Fonctions Graphiques avec R

Pr Roch Giorgi

roch.giorgi@univ-amu.fr

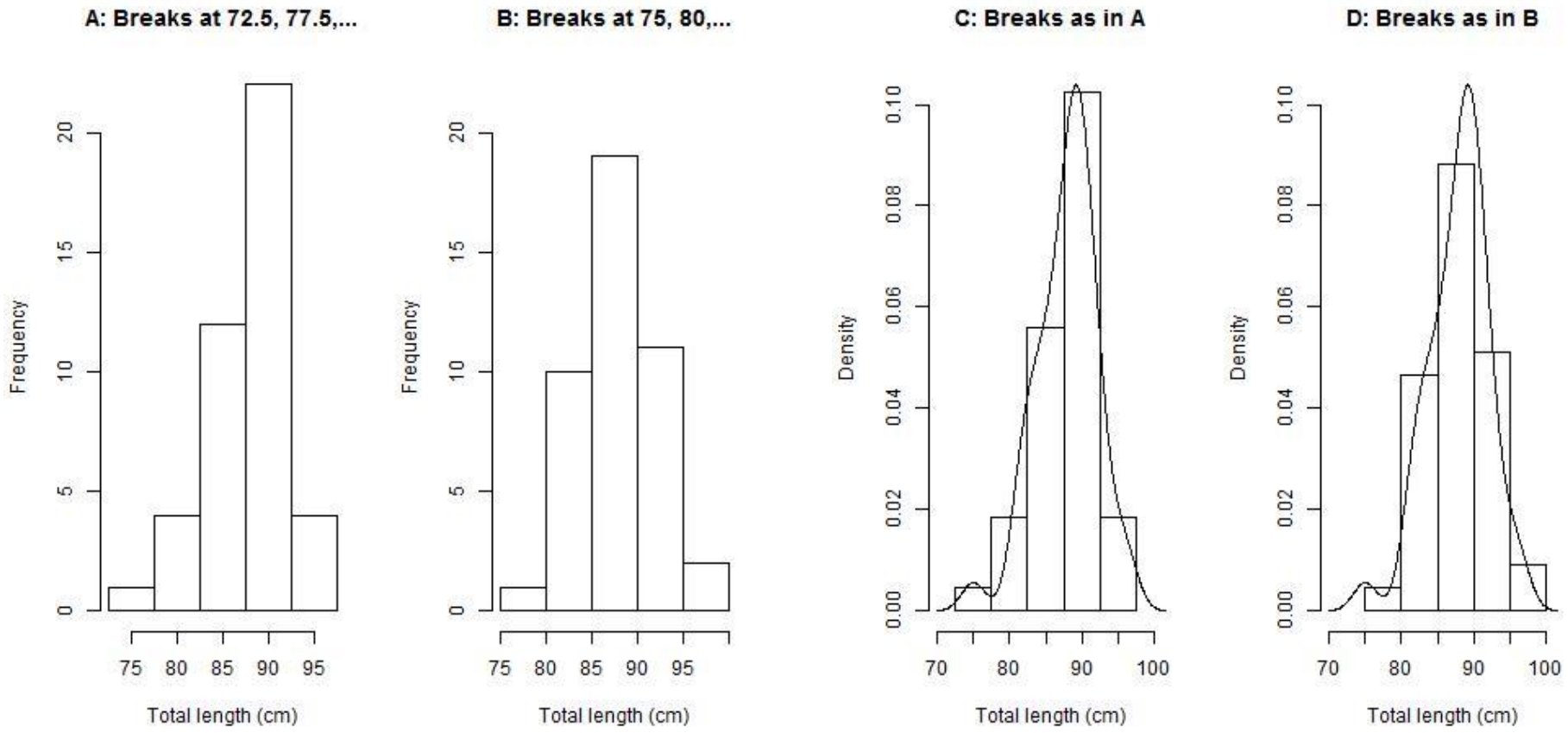
SESSTIM, Faculté des sciences médicales et paramédicale, Aix-Marseille Université, France
http://sesstim.univ-amu.fr/

