```
# A Approche discrète
# En première approche, on compte le nombre d'attaques dans chacune des quatre zone du champ
dataD= c(1,3,12,5)
ntot=sum(dataD)
# le côté du champ fait r = 300 m
r=300


# le modèle nul est "proportionnalité à la surface du champ"
surfacescercles = 3.14*c(r/6,r/3,r/2)^2
probasnull = c(r^2-surfacescercles[3],surfacescercles[3]-surfacescercles[2],surfacescercles[2]-
surfacescercles[1],surfacescercles[1])/r^2
effh0 = ntot*probasnull[4:1]
plot(dataD,type="l")
lines(effh0, col="red")


# Chi 2
X2=sum((effh0-dataD)^2/effh0)
pchisq(X2,df=3,low=F)
# non - significant, but unmet assumptions
# Avec la fonction intégrée de R
chisq.test(x=dataD,p=probasnull[4:1],sim=T)


# G test à la place
G = 2*sum(dataD*log(dataD/effh0))
pchisq(G,df=3,low=F)
# La p-value est un peu plus faible, mais toujours non significative


# multinomial exact test trop long à faire
# on prend l'exemple du test binomial en  regroupant les zones centrales et périphériques
densEXACT = dbinom(0:21,21,7.33/21)
plot(densEXACT,ty="h")
abline(v=4,col="red")
# the (one sided) p-value is
pbinom(4,21,7.33/21)
# environ 9 %. Toujours pas significatif, mais plus proche
# en test bilatéral :
probpile=dbinom(4,21,7.33/21)
sum(dbinom(0:21,21,7.33/21)[dbinom(0:21,21,7.33/21)<=probpile])


# B Approche continue
# coordinates of points
coordX = c(515,571)
coords=matrix(c(c(511,547,756,726,758,406,293,258,383,610,713,228,451,667,318,751,234,313,4
28,600,781),c(821,783,821,777,678,733,697,593,596,613,608,483,502,497,420,461,396,304,317,34
3,357)),nc=2)

# calculer la distance de chaque plant au point de lâcher
```

```
disti=function(x) sqrt((coords[x,1]-coordX[1])^2+(coords[x,2]-coordX[2])^2)
lesdist = sapply(1:21,disti)
# on recalibre pour se mettre à l'échelle du champ
lesdist = lesdist/2

# histogramme (densité)
hist(lesdist,xlab="Distance",xlim=c(0,150*1.4),asp=0.6)
# proba cumulée
quantile(lesdist, (0:100)/100)-> quant
plot(x=quant,y=(0:100)/100)
# un meilleur moyen de le faire dans R
plot(ecdf(lesdist))

# null model from geometric considerations
geonull=function(r) {
sapply(r, function(d) {
if(d<=150) (pi/4)*(d/150)^2 else
pi/4+(600*d*sqrt(1-22500/(d^2))+pi*(d^2-22500)-4*d^2*acos(150/d))/90000
})
}

# perform KS test in R
ks.test(y="geonull", x=lesdist,exact=T)
ks.test(y="geonull", x=lesdist,exact=T,alternative="less")


#  II. Avec champ témoin
# distances
coordX = c(321,210)
coords=matrix(c(c(316,187,475,
176,154,248,394,339,297,476,446,381,233,327,294,466,360,318,278,419,372,225,420,185,233,257
,133,180,310,258,488),c(24,37, 24,
101,172,79,74,105,122,116,159,155,166,155,178,234,227,241,228,230,294,222,330,234,271,305,2
40,300,344,346,320)),nc=2)
lesdistT = sapply(1:31,disti)
# on recalibre pour se mettre à l'échelle du champ
lesdistT = lesdistT*(150/(sqrt(sum((c(470,302)-coordX)^2))))


# KS test sur les deux echantillons
ks.test(lesdist,lesdistT)
ks.test(lesdist,lesdistT,exact=T,alternative="less")

# tests de rang a la Wilcoxon
# remplacer par les rangs
ra=rank(c(lesdist,lesdistT))
sum1=sum(ra[1:length(lesdist)])
sum2=sum(ra[-(1:length(lesdist))])
# test
wilcox.test(lesdist,lesdistT)
wilcox.test(lesdist,lesdistT,simulated=T,alternative="greater")
```

```
# et si c'etait apparie ?
# signes
as.integer((lesdist-lesdistT[1:length(lesdist)])>0)
# wilcoxon
wilcox.test(lesdist,lesdistT[1:length(lesdist)],simulated=T,alternative="greater")

# ANOVA non parametrique
# on genere des donnees non gaussiennes
moy=c(1,2,4,1)
lapply(moy,function(x) x+ exp(rnorm(n=14,mean=0,sd=1)^3))-> dat
boxplot(lapply(dat,log))
hist(log10(dat[[3]]))

kruskal.test(dat)
# on compare a l'approche classique
a=c(rep(1,14),rep(2,14),rep(3,14),rep(4,14))
anova(lm(unlist(dat)~as.character(a)))
anova(lm(log(unlist(dat))~as.character(a)))

# Correlations
plot(lesdist,lesdistT[1:length(lesdist)])
cor.test(lesdist,lesdistT[1:length(lesdist)],method="spearman",exact=T)
cor.test(ra[1:length(lesdist)],ra[-(1:length(lesdist))][1:length(lesdist)])
cor.test(lesdist,lesdistT[1:length(lesdist)],method="kendall",exact=T)


# Approche discrete a nouveau
dataDT = c(5,9,14,3)
ntotT=sum(dataDT)

conting=matrix(c(dataD,dataDT),nc=2)

chisq.test(conting,sim=T)

fisher.test(conting,sim=T)
```