Introduction au logiciel R

Représenter les données

Laurence Viry

MaiMoSiNE - Collège des écoles doctorales Grenoble

7-16 Février 2017

Plan du cours

Les graphiques de bases

Pour commencer, on peut regarder **quelques exemples de représentations graphiques** en utilisant la fonction **demo()**.

```
> demo(graphics)
```

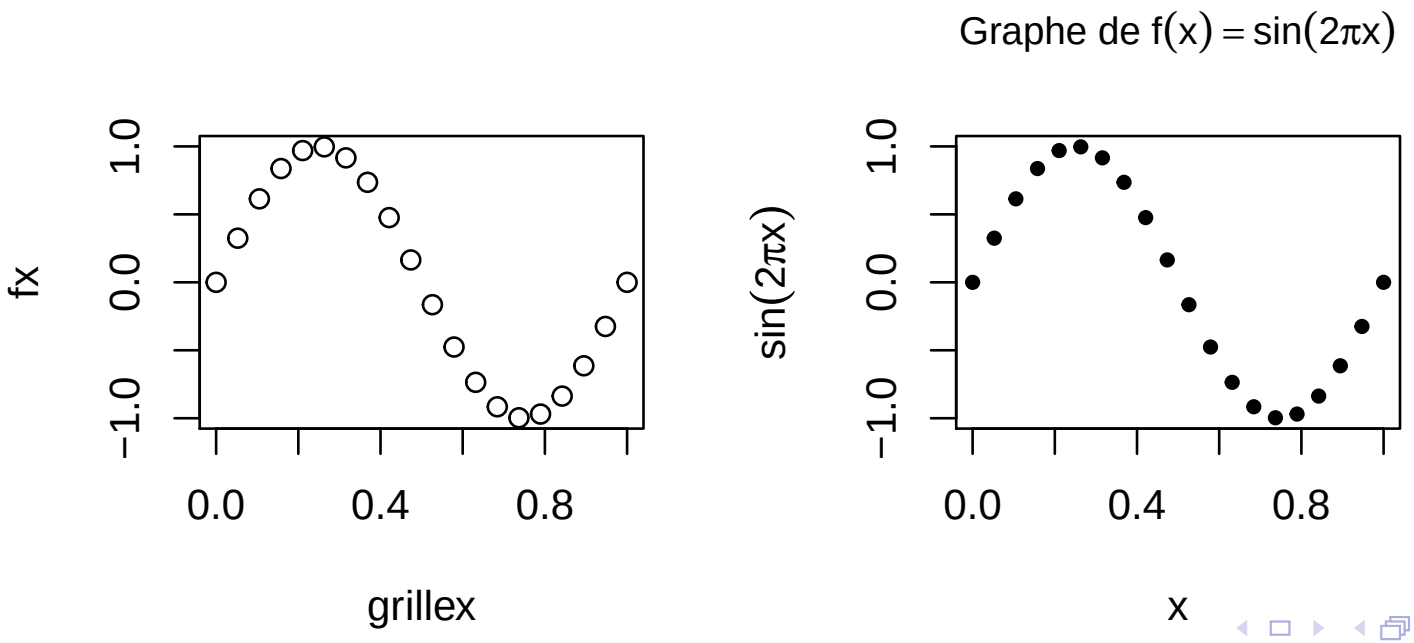
- **Variables qualitatives**
  - pie(x) # diagramme camenbert
  - barplot(x) # diagramme bâton
- **Variables quantitatives**
  - hist(x,nclass) # histogramme de x
  - boxplot(x) # boite à moustache
  - stripchart(x)
- **Graphiques 3D**
  - image(x,y,z) # forme d'image
  - persp(x,y,z) # forme de nappe c
  - ontour(x,y,z) # les contours

  Fonction utile : z=outer(x,y,fonction)
- **Tableau et matrices**
  - pairs(data) # nuage de points colonne par colonne de data
  - matplot(data) # trace chaque colonne de data

La fonction plot (fonction)

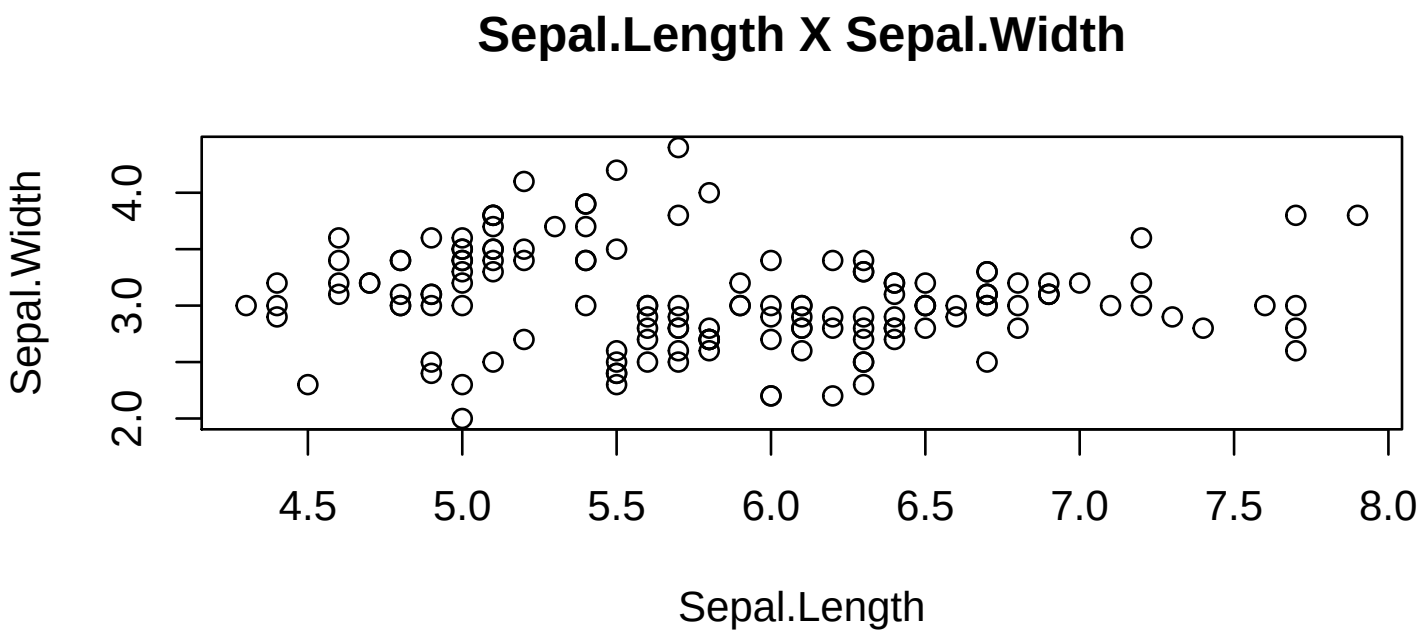
La fonction **plot** est **une fonction générique** de **R** permettant de représenter tout type de données.

```
> par(mfrow=c(1,2)) # 3 figures sur une ligne
> grillex <-seq(0,1,length=20)
> fx <- sin(2*pi*grillex)
> plot(x=grillex,y=fx)   # sans paramètres
> plot(x=grillex,y=fx, main=expression("Graphe de f"(x) == sin(2*pi*
+  ,xlab="x",ylab=expression(sin(2*pi*x)),cex.main=0.9,pch=20)
```



