279 Option 2

Ben Miller

2023-12-14

Contents

Linear helper functions

```
#This function is to be used within my function for linear models. It calculates the standard errors, t

linear_sig2 <- function(X,beta,n,y){
  yhats <- X%*%beta
  sigma2 <- (1/(n-ncol(X)))*(sum((y-yhats)^2))
  Diag <- diag(sigma2*solve(t(X) %*% X))
  se <- sqrt(Diag) #calculate SE's from the above info
  t_score <- beta/se #calc t_scores
  t_score1 <- abs(t_score) #need absvalue when calculating pvalues or the p vals will be wrong
  p_value <- 2*pt(t_score1,(n-ncol(X)),lower.tail=F)
  coeff_df <- data.frame(beta,se,t_score,p_value) #turns into data frame for output
  return(coeff_df)
}
```

Linear with simulated data

```
# Simulate some data
set.seed(279)
n <- 100
x1 <- runif(n)
x2 <- rnorm(n)
y <- rnorm(n, 1 + x1 + x2, 2)

# create the data
data <- data.frame(y, x1, x2)

#Below is a helper function for linear models. It calculates coefficients and then uses the helper func
```

```r
myglm_linear2 <- function(formula,family,df){
  X <- model.matrix(formula, df) #create design matrix from df and formula
  response <- all.vars(formula)[1]
  y <- df[[response]] #this and the above line extract the response variable's values and store it as y
  beta <- solve(t(X) %*% X) %*% t(X) %*% y #creates coefficents
  mean_vector <- X%*%beta
  deviance <- sum((y - mean_vector)^2) #calculates deviance
  n <- nrow(df) #calculates n of df... to be used as input in linear_sig2
  coeff_df <- linear_sig2(X,beta,n,y)
  return(list(coefficients = coeff_df, deviance = deviance,iterations = 1,family = family))
}


m1 <- myglm_linear2(y ~ x1 + x2,"guassian",data)
m1$coefficients
```

```
##                  beta        se  t_score      p_value
## (Intercept) 1.1182111 0.4321904 2.587311 1.115660e-02
## x1          1.4126586 0.7134387 1.980070 5.052892e-02
## x2          0.9552916 0.1878524 5.085330 1.788912e-06
```

```r
m1$deviance
```

```
## [1] 397.8476
```

```r
m1$iterations
```

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

```r
m1$family
```

```
## [1] "guassian"
```

Linear with real data

```r
library(palmerpenguins)
```

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

```r
library(tidyverse)
```

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

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

## Warning: package 'tibble' was built under R version 4.2.3

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

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

## Warning: package 'purrr' was built under R version 4.2.3

## Warning: package 'dplyr' was built under R version 4.2.3

## Warning: package 'forcats' was built under R version 4.2.3

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

```
data <- penguins |>
  drop_na()


m1 <-  myglm_linear2(body_mass_g ~ flipper_length_mm + bill_length_mm,"guassian",data)
m1$coefficients
```

```
##                          beta         se    t_score      p_value
## (Intercept)       -5836.298732 312.603503 -18.669972 1.341791e-53
## flipper_length_mm    48.889692   2.034204  24.033815 1.737931e-74
## bill_length_mm        4.958601   5.213505   0.951107 3.422461e-01
```

```
m1$deviance
```

```
## [1] 51071963
```

```
m1$family
```

```
## [1] "guassian"
```

```
m1$iterations
```

