```
# AMPLIACÍON DE INVESTIGACIÓN OPERATIVA
# TEMA 6: SIMULACIÓN

# Simulación de las principales distribuciones de probabilidad

# DISTRIBUCIÓN UNIFORME en el intervalo [a,b]: runif(número,a=0,b=1)
# Generación de números aleatorios
  suniforme<-runif(1000,0,1)
# Histograma de los datos simulados
  hist(suniforme,prob=T)
  abline(h=1,col=2,lty=2)

# Contraste de hipótesis
  ks.test(suniforme,"punif")


# DISTRIBUCIÓN UNIFORME discreta 0:n
# Crear la función para la generación a partir de la uniforme
 runiformed<-function(n,nmax){trunc(runif(n,0,nmax+1))}
# Generación de números aleatorios
 suniformed<-runiformed(1000,3)
# Histograma de los datos simulados
  hist(suniformed,prob=T)
# Contraste de hipótesis
  chisq.test(table(suniformed))


# MÉTODO DE INVERSIÓN
# DISTRIBUCIÓN EXPONENCIAL de parámetro "landa": rexp(número,landa)
# Simulación de una distribución exponencial
  rexponencial<-function(n,landa){-(1/landa)*log(1-runif(n))}

# Generación de 10000 números aleatorio según una distribución exponencial
  exponencial<-rexponencial(10000,8)

# Histograma de los datos simulados
  hist(exponencial,prob=T,br=30)

# Estimación de la función de densidad
  lines(density(exponencial),col=2)

# Función de densidad teórica
  lines(density(exponencial)$x,dexp(density(exponencial)$x,8),col=4)

# Contraste de hipótesis
```



```
  ks.test(exponencial,"pexp",8)


# DISTRIBUCIÓN DISCRETA CON ESPACIO DE ESTADOS FINITOS: estados, probabilidades
# Crear la función para la generación a partir de la uniforme

        # n: números a generar
        # estados: estados de la distribución
        # prob: probabilidad de cada estado

        rnp<-function (n, estados, prob)
{
    comparacion <- function(a, estados, prob) {
        cumprob <- c(0, cumsum(prob))
        estados[sum(cumprob <= a)]
    }
     apply(cbind(runif(n)), 1, comparacion, estados, prob)
    }

        estados<-c(1,3)

        prob<-c(0.4,0.6)

        rnp(10,estados,prob)

# Generación de números aleatorios

        simulacionnp<-rnp(10000,estados,prob)

# Histograma de los datos simulados
  hist(simulacionnp,prob=T,xlab="")
# Contraste de hipótesis
  chisq.test(table(simulacionnp),p=prob)


# DISTRIBUCIÓN BINOMIAL (n,p): rbinom(n,p)
# Generación de números aleatorios
  binomial<-rbinom(1000,5,0.4)
# Histograma de los datos simulados
  hist(binomial,prob=T)
# Contraste de hipótesis
  chisq.test(table(binomial),p=dbinom(0:5,5,0.4))


# DISTRIBUCIÓN NORMAL (media, var). rnorm(números,media,sqrt(var))
# Generación de números aleatorios
  rnorm(10,3,2)
# Método Teorema Central del Límite (n=12)
# Programación de generación de números aleatorios
  rnormal<-function(n,nu=0,var=1){A<-matrix(runif(12*n),n,12);
  sqrt(var)*(apply(A,1,sum)-6)+nu}

# Generación de 1000 números aleatorios normales.
  normal<-rnormal(1000,3,4)
```

```
# Histograma de los datos simulados

  hist(normal,prob=T,br=20)

# Función de densidad estimada

  lines(density(normal),col=2)

# Función de densidad teórica

  lines(density(normal)$x,dnorm(density(normal)$x,3,2),col=4)

# Dibujo conjunto

  plot(density(normal)$x,dnorm(density(normal)$x,3,2),xlab="",ylab="",type="l",col=2)

  lines(density(normal)$x,density(normal)$y,col=4)

# Contraste de hipótesis

  shapiro.test(normal)

  ks.test(normal,"pnorm",3,2) # No es muy adecuado


# EJEMPLO DE UN PROCESO DE SIMULACIÓN:

# Considérese un ordenador simplificado, compuesto por un
# sistema de entrada-salida (E/S) y una unidad central de proceso (CPU).
# El ordenador falla cuando lo hace alguno de los dos componentes.
# Estudiar el tiempo medio hasta el fallo del sistema.

# SOLUCIÓN:

# Generación del sistema de tiempo de fallo de un ordenador:

# X1: tiempo de fallo de la E/S

# X2: tiempo de fallo de la CPU

# T=min{X1,X2}: tiempo de fallo del computador

# Estudio del sistema

# Las variables siguen modelos exponenciales con parámetros nu1 y nu2.

# Simulamos "n" réplicas del proceso

# X1: rexp(1,nu1)

# X2: rexp(1,nu2)

# T=min{X1,X2}

# Programación del Sistema

  tiempofallo<-function(n,nu1,nu2){X1<-rexp(n,nu1);
                                   X2<-rexp(n,nu2);
                                   apply(cbind(X1,X2),1,min)}

# Considérese n1=1, nu2=2

# Simulación de 10000 réplicas del sistema

  res<-tiempofallo(10000,1,2)

# Análisis de resultados

# Histograma
```



```
  hist(res,prob=T,br=100)

# Función de densidad Estimada

  lines(density(res))

# Comparación con la función de densidad teórica

  plot(density(res)$x,dexp(density(res)$x,3),xlab="T",ylab="",type="l")

  lines(density(res)$x,density(res)$y,lty=2)

