- log(Nprob.glm$fit)
glmLogProbs <- log(Nprob.glmLog$fit)
gamProbs <- log(Nprob.gam$fit)

# Returning the log-transformed probabilities from the chosen model
if (chosen == "GLM") {
  return(glmProbs)
} else if (chosen == "GLM log") {
  return(glmLogProbs)
} else {
  return(gamProbs)
}
}
}

# Prepare lists for looping
sample_list <- list(newData)
name_vector <- c("sample")

# Calculate logProb data for all characteristics
for(loopIndex in 1:length(sample_list)){
  # Loading data corresponding to current iteration
  sampleData <- sample_list[[loopIndex]]

  # Calculate logProbs for all traits
  for (col in 1:length(initialColumns)) {
    columnL <- initialColumns[col]
    bilColumns <- c(columnL)

    for(column in bilColumns){
      data2 <- sampleData[!is.na(sampleData[,column]),]
    }
  }
}

```

```
# Column title
title.text <- names(data2)[column]

# Calculating logProbs
outputData <- csvOutput(column, data2)

# Add calculated logProbs to the dataset
csvOutputs <- cbind(csvOutputs, outputData)
names(csvOutputs)[length(names(csvOutputs))] <- title.text
}
}
}

# Generate filename inside the loop
filename <- "results.csv"

# Printing logProbs data
write.table(csvOutputs, file = filename, row.names = FALSE, quote = FALSE, sep = ";")
```