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

Logistic helper functions

```
#The below function is a helper function used to take initialized logit coefficients, among other input

UpdateCoefficients <- function(X,y,beta,max_iter){
  CoeffVector <- X%*%beta
  ProbVector <- (exp(CoeffVector)/(1+exp(CoeffVector)))
  Past_Deviance <--2*sum(dbinom(y, 1, ProbVector, log=T))
  for(a in 1:max_iter){ #Everything within this loop uses Fisher scoring to iteratively update the mode
  CoeffVector <- X%*%beta
  ProbVector <- (exp(CoeffVector)/(1+exp(CoeffVector)))
  score <- (t(X) %*% (y-ProbVector)) #calc score
  diag_vector <- rep(NA,length(ProbVector))
    for(i in 1:length(ProbVector)){
```

```
        diag_vector[i] <- ProbVector[i]*(1 - ProbVector[i])}
  Diag <- diag(diag_vector) #create diag matrix
  info <- (t(X) %*% Diag %*% X) #calc info
  beta <- (beta + solve(info) %*% score) #update coeffs
  CoeffVector <- X%*%beta #The below code calculates difference in deviances to see if the break condit
  ProbVector <- (exp(CoeffVector)/(1+exp(CoeffVector)))
  Deviance <- -2*sum(dbinom(y, 1, ProbVector, log=T))
  Diff <- Past_Deviance - Deviance
  Past_Deviance <- Deviance
  if(Diff < .001){
    break
  }
  }
  diag_vector <- rep(NA,length(ProbVector)) #The below code creates a data frame containing the model's
    for(i in 1:length(ProbVector)){
        diag_vector[i] <- ProbVector[i]*(1 - ProbVector[i])}
  Diag <- diag(diag_vector) #create diag matrix
  DiagSE <- diag(solve(t(X) %*% Diag %*% X))
  se <- sqrt(DiagSE)
  z_score <- beta/se
  z_score1 <- abs(z_score)
  p_value <- 2*pnorm(z_score1,lower.tail = F)
  coeff_df <- data.frame(beta,se,z_score,p_value)
  return(list(deviance = Deviance,coeff_df = coeff_df,iterations = a))
}
```

Logistic with simulated data

```
set.seed(123)
n <- 200
x <- rnorm(n)
p <- exp(1 + x)/(1 + exp(1 + x))
y <- rbinom(n, 1, p)


data <- data.frame(y,x)

#Below is a helper function for logistic models. It initializes coefficients and then uses the helper f

my_glm_log2 <- function(formula,family,max_iter,df){
    response <- all.vars(formula)[1]
    y <- df[[response]] #145 and 144 extract the values of the formulas response variable
    X <- model.matrix(formula, df) #creates design matrix from formula and df
    BetaNum <- ncol(X)
    beta <- rep(0,BetaNum)
    ybar <- mean(y)
    beta[1] <- log(ybar/(1-ybar)) #Initialize coeffs
    model <- UpdateCoefficients(X,y,beta,max_iter) #Helper function output is stored as model so its ou
    return(list(coefficients = model$coeff_df, family = family, iterations = model$iterations, deviance
}

m1 <- my_glm_log2(y ~ x,"binomial",50,data)
m1$coefficients
```

```
##                 beta        se  z_score      p_value
## (Intercept) 1.189801 0.1839126 6.469385 9.840288e-11
## x           0.922279 0.2143541 4.302595 1.688094e-05
```

```r
m1$family
```

```
## [1] "binomial"
```

```r
m1$iterations
```

```
## [1] 4
```

```r
m1$deviance
```

```
## [1] 208.6589
```

Logistic with real data

```r
data <- penguins |>
  drop_na() |>
  mutate(sex = ifelse(sex == "female", 1, 0))

m1 <- my_glm_log2(sex ~ flipper_length_mm + bill_length_mm,
               "binomial", 10, data)
m1$coefficients
```

```
##                           beta         se    z_score      p_value
## (Intercept)        7.005985900 1.72762038  4.0552809 5.007409e-05
## flipper_length_mm -0.007736914 0.01108136 -0.6981915 4.850574e-01
## bill_length_mm    -0.124374714 0.02952819 -4.2120671 2.530444e-05
```

```r
m1$family
```

```
## [1] "binomial"
```

```r
m1$deviance
```

```
## [1] 419.9377
```

```r
m1$iterations
```

```
## [1] 3
```

