

Les graphiques : introduction à la fonction plot

On représente facilement une fonction avec la fonction **plot()**, et deux vecteurs de même taille en entrée :

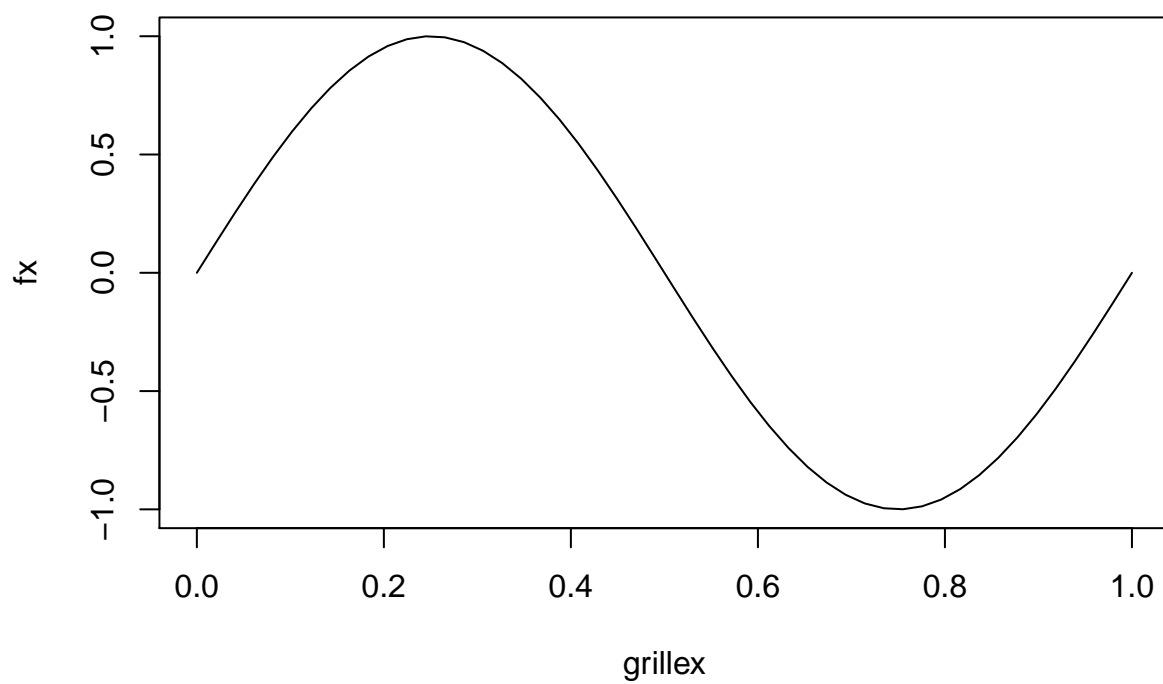
```
grillex <- seq(0, 1, length=50)
fx <- sin(2*pi*grillex)
plot(x = grillex, y = fx) # plot(fx-grillex) notation formule
```



Type de graphique : ligne ou points ?

l'argument **type** contrôle le type du graphique. Les plus courants sont : **p** (*points*), **l** (*ligne*), **b** (*ligne + points*), **s** (*step*),...

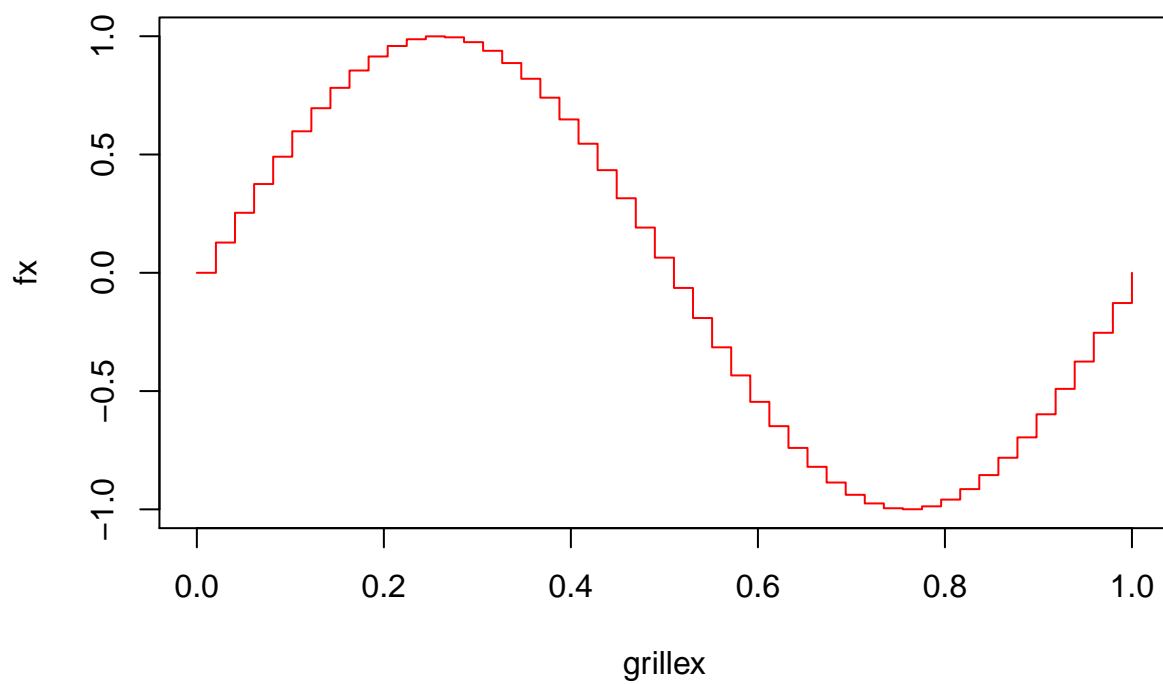
```
plot(x = grillex, y = fx, type = "l")
```



Les couleurs

En utilisant l'argument **col** :

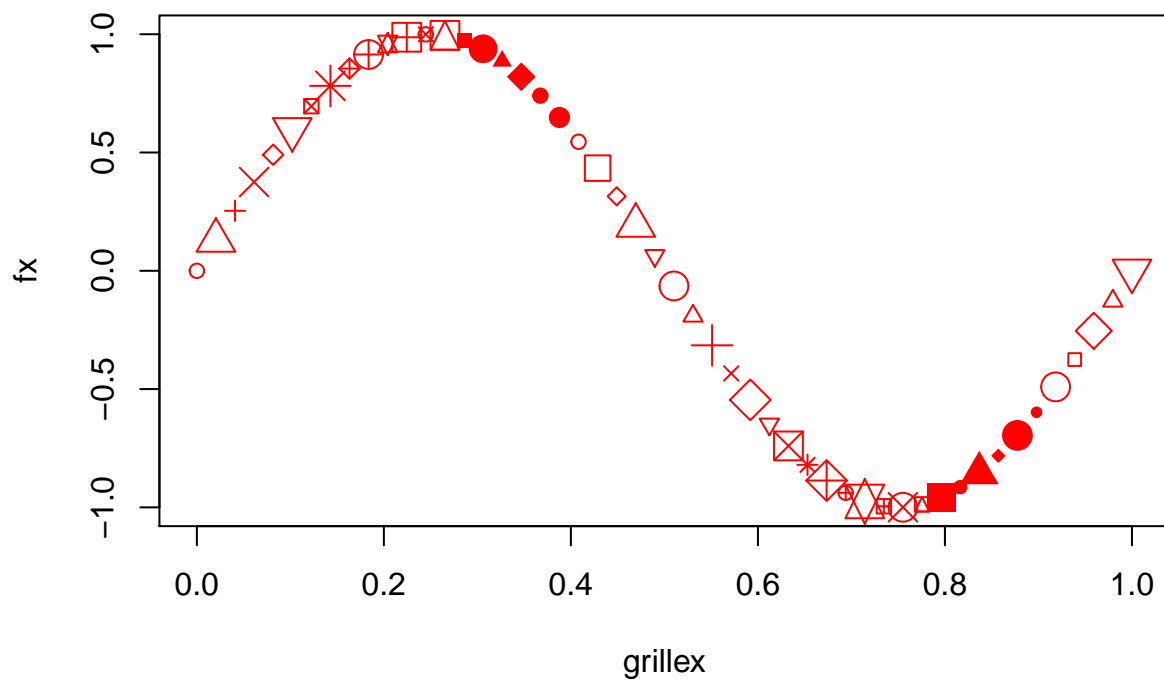
```
plot(x = grillex, y = fx, type = "s", col = "red")
```



