Introduction à R partie 3

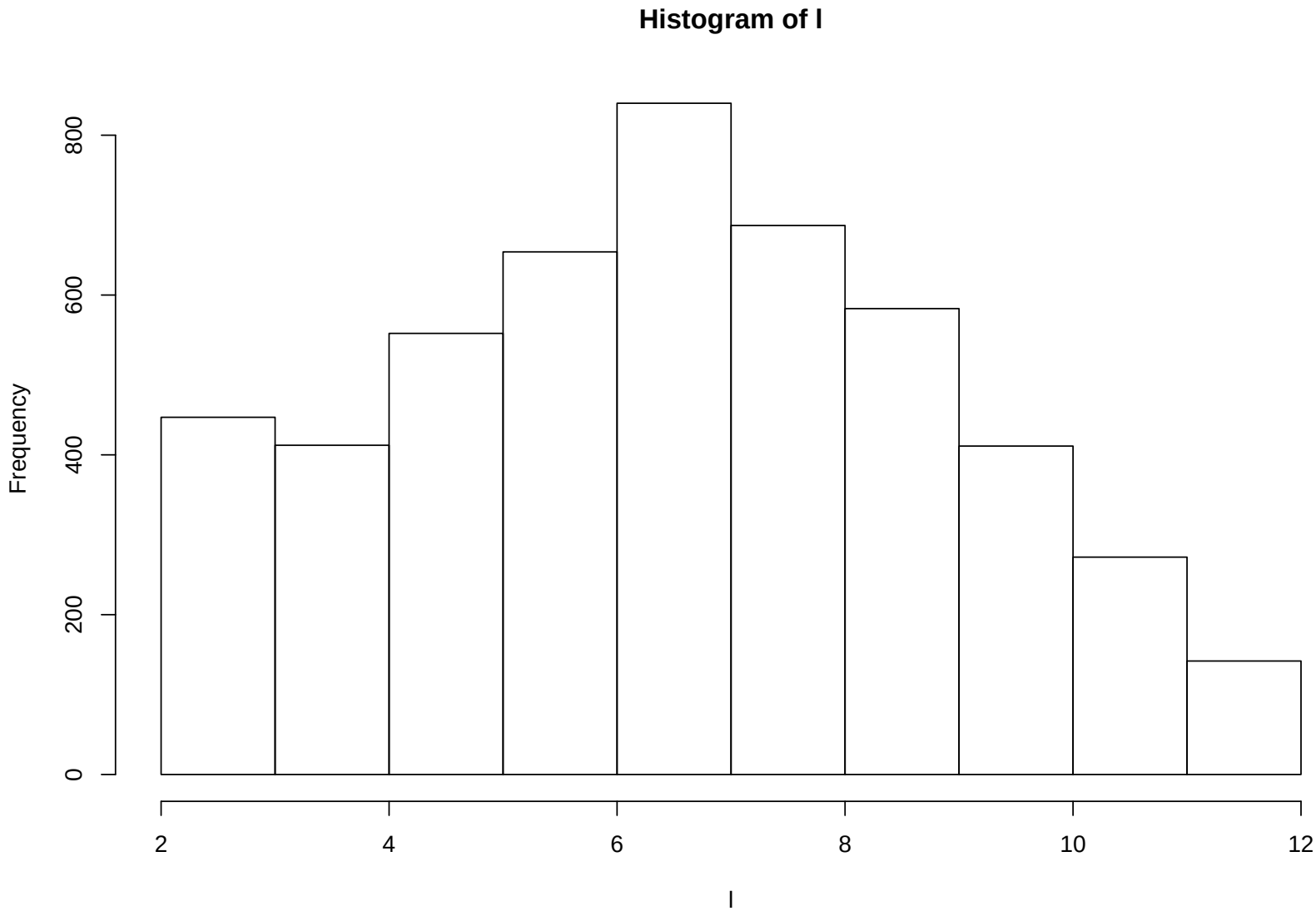
Générer des données

```r
norm<-rnorm(1000, mean=5, sd=10) #Loi normale
```

Générer des données

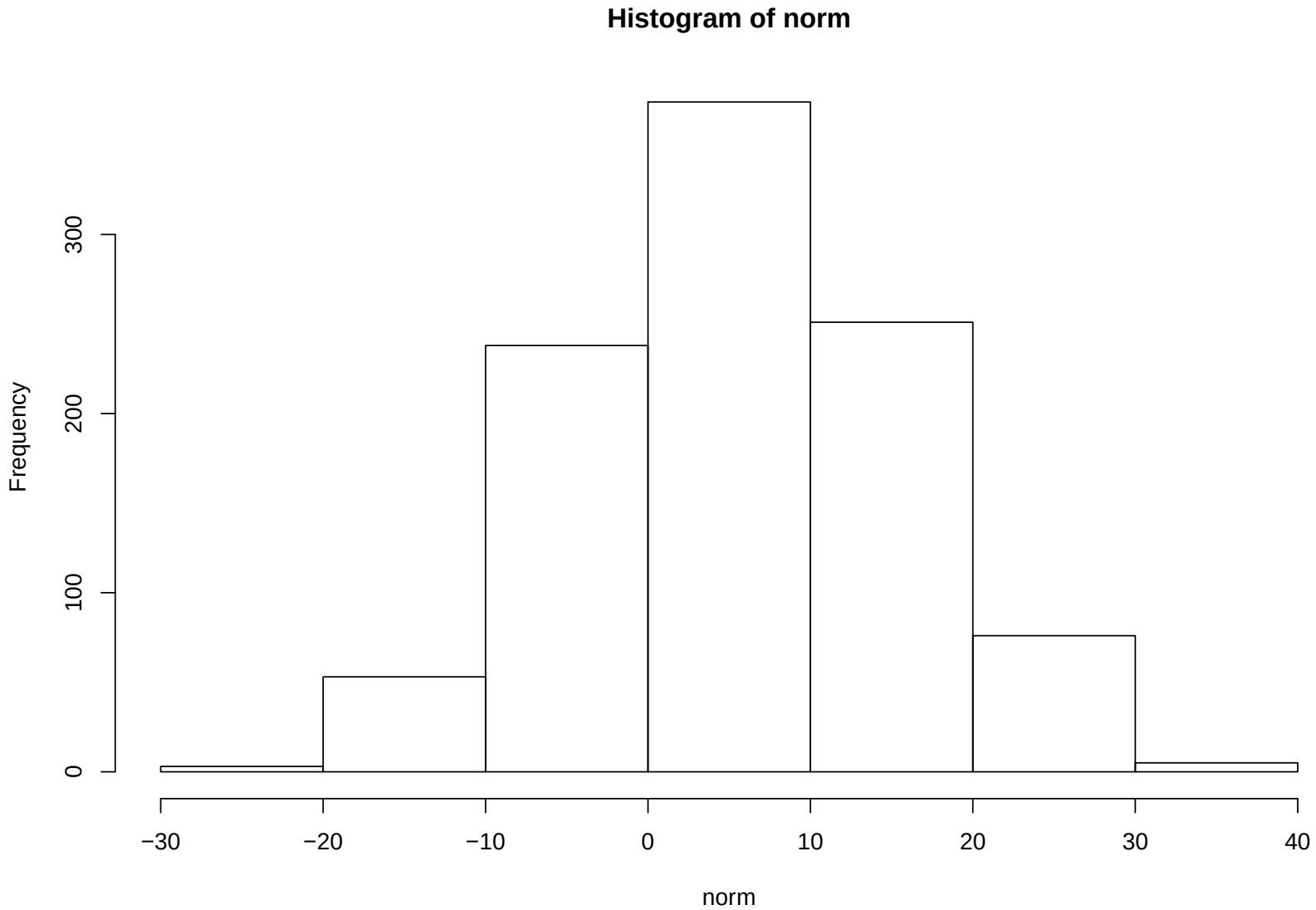
```
seq <- rep(1:6, 10000)
d1 <- sample(seq, 5000)
d2 <- sample(seq, 5000)
l <- d1+d2
```

```r
hist(l)
```



