Appendix

Appendix A1

R Source Code for Confidence interval using direct and inverse binomial sampling method when asympotic of the variance is used.

```
projectDI=function (n,p1,p2,M)
{
set.seed(1234)
 lower=rep(0,M)
 upper=rep(0,M)
 coverage=rep(0,M)
 length=rep(0,M)
 x=rep(0,n)
 m=n*p1
 seta=p1/p2
 for(k in 1:M){
 y=c()
 #--- generate x----------
 for(i in 1:n){
   x[i]=runif(1)
   if(x[i]<p1){
      x[i]=1
```

```
      }
     else {
      x[i]=0
     }
   }
#-----generate y----------
j=0
v=0
while(j < m){
   y[v+1]=runif(1)
   if(y[v+1]<p2){
     y[v+1]=1
     j=j+1
     }
     else {
      y[v+1]=0
     }
     v=v+1
   }
#------calculate Seta_hat----
y_bar=v/m
seta_hat= mean(x)*y_bar
```

```
#------compute CI ----------

sd_est=1.96*sqrt( seta_hat*((y_bar*(1-mean(x))/n )+( mean(x)*(y_bar-1)/m ) ) )

lower[k]=seta_hat-sd_est

upper[k]=seta_hat+sd_est

length[k]=upper[k]-lower[k]

if(seta>=lower[k] && seta<=upper[k]){

  coverage[k]=1

} else {

  coverage[k]=0

}

} #--- for k in M ---

return(mean(coverage),mean(length),median(length),sqrt(var(length)))

}
```

Appendix A2

R Source Code for Confidence interval using direct and inverse binomial sampling method when the true value of the variance is used.

```
projectDI=function (n,p1,p2,M)
{
set.seed(1234)
 lower=rep(0,M)
 upper=rep(0,M)
 coverage=rep(0,M)
 length=rep(0,M)
 x=rep(0,n)
 m=n*p1
 seta=p1/p2
 for(k in 1:M){
 y=c()
 #--- generate x-----------
 for(i in 1:n){
   x[i]=runif(1)
   if(x[i]<p1){
      x[i]=1
```

```
      }
     else {
      x[i]=0
     }
   }
#-----generate y----------
j=0
v=0
while(j < m){
   y[v+1]=runif(1)
   if(y[v+1]<p2){
     y[v+1]=1
     j=j+1
     }
     else {
      y[v+1]=0
     }
     v=v+1
   }
#------calculate Seta_hat----
y_bar=v/m
seta_hat= mean(x)*y_bar
```

```
#------compute CI ----------
sd_est=1.96*seta_hat*sqrt((mean(x)*y_bar)*(((y_bar-1)*(1-mean(x)))/(n*m)
        +(y_bar*(1-mean(x)))/n+(mean(x)*(y_bar-1))/m) )
lower[k]=seta_hat-sd_est
upper[k]=seta_hat+sd_est
length[k]=upper[k]-lower[k]
if(seta>=lower[k] && seta<=upper[k]){
  coverage[k]=1
} else {
  coverage[k]=0
}
} #--- for k in M ---
return(mean(coverage),mean(length),median(length),sqrt(var(length)))
}
```

Appendix A3

R Source Code for Confidence interval using direct and inverse binomial sampling method when the number of successes if fixed as in the first experiment.

```
projectDI=function (n,p1,p2,M)
{
set.seed(1234)
 lower=rep(0,M)
 upper=rep(0,M)
 coverage=rep(0,M)
 length=rep(0,M)
 x=rep(0,n)
 seta=p1/p2
 for(k in 1:M){
 y=c()
 m=0
 #--- generate x----------
 for(i in 1:n){
   x[i]=runif(1)
   if(x[i]<p1){
      x[i]=1
      }
```

```
      else {
       x[i]=0
      }
    if(x[i]==1){
       m=m+1
      }
    }
#-----generate y----------
j=0
v=0
while(j < m){
    y[v+1]=runif(1)
    if(y[v+1]<p2){
       y[v+1]=1
       j=j+1
       }
      else {
       y[v+1]=0
      }
      v=v+1
    }
#------calculate Seta_hat----
```

```
y_bar=v/m

seta_hat= mean(x)*y_bar

#------compute CI ----------

sd_est=1.96*sqrt( seta_hat*((y_bar*(1-mean(x))/n )+( mean(x)*(y_bar-1)/m ) ) )

lower[k]=seta_hat-sd_est

upper[k]=seta_hat+sd_est

length[k]=upper[k]-lower[k]

if(seta>=lower[k] && seta<=upper[k]){

  coverage[k]=1

} else {

  coverage[k]=0

}

} #--- for k in M ---

return(mean(coverage),mean(length),median(length),sqrt(var(length)))

}
```