Type de points

On gère le type des points avec **pch** (`__?points`). **cex** fixe la taille des points :

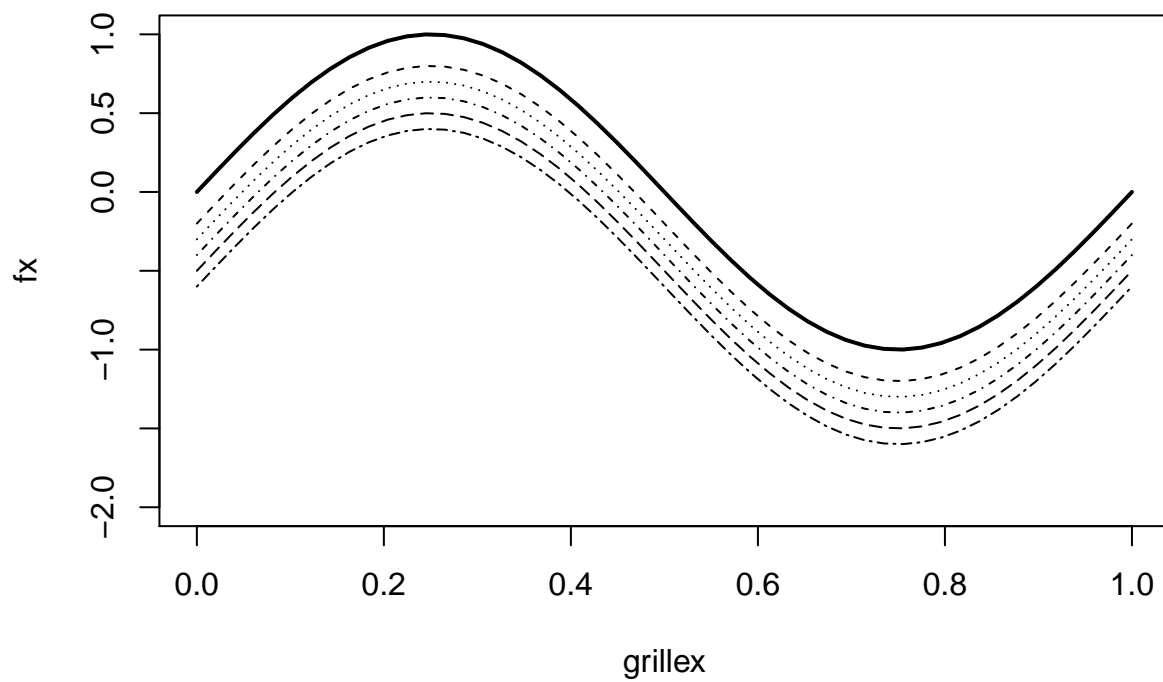
```
plot(x = grillex, y = fx, col = "red", pch = 1:25, cex = c(1:2))
```



Type de lignes

On gère le type des lignes avec **lty** (`_?par`). **lwd** fixe l'épaisseur :

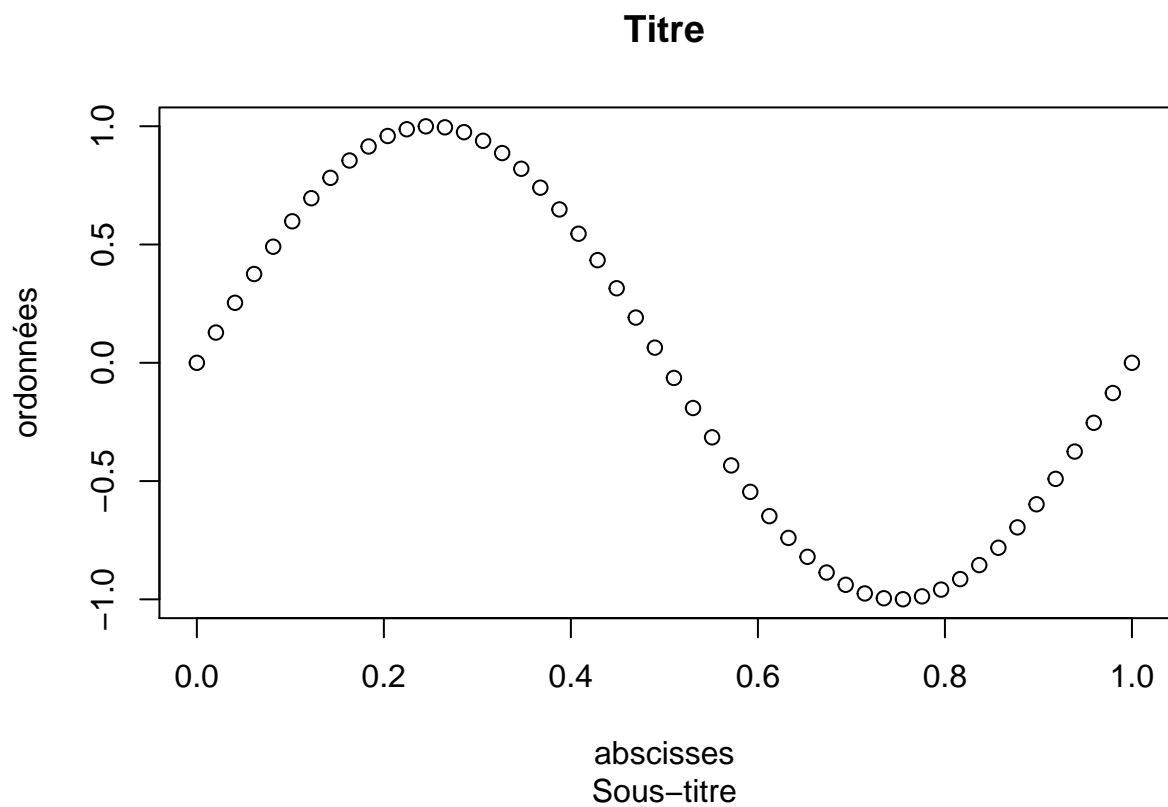
```
plot(x = grillex, y = fx, type = "l", lty = 1, lwd = 2, ylim = c(-2, 1))
for(i in 2:6) lines(x = grillex, y = fx - (i/10), lty = i)
```