Générer des données

```
hist(norm, breaks = 8)
```

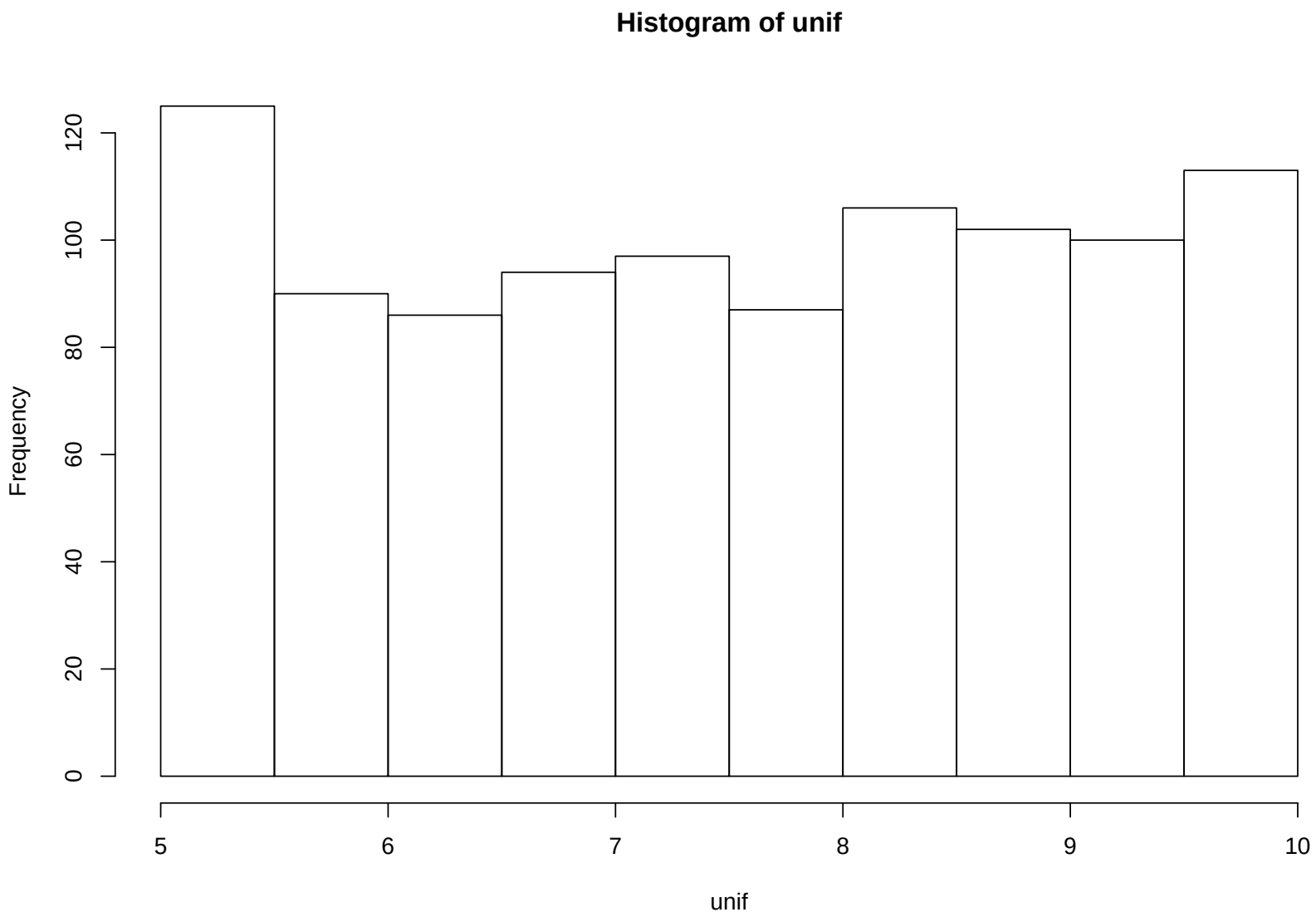


Générer des données

```r
unif<-runif(1000, min=5, max=10) #Loi uniforme
```

Générer des données

```
hist(unif)
```



Manipulation des données

```
tableau <- matrix(c(84, 97, 60, 55, 40, 75, 45, 45, 96),
                  ncol=3)

rownames(tableau) <- c("Site1", "Site2", "Site3")

colnames(tableau) <- c("sp1","sp2","sp3")
```

Manipulation des données

```r
tab1 <- apply(tableau, 1, sum)

tab2 <- apply(tableau, 2, mean)
```

Manipulation des données

```
tab3 <- apply(tableau, 2, function(x) x/2)
```

Manipulation des données

```
coef <- c(2, 0, 1)

tab4 <- apply(tableau, 2, function(x) x/coef)
```

Programmation dans R

Les boucles

Example 1

```
for (i in 1:10){
  print(i)
}
```

```
## [1] 1
## [1] 2
## [1] 3
## [1] 4
## [1] 5
## [1] 6
## [1] 7
## [1] 8
## [1] 9
## [1] 10
```