La fonction plot sur un data-frame

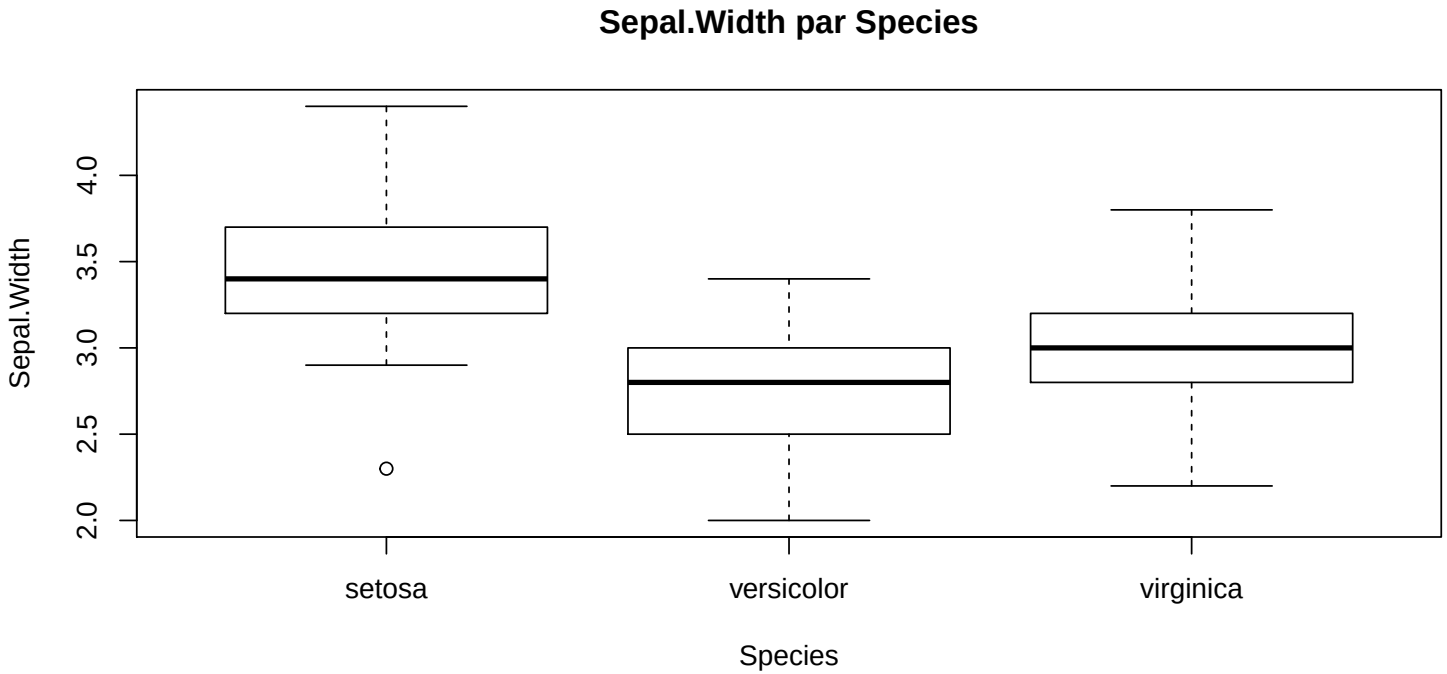
```
> data("iris")
> summary(iris)

> plot(Sepal.Width~Sepal.Length,data=iris,
+      main="Sepal.Length X Sepal.Width")
```



plot: variable quantitative sur des groupes

Pour représenter **une variable quantitative en fonction d'une variable qualitative**:

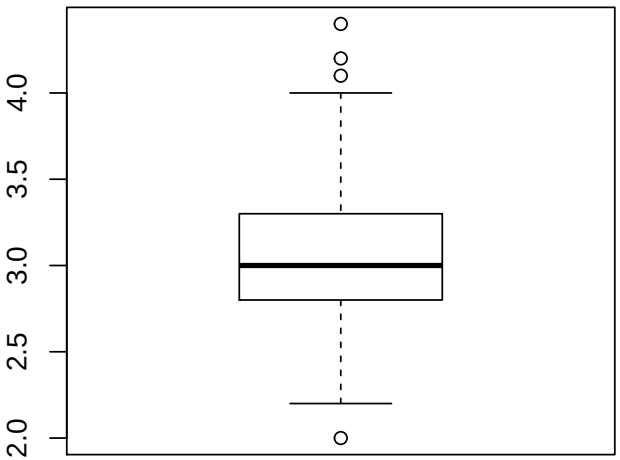
```
> plot(Sepal.Width~Species,data=iris,
+      main="Sepal.Width par Species")
```

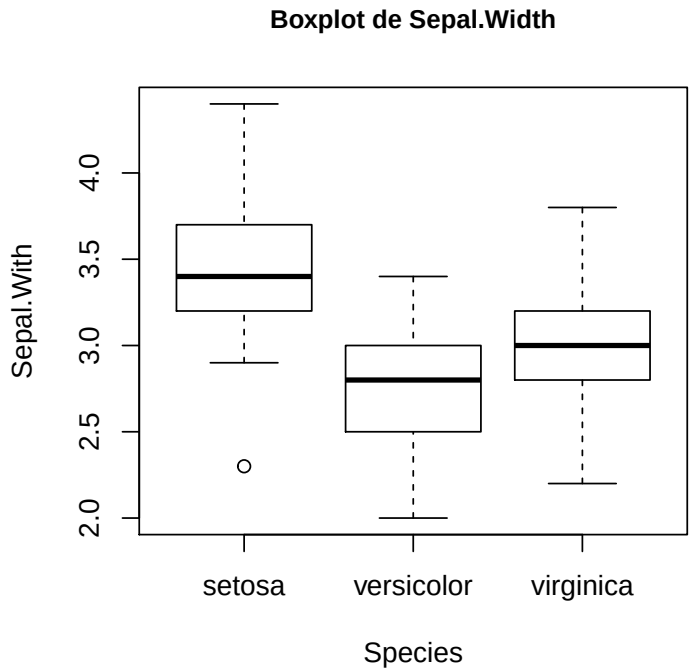


La fonction **plot** retourne des **boxplot** par modalité de la variable qualitative.

