Question 1

(a)

The likelihood function is equivalent to the joint density.

$$\begin{aligned}
\mathcal{L}(\lambda|\boldsymbol{x}) &= p_{\mathbf{X}}(\boldsymbol{x}|\lambda) \\
&= \prod_{i=1}^{n} p_{X_i}(x_i|\lambda) \quad (\because \ i.i.d.) \\
&= \prod_{i=1}^{n} \frac{\lambda^{x_i} e^{-\lambda}}{x_i!} \\
&= e^{-n\lambda} \prod_{i=1}^{n} \frac{\lambda^{x_i}}{x_i!} \\
&= e^{-n\lambda} \lambda^{\sum_{i=1}^{n} x_i} \prod_{i=1}^{n} \frac{1}{x_i!}.
\end{aligned}$$

Then the log likelihood is

$$\ell(\lambda|\boldsymbol{x}) = -n\lambda + \log(\lambda) \sum_{i=1}^{n} x_i - \sum_{i=1}^{n} \log(x_i!).$$

(b)

Since log is a monotonic transformation, $\arg\max_\lambda \mathcal{L}(\lambda|\boldsymbol{x}) = \arg\max_\lambda \ell(\lambda|\boldsymbol{x}) = \hat{\lambda}$ (i.e. MLE). So we take the first order conditions of the log likelihood function.

$$\left.\frac{\partial \ell(\lambda|\boldsymbol{x})}{\partial \lambda}\right|_{\hat{\lambda}} = -n + \frac{\sum_{i=1}^{n} x_i}{\hat{\lambda}} = 0$$

$$\implies \hat{\lambda} = \frac{\sum_{i=1}^{n} x_i}{n}.$$

(c)

By the Central Limit Theorem, we can say that $\hat{\lambda}$ is normally distributed for large samples even if $\mathcal{L}(\lambda|\boldsymbol{x})$ has some other distribution. That is, for infinitely large samples,

$$\hat{\lambda} \sim \mathcal{N}(\lambda, \mathcal{I}(\lambda)^{-1}),$$

where $\mathcal{I}(\lambda) = -E\left[\frac{\partial^2 \ell(\lambda|\boldsymbol{X})}{\partial \lambda^2}\right]$ is the Fisher information. Equivalently, to prevent the distribution from collapsing to a point we can inflate the variance and express as

$$\sqrt{n}(\hat{\lambda} - \lambda) \sim \mathcal{N}(0, n\mathcal{I}(\lambda)^{-1}).$$

(d)

We first derive the asymptotic standard deviation of $\hat{\lambda}$. By Central Limit Theorem, if we assume large sample,

$$\begin{aligned} Var(\hat{\lambda}) = \frac{1}{\mathcal{I}(\lambda)} &\approx \frac{1}{\widehat{\mathcal{I}}(\lambda)} = \frac{\lambda^2}{\sum_{i=1}^n x_i} \\ &\approx \frac{1}{\widehat{\mathcal{I}}(\hat{\lambda})} = \Big(\frac{\sum_{i=1}^n x_i}{n}\Big)^2 \frac{1}{\sum_{i=1}^n x_i} = \frac{\sum_{i=1}^n x_i}{n^2} \\ \implies sd(\hat{\lambda}) &\approx \frac{\sqrt{\sum_{i=1}^n x_i}}{n}. \end{aligned}$$

The fraction of estimates that fall within a two standard deviation interval is 0.946. See `R` code in Appendix.

Question 2

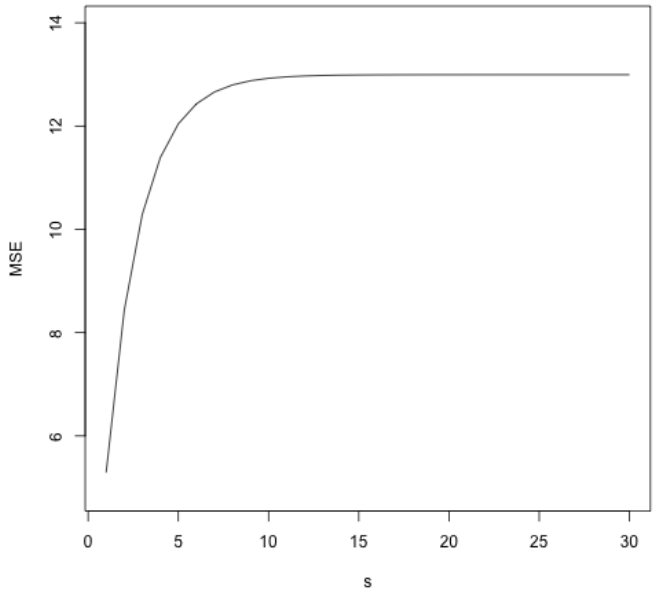
(a)

For $AR(1)$, the formula for the Mean Squared Error of an s-step ahead forecast is

$$MSE = \sigma^2 \sum_{j=0}^{s-1} \phi^{2j}.$$

Figure 1 plots the MSE for $s = 1, 2, \ldots, 30$. See Appendix for `R` code.