Programmation dans R

Les boucles

Example 2

```
for(i in 1:5){
print(paste("Working on step", i))
}
```

```
## [1] "Working on step 1"
## [1] "Working on step 2"
## [1] "Working on step 3"
## [1] "Working on step 4"
## [1] "Working on step 5"
```

Programmation dans R

Les boucles

```
j <- rep(NA, 10)
for (i in 1:10){
  j[i] <- 3*i^2
  print(j)
}
```

```
##  [1]  3 NA NA NA NA NA NA NA NA NA
##  [1]  3 12 NA NA NA NA NA NA NA NA
##  [1]  3 12 27 NA NA NA NA NA NA NA
##  [1]  3 12 27 48 NA NA NA NA NA NA
##  [1]  3 12 27 48 75 NA NA NA NA NA
##  [1]   3  12  27  48  75 108  NA  NA  NA  NA
##  [1]   3  12  27  48  75 108 147  NA  NA  NA
##  [1]   3  12  27  48  75 108 147 192  NA  NA
##  [1]   3  12  27  48  75 108 147 192 243  NA
##  [1]   3  12  27  48  75 108 147 192 243 300
```

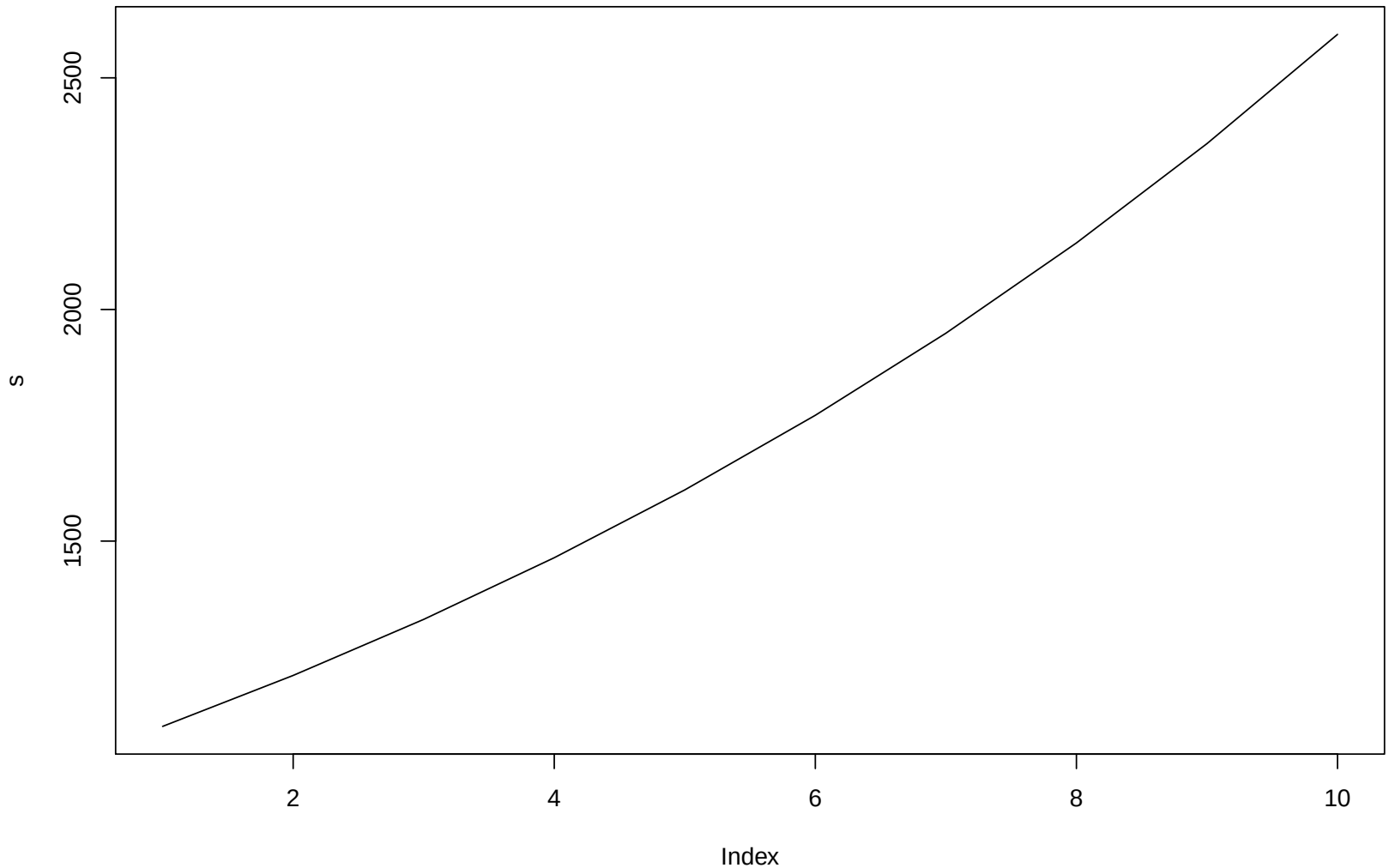
Programmation dans R

Les boucles

```r
j <- 1000
s <- vector()
for (i in 1:10){
  j <- j + j*0.1
  print(j)
  s[[i]] <- j
}
```

```
## [1] 1100
## [1] 1210
## [1] 1331
## [1] 1464.1
## [1] 1610.51
## [1] 1771.561
## [1] 1948.717
## [1] 2143.589
```

```r
plot(s, type = "l")
```



Programmation dans R

Les boucles

- Boucles imbriquées

```
for(i in 1:4){
for(j in 1:3){
print(c(i,j))
}
}
```

```
## [1] 1 1
## [1] 1 2
## [1] 1 3
## [1] 2 1
## [1] 2 2
## [1] 2 3
## [1] 3 1
## [1] 3 2
## [1] 3 3
```

Programmation dans R

Les expressions conditionnelles

Souvent il est utile d'évaluer une expression ou une commande avec des opérateurs booléen (opérateurs logiques) pour vérifier si l'expression est Vraie (True) ou Fausse (False).
Dans R on utilise :

- Equals : $==$
- Not equals : $!=$
- Greater than : $>$
- Less than : $<$
- Greater than or equals ; $>=$
- AND : &
- OR : |

Programmation dans R

Les expressions conditionnelles

```
x <- c(1,2,3)
y <- c(3,2,1)
x==y
```

```
## [1] FALSE  TRUE FALSE
```

```
x>y
```

```
## [1] FALSE FALSE  TRUE
```

```
x>y | x<y
```

```
## [1]  TRUE FALSE  TRUE
```