Titre et intitulés des axes

main : titre, **sub** : sous-titre, **xlab** : intitulé des abscisses, **ylab** : intitulé des ordonnées

```
plot(x = grillex, y = fx, main = "Titre", sub = "Sous-titre",  
     xlab = "abscisses", ylab = "ordonnées")
```



Ajout de lignes

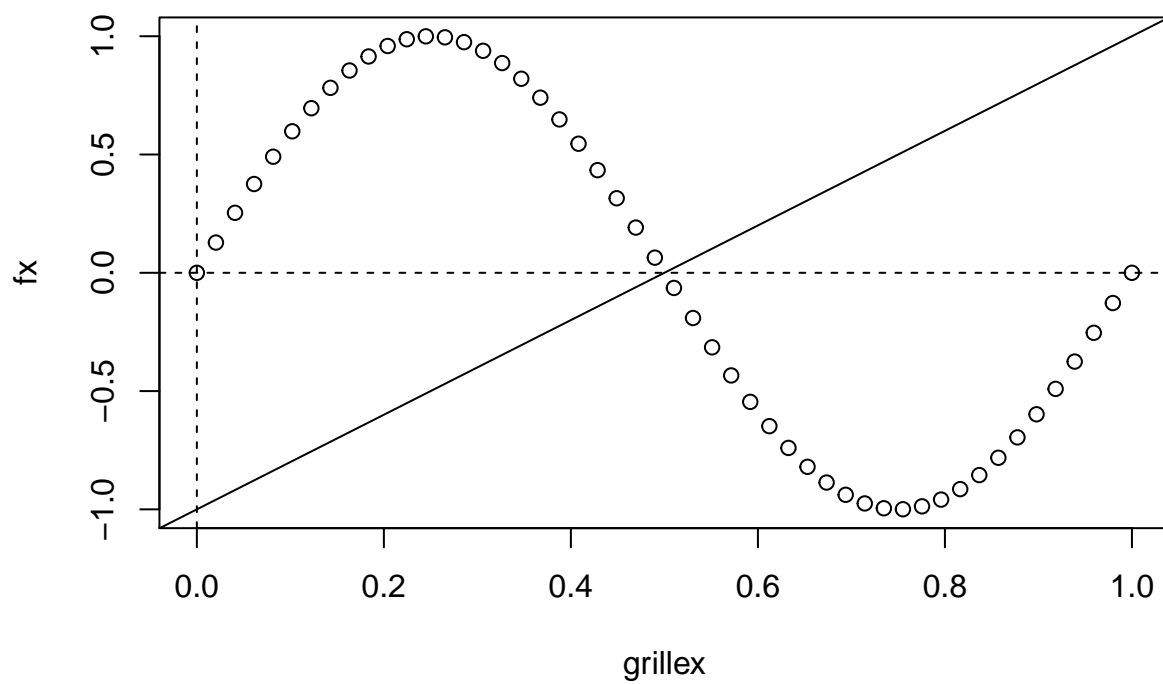
lines() : ajout d'une nouvelle ligne

```
plot(x = grillex, y = fx, ylim = c(-2, 1))  
lines(x = grillex, y = fx - 0.2)
```



abline() : ajout de droites (**h** horizontales, **v** verticales, **a** & **b** avec coefficients)

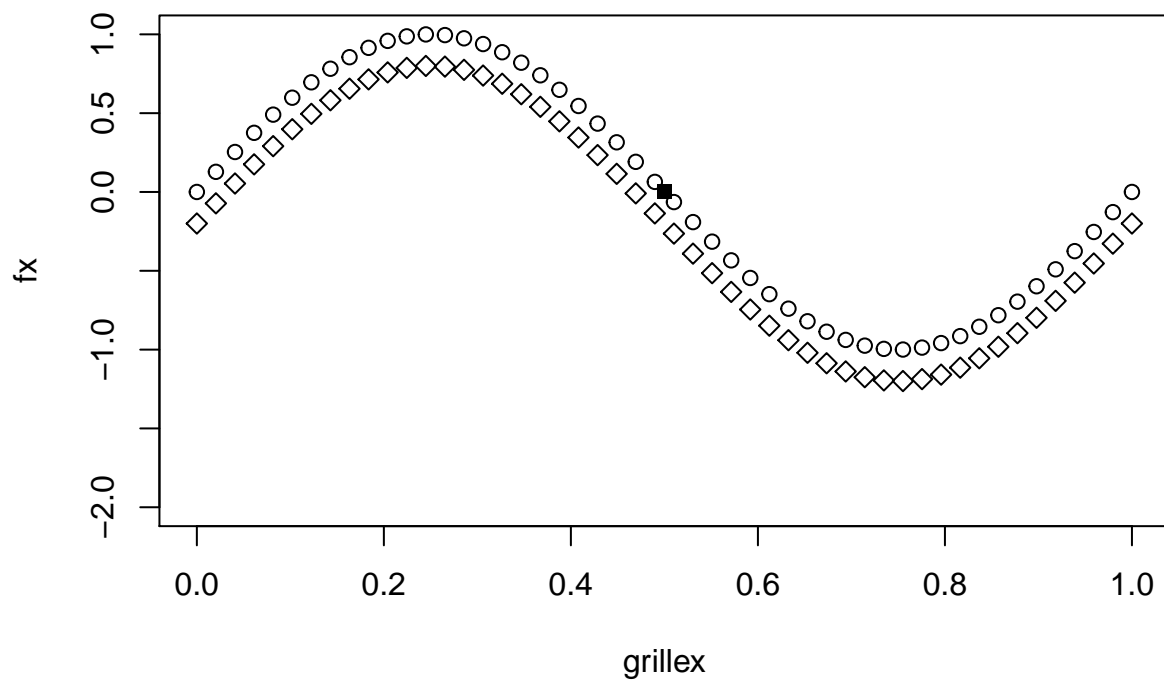
```
plot(x = grillex, y = fx)
abline(h = 0, lty = 2) ; abline(v = 0, lty = 2)
abline(a = -1, b = 2)
```



Ajout de points

fonction **points()**

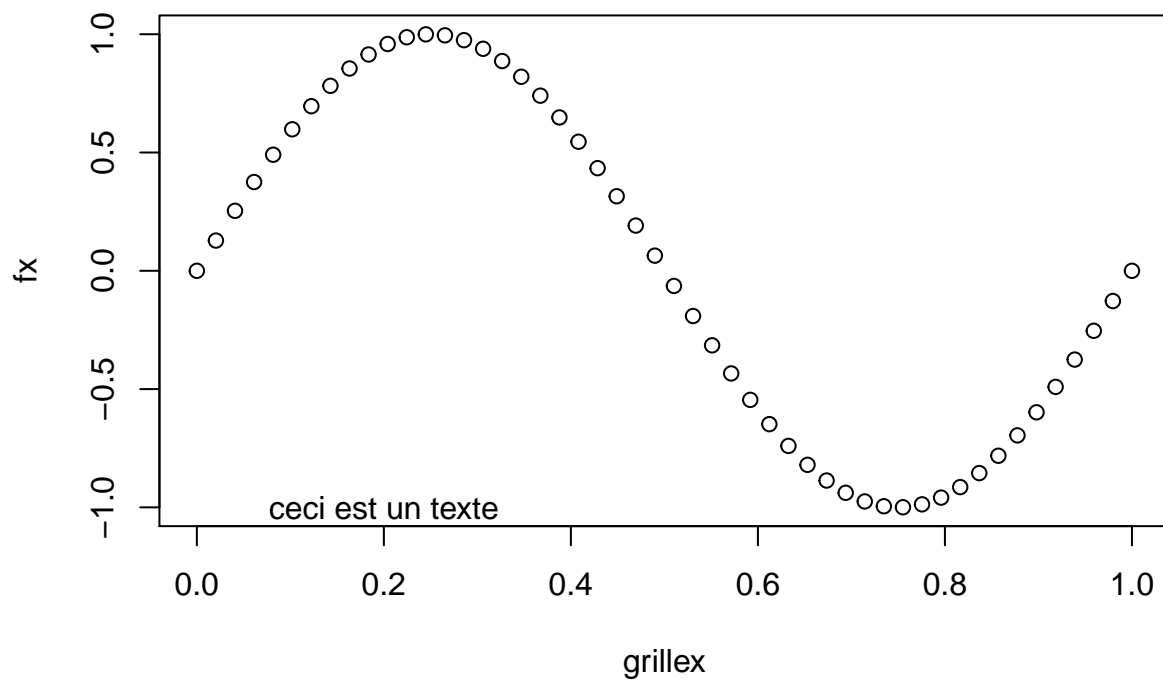
```
plot(x = grillex, y = fx, ylim = c(-2, 1))  
points(x = grillex, y = fx - 0.2, pch = 5)  
points(x = 0.5, y = 0, pch = 15)
```