Appendix A4

R Source Code for Confidence intervals using

direct binomial sampling method

```
projectDD = function (n,m,p1,p2,M)
{
set.seed(123)
 lower=rep(0,M)
 upper=rep(0,M)
 coverage=rep(0,M)
 length=rep(0,M)
 x=rep(0,n)
 y=rep(0,m)
 seta=p1/p2
 for(k in 1:M){
 #--- generate x----------
 for(i in 1:n){
   x[i]=runif(1)
   if(x[i]<p1){
      x[i]=1
      }
     else {
```

```
      x[i]=0
     }
   }
#-----generate y----------
for(j in 1:m){
   y[j]=runif(1)
   if(y[j]<p2){
     y[j]=1
     }
    else {
     y[j]=0
     }
   }
#------calculate Seta_hat----
seta_hat= {mean(x)*(m+1)}/{(m*mean(y))+1}
#------compute CI ----------
lower[k]=seta_hat-1.96*sqrt(seta_hat*((1-mean(x))/(n*mean(y))+
            (seta_hat*(1-mean(y))/(m*mean(y)))))
upper[k]=seta_hat+1.96*sqrt(seta_hat*((1-mean(x))/(n*mean(y))+
             (seta_hat*(1-mean(y))/(m*mean(y)))))
length[k]=upper[k]-lower[k]
if(seta>=lower[k]&&seta<=upper[k])
```

```
            coverage[k]=1
          else coverage[k]=0
}
return(mean(coverage),mean(length),median(length),sqrt(var(length)))
}
```

Appendix A5

R Source Code for Confidence intervals using

inverse binomial sampling method

```
projectII=function (n,p1,p2,M)
{
set.seed(1234)
 lower=rep(0,M)
 upper=rep(0,M)
 coverage=rep(0,M)
 length=rep(0,M)
 m1=n*p1
 m2=n*p2
 seta=p1/p2
 for(k in 1:M){
 x=c()
 y=c()
 #--- generate x----------
 j=0
 v1=0
while(j < m1){
   x[v1+1]=runif(1)
```

```
    if(x[v1+1]<p1){
      x[v1+1]=1
      j=j+1
      }
     else {
      x[v1+1]=0
     }
     v1=v1+1
    }
#-----generate y-----------
j=0
v2=0
while(j < m2){
    y[v2+1]=runif(1)
    if(y[v2+1]<p2){
      y[v2+1]=1
      j=j+1
      }
     else {
      y[v2+1]=0
     }
     v2=v2+1
```

```
    }
#------calculate Seta_hat----
p_hat1=(m1-1)/(v1-1)
p_hat2=(m2-1)/(v2-1)
seta_hat= {v2*(m1-1)}/{(v1-1)*m2}
#------compute CI ----------
sd_est=1.96*sqrt( seta_hat*((p_hat1*(1-p_hat2)/m2)+seta_hat*((1-p_hat1)/m1)) )
lower[k]=seta_hat-sd_est
upper[k]=seta_hat+sd_est
length[k]=upper[k]-lower[k]
if(seta>=lower[k] && seta<=upper[k]){
  coverage[k]=1
} else {
  coverage[k]=0
}
} #--- for k in M ---
return(mean(coverage),mean(length),median(length),sqrt(var(length)))
}
```