# HW2 Stat-comp (due Sunday Oct. 26th 8pm in D2L)

## 1) Predicting confidence bands for logistic regression using Bootsrap

Include here your code and results from INCLASS-10

**Fitting the model to the data set and extracting coefficients**:

```
DATA=read.table('https://raw.githubusercontent.com/gdlc/STAT_COMP/master/goutData.txt',
                   header=TRUE)
DATA$y=ifelse(DATA$gout=="Y",1,0)
fm=glm(y~su,data=DATA,family='binomial')
bHat=coef(fm)
```

**Predictions**:

```
su.grid=seq(from=4,to=10,by=.1)

phat=predict(fm,newdata=data.frame(su=su.grid),type='response')
```

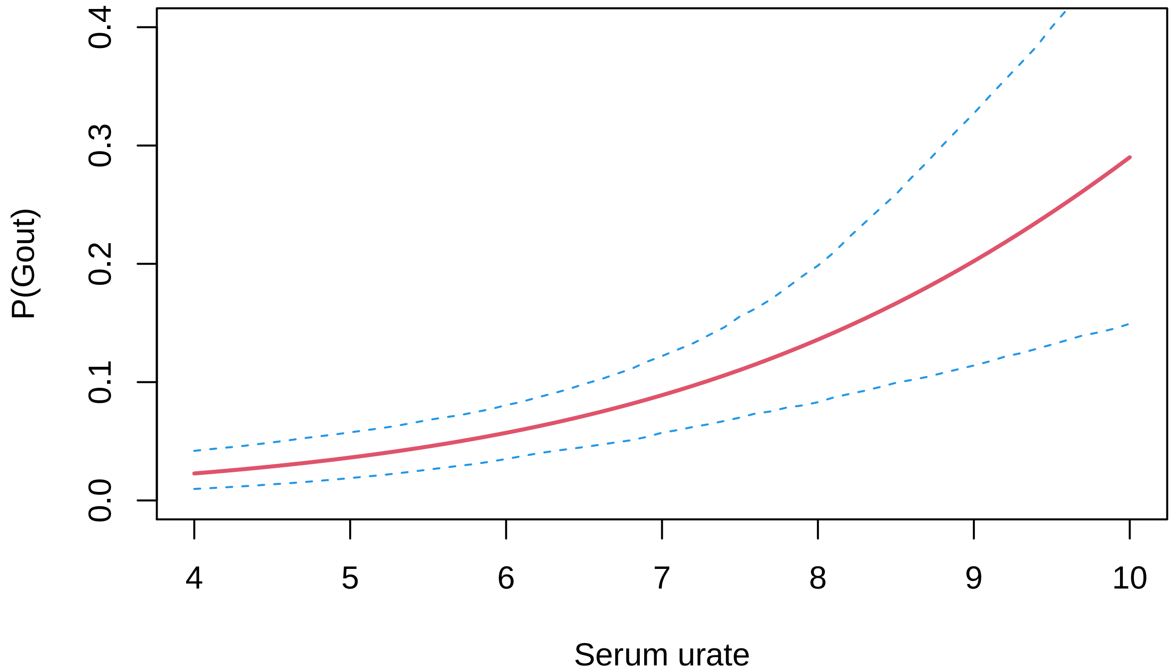
**Bootstrap**:

```
n=nrow(DATA)
B=1000
PHAT=matrix(nrow=length(phat),ncol=B,NA)
for(i in 1:B){
  tmp=sample(1:n,size=n,replace=TRUE)
  boostrapSample=DATA[tmp,]
  fm=glm(y~su,data=boostrapSample,family='binomial')
  PHAT[,i]=predict(fm,newdata=data.frame(su=su.grid),type='response')
}

BANDS=apply(FUN=quantile,prob=c(.025,.975),X=PHAT,MARGIN=1)
```

**Plot**

```
# Plot
 plot(phat~su.grid,col=2,xlab='Serum urate',ylab='P(Gout)',type='l',ylim=c(0,.4),lwd=2)
 lines(x=su.grid,y=BANDS[1,],lty=2,col=4)
 lines(x=su.grid,y=BANDS[2,],lty=2,col=4)
```




## 2)

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 $\lambda$ 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

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

**Using numerical optimization to estimate** $\lambda$**:**

Since $\lambda > 0$, we need to be careful using `optim()` because this function may report an estimate smaller than zero. Furthermore, for models involving a single parameter, `optimize()` is preferred relative to `optim()`; `optimize()` allows you to provide an interval for the optimization.

**2.1**) Use `optimize()` to estimate $\lambda$ compare your estimate with $\frac{1}{\bar{x}}$.

```
negLogLik=function(x,lambda){
  log_lik=sum(dexp(rate = lambda,x=x,log=TRUE))
  return(- log_lik)
}
fm=optimize(f=negLogLik,x=x,interval=c(0,100))
MLE=fm$minimum
round(c('optimize'=MLE,'1/mean'=1/mean(x)),4)
```

```
## optimize   1/mean
```

```
##   3.2472   3.2472
```

**2.2** Use numerical methods to proivde 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)

# I need to rename the data because hessian( ) has an argument called xc
negLogLik=function(z,lambda){
 log_lik=sum(dexp(rate = lambda,x=z,log=TRUE))
 return(- log_lik)
}

H=hessian(f=negLogLik,z=x,x=fm$minimum)
VAR=1/H
SE=sqrt(VAR)
print(SE)
```

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

```
CI_1=c('Low'=MLE-1.96*SE,'Up'=MLE+1.96*SE)
print(round(CI_1,5))
```

```
##     Low      Up
## 2.34712 4.14728
```

## 3) Bootstrap

**3.1)** Use 10,000 bootstrap samples to estimate the SE. Compare your results with those reported in the previous question.

```
B=10000
estimates=rep(NA,B)
n=length(x)
for(i in 1:B){
  tmp=sample(1:n,size=n,replace=TRUE)
  boostrapSample=x[tmp]
  fm=optimize(f=negLogLik,z=boostrapSample,interval=c(0,100))
  estimates[i]=fm$minimum
}
BOOTSTRAP_SE=sd(estimates)
round(BOOTSTRAP_SE,4)
```

```
## [1] 0.4927
```

```
CI_2=c('Low'=MLE-1.96*BOOTSTRAP_SE,'Up'=MLE+1.96*BOOTSTRAP_SE)
```

The Bootstrap SE ( 0.4927) is higher than the asymptotic SE (0.4592)

**3.2)** Report 95% CI assuming normality using the SE from question 2 and the SE from 3.1, and compare these CIs with those obtained with the percentile method (i.e., applying `quantile(x=bootstrap_estimates,prob=c(.025,.975))` to the bootstrap samples).

```r
CI_3=quantile(estimates,prob=c(.025,.975))
round(rbind('Asym'=CI_1,'Asym(w/bootstrap SE'=CI_2,'Bootstrap(percentile)'=CI_3),4)
```

```
##                          Low     Up
## Asym                  2.3471 4.1473
## Asym(w/bootstrap SE   2.2815 4.2129
## Bootstrap(percentile) 2.5049 4.4435
```

**3.3)** Compare the estimate obtained with the sample, with the average Bootstrap estimate. Do we have any evidence that the estimator may be biased?

The ML estimate was 3.2472; the average bootrstrap estiamte was 3.3146. It seems that the estimator is upwardly biased.