Ajout de texte

En utilisant la fonction `text()`

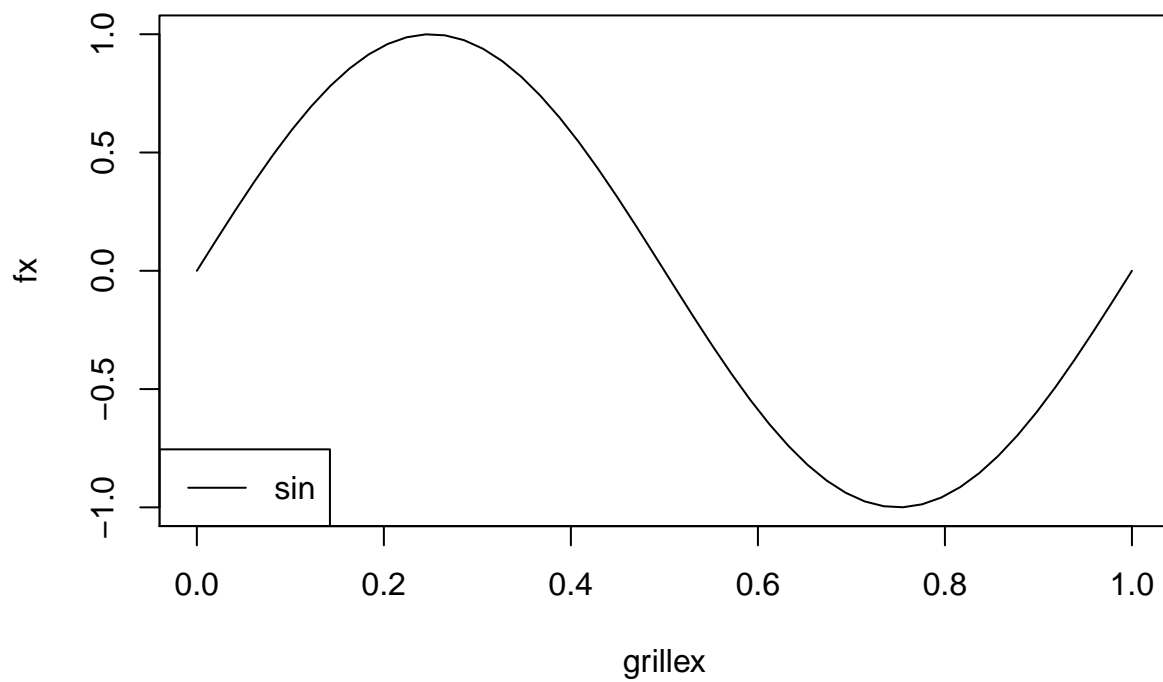
```
plot(x = grillex, y = fx)
text(x = 0.2, y = -1, labels = "ceci est un texte")
```



Ajout d'une légende

Simplement avec la fonction **legend()**

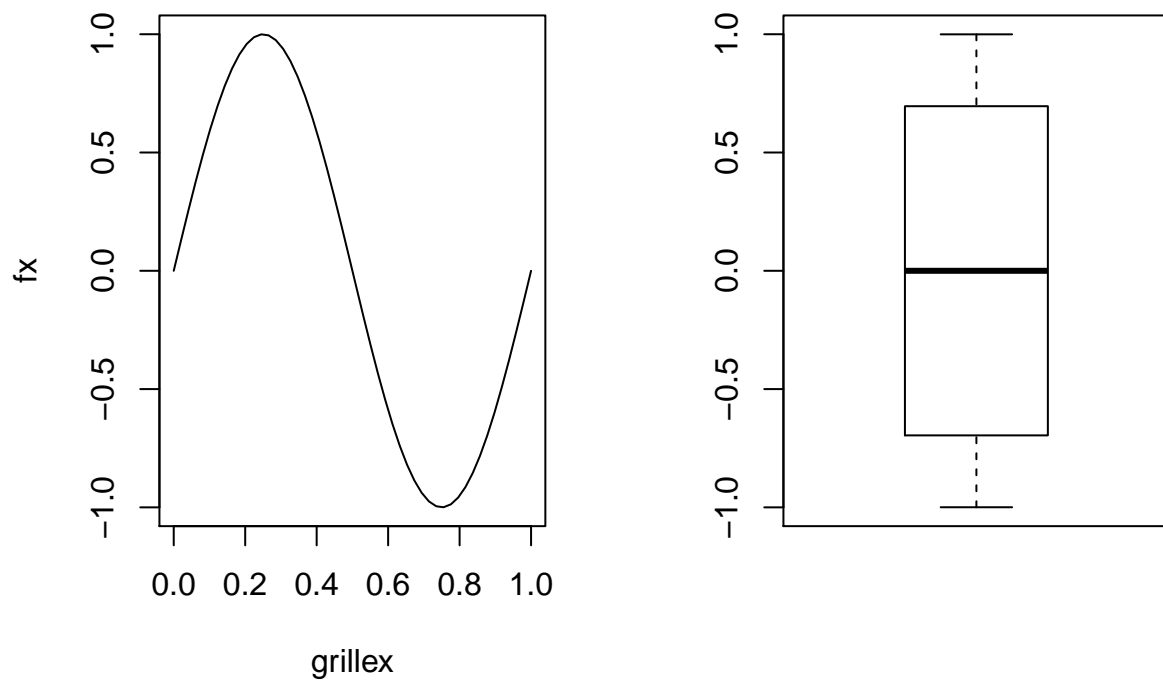
```
plot(x = grillex, y = fx, type = "l")  
legend("bottomleft", legend="sin", col=1, lty=1)
```



Plusieurs graphiques

Utilisation de la fonction **par()** et de l'argument **mfrow()**. Retour à une fenêtre unique : **dev.off()**

```
par(mfrow = c(1, 2))  
plot(x = grillex, y = fx, type = "l")  
boxplot(fx)
```



Les graphiques de statistique univariés

Les données d'exemples :

- deux variables quantitatives :
 - la température *T12*
 - l'ozone *maxO3*
- deux variables qualitatives / facteurs :
 - le vent
 - la pluie