Préambule (1)

- Ce diaporama présente, de manière non exhaustive, certaines fonctions graphiques de R

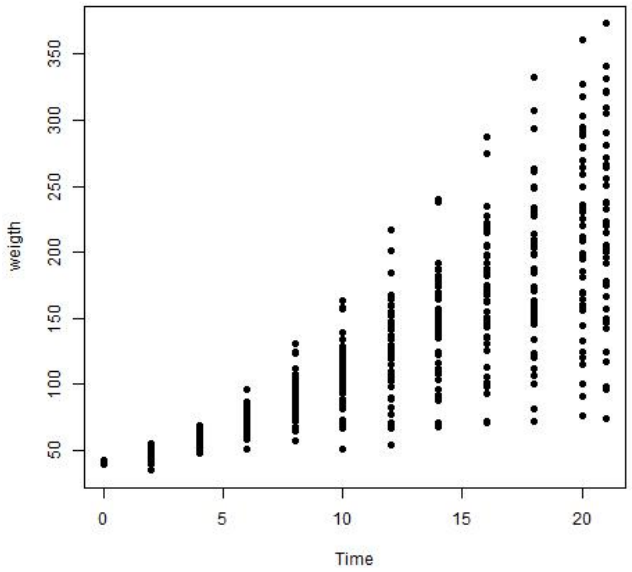
- Au-delà de la simple utilisation de fonctions, l'objectif est de faire comprendre l'intérêt de la production de graphiques adaptés lors de la réalisation d'une l'analyse statistique

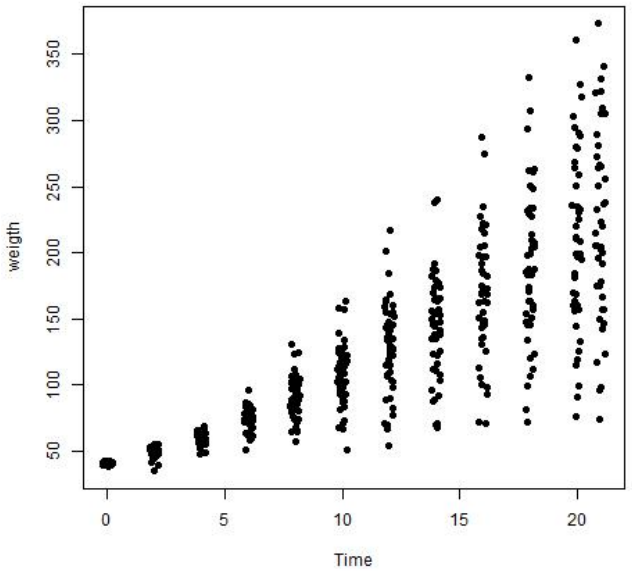
Préambule (2)



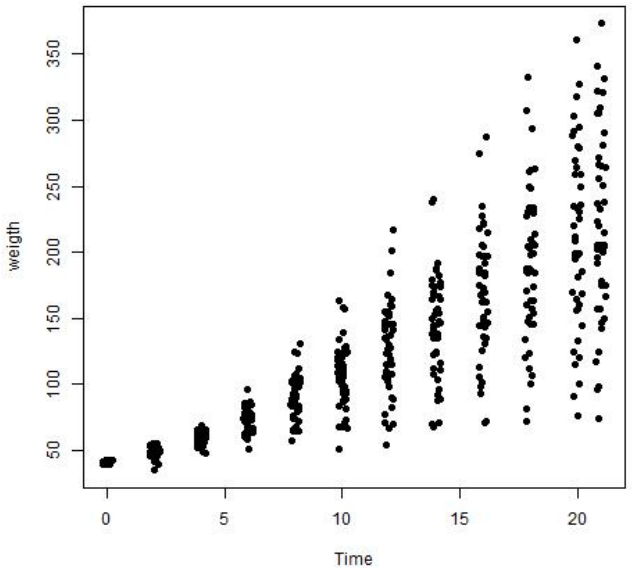
Source : Data Analysis and Graphics Using R: An Example-Based Approach. Cambridge Series in Statistical and Probabilistic Mathematics. JH Maindonald, WJ Braun. Third edition.

