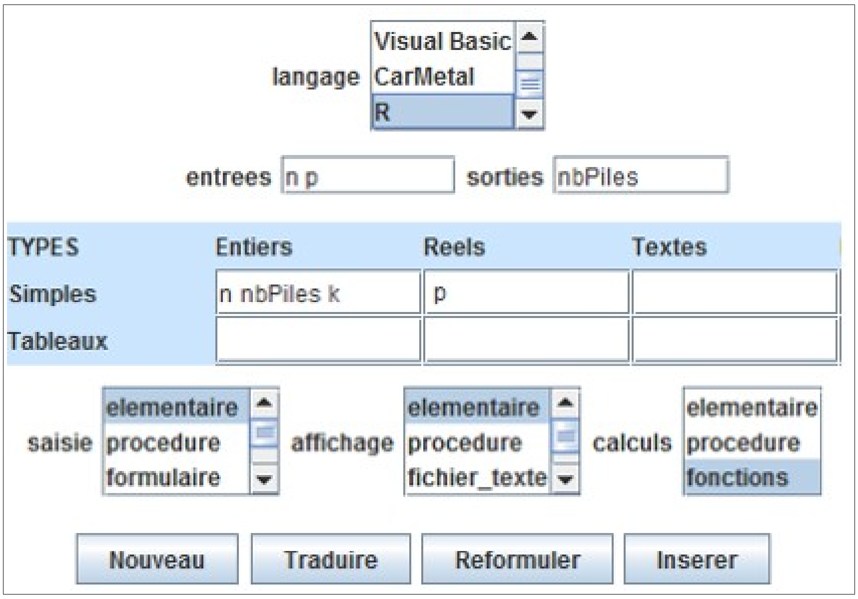
Loi binomiale en R avec PluriAlgo

Ce document reprend (et prolonge) en R le problème de synthèse traité en Xcas dans l'article.

Etape 1 : calcul du nombre de piles

L'onglet Principal permet de créer une fonction comptant le nombre de piles (nbPiles) lors d'une simulation de n lancers d'une pièce ayant la probabilité p de tomber sur pile.



Deux solutions sont envisageables :

- recopier dans l'éditeur la solution Xcas (disponible dans le fichier zippé contenant les programmes de l'article), puis cliquer sur le bouton **Traduire**.
- créer la fonction en R (bouton **Insérer**), en la complétant avec l'onglet Boucles.

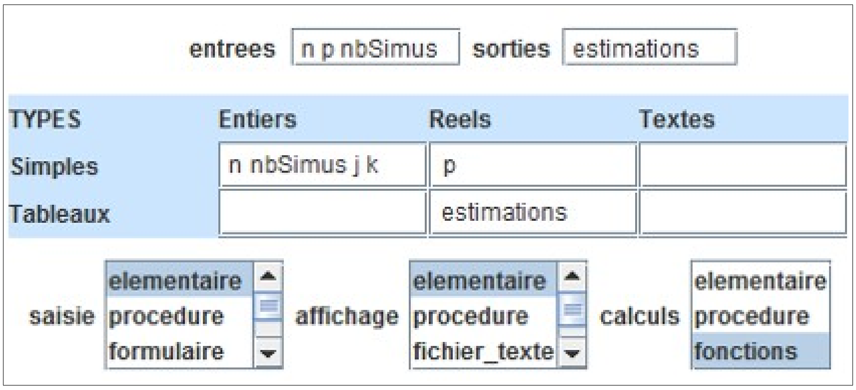
Si vous optez pour la première solution, il faut adapter la traduction obtenue en remplaçant l'instruction Xcas calculant le réel aléatoire compris entre 0 et 1 :

```
calculer_nbPiles = function(n, p) {
     nbPiles=0
     for (k in seq(1, n)) {
          if (alea(0,1)<=p) {      alea(0,1) à remplacer par runif(1)
               nbPiles=nbPiles+1
          }
     }
     return(nbPiles)
}
```

Si vous optez pour la deuxième solution, voici les manipulations à effectuer dans l'onglet Boucles :

Etape 2 : répétition de la simulation

La seconde étape consiste à répéter la simulation de n lancers, afin de construire le tableau estimant les diverses probabilités relatives à la loi binomiale. Ses entrées sont donc les paramètres de la loi (n et p) et le nombre de simulations (nbSimus), ce qui est précisé dans l'onglet Principal :



Comme pour l'étape 1, deux solutions sont envisageables : traduction de la solution Xcas (voir résultat ci-dessous) ou création de la fonction en R.

```
calculer_estimations = function(n, p, nbSimus) {
    estimations = numeric(0)
    for (k in seq(0, n)) {
        estimations[k+1]=0
    }
    for (j in seq(1, nbSimus)) {
        k=calculer_nbPiles(n, p)
        estimations[k+1]=estimations[k+1] + 1/nbSimus
    }
    return(estimations)
}
```

On constate que le traducteur gère correctement la numérotation des tableaux qui démarre à 1 en R, mais pas la déclaration de la variable locale « estimations » qui est à ajouter.

Compléments sur R...

