HW2 Stat-comp (Solution)

1) Maximum likelihood estimation and inference with the exponential distribution

The density function of an exponential random variable is

$f(x_i|\lambda) = \lambda e^{-\lambda x_i}$

where $x_i \geq 0$ is the random variable, and $\lambda > 0$ is a rate parameter.

The expected value and variance of the random variables are $E[X] = \frac{1}{\lambda}$ and $Var[X] = \frac{1}{\lambda^2}$.

The following code simulates 50 IID draws from an exponential distribution

```
set.seed(195021)
x=rexp(n=50,rate=2)
```

The maximum likelihood estimate of λ has a closed form, indeed

$L(\lambda|x) = \lambda^n e^{-\lambda n\bar{x}}$

Thus, $l(\lambda|x) = nlog(\lambda) - \lambda n\bar{x}$, therefore

$\frac{dl}{d\lambda} = \frac{n}{\lambda} - n\bar{x}$. Setting this derivative equal to zero, and solving for $\hat{\lambda}$ gives $\hat{\lambda} = \frac{1}{\bar{x}}$

1.1) Use `optimize()` to estimate λ compare your estimate with $\frac{1}{\bar{x}}$.

```
negLogLik=function(y,lambda){
    n=length(x)
    xBar=mean(x)
    logLik=n*log(lambda)-lambda*xBar*n
    return(-logLik)
}
fm=optimize(f=negLogLik,y=x,interval=c(0,100))

fm$minimum
```

```
## [1] 3.247199
```

```
1/mean(x)
```

```
## [1] 3.2472
```

```
fm$objective
```

```
## [1] -8.889658
```

```
negLogLik(y=x,lambda=fm$minimum)
```

```
## [1] -8.889658
```

1.2) Use numerical methods to provide an approximate 95% CI for your estimate.

Hint: `optimize()` does not provide a Hessian. However, you can use the `hessian()` function of the `numDeriv` R-package to obtain a numerical approximation to the second order derivative of the logLikelihood at the ML estiamte. To install this package you can use

```
#install.packages(pkg='numDeriv',repos='https://cran.r-project.org/')
library(numDeriv)
H=hessian(func=negLogLik,y=x,x=fm$minimum)
H
```

```
##          [,1]
## [1,] 4.741898
```

```
VAR=1/H # since H is scalar, we just use 1/H
SE=sqrt(VAR)

SE
```

```
##           [,1]
## [1,] 0.4592233
```

```
CI=fm$minimum+c(-1,1)*as.vector(1.96*SE)
round(CI,3)
```

```
## [1] 2.347 4.147
```

2) CIs for Predictions from Logistic Regression

Recall that in a logistic regresion model, the log-odds are parameterized as

$$log[\frac{\theta_i}{(1-\theta_i)}] = \mathbf{x}_i'\beta = \eta_i \tag{1}$$

The sampling variance of $\mathbf{x}_i'\beta = \eta_i$ is $Var(\eta_i) = \mathbf{x}_i'\mathbf{V}\mathbf{x}_i$, where $\mathbf{V}$ is the (co)variance matrix of the estimated effects; therefore, a SE and an approximate 95%CI for η_i can be obtained using

$SE(\eta_i) = \sqrt{\mathbf{x}_i'\mathbf{V}\mathbf{x}_i}$ and $CI : \mathbf{x}_i'\hat{\beta} + / - 1.96 \times SE(\eta_i)$.

Because the inverse-logit is a monotonic map, we can then obtain a 95% CI for the predicted probabilities by applying the inverse logit,$\theta_i = \frac{e^{\eta_i}}{1+e^{\eta_i}}$, to the bounds of the CI for the linear predictor.

- Using the gout data set, fit a logistic regression for gout using sex, age, and race as predictors (for this you can use `glm()`, don't forget the link!).
- From the fitted model, and using the formulas presented above, report predictions and 95% CIs in the scale of the linear predictor and in the probability scale.

Race	Sex	Age	Predicted Risk	95%CI
White	Male	55		
White	Female	55		
Black	Male	55		
Black	Female	55		

```
DATA=read.table('https://raw.githubusercontent.com/gdlc/STAT_COMP/master/DATA/goutData.txt',header=TRUE)
DATA$y=ifelse(is.na(DATA$gout),NA,ifelse(DATA$gout=='Y',1,0))
table(DATA$y,DATA$gout)
```

```
##
##       N   Y
##   0 370   0
##   1   0  30
```

```
fm=glm(y~race+sex+age,data=DATA,family=binomial)
```

Once we fitted the model, we:

- create the incidence matrix (X) for the cases we want to predict,
- evaluate the linear predictor (eta=Xb), and its SE,
- use the previous results to get a CI for eta
- map it, using the inverse-logit, into a CI for the predicted probability

```
## Incidence matrix
X=cbind('int'=1,'raceW'=c(1,1,0,0),'sexM'=c(1,0,1,0),'age'=55)
X
```

```
##      int raceW sexM age
## [1,]   1     1    1  55
## [2,]   1     1    0  55
## [3,]   1     0    1  55
## [4,]   1     0    0  55
```

```
eta=X%*%coef(fm)
VCOV.ETA=X%*%vcov(fm)%*%t(X)
SE.ETA=sqrt(diag(VCOV.ETA))

invLogit=function(eta){
    exp(eta)/(1+exp(eta))
}

LOW=invLogit(eta-1.96*SE.ETA)
UP=invLogit(eta+1.96*SE.ETA)

ANS=data.frame('race'=c('W','W','B','B'),'sex'=c('M','F','M','F'),'age'=55,'LP'=eta,'LB-LP'=eta-1.96*SE

ANS
```

```
##   race sex age        LP     LB.LP     UB.LP       prob     LB.prob    UB.prob
## 1    W   M  55 -3.296722 -4.230460 -2.362984 0.03568382 0.014337159 0.08603923
## 2    W   F  55 -3.745870 -4.705597 -2.786143 0.02307028 0.008963444 0.05807762
## 3    B   M  55 -2.553434 -3.572308 -1.534560 0.07219612 0.027323395 0.17732751
## 4    B   F  55 -3.002582 -3.975647 -2.029517 0.04730937 0.018421435 0.11613854
```

You can also obtain predictions in the probability scale using the `predict()` function, specifying `type=response` and `se.fit=TRUE`. You can then build a 95% CI using prediction +/1 1.96*SE. This method can give you predictions outside tye [0,1] interval which make no sense. You can then set the lower bound to zero (if the original lower bound was <0) and the upper bound to 1 (if the original upper bound was >1).

```
newData=data.frame('race'=c('W','W','B','B'),'sex'=c('M','F','M','F'),'age'=55)
TMP=predict(fm,type='response',se.fit=TRUE,newdata = newData)
ANS2=cbind(newData,'pred.prob'=TMP$fit,'SE'=TMP$se.fit,'LB'=TMP$fit-1.96*TMP$se.fit,'UB'=TMP$fit+1.96*

# Fixing possible bounds below 0 or above 1
ANS2$LB=ifelse(ANS2$LB<0,0,ANS2$LB)
ANS2$UB=ifelse(ANS2$UB>1,1,ANS2$UB)

ANS2
```

```
##   race sex age  pred.prob         SE          LB         UB
## 1    W   M  55 0.03568382 0.01639304 0.003553453 0.06781418
```

```
## 2    W   F  55 0.02307028 0.01103590 0.001439908 0.04470064
## 3    B   M  55 0.07219612 0.03482046 0.003948008 0.14044423
## 4    B   F  55 0.04730937 0.02237613 0.003452156 0.09116659
```