Préambule (3)





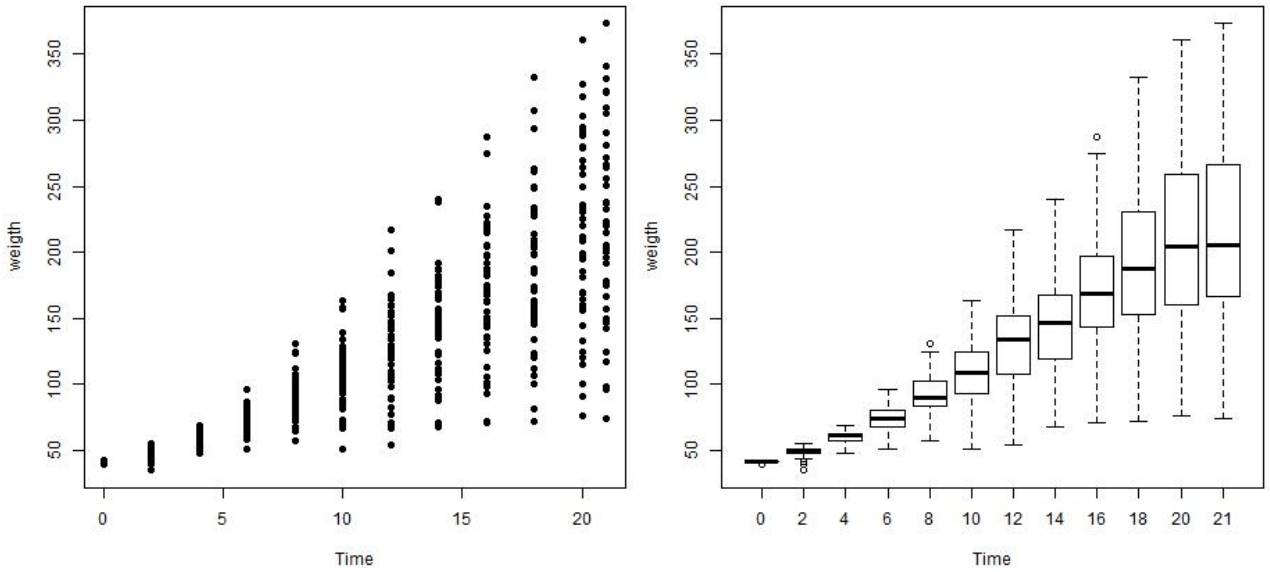
« jittered »



« jittered » horizontal

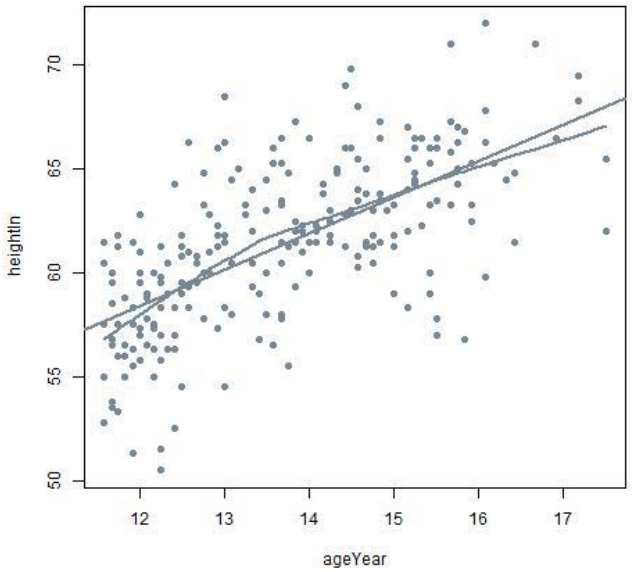
Source : R Graphics Cooking. Practical recipes for visualizing data. W Chang. O'Reilly, First Edition.

