1. For $X \sim f(x;\theta)$, show that

$$E\left\{\frac{\partial}{\partial\theta}\log f(x;\theta)\right\}=0, \quad \text{and} \quad -E\left\{\frac{\partial^2}{\partial\theta^2}\log f(x;\theta)\right\}=E\left[\left\{\frac{\partial}{\partial\theta}\log f(x;\theta)\right\}^2\right].$$

$$\text{(i)}\quad E\left[\frac{\partial}{\partial\theta}\log f(x;\theta)\right]=E\left[\frac{\frac{\partial}{\partial\theta}f(x;\theta)}{f(x;\theta)}\right]=\int\frac{\frac{\partial}{\partial\theta}f(x;\theta)}{f(x;\theta)}f(x;\theta)\,dx=\int\frac{\partial}{\partial\theta}f(x;\theta)\,dx=\frac{\partial}{\partial\theta}\underbrace{\int f(x;\theta)\,dx}_{1}=0$$

$$\text{(ii)}\quad E\left[\frac{\partial^2}{\partial\theta^2}\log f(x;\theta)\right]=E\left[\frac{\partial}{\partial\theta}\left(\frac{\frac{\partial}{\partial\theta}f(x;\theta)}{f(x;\theta)}\right)\right]=E\left[\frac{\frac{\partial^2}{\partial\theta^2}f(x;\theta)-\left(\frac{\partial}{\partial\theta}f(x;\theta)\right)^2}{f^2(x;\theta)}\right]=E\left[\frac{\frac{\partial^2}{\partial\theta^2}f(x;\theta)}{f(x;\theta)}\right]-E\left[\frac{\left(\frac{\partial}{\partial\theta}f(x;\theta)\right)^2}{f^2(x;\theta)}\right]$$

$$E\left[\frac{\frac{\partial^2}{\partial\theta^2}f(x;\theta)}{f(x;\theta)}\right]=\int\frac{d^2}{d\theta^2}f(x;\theta)\,dx=0$$

$$E\left[\frac{\left(\frac{\partial}{\partial\theta}f(x;\theta)\right)^2}{f^2(x;\theta)}\right]=E\left[\frac{\left(\frac{\partial}{\partial\theta}f(x;\theta)\right)}{f(x;\theta)}\right]^2=E\left[\left(\frac{\partial}{\partial\theta}\log f(x;\theta)\right)^2\right]$$

$$\therefore\ -E\left[\frac{\partial^2}{\partial\theta^2}\log f(x;\theta)\right]=E\left[\left(\frac{\partial}{\partial\theta}\log f(x;\theta)\right)^2\right]$$

2. Suppose we have $y_i \overset{\text{ind}}{\sim} Poisson(\mu_i), i=1,\cdots,n$.

   (a) Show that the Poisson distribution belongs to the exponential dispersion family to identify θ_i, $b(\theta_i)$, $a(\phi)$ under the exponential dispersion family form given in Lecture note.

   (b) Derive the canonical link function $g(\mu_i)$ to be modeled via $\eta_i=\mathbf{x}_i\boldsymbol{\beta}, i=1,\cdots,n$ for the Poisson regression.

$$f(y_i)=\frac{e^{-\mu_i}\mu_i^{y_i}}{y_i!}, \quad y_i=0,1,2,\cdots$$

$$\text{(a)}\quad f(y_i)=\frac{e^{-\mu_i}\mu_i^{y_i}}{y_i!}=\exp\left(\log\frac{e^{-\mu_i}\mu_i^{y_i}}{y_i!}\right)=\exp\left(-\mu_i+y_i\log\mu_i-\log y_i!\right)$$

$\log\mu_i=\theta_i$, $b(\theta_i)=e^{\theta_i}=\mu_i$, $a(\phi)=1$, $c(y_i,\phi)=-\log y_i!$

$$=\exp\left(\frac{y_i\theta_i-b(\theta_i)}{a(\phi)}+c(y_i,\phi)\right)$$

$$\text{(b)}\quad g(\mu_i)=\eta_i \quad\leftrightarrow\quad \mu_i=g^{-1}(\eta_i)$$

$$\eta_i=x_i^T\beta=\theta_i,\quad i=1,2,\cdots,n$$

$$=\log\mu_i$$

$$\therefore\ g(\cdot)=\log$$

ST509_HW3_Group 2

3. Write your own code to estimate the parameters β in the Poisson regression based on Newton-Raphson method. Namely, we have the following model for $(y_i, \mathbf{x_i})$, $i = 1, \cdots, n$:

$$y_i \,|\, \mathbf{x_i} \sim \text{Poisson}(\mu_i), \quad i = 1, \cdots, n$$

where

$$g(\mu_i) = \eta_i = \mathbf{x_i}\beta$$

with g being the canonical link function obtained in Problem 2-(b).

