#######################

# PRUEBAS - PC1 #

#######################

#\*\*PARA UNA VARIABLE DICOTÓMICA

#################

#Prueba Binomial#

#################

#Se necesita la cantidad de éxitos

##table: para que arroje la cantidad de TRUE or FALSE de la condición dada

table(resta$Concurrencia>5)

# Prueba de Hipotesis

#Método exacto

#binom.test(cant.éxitos, #obs. totales, val. hipotético(pi0),"g" )

#tipo de prueba:

#"t" : bilateral(H1:pi!=pi0)

#"l" : unilateral(H1:pi<pi0)

#"g" : unilateral(H1:pi>pi0)

binom.test(19,50,0.4,"g")

#Intervalo de confianza

#TIPO DE PRUEBA: Siempre: "t"

binom.test(19,50,0.4,"t",0.97)

#\*\*PARA EVAÑUAR SUPUESTOS

#\*a)Para determinar la distribución de los datos

##############################

#Prueba de Kolmogorov-Smirnov#

##############################

#Gráficos opcionales

library(vioplot)

par(mfrow=c(2,1))

boxplot(resta$Monto,col=4)

plot(density(resta$Monto),col=4)

#Gráfico de violín:

par(mfrow=c(1,1))

vioplot(resta$Monto,col=4,horizontal=T)

#Nota: por las curvas es probable de que los datos no se ajusten a una uniforme

#Gráfica de la dist. acumulada empírica

plot.ecdf(resta$Monto,col=3)

###PRUEBA

#ks.test(variable,"dist teórica",como estimarías los parámetros en base a esa dist)

ks.test(resta$Monto,"punif",min(resta$Monto),

max(resta$Monto))

#########################################################

#Prueba Chi Cuadrado de Pearson: Ajuste a la multinomial#

#########################################################

tabla<-table(resta$Menú)

prob<-c(1,1,1,1)/4

# Método exacto

library(RVAideMemoire)

#Pasar la tabla inicial a vector

tabla1<-as.vector(tabla)

multinomial.test(tabla1,prob)

##############################################################

# Prueba Chi Cuadrado de Pearson - Ajuste a una dist. teórica#

##############################################################

tabla2<-table(resta$Postres);tabla2

#Gráfico previo:

plot(0:5,tabla2,type="h",col=6,main="Gráfico de líneas")

#Probabilidades

#prob<-dbinom(dimensión de x, máximo valor de x,prob éxito(si no te lo dan se estima))

prob<-dbinom(0:5,5,0.5)

chisq.test(tabla2,p=prob)

#Comentario: esperados menores a 5

res$expected # si tienen valores esperados menores a 5 se agrupan

#Se sumarán los valores de las frecuencias menor a 5 con la siguiente (posición) columna

#Fijarse en la tabla inicial, esos valores se suman

#La nueva tabla sería:

obs<-c(18,11,9,12)

#Tambipen se suman según la posición hallada antes

probc<-c(prob[1]+prob[2],prob[3],prob[4],prob[5]+prob[6])

#Aplicar nuevamente la prueba, con las correcciones:

chisq.test(obs,p=probc)

#\*b) Para probar normalidad

############################

#Prueba deShapiro Wilk #

############################

############################

#Prueba de Anderson-Darling#

############################

primer<-subset(feria,Dia=="Primer")

library(goftest)

goftest::ad.test(primer$Tiempo,"pexp",1/mean(primer$Tiempo), estimated = F)

##################

#Prueba D'Agostino

##################

install.packages("PoweR")

library(PoweR)

#statcompute(Tipo de prueba: 6 (siempre), variable)

statcompute(6, estatura)

######################

#Prueba de Jarque-Bera

######################

install.packages("moments")

library(moments)

jarque.test(estatura)

#\*c) Para evaluar aleatoriedad

#########################

# Prueba de Rachas

#########################

library(tseries)

runs.test(as.factor(primer$Gasto>100))

#\*c) Para evaluar simetría

#\*

####################

#Prueba de Triadas #

####################

###################

# Prueba de simetría

###################

library(lawstat)

#Poner boot=F, para quitar bootstrap

#Prueba MGG (Usar esta!-Máas poderosa)

symmetry.test(resta$Satisfacción,option="MGG",boot=F)

#Prueba Cabilio-Masaro

symmetry.test(resta$Satisfacción,option="CM",boot=F)

#\*\*PARA EVALUAR UN PARÁMETRO DE POSICIÓN

#\*

#\*#Primero hacer pueba de simetría

#Si existe simetría: Uso Wilcoxon

#No existe simetría: Uso signos

###################

#Prueba de Wilcoxon

###################

#Método exacto (Usar este!)

library(exactRankTests)

#wilcox.exact(resta$Satisfacción,mu=valor hipotético,alternative="ltipo de prueba")

wilcox.exact(resta$Satisfacción,mu=7,alternative="l")

# Intervalo de confianza

#Siempre: tipo de prueba ="t"

wilcox.exact(resta$Satisfacción,mu=7,alternative="t",

conf.int=T,conf.level=0.97)

#################

#Prueba de Signos

#################

library(BSDA)

#md=valor hipotético

SIGN.test(resta$Satisfacción,md=7,alternative="l")

#Intervalo de confianza

SIGN.test(resta$Satisfacción,alternative="t",

conf.level = 0.97)

#\*\*PARA DETECTAR OUTLIERS

##Probamos con Dixon

library(outliers)

dixon.test(resta$Monto)

#No se puede realizar la prueba de Dixon

#porque n>30

#Análisis exploratorio

boxplot(resta$Monto)

#Aparentemente no hay outliers

library(vioplot)

vioplot(resta$Monto,col=4,horizontal=T)

#########################

# Prueba de Grubbs

#########################

library(outliers)

#Two.side=T: superior o inferior

grubbs.test(resta$Monto,type=10,two.sided=T)