La fonction boxplot

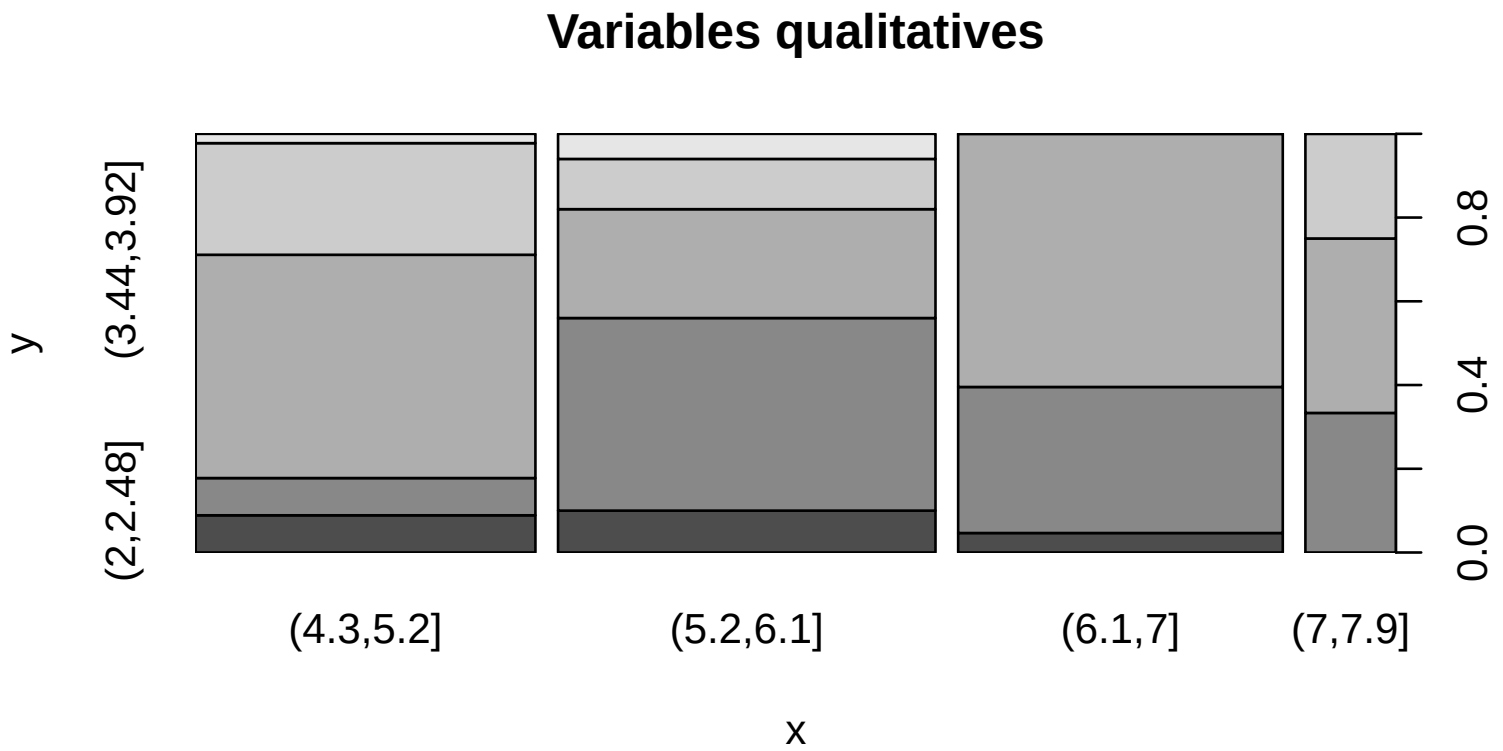
```
> par(mfrow=c(1,2))
> par(cex.main=0.9)
> boxplot(iris$Sepal.Width)
> boxplot(iris$Sepal.Width~iris$Species,
+         xlab="Species",ylab="Sepal.With",
+         main="Boxplot de Sepal.Width")
```



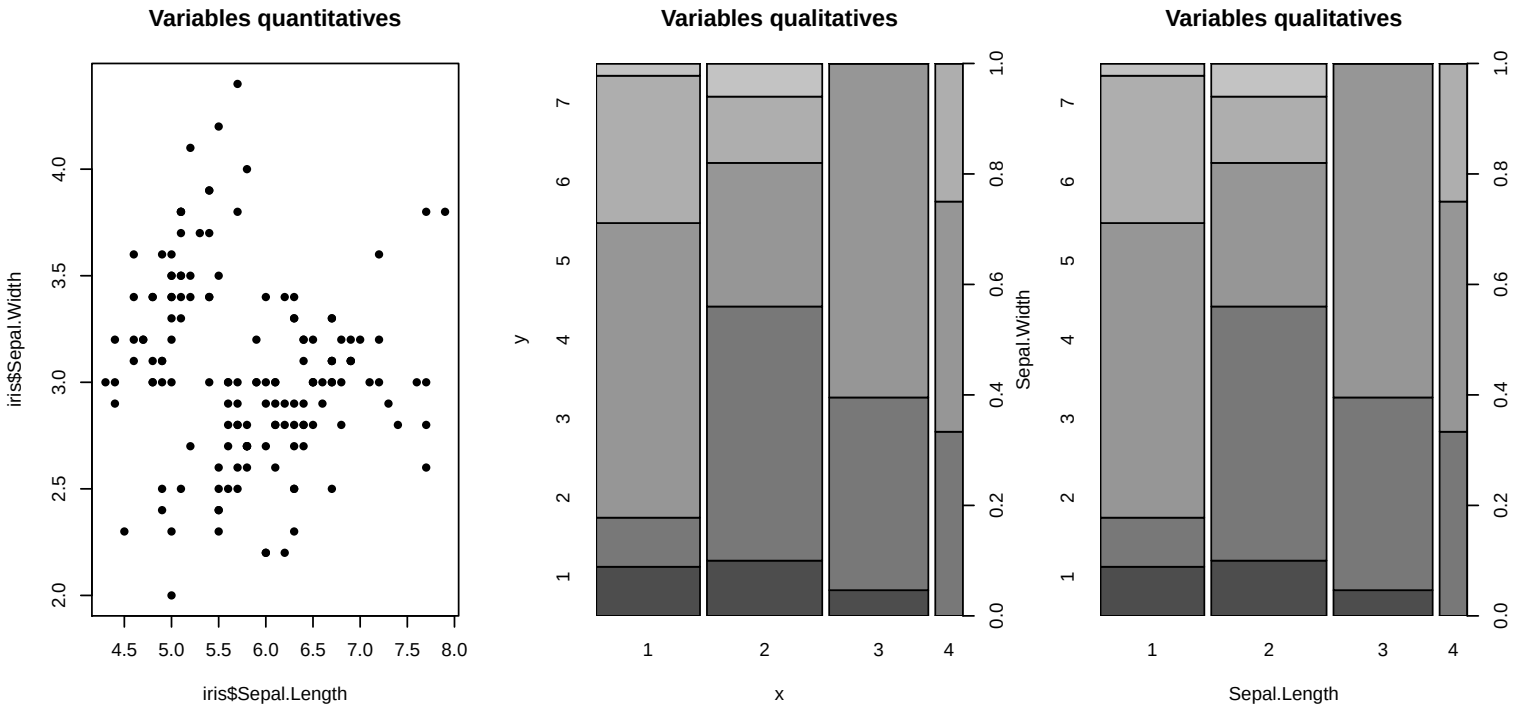


plot: deux variables qualitatives

```
> Sepal.Length <- cut(iris$Sepal.Length,4) # découpe en classes
> Sepal.Width <- cut(iris$Sepal.Width,5)
> plot(Sepal.Length,Sepal.Width,main="Variables qualitatives")
> levels(Sepal.Length) <- 1:4
> levels(Sepal.Width) <- 1:7
```



```
> par(mfrow=c(1,3))
> plot(iris$Sepal.Length,iris$Sepal.Width,main="Variables quantitati
> plot(Sepal.Length,Sepal.Width,main="Variables qualitatives")
> plot(Sepal.Length,Sepal.Width,main="Variables qualitatives",
+      xlab="Sepal.Length",ylab="Sepal.Width")
```