in R

```
my_poi <- function(X, y, init = NULL, max_iter = 1000, eps = 1e-5, offset = NULL) {
    # Scaling
    y_sd <- sd(y); X_sd <- apply(X, 2, sd)
    y <- y /y_sd; X <- t(t(X)/ X_sd)

    # Add intercept term
    X <- cbind(1, X) ; n <- nrow(X); p <- ncol(X)

    # INIT
    if (is.null(init)) init <- rep(0, p); beta <- init

    # Offset
    if (is.null(offset)) offset <- rep(0, n)

    # Iteration
    for (iter in 1:max_iter) {
        eta <- X %*% beta
        w <- exp(eta + offset) # Mean = Variance
        z <- eta + (y-w)/w
        # IRLS
        X_tilde <- diag(c(sqrt(w))) %*% X
        z_tilde <- diag(c(sqrt(w))) %*% z
        qr_obj <- qr(X_tilde)
        new_beta <- backsolve(qr_obj$qr, qr.qty(qr_obj, z_tilde))
        # Check Convergence
        if (max(abs(new_beta - beta))/ max(abs(beta)) < eps) break
        beta <- new_beta
    }

    # Warning
    if (iter == max_iter) warning("Algorithm may not have converged!")
    # Restore beta coef
    beta <- c(beta) * c(1, 1/X_sd) + c(log(y_sd), rep(0, p-1))
    # Result
    list(est = t(beta), iterations = iter)
}
```

in Python

```python
import numpy as np
import pandas as pd
import statsmodels.api as sm
from statsmodels.genmod.families import Poisson
from scipy.linalg import solve_triangular


def my_poi(X, y, init=None, max_iter=1000, eps=1e-5, offset=None):

    # Scaling
    y_sd = np.std(y); X_sd = np.std(X, axis=0)
    y = y / y_sd; X = X / X_sd[np.newaxis, :]

    # Add intercept term
    X = np.column_stack((np.ones(len(X)), X))
    n, p = X.shape

    # INIT
    if init is None:
        init = np.zeros(p)
    beta = init

    # Offset
    if offset is None:
        offset = np.zeros(n)

    # Iteration
    for iter in range(max_iter):
        eta = X @ beta
        w = np.exp(eta + offset)
        z = eta + (y - w)/w
        # IRLS
        X_tilde = np.sqrt(w)[:, np.newaxis] * X
        z_tilde = np.sqrt(w) * z
        Q, R = np.linalg.qr(X_tilde)
        new_beta = solve_triangular(R, Q.T @ z_tilde, lower=False)
        # Check convergence
        if np.max(np.abs(new_beta - beta)) / np.max(np.abs(beta)) < eps :
            break
        beta = new_beta

    # Warning
    if iter == max_iter - 1:
        print("Algorithm may not have converged!")
    # Restore beta coefficients
    beta = beta * np.concatenate(([1], 1/X_sd)) + np.concatenate(([np.log(y_sd)], np.zeros(p-1)))

    return beta, iter
```

I. Scaling

exp term으로 인해 무한대로 발산하는 경우를 방지하기 위해, Data에 스케일링을 진행하여 회귀 계수를 예측하고 이를 복원하는 과정을 거친다.