Poisson helper functions

```r
#The below function is a helper function used to take initialized poisson coefficients, among other inp

update_pois_coeffs <- function(X,y,beta,max_iter){
  CoeffVector <- X%*%beta
  ProbVector <- (exp(CoeffVector)/(1+exp(CoeffVector)))
  Past_Deviance <- 2*(sum(dpois(y, y, log=T)) - sum(dpois(y, ProbVector, log=T)))
  for(a in 1:max_iter){ #Everything within this loop uses Fisher scoring to iteratively update the mode
  CoeffVector <- X%*%beta
  ProbVector <- exp(CoeffVector) #get means vector
  score <- (t(X) %*% (y-ProbVector)) #calc score
  diag_vector <- rep(NA,length(ProbVector))
    for(i in 1:length(ProbVector)){
       diag_vector[i] <- ProbVector[i]}
  Diag <- diag(diag_vector) #create diag matrix
  info <- (t(X) %*% Diag %*% X) #calc info
  beta <- (beta + solve(info) %*% score)#update coeffs
  CoeffVector <- X%*%beta #The below code calculates difference in deviances to see if the break condit
  ProbVector <- exp(CoeffVector)
  Deviance <- 2*(sum(dpois(y, y, log=T)) - sum(dpois(y, ProbVector, log=T)))
  Diff <- Past_Deviance - Deviance
  Past_Deviance <- Deviance
  if(Diff < .001){
    break
  }
  }
  diag_vector <- rep(NA,length(ProbVector)) #The below code creates a data frame containing the model's
    for(i in 1:length(ProbVector)){
       diag_vector[i] <- ProbVector[i]}
  Diag <- diag(diag_vector) #create diag matrix
  DiagSE <- diag(solve(t(X) %*% Diag %*% X))
  se <- sqrt(DiagSE)
  z_score <- beta/se
  abs_z <- abs(z_score)
  p_value <- 2*pnorm(abs_z,lower.tail = F)
  coeff_df <- data.frame(beta,se,z_score,p_value)
  return(list(deviance = Deviance,coeff_df = coeff_df,iterations = a))
}
```

Poisson with simulated data

```r
set.seed(214)
n <- 300
x1 <- rbinom(n, 1, 0.5)
x2 <- runif(n)
x3 <- runif(n)
y <- rpois(n, exp(0.5 - x1 + x2 - 0.5*x3))

data <- data.frame(y,x1,x2,x3)

#Below is a helper function for poisson models. It initializes coefficients and then uses the helper fu

my_glm_poiss2 <- function(formula,family,max_iter,df){
    X  <- model.matrix(formula, df) #create design matrix from formula and df
```

```
    response <- all.vars(formula)[1]
    y <- df[[response]] #the above code extracts the values of the formula's response variable
    BetaNum1 <- ncol(X)
    beta <- rep(0,BetaNum1)
    ybar <- mean(y)
    beta[1] <- log(ybar) #initializes coefficients
    model <- update_pois_coeffs(X,y,beta,max_iter) #Helper function output is stored as model so its ou
    return(list(coefficients = model$coeff_df, family = family, deviance = model$deviance,iterations = m
}


m1 <- my_glm_poiss2(y ~ x1 + x2 + x3,"poisson",12,data)
m1$coefficients
```

```
##                   beta        se   z_score      p_value
## (Intercept)  0.4096576 0.1288543  3.179232 1.476660e-03
## x1          -0.9375865 0.1000650 -9.369779 7.268426e-21
## x2           1.0644028 0.1686682  6.310630 2.779011e-10
## x3          -0.3281246 0.1686991 -1.945029 5.177144e-02
```

```
m1$family
```

```
## [1] "poisson"
```

```
m1$deviance
```

```
## [1] 358.0896
```

```
m1$iterations
```

```
## [1] 4
```

The my_glm2 function

```
#Finally, I use all of the above helper functions to create my_glm2. The function takes inputs of a for

my_glm2 <- function(formula,family,max_iter=50,df){
  if(family == "guassian"){
    return(myglm_linear2(formula,family,df))
  }
  if(family == "binomial"){
    return(my_glm_log2(formula,family,max_iter=50,df))
  }
  if(family == "poisson"){
    return(my_glm_poiss2(formula,family,max_iter=50,df))
  }

}
```