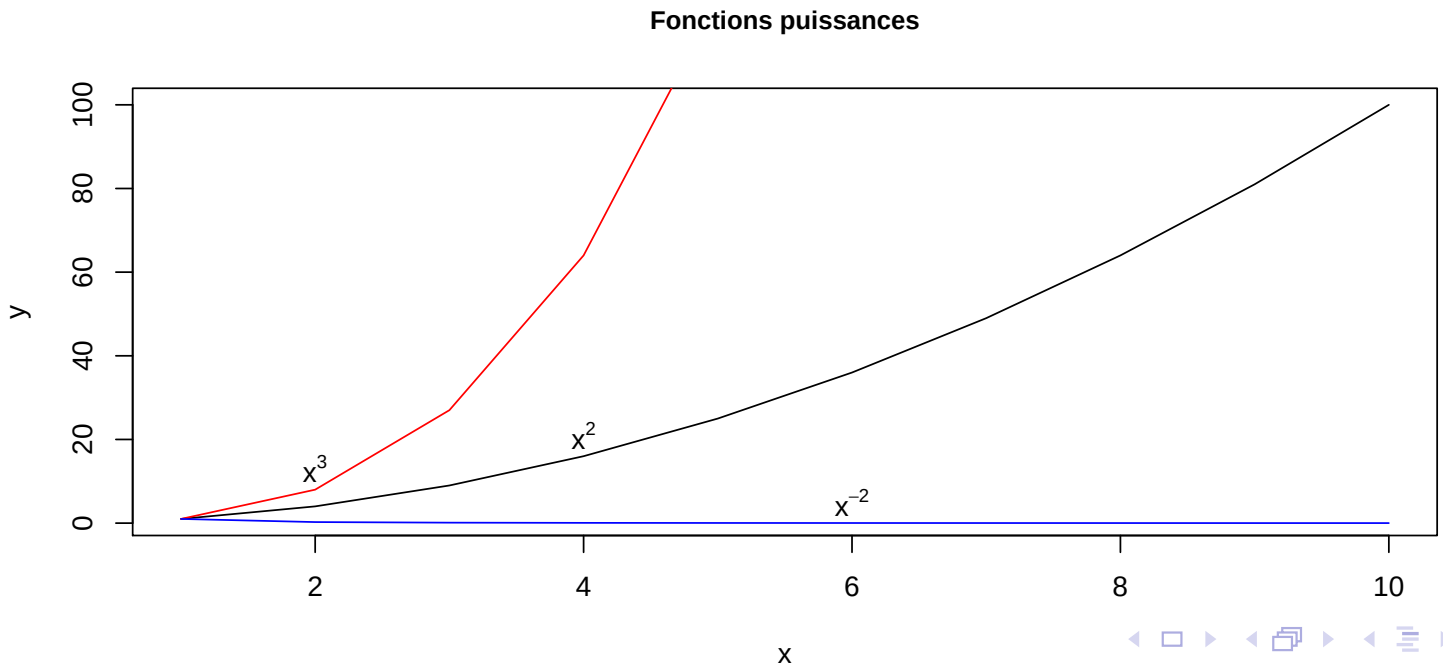


Sur des variables qualitatives, ces graphiques peuvent être obtenues avec la fonction **spineplot**

Une fois le graphique tracé, on peut le compléter par d'autres informations:

- **ajout de lignes** avec la fonction **lines**
- **ajout de text** avec la fonction **text**

```
> x<- 1:10;y <-x^2;z <-x^3;w<-x^(-2)
> plot(x,y,type="l",main="Fonctions puissances",cex.main=0.9)
> text(4,y[4]+5,expression(x^2))
> lines(x,z,col="red")
> text(2,z[2]+5,expression(x^3))
> lines(x,w,col="blue")
> text(6,w[6]+5,expression(x^-2))
```



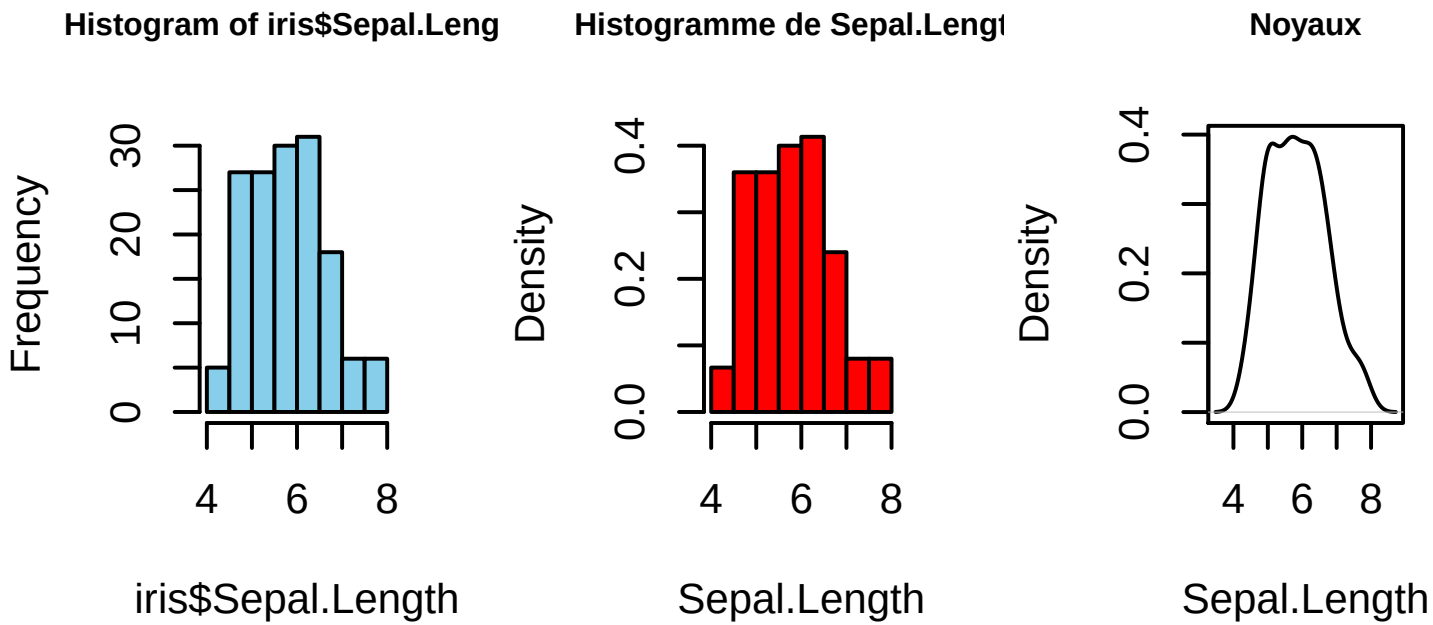
Ajout à un graphique

Une fois le graphique tracé, on peut le compléter par d'autres informations:

- **ajout de points** par la fonction **point**
- **ajout de flèches** par la fonction **arrow**
- **ajout de segments** par la fonction **segments**
- **ajout de polygones** par la fonction **polygone**

Histogramme avec la fonction **hist**

```
> par(mfrow=c(1,3))
> hist(iris$Sepal.Length,col="skyblue",cex.main=0.8)
> hist(iris$Sepal.Length,col="red",main="Histogramme de Sepal.Length
+      prob=TRUE,xlab="Sepal.Length",cex.main=0.8)
> plot(density(iris$Sepal.Length),cex.main=0.8,main="Noyaux",
+      xlab="Sepal.Length")
```

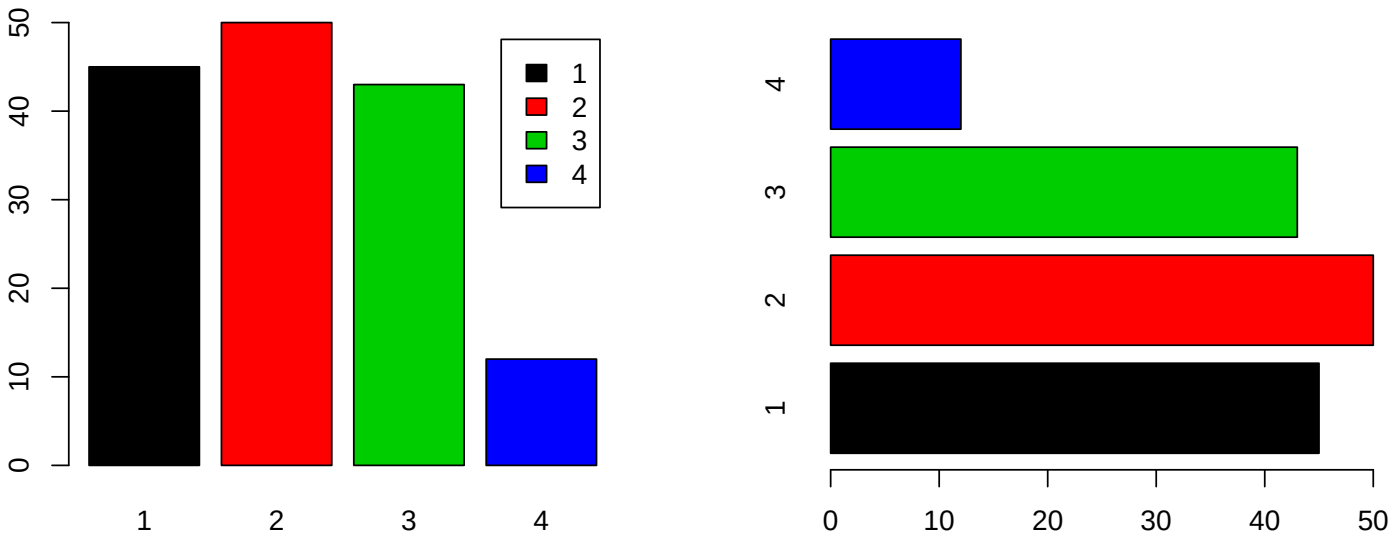


La liste des couleurs disponibles est fournie par la commande **color()**

barplot

Sepal.Length est de class **factor**: iris$Sepal.Length découpée en classes.

```
> par(mfrow=c(1,2))
> barplot(table(Sepal.Length),col=1:nlevels(Sepal.Length),
+         legend=TRUE)
> barplot(table(Sepal.Length),col=1:nlevels(Sepal.Length),
+         horiz=TRUE)
```



pie: diagramme circulaire (camembert)

Sepal.Length est de class **factor**: iris$Sepal.Length découpée en classes.

```
> Sepal.Length <- cut(iris$Sepal.Length,4)
> pie(table(Sepal.Length),col=1:nlevels(Sepal.Length),radius = 1.)
```

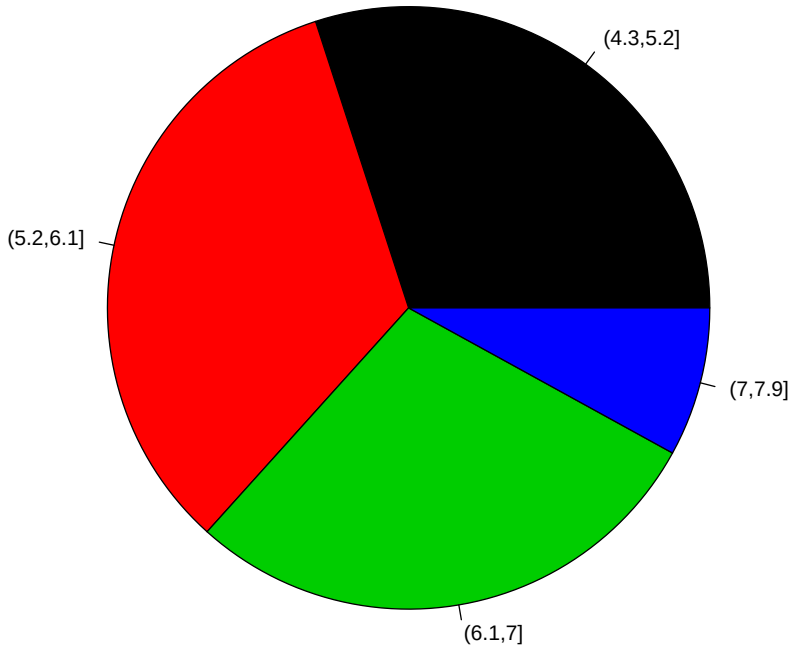
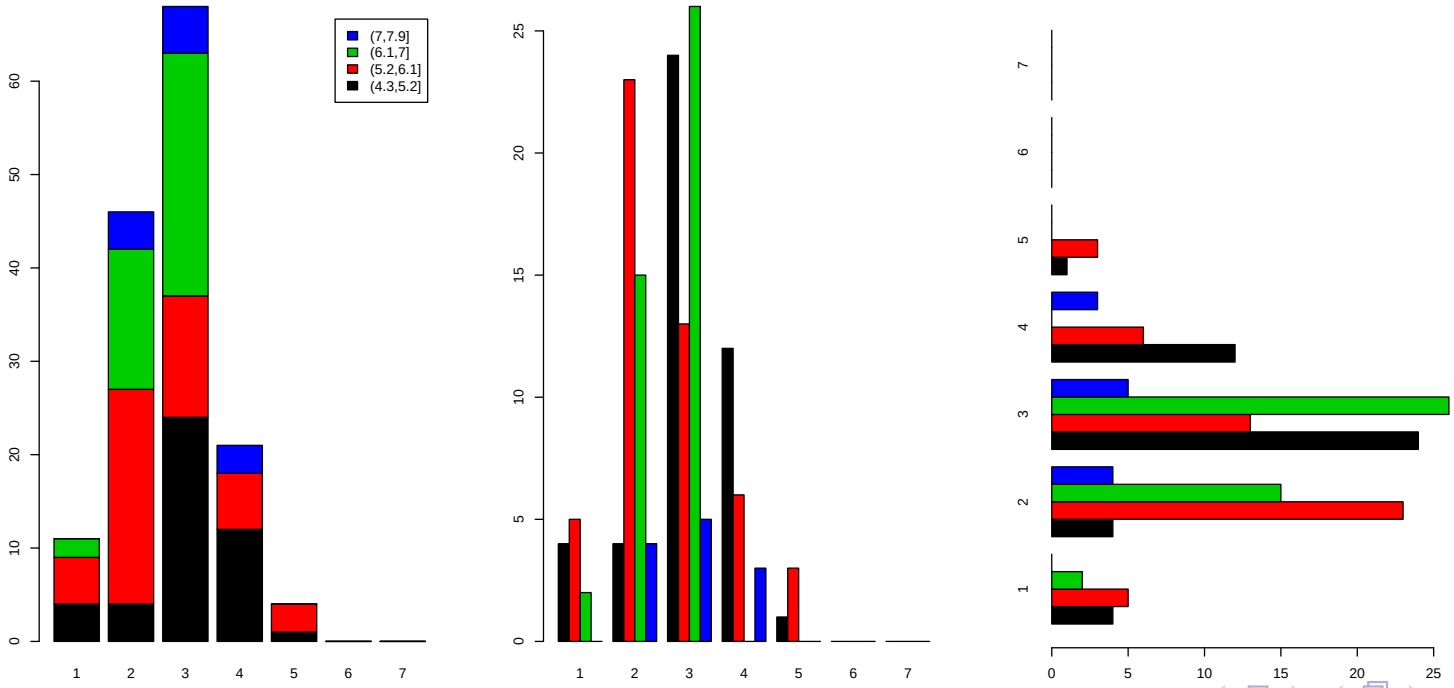