# Test de hipótesis: Comprobación de la distribución del tiempo de fallo.

  ks.test(res,"pexp",3)

# Estimación del tiempo de fallo

  mean(res)

# Intervalo de confianza

  n<-10000

  mean(res)-qt(1-0.05/2,n-1)*sqrt(var(res)/n)

  mean(res)

  mean(res)+qt(1-0.05/2,n-1)*sqrt(var(res)/n)

# Función que determina el intervalo de confianza

  intervalo<-function(n,nu1,nu2,alfa=.05){res<-tiempofallo(n,nu1,nu2);
  c(mean(res)-qt(1-0.05/2,n-1)*sqrt(var(res)/n),mean(res),
  mean(res)+qt(1-0.05/2,n-1)*sqrt(var(res)/n))}

  intervalo(10000,1,2,0.05)

# Análisis de sensibilidad: Variando los parámetros de la exponencial de X1

  rangonu1<-function(n,nu1min,nu1max,nu2,densidad=0.01,alfa=0.5){
  nu<-seq(nu1min,nu1max,densidad);
  apply(cbind(nu),1,intervalo,n=n,nu2=nu2,alfa=alfa)}

# Variamos nu1 entre 1 y 2 distadno 0.1, simulando 10000 procesos para cada uno

  int<-rangonu1(10000,1,2,2,densidad=0.1)


  plot(x<-seq(1,2,0.1),int[1,],type="l",xlab="",ylab="",lty=2,ylim=c(0.25,0.34))

  lines(x,int[3,],lty=2)

  lines(x,int[2,],lty=4)

  lines(x,1/(x+2))

# Con el fin de observar la convergencia de los intervalos de confianza,
# variamos el "n" (tamaño muestral) en la simulación.


  res<-tiempofallo(1000,1,2)

  varitamaño<-function(res,alfa=.05){n<-length(res);A<-matrix(0,3,n);
  for(i in 2:n){A[,i]<-c(mean(res[1:i])-qt(1-0.05/2,i-1)*sqrt(var(res[1:i])/i),
  mean(res[1:i]),mean(res[1:i])+qt(1-0.05/2,i-1)*sqrt(var(res[1:i])/i))};A}

  int2<-varitamaño(res)
```

```
  plot(1:1000,int2[1,],type="l",xlab="",ylab="",lty=2,ylim=c(min(int2),max(int2)),col=4)
  lines(1:1000,int2[3,],lty=2,col=4)
  lines(1:1000,int2[2,],lty=4,col=2)
  lines(1:1000,rep(1/3,1000))


# EJERCICIO A SIMULAR:


# Simular el número de lanzamientos necesarios a realizar de una moneda perfecta, hasta
conseguir 3 caras.

# Generar una realización del sistema

        # m: número de caras a obtener
        # prob: probabilidad de cara


lanzmoneda<-function(m,prob=0.5){i<-0;cont<-0;while(i<m){i<-i+rbinom(1,1,prob);cont<-cont+
1}
;cont}

        lanzmoneda(3)


        # Repertir el proceso anterior "n" veces


        rlanzmoneda<-function(n,m,prob=.5){res<-numeric();for(i in
1:n){res<-c(res,lanzmoneda(m,prob))};res}

        simmoneda<-rlanzmoneda(1000,3)

# Determinar la probabilidad de algún estado en concreto

        # simmoneda: salida de la función rlanzmoneda


probestado<-function(simmoneda,estados){n<-length(simmoneda);sum(simmoneda==estados)/n}

        probestado(simmoneda, estados=3)

        dnbinom(3-3,3,0.5)


# Determinación de intervalo de confianza.

        estados<-3

        n<-length(simmoneda)

        probestado(simmoneda, estados)-qt(1-0.05/2,n-1)*sqrt(var(simmoneda==estados)/n)

        probestado(simmoneda, estados)

        probestado(simmoneda, estados)+qt(1-0.05/2,n-1)*sqrt(var(simmoneda==estados)/n)


# Determinar prob para que la probabilidad de ser 3 (estado) sea mayor de 0.75 (alfa)

        alpha<-0.75


probmoneda<-function(n,m,densidad=0.01){prob<-seq(0+densidad,1,densidad);res<-numeric();
        long<-length(prob);for(i in 1:long){res<-rbind(res,rlanzmoneda(n,m,prob[i]))};res}

        simprobmoneda<-probmoneda(2000,3,0.01)



probmonedaestado<-function(probmoneda,estado){long<-length(probmoneda[,1]);res<-numeric();
        for(i in 1:long){res<-c(res,probestado(probmoneda[i,],estado))};res}

        simprobmonedaestado<-probmonedaestado(simprobmoneda,estados)
```



```
        long<-length(simprobmonedaestado)

        plot(p<-seq(1/(long),1,1/(long)),simprobmonedaestado,type="l",col=2)

# Dibujo de la teórica

        p3<-function(p)p^3

        lines(p,p3(p),col=4)

        abline(h=alpha)

# Determinación de la probabilidad

        pmax<-max(p[simprobmonedaestado<alpha])

        abline(v=pmax);pmax


# EJERCICIO

# Simular el número de lanzamientos necesarios a realizar de una moneda perfecta, hasta #
conseguir 3 caras consecutivas.
```