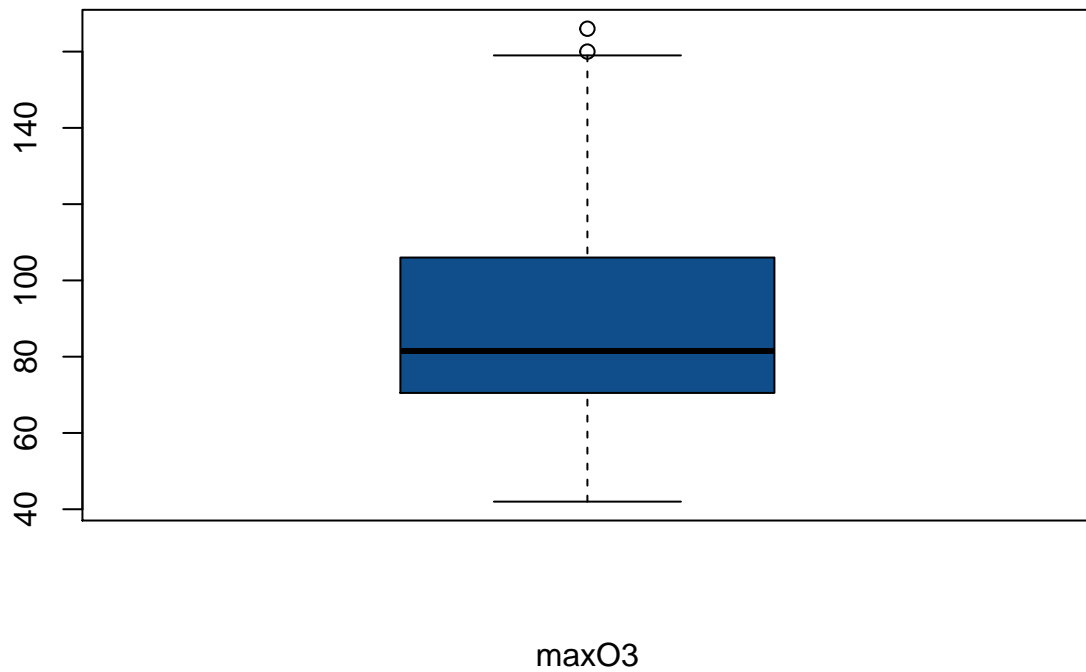
```
ozone <- read.csv("ozone_plot.txt", sep="")
summary(ozone)
```

##	T12	maxO3	vent	pluie
##	Min. :14.00	Min. : 42.00	Est :10	Pluie:43
##	1st Qu.:18.60	1st Qu.: 70.75	Nord :31	Sec :69
##	Median :20.55	Median : 81.50	Ouest:50	
##	Mean :21.53	Mean : 90.30	Sud :21	
##	3rd Qu.:23.55	3rd Qu.:106.00		
##	Max. :33.50	Max. :166.00		

Boxplot

Disponible avec la fonction `boxplot()`

```
boxplot(ozone$max03, col = "dodgerblue4", xlab = "max03")
```

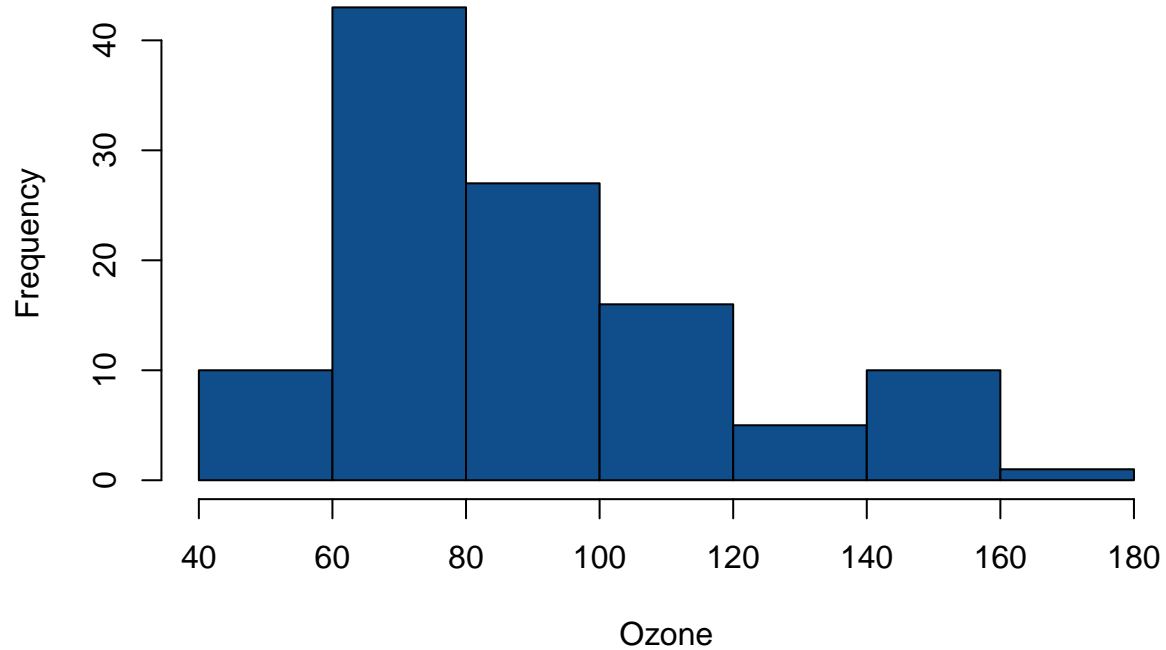


Histogramme

Disponible avec la fonction `hist()`

```
hist(ozone[, "max03"], main = "Histogramme", xlab = "Ozone", col = "dodgerblue4")
```

Histogramme



Ajout de l'estimateur à noyaux, avec la fonction `density()`

```
hist(ozone[, "T12"], prob = TRUE, col = "olivedrab3")  
lines(density(ozone[, "T12"]))
```