Diagramme en barre par groupe

```
> par(mfrow=c(1,3))
> barplot(table(Sepal.Length,Sepal.Width),
+         col=1:nlevels(Sepal.Length),legend=TRUE)
> barplot(table(Sepal.Length,Sepal.Width),
+         col=1:nlevels(Sepal.Length),beside=TRUE)
> barplot(table(Sepal.Length,Sepal.Width),
+         col=1:nlevels(Sepal.Length),beside=TRUE,horiz=TRUE)
```



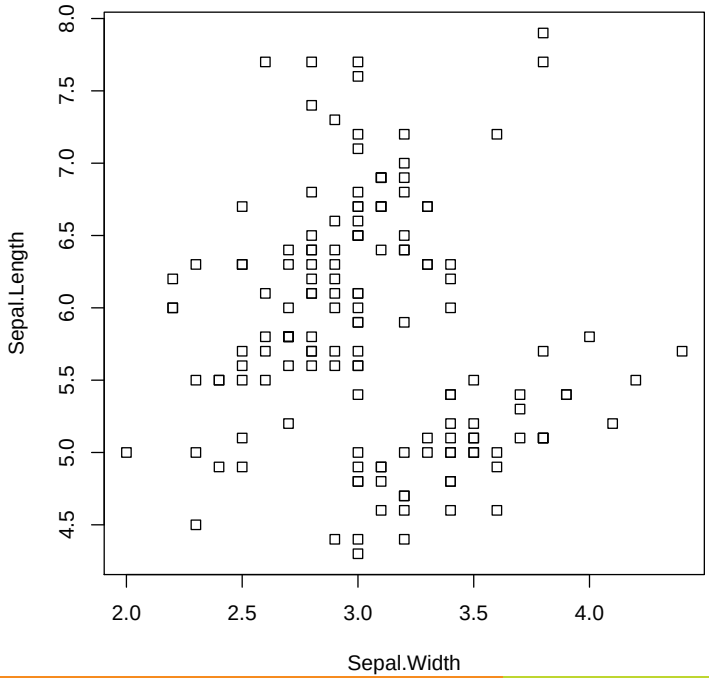
Plusieurs graphiques dans la même fenêtre

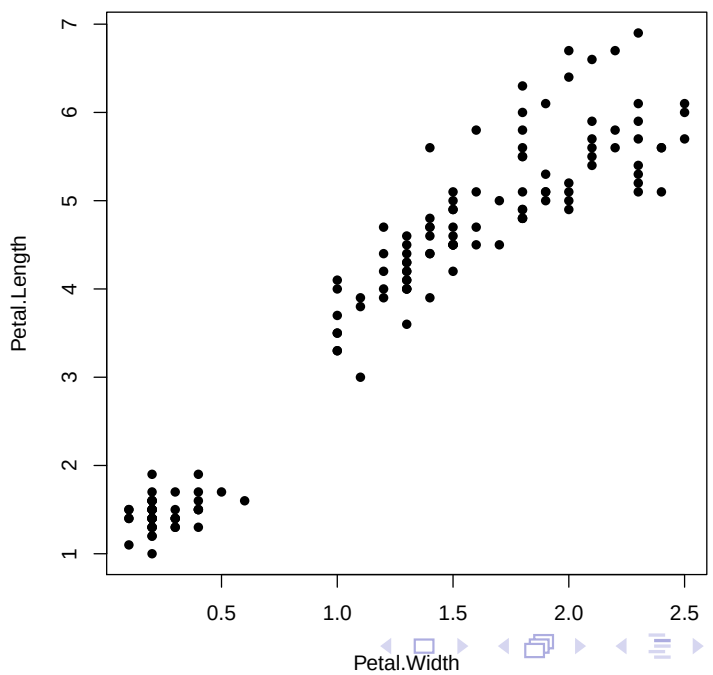
On veut faire figurer plusieurs graphiques dans une même fenêtre.

- Pour des **graphiques de même taille**: on utilise la fonction **par**.

  L'instruction **par**(**mfrow**=c(n,p)) organise **np** graphiques en **n** lignes et **p** colonnes.

```
> par(mfrow=c(1,2))
> plot(Sepal.Length~Sepal.Width,data=iris,pch=0)
> plot(Petal.Length~Petal.Width,data=iris,pch=16)
```





Plusieurs graphiques dans la même fenêtre

Pour **des graphiques de taille différentes**, on utilisera la fonction **layout**.