Programmation dans R

Les expressions conditionnelles

```
if(TRUE){
## do this
}else{
## do this
}
```

```
## NULL
```

Programmation dans R

Les expressions conditionnelles

```
for(i in 1:3){
if(x[i]>y[i]){
print(paste(x[i], "is greater than", y[i]))
}else{
print(paste(x[i], "is not greater than", y[i]))
}
}
```

```
## [1] "1 is not greater than 3"
## [1] "2 is not greater than 2"
## [1] "3 is greater than 1"
```

Programmation dans R

Les fonctions

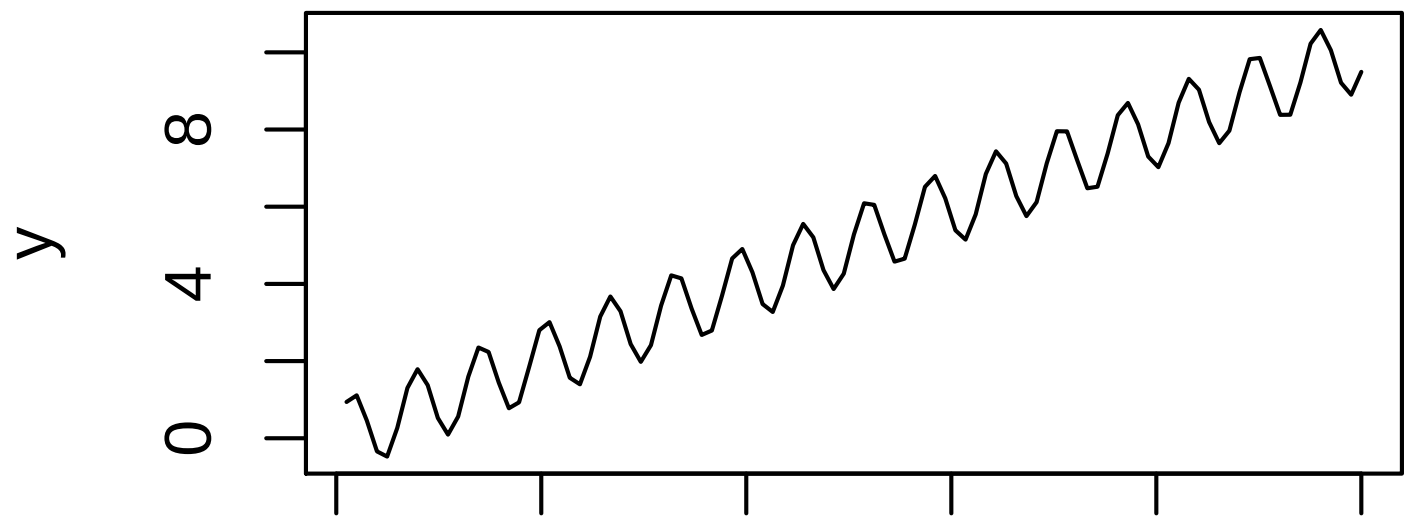
If you have to do something twice, write a function to do it.

Programmation dans R

Les fonctions

```r
my_function <- function(x){ 0.1*x + sin(x) }

# plot the function
curve(my_function, from = 1, to = 100, ylab = "y")
```



Programmation dans R

Les fonctions

Il n'y a pas de racine cubique dans R.

```
cube.root<-function(x){
if (x<0) {
stop("Value less than zero,
not dealing with complex numbers here")
} else {
result <- (x)^(1/3)
return(result)
}
}
```

Programmation dans R

Les fonctions

- Créer une fonction qui calcule la croissance de *Sardina pilchardus* à partir de l'équation de *Von Bertalanfy*.

Rappel de l'équation :

$$L_t = L_{inf}.(1 - exp^{-K(t-t_0)})$$

- représenter graphiquement l'équation pour t allant de 0 à 10

Exercice

1/ créer un vecteur *taille_poisson* composé de 100 observations ayant une distribution normale et de taille moyenne de 15 cm avec un écart-type de 5

2/ faire un plot de la distribution des tailles de poissons

3/ créer un vecteur *poids_poisson* composé de 100 observations. Les poids des poissons doivent être calculer à partir de la la relation taille-poids $W = aL^b$ avec $a = 0.2$ et $b = 3$

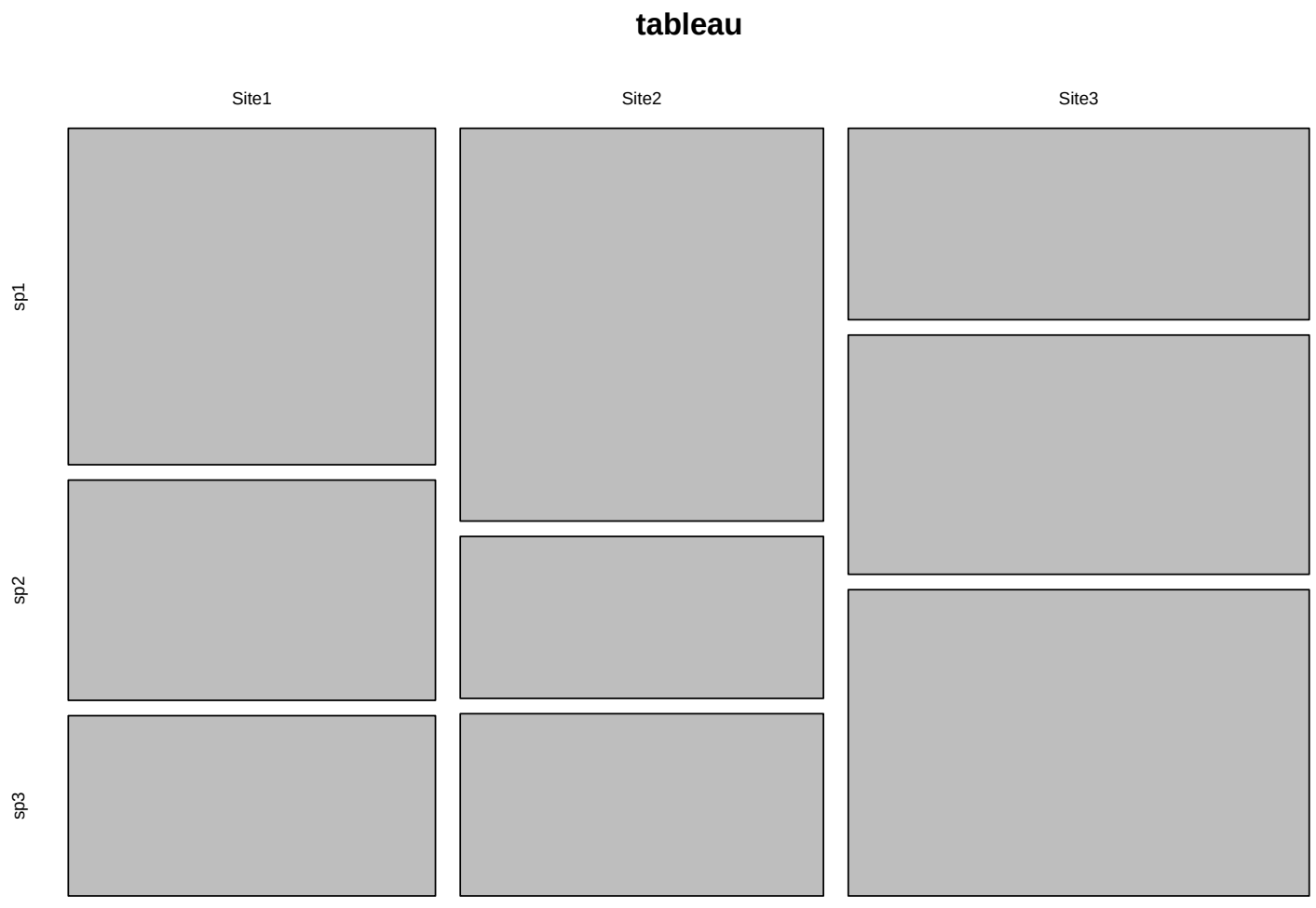
4/ créer un tableau avec les données de taille et de poids et enregistrer le dans le répertoire de travail en format *csv* sous le nom *data_poisson*