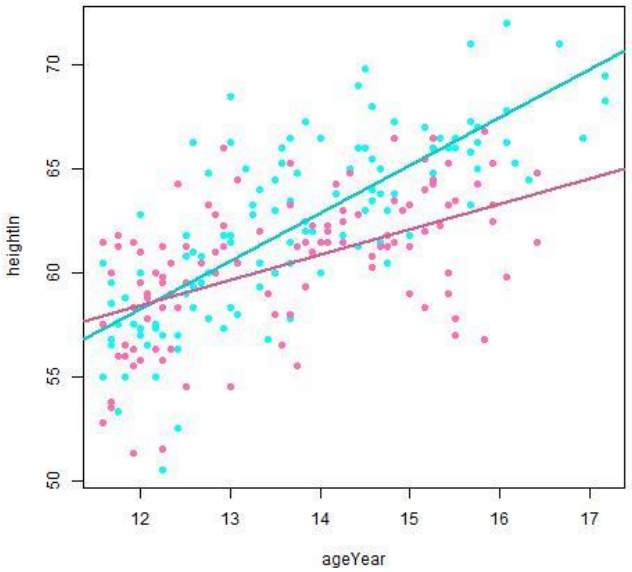
Préambule (4)



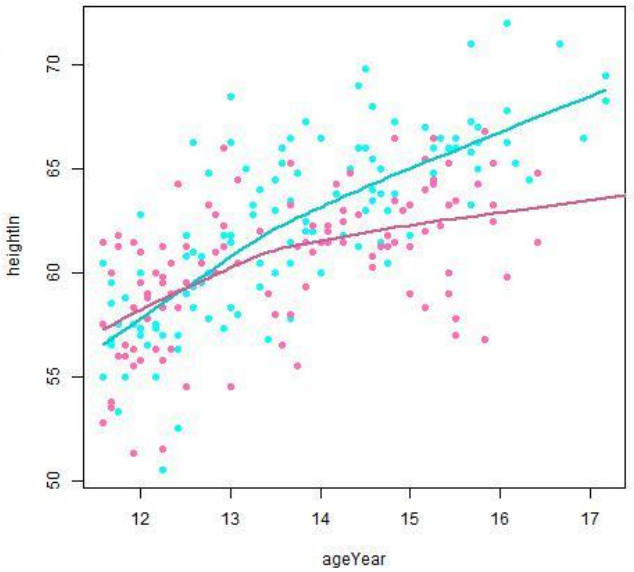
Source : R Graphics Cooking. Practical recipes for visualizing data. W Chang. O'Reilly, First Edition.

Préambule (5)





Selon le sexe
Effet linéaire



Selon le sexe
Effet souple

Source : R Graphics Cooking. Practical recipes for visualizing data. W Chang. O'Reilly, First Edition.

