Q1. (20%) The probability density P(X=k) of a random variable X ~ Poisson-Tweedie family (α, β, γ) can be calculated through a double recursion algorithm.

The first recursion exists in p_k:

$$p_0 = \begin{cases} e^{\beta[(1-\gamma)^{\alpha}-1]/\alpha}, \alpha \neq 0 \\ (1-\gamma)^{\beta}, \alpha = 0 \end{cases}$$

$$p_1 = \beta\gamma * p_0$$

$$p_{k+1} = \frac{1}{k+1}(\beta\gamma * p_k + \sum_{j=1}^{k} j * r_{k+1-j}p_j), k = 1, 2...$$

The second recursion exists in r_j:

$$r_1 = (1-\alpha)\gamma$$

$$r_{j+1} = (\frac{j-1+\alpha}{j+1})\gamma * r_j, j = 1, 2...$$

$$where\ \alpha \in (\infty, 1], \beta \in (0, +\infty], \gamma \in [0, 1]$$

Write a function PTF that has four inputs (k, a, b, g) (a for α, b for β, and g for γ) and returns pk.
Use the function to calculate PTF(9, -3, 2, 0.5) (The answer should be close to 0.04235).

```
PTF=function(k,a,b,g){
p.list=PTF.initList(a,b,g)
r.list=c(PTF.r0(a,g))
for(i in 2:k){
j.list=c(1:(i-1))
pn=(b*g*p.list[i-1]+sum(rev(r.list)*p.list*j.list))/i
# add p to p.list
p.list[i]=pn
# add r to r.list
rn=(i-2+a)/i*g*r.list[i-1]
r.list[i]=rn
}
tail(p.list,n=1)
}

PTF.initList=function(a,b,g){
p0=PTF.p0(a,b,g)
p1=b*g*p0
c(p1)
}


PTF.p0=function(a,b,g){
p0=0
if(a==0){
p0=(1-g)^b
}
else{
p0=exp(b*((1-g)^a-1)/a)
}
p0
}
PTF.r0=function(a,g){
(1-a)*g
```

```
}
PTF(9,-3,2,.5)#0.04235393
```

Q2. (20%) MLE and Simulation

(a) For the Bernoulli distribution $P(X = x_i|p) = p^{x_i}(1-p)^{1-x_i}$,derive$\hat{p}_{MLE}$

$$loglik(p) = log(p^{\Sigma_{i=1}^n}(1-p)^{n-\Sigma_{i=1}^n x_i})$$
$$= \Sigma_{i=1}^n x_i logp + (n - \Sigma_{i=1}^n)log(1-p)$$

找極值,loglik(p)一階微分等於0

$$\frac{dloglik(p)}{dp} = \frac{1}{p}\Sigma_{i=1}^n x_i - \frac{1}{1-p}(n - \Sigma_{i=1}^n) = 0$$
$$\hat{p}_{MLE} = \Sigma_{i=1}^n x_i/n$$

(b) Continuing (a), given p=0.5, simulate n=50, n=5000, n=500,000 Bernoulli random numbers. For each simulated sample of size n, calculate pˆMLE from the sample and compare pˆMLE to the TRUE p=0.5, what have you observed?

```
p=0.5
n.list=c(50,5000,500000)
p.mle=c()
for(i in 1:length(g)){
        n=n.list[i]
        mle=sum(rbinom(n,1,0.5))/n
        p.mle[i]=mle
}
p.mle-p
```

抽樣的次數越多,$\hat{p}_{MLE}$ 與p的誤差越小,也就是說樣本數越多,MLE會越接近理論值。

(c) For the exponential distribution $f(x_i) = \lambda e^{-\lambda x_i}$,derive$\hat{\lambda}_{MLE}$ and prove the Markov Property:

$$loglik(\lambda) = log(\Pi_1^n \lambda e^{-\lambda x_i})$$
$$= n * log\lambda + (-\lambda \sum_1^n x_i)$$

找極值,loglik(λ)一階微分等於0

$$\frac{dloglik(\lambda)}{d\lambda} = n/\lambda - \sum_1^n x_i = 0$$
$$n/\lambda = \sum_1^n x_i$$
$$\hat{\lambda}_{MLE} = \frac{n}{\sum_1^n x_i}$$