5/ lire le fichier data_poisson et calculer les valeurs moyennes de la taille et du poids

6/ faire une figure qui présente le poids en fonction de la taille du poisson en utilisons le fonction *plot*

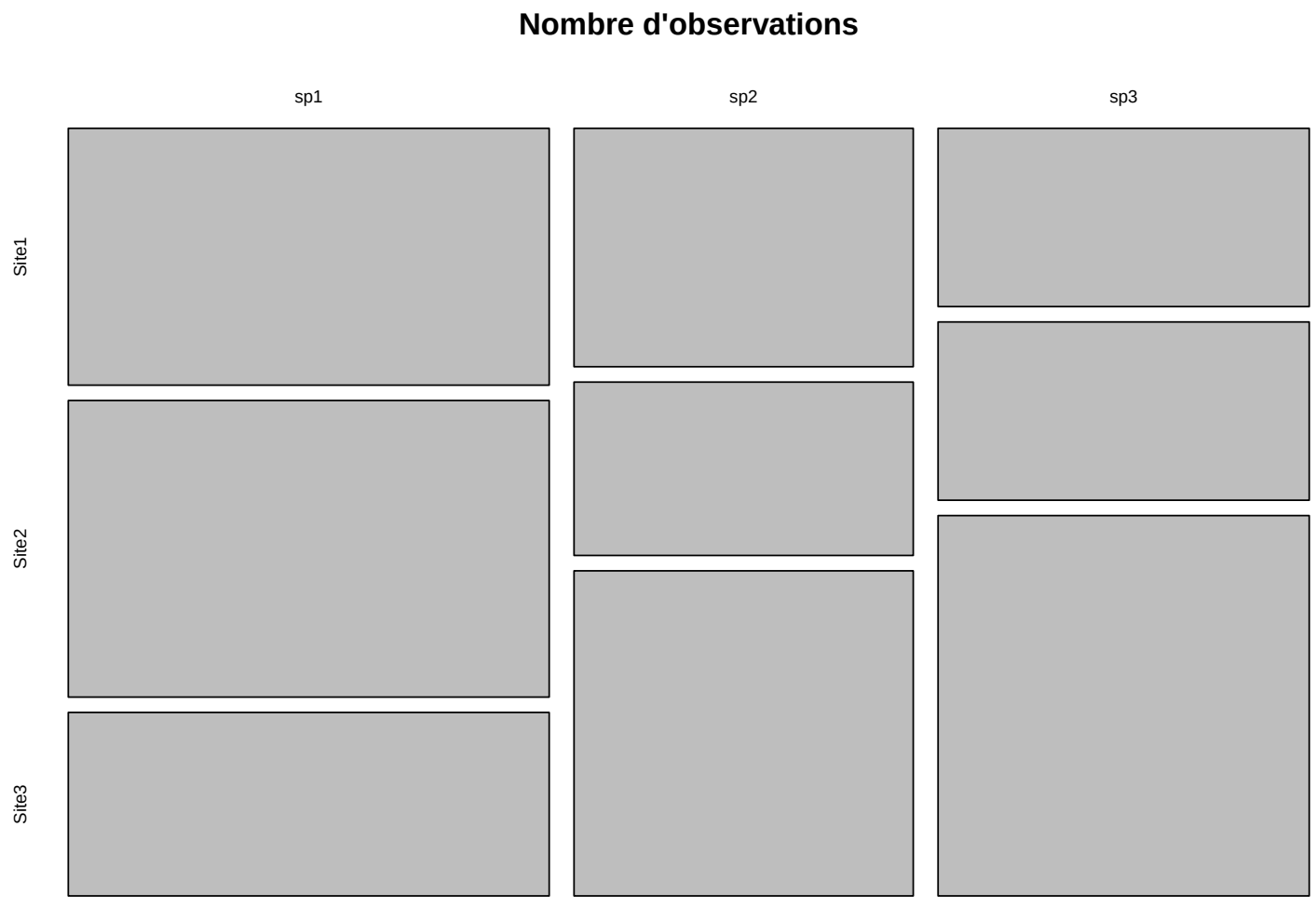
Les plots sur R

```
mosaicplot(tableau)
```