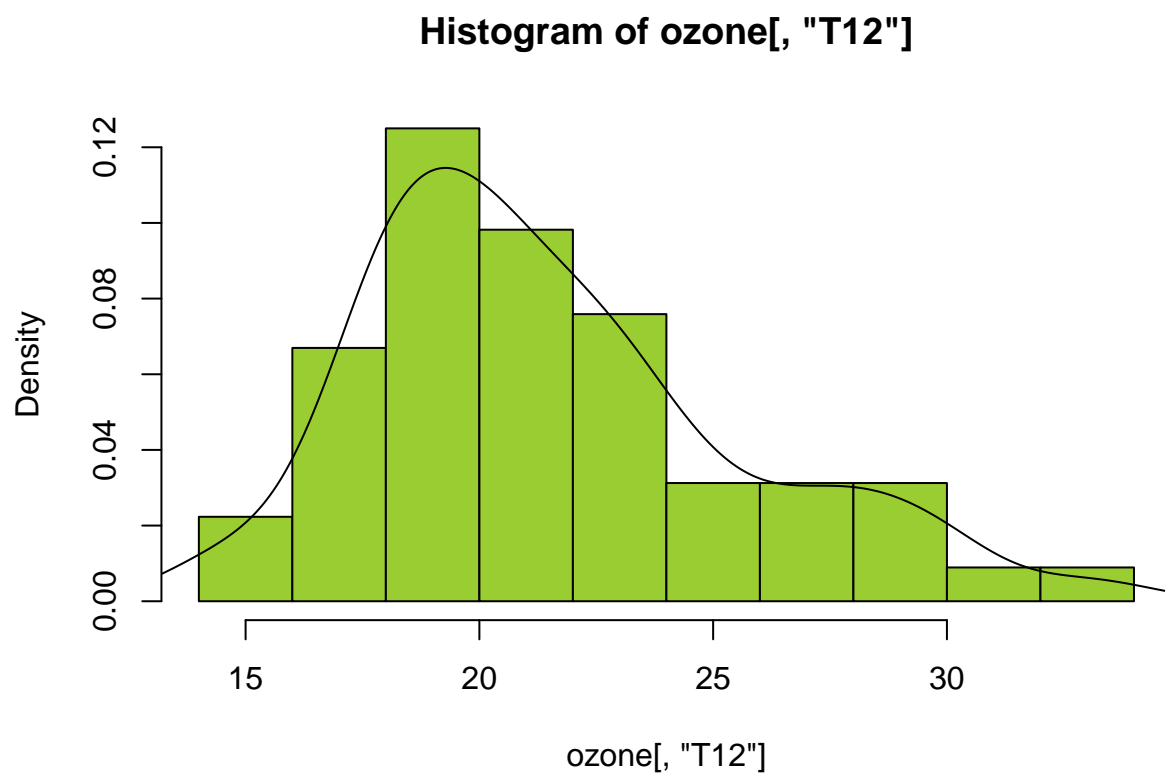
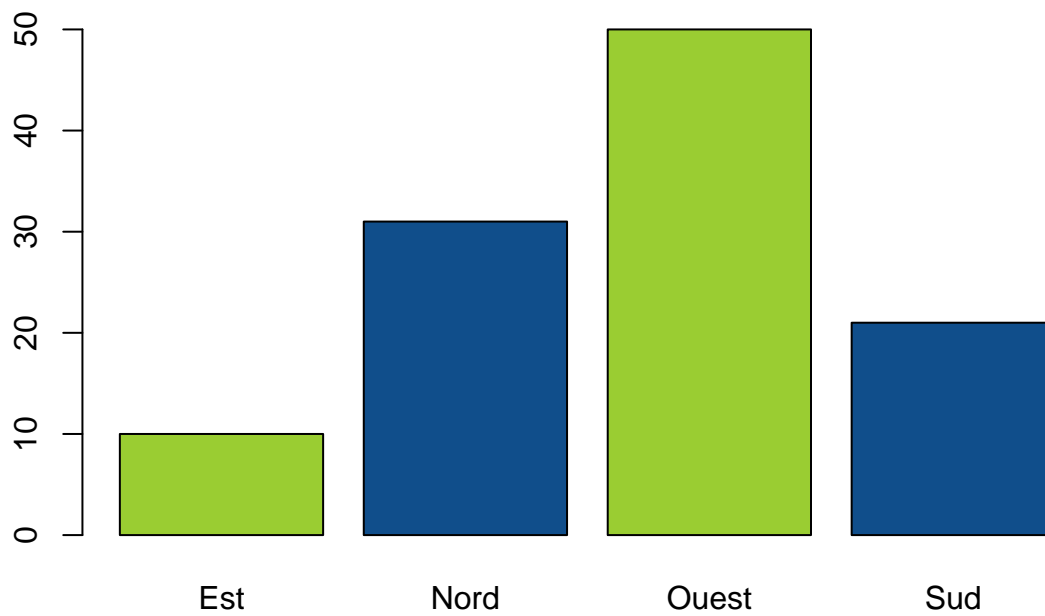


Diagramme en barres

Disponible avec la fonction **barplot()**

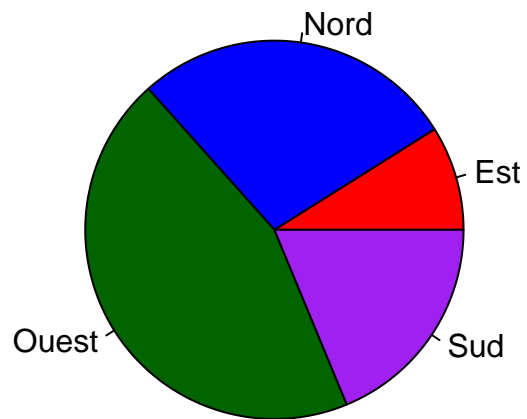
```
barplot(table(ozone[, "vent"]), col = c("olivedrab3", "dodgerblue4"))
```



Camenbert...

Disponible avec la fonction **pie()**

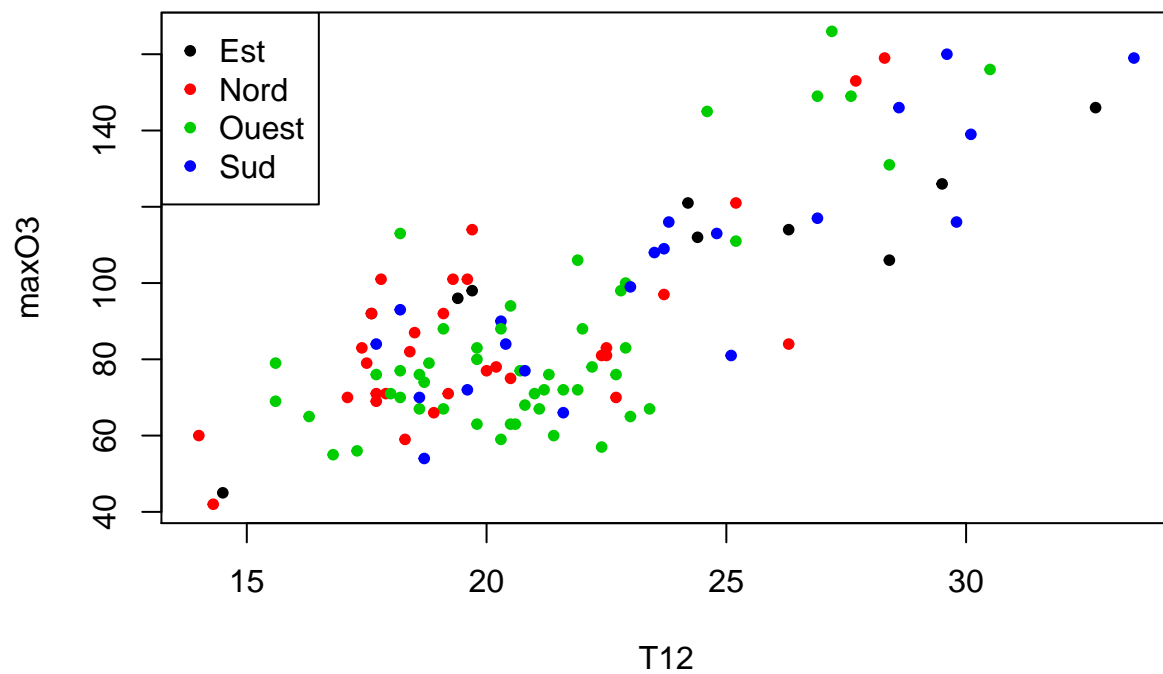
```
pie(table(ozone[, "vent"]), col = c("red", "blue", "darkgreen", "purple"))
```