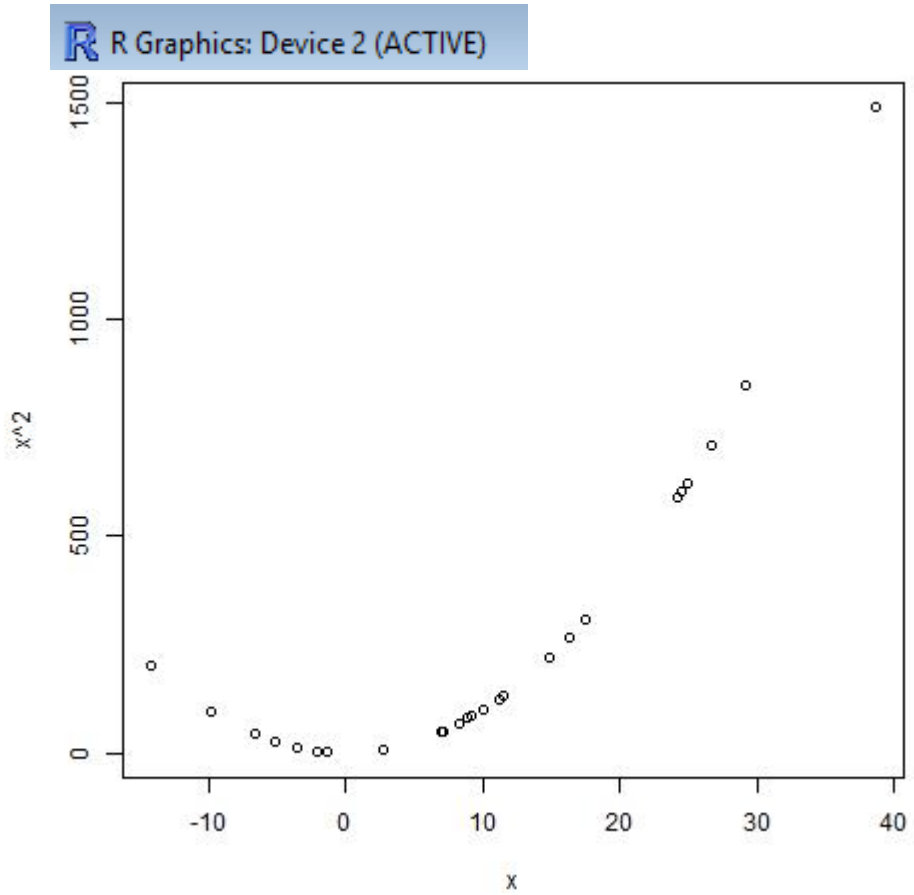
Introduction (1)

- Graphique → Visualisation
  - ✓ des données
  - ✓ des résultats
- Graphique → Compréhension
  - ✓ de la structure des données avant leur analyse
  - ✓ des résultats obtenus suite à l'analyse
- Graphique → Présentation
  - ✓ simplifiée de messages pouvant être complexes
  - ✓ esthétique

Introduction (2)

- Réalisation de graphique(s)
  - ✓ Une des étapes de toute analyse statistique
- Vaste types de graphiques
  - ✓ Adaptés au(x) type(s) de données
  - ✓ Standards et personnels
  - ✓ Pour représenter au mieux ce que l'on veut
- Graphiques avec R
  - ✓ Nombreuses fonctions, nombreuses options
  - ✓ Nombreux packages (`graphics`, `stats`, `lattice`, `grid`, `ggplot2`,...)
  - ✓ Possibilités de programmer pour obtenir ses propres graphiques souhaités

Fenêtre Graphique (1)




```
> set.seed(245) # Pour rendre l'exemple reproductible
> x <- rnorm(n=25, mean=10, sd=10)
> plot(x, x^2)
>
```

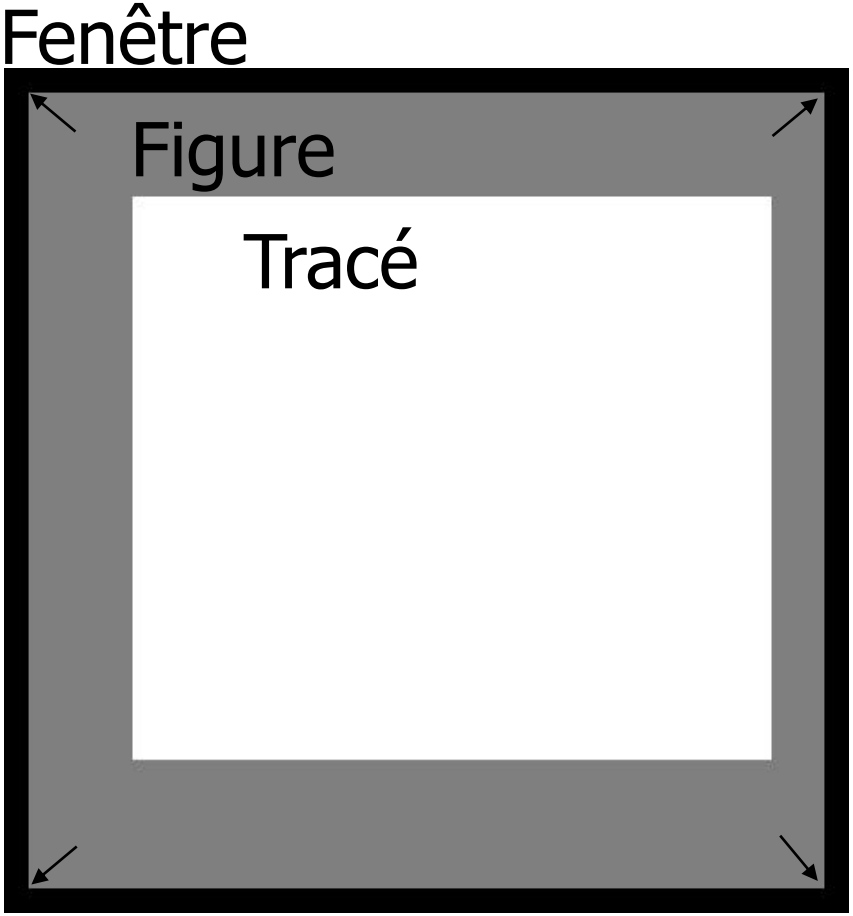
Fenêtre Graphique (2)