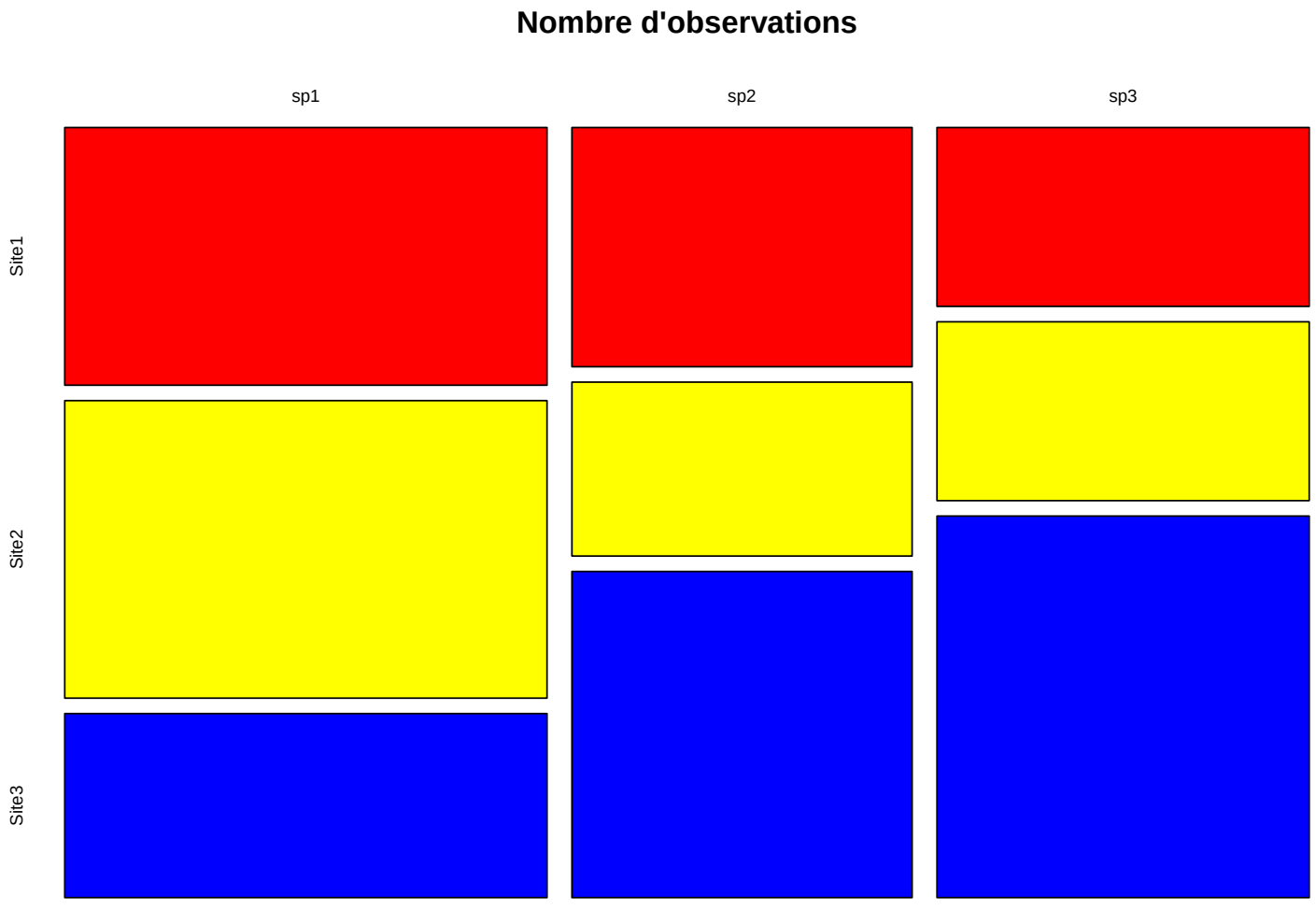
Les plots sur R

```
mosaicplot(t(tableau), main = "Nombre d'observations")
```



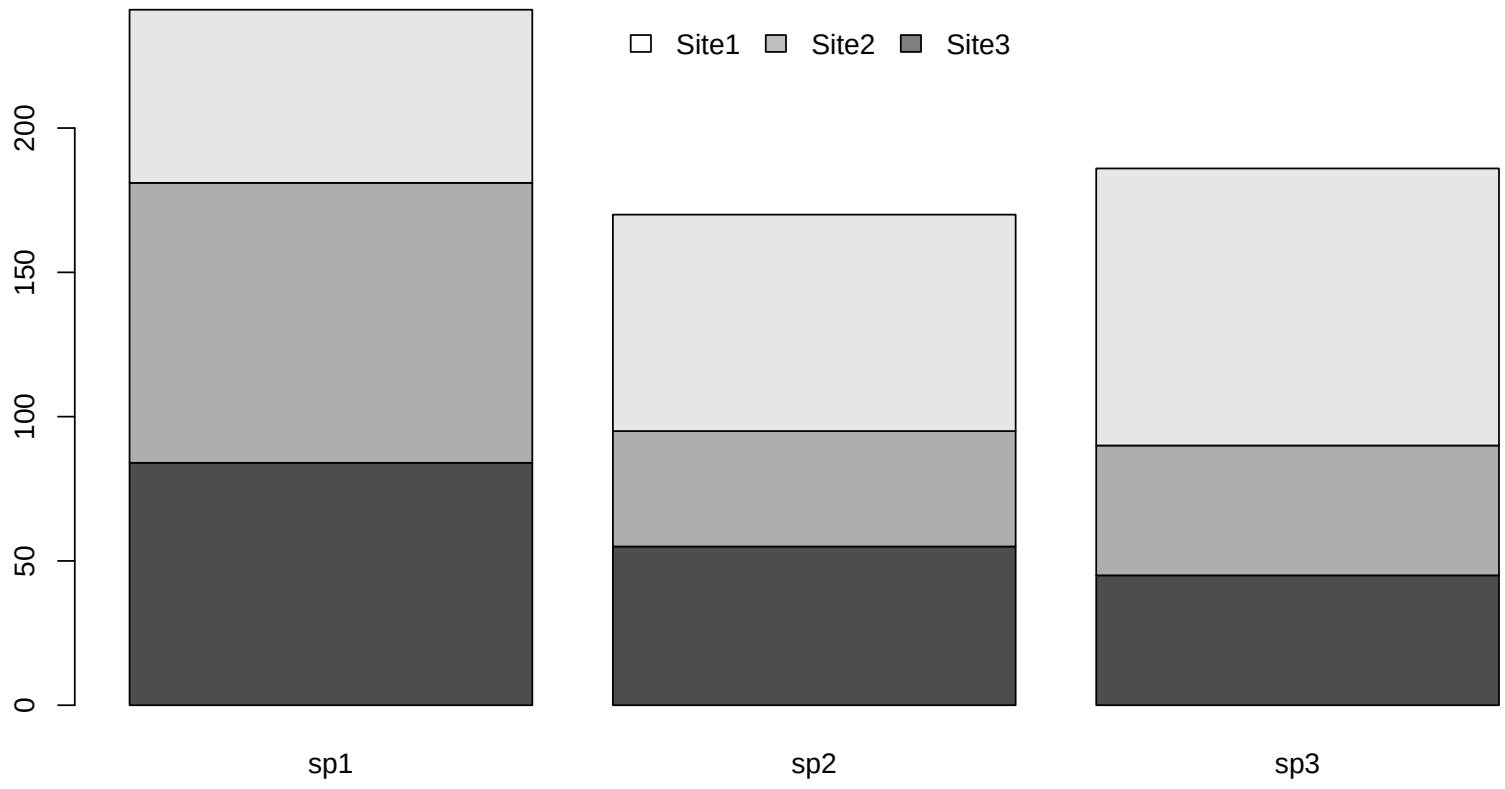
Les plots sur R

```
mosaicplot(t(tableau), col=c("red", "yellow", "blue"),
           main = "Nombre d'observations")
```