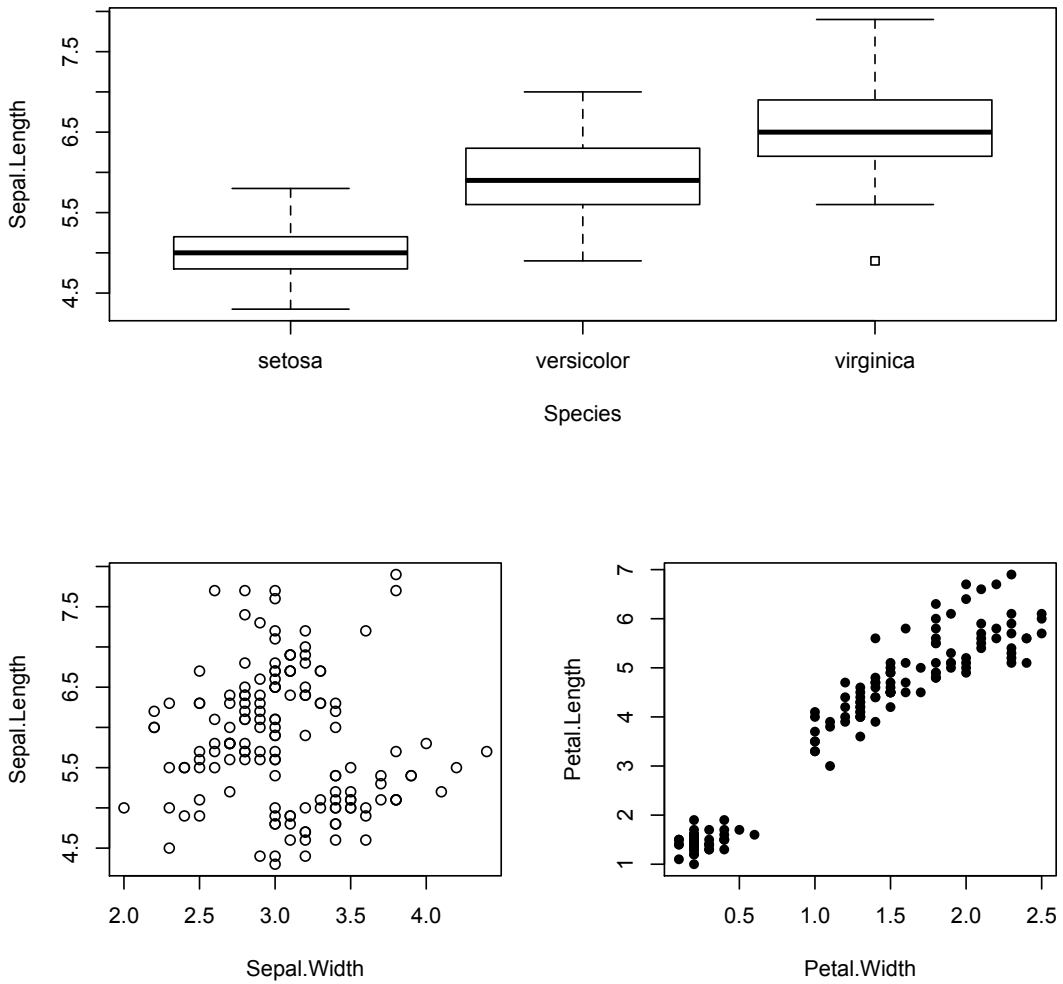
Elle admet comme argument, **une matrice de taille nrow x ncol**,les valeurs de la matrice correspondent aux numéros des graphiques qui doivent être dessinés dans chaque case.

Pour disposer trois graphiques sur deux lignes:

$$mat = \begin{pmatrix} 1 & 1 \\ 2 & 3 \end{pmatrix}$$

```
> mat <-matrix(c(1,2,1,3),nrow=2)
> layout(mat)
> plot(Sepal.Length~Species,data=iris,pch=0)
> plot(Sepal.Length~Sepal.Width,data=iris,pch=21)
> plot(Petal.Length~Petal.Width,data=iris,pch=16)
```

Trois graphiques sur deux lignes avec **layout**



Quelques paramètres pour améliorer un graphique

Certains paramètres sont modifiables **directement dans la commande graphique**, d'autres sont accessibles dans la fonction **par** qui **gère tous les paramètres graphiques du device**.

- Utiliser des couleurs avec **col="red"**, en utilisant des chiffres ou un code RGB.
- **contrôler l'aspect des axes et de leur label**

```
> x <- 1:10
> y <- x^2
> plot(x,y,type="l",axes=FALSE,xlab="",ylab="",
+      main=expression(f(x) == x^2),cex.main=0.9)
> axis(1,at=c(1,5,10),label=c("coord 1","coord 2","coord 3"),
+      cex.axis=0.8)
> axis(2,at=seq(1,100,length=10),cex.axis=0.8)
```

Quelques paramètres pour améliorer un graphique

- **ajouter une légende**

```
> ecarty <- range(iris[,"Sepal.Length"])
> plot(iris[1:75,"Sepal.Length"],type="l")
> lines(iris[76:150,"Sepal.Length"],
+       ylim=ecarty,col="red")
> legend("topleft",legend=c("sem1","sem2"),
+        col=c("black","red"),lty=1)
```

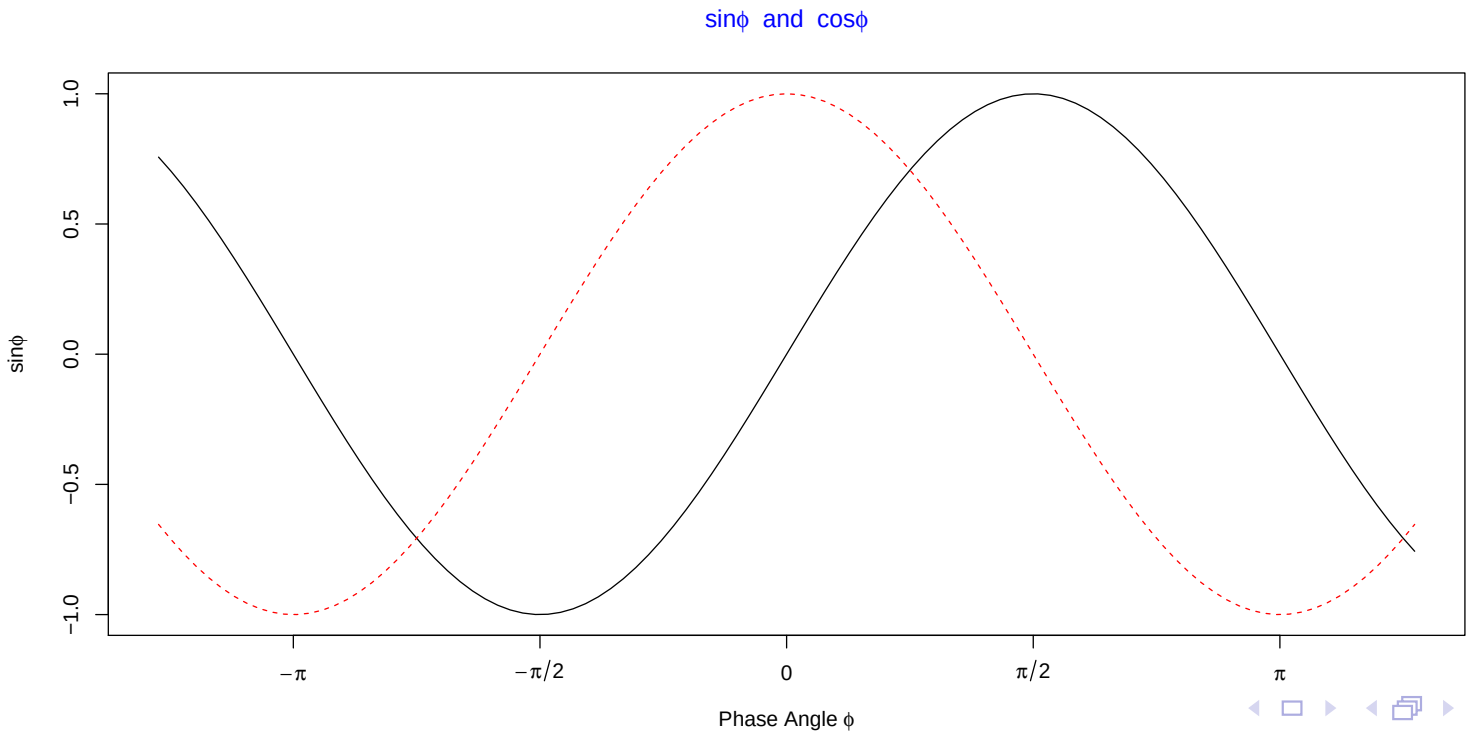
- **insérer des symboles ou des formules mathématiques**:

```
> plot(1,1,cex.lab=0.8,
+      xlab=expression(bar(x) == sum(frac(x[i],10),i==1,10)))
```