$$\log\left(E\left[\frac{\mathbf{Y}}{\sigma_Y}\right]\right) = \left[1 \mid \tfrac{X_1}{\sigma_1} \mid \cdots \mid \tfrac{X_p}{\sigma_p}\right] \begin{bmatrix}\beta_0 \\ \beta_1 \\ \vdots \\ \beta_p\end{bmatrix}$$

which means

$$\log\begin{bmatrix}E(y_1/\sigma_Y) \\ E(y_2/\sigma_Y) \\ \vdots \\ E(y_n/\sigma_Y)\end{bmatrix} = \left[1 \mid \tfrac{X_1}{\sigma_1} \mid \cdots \mid \tfrac{X_p}{\sigma_p}\right] \begin{bmatrix}\beta_0 \\ \beta_1 \\ \vdots \\ \beta_p\end{bmatrix}$$

equal to

$$\log\left(E\left[\mathbf{Y}\right]\right) = \left[1 \mid X_1 \mid \cdots \mid X_p\right] \begin{bmatrix}\beta_0^* \\ \beta_1^* \\ \vdots \\ \beta_p^*\end{bmatrix} + \log\left(\sigma_Y\right)$$

II. Offset

A response variable Y has an index t_i s.t. its expected value is proportional to t_i, e.g. *an amount of time or spatial area* over which the count is made.

Sample rate $= y_i/t_i$ and $E\left(Y_i/t_i\right) = \mu_i/t_i$

Poisson regression model for rate date will be

$$\log(\mu_i/t_i) = \log(\mu_i) - \log(t_i) = \beta_0 + \beta_1 x_{1i} + \cdots + \beta_p x_{pi}$$

$$\log(\mu_i) = \beta_0 + \beta_1 x_{1i} + \cdots + \beta_p x_{pi} + \beta_{p+1} x_{(p+1)i}$$

where $\beta_{p+1} = 1$ and $x_{(p+1)i} = \log(t_i)$: offset

예시) 병원에서 일정 기간 동안 발생한 질병의 건수를 예측하는 모델을 만든다고 가정. 이때 각 병원의 환자 수는 예측에 중요한 요소가 된다. 만약 한 병원에서 한 달 동안 질병이 10건 발생했고, 다른 병원에서는 같은 기간 동안 20건 발생했다면, 이 차이는 단순히 두 번째 병원에서 질병이 더 많이 발생하기 때문일 수도 있지만, 환자 수가 더 많아서일 수도 있다. 이 경우, 환자 수를 오프셋으로 사용하여 모델을 조정할 수 있다. 즉, 모델이 각 병원의 환자 수에 따라 조정되어, 환자 한 명당 질병 발생 건수를 더 잘 예측할 수 있게 된다.

4. Apply your code and report your parameter estimates of the Poisson regression to "Smoking and Lung Cancer data".

해당 문제는 일반적인 포아송 회귀를 적용하여 사망자 수를 예측할 수도 있지만, offset을 이용해서 인구수를 조정하여 사망 비율을 예측할 수도 있다. 또한, 범주형 변수는 홈페이지의 설명대로 label encoding을 진행하여 적합하였다.

in R

```
setwd("C:/Users/jhmok/OneDrive/바탕 화면/학교/2024-1/통계계산방법론/과제/3")
dat <- read.table("smoking.dat")
dat$age <- as.numeric(factor(dat$age))
dat$smoke <- as.numeric(factor(dat$smoke, levels=c("no", "cigarPipeOnly", "cigarrettePlus", "cigarrette

X <- as.matrix(dat[,-4]); y <- as.matrix(dat[,4])
X_offset <- as.matrix(dat[,1:2]); pop <-as.matrix(dat[,3])
```

```
# Model (w/o offset)
my_poi(X, y)
```

```
## $est
##                     age     smoke          pop
## [1,] 2.425885 0.2618248 0.2689611 0.0004433254
##
## $iterations
## [1] 6
```

```
glm(dead ~ age + smoke + pop, data = dat, family = "poisson")
```

```
##
## Call:  glm(formula = dead ~ age + smoke + pop, family = "poisson", data = dat)
##
## Coefficients:
## (Intercept)          age        smoke          pop
##   2.4258846    0.2618248    0.2689611    0.0004433
##
## Degrees of Freedom: 35 Total (i.e. Null);  32 Residual
## Null Deviance:      8434
## Residual Deviance: 1304   AIC: 1552
```

```r
# Model (with offset)
my_poi(X_offset,y, offset=log(pop))
```

```
## $est
##                     age     smoke
## [1,] -3.967813 0.331192 0.1638844
##
## $iterations
## [1] 12
```

```r
glm(dead ~ age + smoke, data=dat, family="poisson", offset = log(pop))
```

```
##
## Call:  glm(formula = dead ~ age + smoke, family = "poisson", data = dat,
##     offset = log(pop))
##
## Coefficients:
## (Intercept)          age        smoke
##     -3.9679       0.3312       0.1639
##
## Degrees of Freedom: 35 Total (i.e. Null);  33 Residual
## Null Deviance:       4056
## Residual Deviance: 85.97     AIC: 332
```