Final Product + Examples

Linear With Simulated Data

```r
set.seed(279)
n <- 100
x1 <- runif(n)
x2 <- rnorm(n)
y <- rnorm(n, 1 + x1 + x2, 2)

# create the data
data <- data.frame(y, x1, x2)

m1 <- my_glm2(y ~ x1 + x2,"guassian",max_iter=50,data)
m1$coefficients
```

```
##                  beta        se  t_score      p_value
## (Intercept) 1.1182111 0.4321904 2.587311 1.115660e-02
## x1          1.4126586 0.7134387 1.980070 5.052892e-02
## x2          0.9552916 0.1878524 5.085330 1.788912e-06
```

```r
m1$deviance
```

```
## [1] 397.8476
```

```r
m1$iterations
```

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

```r
m1$family
```

```
## [1] "guassian"
```

Linear With Real Data

```r
library(palmerpenguins)
library(tidyverse)

data <- penguins |>
  drop_na()

m1 <- my_glm2(body_mass_g ~ flipper_length_mm + bill_length_mm,"guassian",50,data)
m1$coefficients
```

```
##                          beta         se    t_score      p_value
## (Intercept)       -5836.298732 312.603503 -18.669972 1.341791e-53
## flipper_length_mm    48.889692   2.034204  24.033815 1.737931e-74
## bill_length_mm        4.958601   5.213505   0.951107 3.422461e-01
```

```r
m1$deviance
```

```
## [1] 51071963
```

```r
m1$iterations
```

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

```r
m1$family
```

```
## [1] "guassian"
```

Logistic With Simulated Data

```r
set.seed(123)
n <- 200
x <- rnorm(n)
p <- exp(1 + x)/(1 + exp(1 + x))
y <- rbinom(n, 1, p)

data <- data.frame(y,x)


m1 <- my_glm2(y ~ x,"binomial",50,data)
m1$coefficients
```

```
##                 beta        se  z_score      p_value
## (Intercept) 1.189801 0.1839126 6.469385 9.840288e-11
## x           0.922279 0.2143541 4.302595 1.688094e-05
```

```r
m1$family
```

```
## [1] "binomial"
```

```r
m1$iterations
```

```
## [1] 4
```

```r
m1$deviance
```

```
## [1] 208.6589
```

Logistic With Real Data

```r
data <- penguins |>
  drop_na() |>
  mutate(sex = ifelse(sex == "female", 1, 0))

m1 <- my_glm2(sex ~ flipper_length_mm + bill_length_mm,
               "binomial", 10, data)
m1$coefficients
```

```
##                           beta         se    z_score      p_value
## (Intercept)        7.005985900 1.72762038  4.0552809 5.007409e-05
## flipper_length_mm -0.007736914 0.01108136 -0.6981915 4.850574e-01
## bill_length_mm    -0.124374714 0.02952819 -4.2120671 2.530444e-05
```

```r
m1$family
```

```
## [1] "binomial"
```

```r
m1$deviance
```

```
## [1] 419.9377
```

```r
m1$iterations
```

```
## [1] 3
```

Poisson With Simulated Data

```r
set.seed(214)
n <- 300
x1 <- rbinom(n, 1, 0.5)
x2 <- runif(n)
x3 <- runif(n)
y <- rpois(n, exp(0.5 - x1 + x2 - 0.5*x3))

data <- data.frame(y,x1,x2,x3)

m1 <- my_glm_poiss2(y ~ x1 + x2 + x3,"poisson",16,data)
m1$coefficients
```

```
##                   beta        se   z_score      p_value
## (Intercept)  0.4096576 0.1288543  3.179232 1.476660e-03
## x1          -0.9375865 0.1000650 -9.369779 7.268426e-21
## x2           1.0644028 0.1686682  6.310630 2.779011e-10
## x3          -0.3281246 0.1686991 -1.945029 5.177144e-02
```

```r
m1$family
```

```
## [1] "poisson"
```

```r
m1$deviance
```

```
## [1] 358.0896
```

```r
m1$iterations
```

```
## [1] 4
```