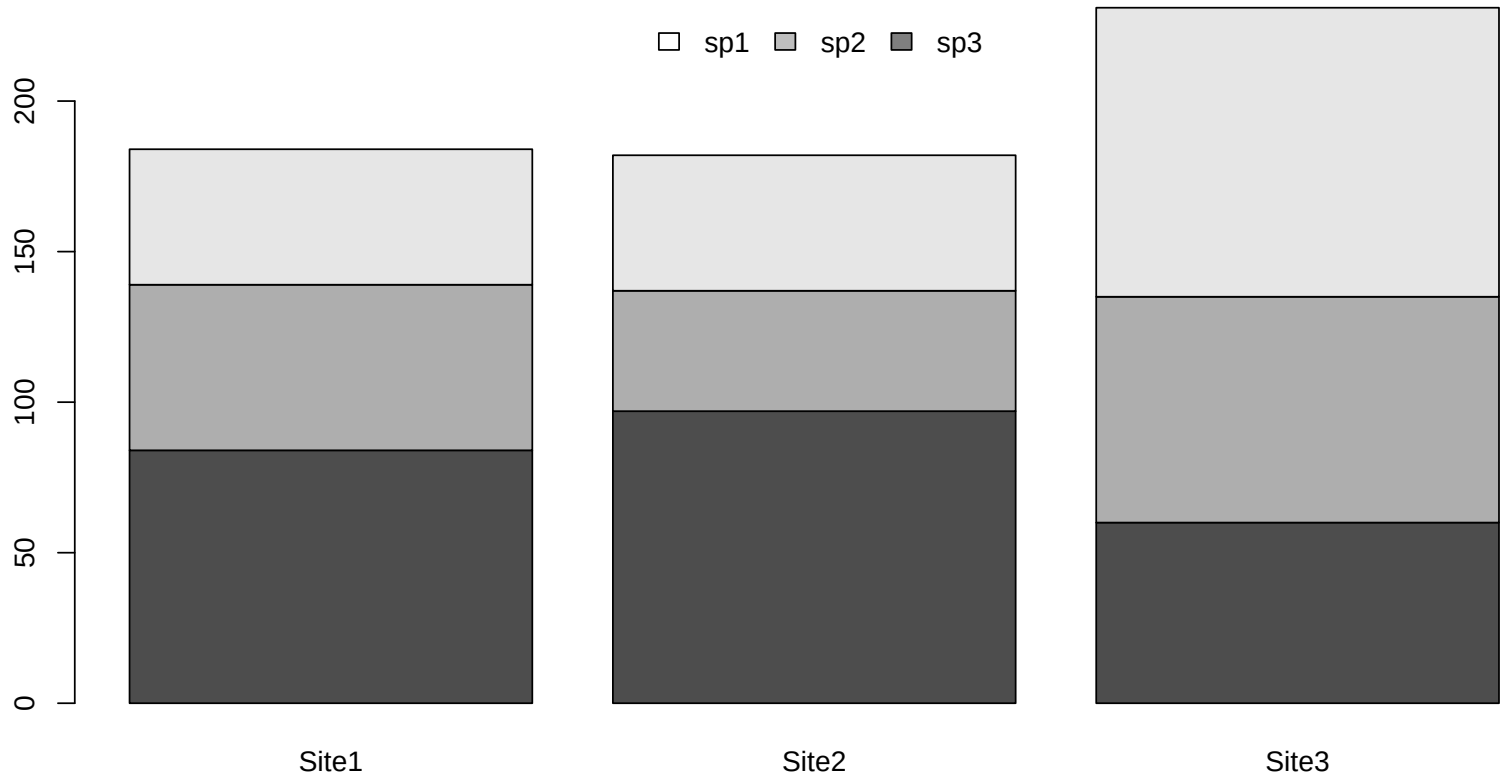
Les plots sur R

```
barplot(tableau)
legend("top", legend=rownames(tableau),
       fill=c("white", "grey", "grey50"),
       ncol = 3, bty = "n")
```