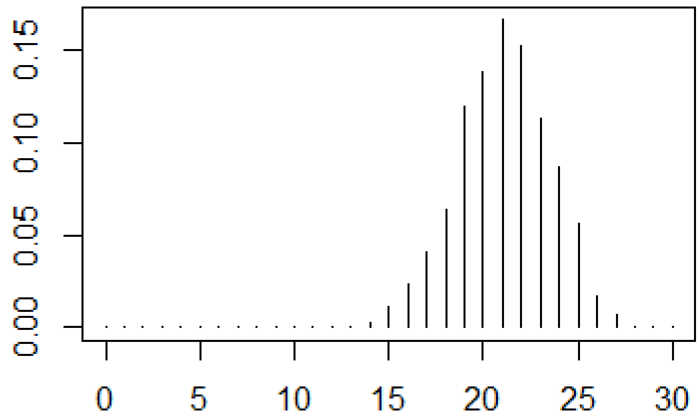
La fonction calculer_estimations peut ensuite être testée avec une simple ligne de commande :

```
> calculer_estimations(10,0.5,1000)
 [1] 0.001 0.012 0.044 0.114 0.192 0.230 0.239 0.106 0.047 0.014 0.001
```

Remarque : les estimations auraient pu être obtenues sans programmation

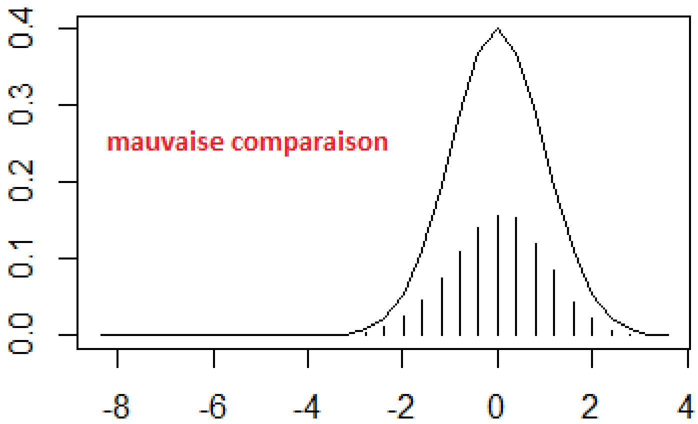
```
> table(rbinom(1000,10,0.5))/1000
    0     1     2     3     4     5     6     7     8     9    10
0.001 0.006 0.051 0.120 0.214 0.272 0.168 0.117 0.042 0.007 0.002
```

Diagramme en bâtons



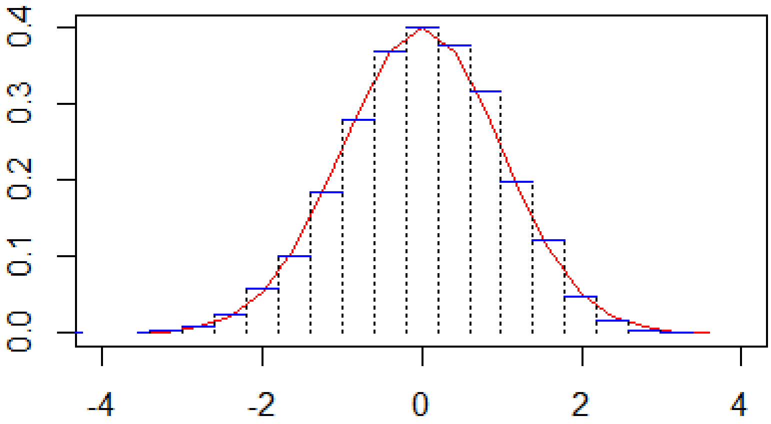

```
n=30
p=0.7
nbSimus=1000
estim1 = calculer_estimations(n,p,nbSimus)
plot(estim1, type="h", xlab="", ylab="")
```

Comparaison loi binomiale centrée réduite et loi normale centrée réduite




```
# diagramme en bâtons de la loi binomiale centree-reduite
sigma = sqrt(n*p*(1-p))
mu = n*p
centre_reduit = (seq(0,n)-mu)/sigma
plot(centre_reduit, estim1, type="h", xlab="", ylab="", ylim=c(0,dnorm(0)))
# ajout sur le graphique de la loi normale centree-reduite
lines(centre_reduit, dnorm(centre_reduit))
```

Pour que la comparaison des deux représentations soit possible, il faut se mettre en situation de comparer deux variables continues (voir [[1]]).




```
# la loi normale
plot(centre_reduit, dnorm(centre_reduit), type="l", col="red")
# la fonction en escalier (en bleu) et l'histogramme
x = centre_reduit - 0.5/sigma
y = sigma*estim1
for (k in seq(1, n)) {
  lines( c(x[k],x[k]) , c(0, y[k]), lty="dotted" )
  lines( c(x[k],x[k+1]), c(y[k], y[k]), col="blue" )
  lines( c(x[k+1],x[k+1]) , c(0, y[k]), lty="dotted" )
}
```

La loi binomiale centrée-réduite peut être assimilée à une variable aléatoire continue dont la densité est une fonction en escalier, représentée ici en bleu. Les rectangles ont pour aire les estimations de la loi binomiale : leur largeur est 1/sigma et leur hauteur est sigma*estimation.

1 Du discret au continu : loi binomiale et loi normale - H. Lample et D. Bernard - (http://pedagogie.ac-toulouse.fr/math/stages/terminales/2011_12/probabilites/02_binomiale_normale_lyon.pdf)