(d) Continuing (c), given λ=0.5, simulate n=50, n=5000, n=500,000 exponential random numbers. For each simulated sample of size n, calculate $\hat{\lambda}_{MLE}$ from the sample and compare $\hat{\lambda}_{MLE}$ to the TRUE λ=0.5, what have you observed?

```
p=0.5
n.list=c(50,5000,500000)
p.mle=c()
for(i in 1:length(g)){
        n=n.list[i]
        mle=sum(rbinom(n,1,0.5))/n
        p.mle[i]=mle
}
p.mle-p
```

抽樣的次數越多,$\hat{\lambda}_{MLE}$ 與λ的誤差越小,也就是說樣本數越多,MLE會越接近理論值。

Q3. (20%) Binomial and Poisson Distributions

(a) The table below records the historical number of car accidents/week in a district. Create a vector called car.accident that stores 109 zeros, 65 ones, 22 twos, 3 threes, and 1 four.

```
car.accident=c(rep(0,109),rep(1,65),rep(2,22),rep(3,3),rep(4,1))
```

(b) Apply the fitdistr( ) function in R (load the MASS library first) to fit the car.accident data with the Poisson distribution.

What is the value of estimated λ? What is the log-likelihood?

```
fitdistr(car.accident,"Poisson")# lambda=0.61000000
fitdistr(car.accident,"Poisson")$loglik# log-likelihood=-206.1067
```

(c) Given the estimated λ, use R to do the computation and finish the 3rd column of table below.Are the predicted frequencies close to the actual frequency?

```
car.lambda=0.61000000
for(i in 0:4){
p=dpois(i,car.lambda)
frequency.ideal=200*p
print(paste0("200*P(X=",i,"| λ)=",frequency.ideal))
}
p.lt4=1-ppois(4,car.lambda)
frequency.lt4=200*p.lt4
print(paste0("200*P(X >4|λ)=",frequency.lt4))
```

Car Accidents	Frequency	Poisson(λ=???)
0	109	108.6701738149
1	65	66.288806027089
2	22	20.2180858382621
3	3	4.1110107871133
4	1	0.626929145034778
>4	0	0.0849943876008563

(d) Given the estimated λ, finish the 4th column of table below using R.

```
for(i in 0:4){
frequency.ideal=200*dbinom(i,200,p=car.lambda/200)
print(paste0("200*P(X =",i,"|n, p)=",frequency.ideal))
}
p2.lt4=1-pbinom(4,200,p=car.lambda/200)
frequency2.lt4=200*p2.lt4
print(paste0("200*P(X >4|n, p)=",frequency2.lt4))
```

Car Accidents	Frequency	Poisson(λ=???)	Binomial(n=200, p= λ/200)
0	109	108.6701738149	108.568924560689
1	65	66.288806027089	66.429654428026
2	22	20.2180858382621	20.2214146923569
3	3	4.1110107871133	4.0830240007738
4	1	0.626929145034778	0.615197595382149
>4	0	0.0849943876008563	0.0817847227718715

Q4. (20%) Binomial, Poisson, and Normal Distributions

(a) Finish the second, third, and fourth column of the table below using R.

```
for(i in 0:8){
 p.binom=dbinom(i,20,0.2)
 p.pois=dpois(i,4)
 p.norm=dnorm(i,4,2)
 print(paste0(i,"|",p.binom,"|",p.pois,"|",p.norm,"|"))
}
```

Defectives	Binomial(n=20, p=0.2)	Poisson(λ=4)	$Normal(\mu = 4, \sigma^2 = 4)$
(x)	P(X=x)	P(X=x)	P(x-0.5< X < x+0.5)
0	0.0115292150460685	0.0183156388887342	0.026995483256594
1	0.0576460752303423	0.0732625555549367	0.0647587978329459
2	0.136909428672063	0.146525111109873	0.120985362259572
3	0.205364143008095	0.195366814813165	0.17603266338215
4	0.218199401946101	0.195366814813165	0.199471140200716
5	0.17455952155688	0.156293451850532	0.17603266338215
6	0.10909970097305	0.104195634567021	0.120985362259572
7	0.0545498504865251	0.0595403626097264	0.0647587978329459
8	0.0221608767601508	0.0297701813048632	0.026995483256594

(b) Based on the probability densities you calculate, generate a plot in which the x-axis is 0:8 and the y-axis lies between [0, 0.25]. The plot should have three lines with different width and colors. (red thin line for Binomial, green thick line for Poisson, blue thicker line for Normal).

```
y.binom=c()
y.pois=c()
y.norm=c()
for(i in 0:8){
        y.binom[i+1]=dbinom(i,20,0.2)
        y.pois[i+1]=dpois(i,4)
        y.norm[i+1]=dnorm(i,4,2)
        print(paste0(i,"|",p.binom,"|",p.pois,"|",p.norm,"|"))
}
x=seq(0,8)
plot(c(0,8),c(0,0.25),type="n")
lines(x,y.binom,col='red')
lines(x,y.pois,col="green",lwd=2)
lines(x,y.norm,col="blue",lwd=5)
```