Les graphiques de statistique bivariés

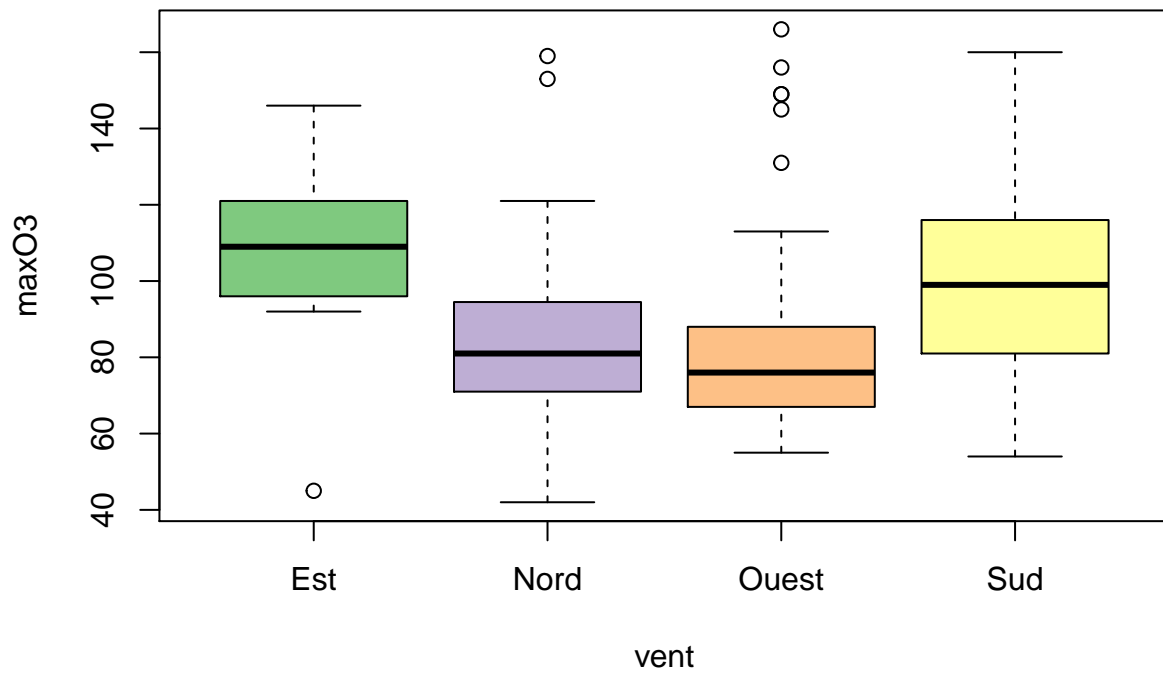
Quantitatif-Quantitatif

```
plot(max03-T12, data = ozone, col = ozone$vent, pch = 20)
legend("topleft", legend = levels(ozone$vent), col = 1:4, pch = rep(20, 4))
```



Quantitatif-Qualitatif

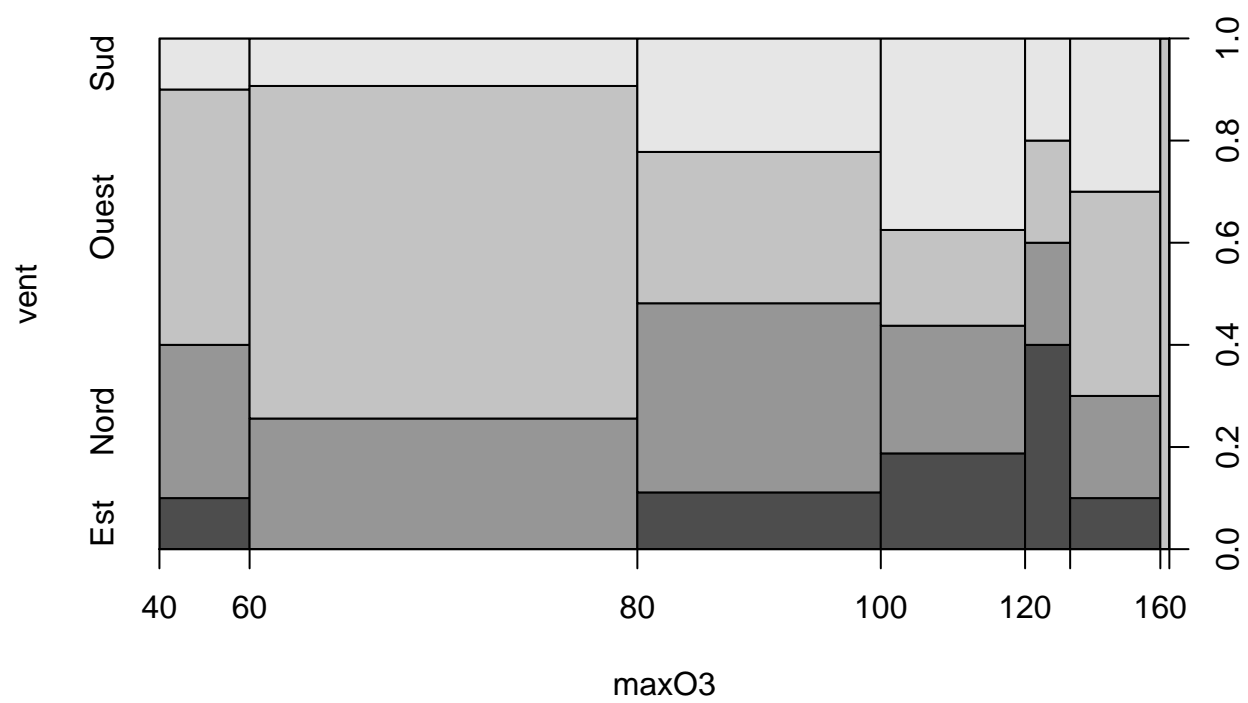
```
plot(maxO3~vent, data = ozone,  
     col = RColorBrewer::brewer.pal(4,"Accent"))
```



```
# (=) boxplot(maxO3~vent,data=ozone)
```

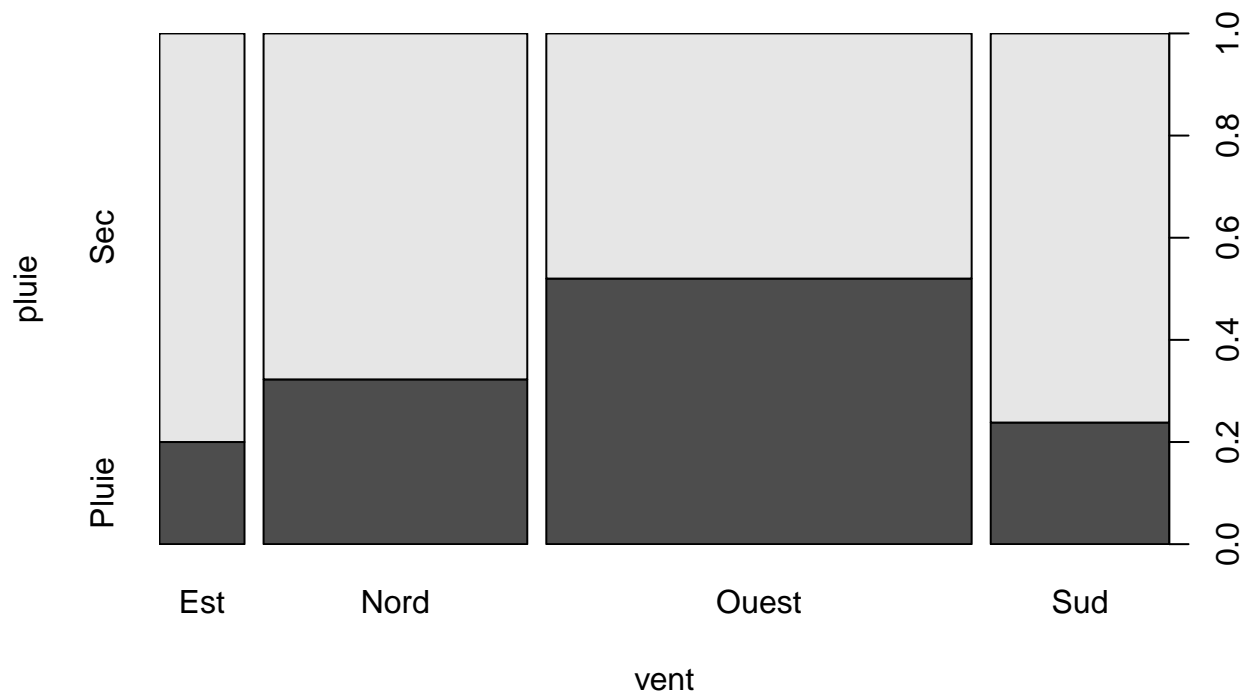
Et dans l'autre sens ? Peut-être un peu dur à lire...!

```
plot(vent~maxO3, data = ozone)
```