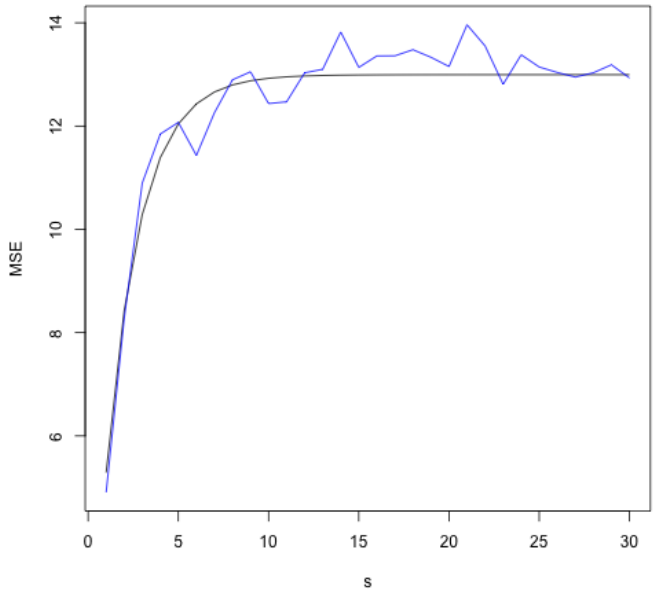
Figure 1: Theoretical MSE of $AR(1)$



(b)

Figure 2 plots the empirical MSE along with the theoretical MSE. The black line indicate theoretical MSE, and the blue line empirical MSE. See Appendix for `R` code.

Figure 2: Theoretical and Empirical MSE of $AR(1)$



Appendix: R Code

```
##################################### Question 1 (d) #####################################
# Set parameters
nObs = 10
nSim = 1000
lambda = 4
X = matrix(0,nObs,nSim)

# Simulate
for(j in 1:nSim){
  X[,j] = rpois(nObs,lambda)
}

# Compute MLE
MLE = apply(X,2,mean)

# Compute two standard deviation
twoStdDev = 2*sqrt(apply(X,2,sum))/nObs

# Compute fraction of estimates within two s.d.
comp_sim = 1*(( MLE < (lambda + twoStdDev) ) & (MLE > (lambda - twoStdDev)))
mean(comp_sim)

##################################### Question 2 (a) #####################################
# Set parameters
sigma = 2.3
phi = 0.77
S = 30

# Compute MSE by s
MSE = (cumsum(c(1,phi^(2*(1:S))))*(sigma^2))[1:S]

# Plot
png(filename="211c_mt_2a.png", height = 500 , width = 500)
plot((1:S),MSE,type="l",xlab="s", ylim = c(yMin,yMax)) #(yMin,yMax later added)
dev.off()

##################################### Question 2 (b) #####################################
# Set parameters
nObs = 130
nSim = 1000
phi = 0.77
mu = 0
sigma = 2.3

# Simulate AR(1)
eps = matrix(0,nObs,nSim)
Y = matrix(0,nObs,nSim)

for(i in 1:nObs){
  for(j in 1:nSim){
    eps[i,j] = rnorm(1,mu,sigma)
  }
```

```
}
for(j in 1:nSim){
  Y[1,j] = eps[1,j]
  for(i in 2:nObs){
    Y[i,j] = phi * Y[(i-1),j] + eps[i,j]
  }
}

# Forecast
S = 30
YHat = matrix(0,S,nSim)

for(j in 1:nSim){
  YHat[,j] =  phi^(1:S)*(Y[100,j])
}

# Compute squared errors
sqrerr = matrix(0,S,nSim)
for(j in 1:nSim){
  sqrerr[,j] = (Y[(nObs-S+1):nObs,j] - YHat[,j])^2
}

# Compute average of squared errors by s
MSE_sim = apply(sqrerr,1,mean)

# Plot
yMax = max(MSE_sim,MSE)
yMin = min(MSE_sim,MSE)
png(filename="211c_mt_2b.png", height = 500 , width = 500)
plot((1:S),MSE,type="l",xlab="s", ylim = c(yMin,yMax))
lines(MSE_sim, col="blue")
dev.off()
```