Q5. (20%) Random Numbers and Monte-Carlo Simulation

(a) Implement the algorithm in the bottom of page 10 in lecture 6. Write a function runi.congru that has five arguments (N, A, B, m, seed) (N is the number of random variates to be simulated). After that, generate five uniform random numbers using A=1217, B=0, m=32767, and seed=1. Save the five numbers as a vector u in R and show me the five numbers.

```
runi.congru =function(N, A, B, m, seed){
```

```
        x=seed
        numbers=c()
        for(i in 1:N){
                x=(A*x+B)%%m
                numbers[i]=x/m
        }
        numbers
}

u=runi.congru(5,1217,0,32767,1)#  0.03714103 0.20062868 0.16510514 0.93295083 0.40116581
```

(b) Implement the inverse transformation method in the bottom of page 11 in lecture 5. Write a function rbinom.invtran that has four arguments (N, n, p, uni) and returns N binomial random numbers. Set n=3, p=0.5, and use the vector u in part (a) as inputs to uni to generate five binom random variates. Show me the simulated numbers too.

```
rbinom.invtran=function(N,n,p,uni){
        numbers=c()
        for(i in 1:N){
                x=0
                while(pbinom(x,n,p) < uni[i]){
                        x=x+1
                }
                numbers[i]=x
        }
        numbers
}
rbinom.invtran(5,3,0.5,u) # 0 1 1 3 1
```

(c) Now, use the function runi.congru in (a) to simulate another 50 numbers by setting seed=2 (A=1217, B=0, m=32767 still) and store the 50 numbers in a vector U.

```
U=runi.congru(50,1217,0,32767,2)
#  [1] 7.428205e-02 4.012574e-01 3.302103e-01 8.659017e-01 8.023316e-01 4.375744e-01 5.280
#  [9] 2.774438e-01 6.490677e-01 9.153417e-01 9.707938e-01 4.560381e-01 9.983520e-01 9.943
# [17] 8.432264e-02 6.206549e-01 3.370464e-01 1.854915e-01 7.431562e-01 4.211249e-01 5.090
# [25] 2.857753e-01 7.885678e-01 6.869716e-01 4.449599e-02 1.516160e-01 5.166173e-01 7.232
# [33] 8.997467e-01 9.917295e-01 9.347819e-01 6.296274e-01 2.565081e-01 1.703238e-01 2.840
# [41] 4.517655e-01 7.986084e-01 9.063692e-01 5.133213e-02 4.712058e-01 4.574419e-01 7.068
# [49] 7.608875e-01 6.103702e-05
```

(d) For the zero-truncated Poission distribution

Following the logic of inverse transformation, write a function rztpois.invtran that has three arguments (N, lambda, uni) (lambda for λ) and returns N zero-truncated Poisson random numbers. Set lambda=4 and use the vector U (part (c)) as inputs to uni to generate 50 non-zero Poisson random variates. Show me the simulated numbers too.

```
dztpois=function(x,lambda){
        lambda^x*exp(-lambda)/(factorial(x)*(1-exp(-lambda)))
}
```

```r
pztpois=function(x,lambda){
        if(x>1){
                return (dztpois(x,lambda)+pztpois(x-1,lambda))
        }
        else{
                return (dztpois(1,lambda))
        }
}

rztpois.invtran=function(N, lambda, uni){
        numbers=c()
        for(i in 1:N){
                x=1
                while(pztpois(x,lambda) < uni[i]){
                        x=x+1
                }
                numbers[i]=x
        }
        numbers
}
rztpois.invtran(50,4,U)
#  [1]  1  3  3  6  6  4  4  4  3  5  7  8  4 11 10  2  2  4  3  2  5  3  4  4  3  6  5  1
# [39]  3  5  4  6  7  1  4  4  5  2  5  1
```

(e)

```r
customers.counts=rztpois.invtran(50,4,U)
seat2=length(customers.counts[customers.counts<=2])#10
seat4=length(customers.counts[customers.counts<=4&customers.counts>2])#20
seat6=length(customers.counts[customers.counts<=6&customers.counts>4])#12
room.private=length(customers.counts[customers.counts>6])#8
```

,四人桌20組,六人桌12組,包廂8組