- ...

Pour connaître l'ensemble des paramètres qui permettent d'améliorer les graphiques, **par()** et **help(par)**

```
> x <- seq(-4, 4, len = 101)
> y <- cbind(sin(x), cos(x))
> matplot(x, y, type = "l", xaxt = "n",
+         main = expression(paste(plain(sin) * phi, "  and  ",
+                                 plain(cos) * phi)),
+         ylab = expression("sin" * phi, "cos" * phi), # only 1st is
+         xlab = expression(paste("Phase Angle ", phi)),
+         col.main = "blue")
> axis(1, at = c(-pi, -pi/2, 0, pi/2, pi),
+      labels = expression(-pi, -pi/2, 0, pi/2, pi))
```



Graphiques en plusieurs dimensions

Les fonctions de représentation 3D sur une grille de points sont les fonctions **persp**(3D avec effet de perspective), **contour** (lignes de niveau) et **image** (lignes de niveau avec effet de couleur).

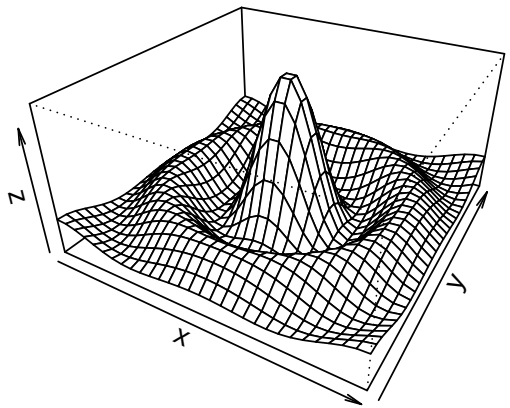
```
> f <- function(x,y) 10*sin(sqrt(x^2+y^2))/sqrt(x^2+y^2)
> x <-seq(-10,10,length=30) # maillage
> y <- x
```

Evaluons la fonction en chaque point de la grille avec la fonction **outer**.

```
> z <-outer(x,y,f)
```

Traçons la fonction 3D avec la fonction **persp**.

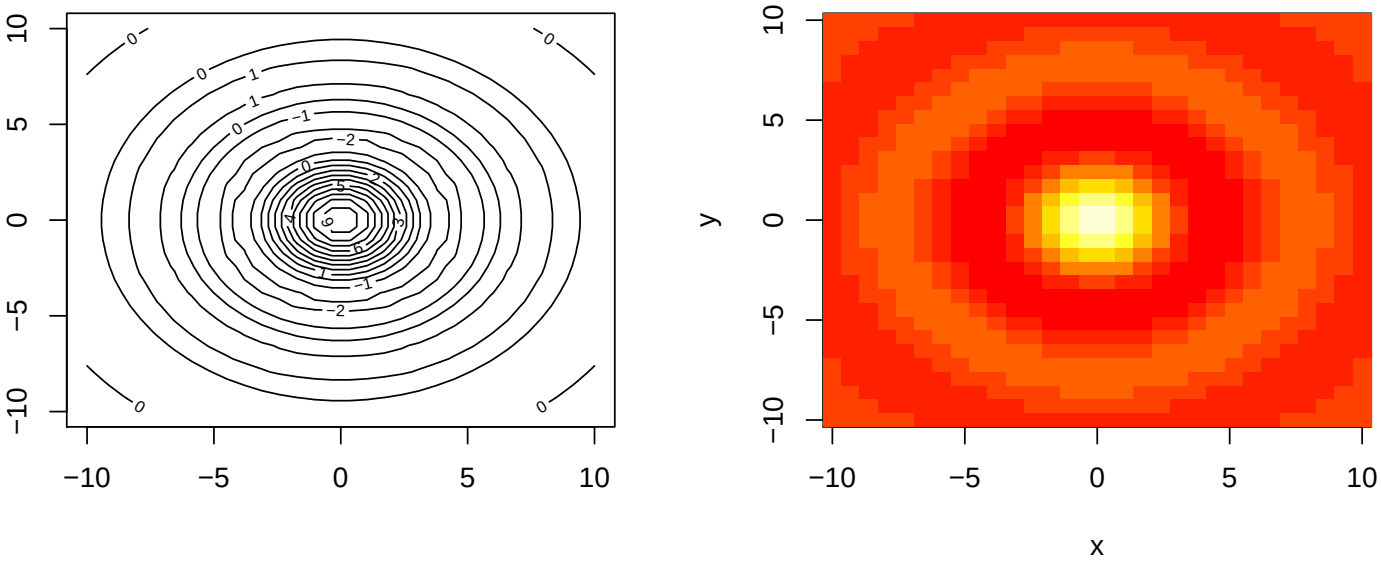
```
> persp(x,y,z,theta=30,phi=30,expand=0.5)
```



Graphiques en plusieurs dimensions

Nous pouvons obtenir les courbes de niveaux en utilisant **contour** (lignes de niveau) ou **image** (lignes de niveau avec effet de couleur).

```
> par(mfrow=c(1,2))
> contour(x,y,z)
> image(x,y,z)
```



Graphiques en plusieurs dimensions

Nous pouvons utiliser le package **rgl** pour construire la surface de réponse du graphique précédent.

```
> library(rgl)
> rgl.surface(x,y,z)
> plot3d(x,y,z)
```