Qualitatif-Qualitatif

```
plot(pluie~vent, data = ozone)
```



Pour aller plus loin

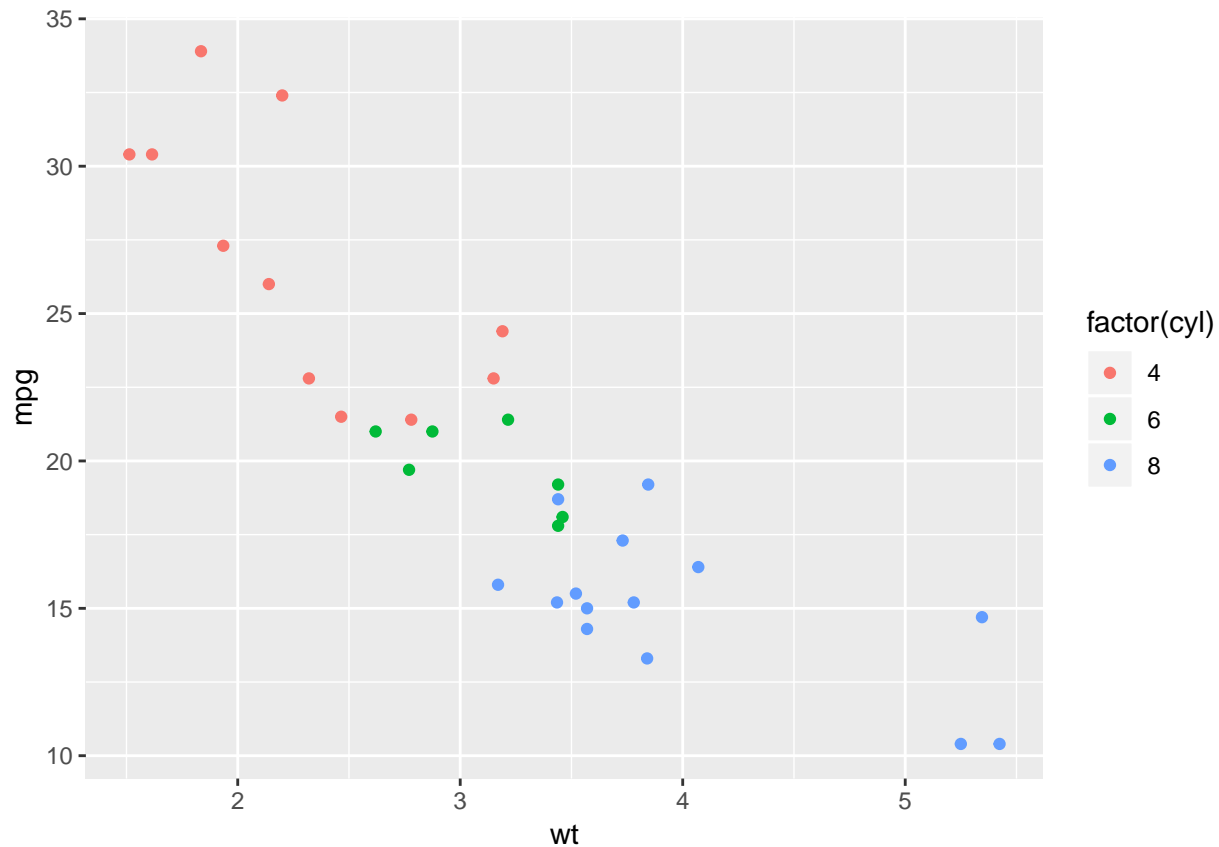
Démo

- `demo(image)`
- `example(contour)`
- `demo(persp)`
- `require("lattice"); demo(lattice)`
- `example(wireframe)`
- `require("rgl"); demo(rgl)`
- `example(persp3d)`
- `demo(plotmath); demo(Hershey)`

ggplot2

ggplot2 a été développé par Hadley Wickham comme une implémentation de *Grammar of Graphics*. C'est un package relativement complet et puissant. (<http://ggplot2.org/>, <http://docs.ggplot2.org/current/>, <https://www.rstudio.com/wp-content/uploads/2015/03/ggplot2-cheatsheet.pdf>)

```
ggplot(mtcars, aes(wt, mpg)) + geom_point(aes(colour = factor(cyl)))
```



ggplot2 : utilisation du addin esquisse

Le package **esquisse** propose une interface _clique bouton- permettant de réaliser des graphiques **ggplot2** pour ensuite les exporter et/ou récupérer le code R correspondant.

```
esquisse:::esquisser(viewer = "browser")
```

Graphiques interactifs

En utilisant des packages faisant le pont entre **R** et des librairies **javascript** :

- **htmlwidgets** (<http://www.htmlwidgets.org/>) et tous ses enfants (<http://gallery.htmlwidgets.org/>) :
 - **dygraphs** pour les séries temporelles : <http://rstudio.github.io/dygraphs>
 - **leaflet** pour les cartes : <http://rstudio.github.io/leaflet/>
 - **visNetwork** (<http://datastorm-open.github.io/visNetwork>) et **networkD3** (<http://christophergandrud.github.io/networkD3/>) pour les réseaux
 - **d3heatmap** pour des heatmaps : <https://github.com/rstudio/d3heatmap>
 - **DT** pour des tableaux interactifs : <http://rstudio.github.io/DT/>
 - **threejs** (<https://github.com/bwlewis/rthreejs>) et **rglwidget** (<http://cran.at.r-project.org/web/packages/rglwidget/index.html>) pour des représentations 3D
 - **plotly** pour passer du **ggplot2** en interactif : <https://plot.ly/r/>
 -



Figure 1:

Graphiques interactifs

- Visibles dans le **Viewer** de RStudio
- Facilement intégrables à des documents **Rmarkdown** et des applications **shiny**
- Exportables en .html

```
suppressWarnings(require(dygraphs, quietly = T))
lungDeaths <- cbind(mdeaths, fdeaths)
g <- dygraph(lungDeaths)
# affichage
g

# exportation
htmlwidgets::saveWidget(g, file = "ex_dygraph.html")
```