in Python

```
file_path = "C:/Users/jhmok/OneDrive/바탕 화면/학교/2024-1/통계계산방법론/과제/3/smoking.dat"
dat = pd.read_csv(file_path, sep='\s+')

dat['age'] = pd.Categorical(dat['age']).codes + 1
dat['smoke'] = pd.Categorical(dat['smoke'], categories=["no", "cigarPipeOnly", "cigarrettePlus", "cigar

X = dat.iloc[:, 0:3].values; y = dat.iloc[:, 3].values
X_offset = dat.iloc[:, 0:2]; pop = dat.iloc[:, 2].values
```

```
# Model (w/o offset)
beta_no_offset, iter_no_offset = my_poi(X, y)
coef_formatted = [f"{coef:.4f}" for coef in beta_no_offset]
print("coef :", coef_formatted, "iter :", iter_no_offset+1)
```

```
## coef : ['2.4259', '0.2618', '0.2690', '0.0004'] iter : 6
```

```
glm_no_offset = sm.GLM(y, sm.add_constant(X), family=Poisson(), offset=None).fit()
print(glm_no_offset.summary())
```

```
##                  Generalized Linear Model Regression Results
## ==============================================================================
## Dep. Variable:                      y   No. Observations:                   36
## Model:                            GLM   Df Residuals:                       32
## Model Family:                 Poisson   Df Model:                            3
## Link Function:                    Log   Scale:                          1.0000
## Method:                          IRLS   Log-Likelihood:                -771.89
## Date:                  월, 01 4 2024   Deviance:                       1303.8
## Time:                        00:33:01   Pearson chi2:                 1.12e+03
## No. Iterations:                     5   Pseudo R-squ. (CS):              1.000
## Covariance Type:            nonrobust
## ==============================================================================
##                  coef    std err          z      P>|z|      [0.025      0.975]
## ------------------------------------------------------------------------------
## const          2.4259      0.057     42.777      0.000       2.315       2.537
## x1             0.2618      0.006     45.608      0.000       0.251       0.273
## x2             0.2690      0.012     22.191      0.000       0.245       0.293
## x3             0.0004   6.29e-06     70.484      0.000       0.000       0.000
## ==============================================================================
```

```python
# Model (with offset)
beta_offset, iter_offset = my_poi(X_offset, y, offset = np.log(pop))
coef_formatted2 = [f"{coef:.4f}" for coef in beta_offset]
print("coef :", coef_formatted2, "iter :", iter_no_offset+1)
```

```
## coef : ['-3.9678', '0.3312', '0.1639'] iter : 6
```

```python
glm_offset = sm.GLM(y, sm.add_constant(X_offset), family=Poisson(), offset=np.log(pop)).fit()
print(glm_offset.summary())
```

```
##                  Generalized Linear Model Regression Results
## ==============================================================================
## Dep. Variable:                      y   No. Observations:                   36
## Model:                            GLM   Df Residuals:                       33
## Model Family:                 Poisson   Df Model:                            2
## Link Function:                    Log   Scale:                          1.0000
## Method:                          IRLS   Log-Likelihood:                -163.00
## Date:                월, 01 4 2024   Deviance:                       85.970
## Time:                        00:33:01   Pearson chi2:                     82.9
## No. Iterations:                     6   Pseudo R-squ. (CS):              1.000
## Covariance Type:            nonrobust
## ==============================================================================
##                  coef    std err          z      P>|z|      [0.025      0.975]
## ------------------------------------------------------------------------------
## const         -3.9679      0.055    -71.593      0.000      -4.077      -3.859
## age            0.3312      0.005     60.423      0.000       0.320       0.342
## smoke          0.1639      0.012     13.398      0.000       0.140       0.188
## ==============================================================================
```