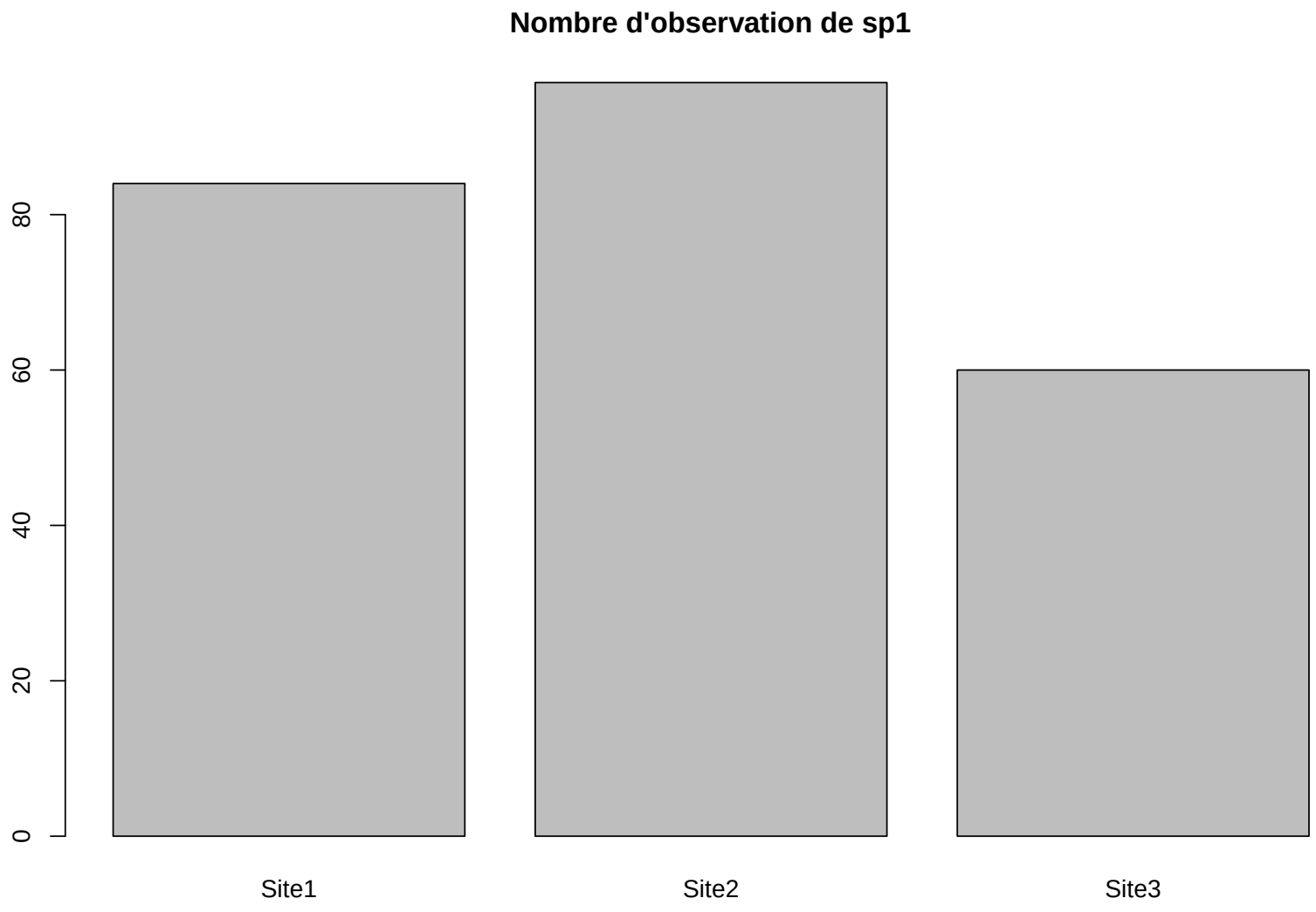
Les plots sur R

```
barplot(t(tableau))
legend("top", legend=colnames(tableau),
       fill=c("white", "grey", "grey50"),
       ncol = 3, bty = "n")
```



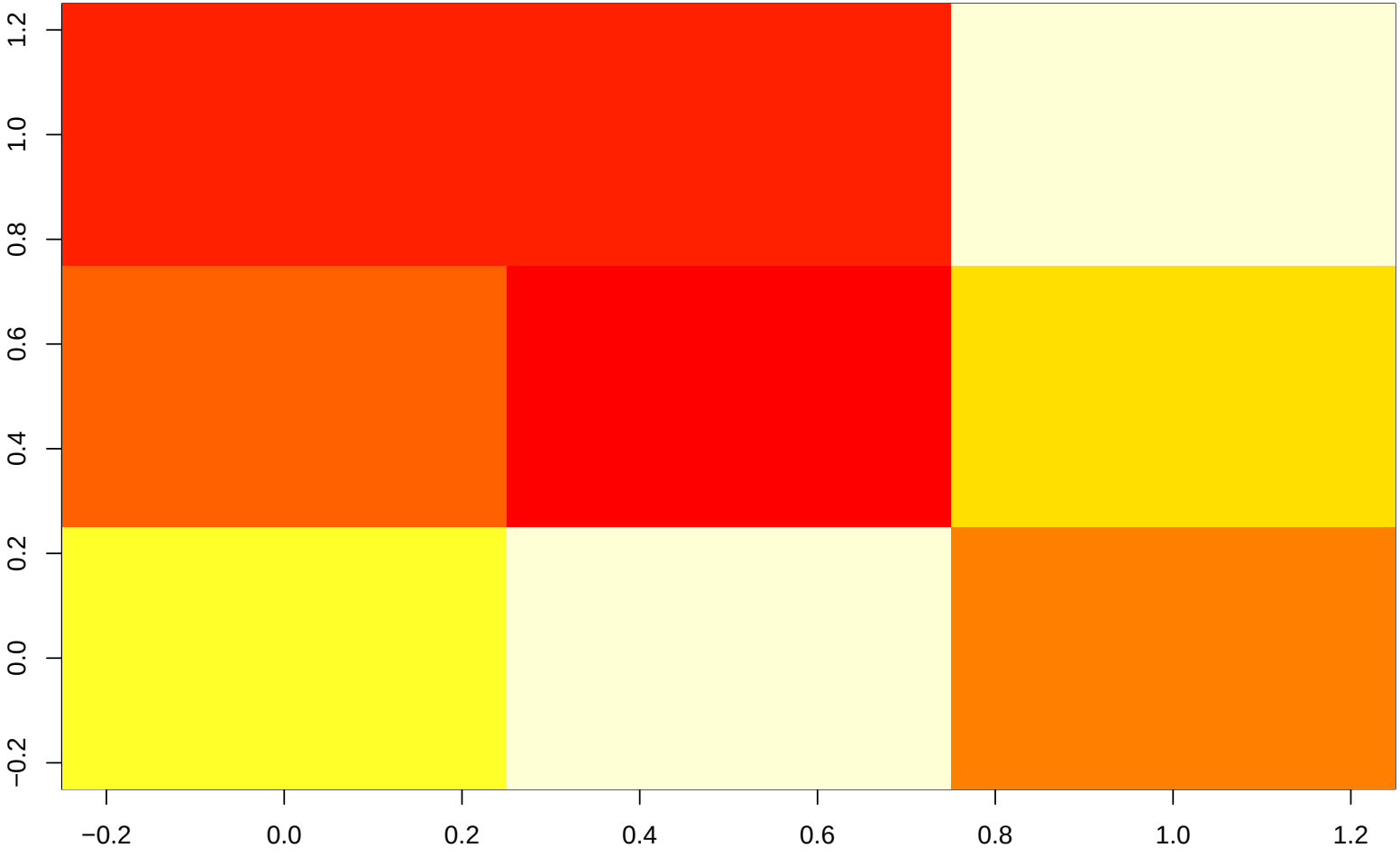
Les plots sur R

```
barplot(tableau[,1], main = "Nombre d'observation de sp1")
```



Les plots sur R

```
image(tableau)
```



Les plots sur R

Sauvegarder une figure en format pdf

```
pdf("My_First_Plot.pdf")
barplot(tableau[1,], main = "Nombre d'observation dans le s
        col=c("hotpink", "sandybrown", "turquoise"),
        border=NA)
dev.off()
```