- Exécution d'une commande graphique
  - ✓ Si aucune fenêtre (ou périphérique) graphique n'est ouverte
  - ✓ R ouvre une fenêtre où sera affiché le graphique
  - ✓ Cette dernière reste active tant qu'elle n'est pas fermée
    - L'exécution d'une commande graphique fait afficher le graphique dans cette fenêtre

Fonction	Action
`x11()`	ouverture d'une fenêtre graphique
`dev.list()`	liste des fenêtres (périph.) ouvertes
`dev.cur()`	fenêtre (ou périph.) active
`dev.set(i)`	activation de la fenêtre (ou périph.) i
`dev.off(i)`	fermeture de la fenêtre (ou périph.) i

Fenêtre Graphique (3)

- Divisée en régions



Fonctions Graphiques (1)

- De haut niveau
  - ✓ Créent un nouveau tracé dans la fenêtre graphique
  - ✓ `plot()`, `boxplot()`,...
- De bas niveau
  - ✓ Ne créent pas un nouveau tracé
  - ✓ Permettent de retoucher un graphique déjà existant
  - ✓ `points()`, `lines()`, `title()`,...
- Liées au format de sortie des graphiques
  - ✓ Pour ouvrir un nouveau dispositif graphique
  - ✓ `x11()`, `jpeg()`, `pdf()`,...
- Permettant d'interagir avec les graphiques
  - ✓ Pour retoucher « à la main » un graphique
  - ✓ `locator()`, `identify()`

Fonctions Graphiques (2)

- Représentation d'une variable

Fonction	Type de graphique
`plot()`	Tracé en ordonné suivant différents styles (points, lignes, barres,…)
`hist()`	Histogramme
`barplot()`	Barres
`Barplot2()`	Barres avec intervalles
`stripchart()`	Nuage de point à une dimension
`boxplot()`	Boîtes à moustaches
`mosaicplot()`	Mosaïque
`qqnorm()`	Quantile-Quantile plot sur la loi Normale

Fonctions Graphiques (3)

- Représentation de deux variables

Fonction	Type de graphique
`plot()`	Tracé suivant différents styles (points, lignes, barres,...)
`mosaicplot()`	Mosaïque
`qqplot()`	Quantile-Quantile plot

- Données soumises à la fonction peuvent être de différentes classes
- Type de graphique produit différent selon le nombre et la nature des données

Fonctions Graphiques (4)

- Utilisez la fonction plot avec 1 ou 2 données

```
> set.seed(245) # Pour rendre l'exemple reproductible
> x <- rnorm(n=25, mean=10, sd=10)  # Vecteur
> mat1 <- matrix(cbind(x, x^2), ncol=2)  # matrice
> fact1 <- factor(x>0)  # Facteur
> fact2 <- factor(x^2>100)  # Facteur
> tab1 <- table(fact1)  # Table
> tab2 <- table(fact2)  # Table
> dataf1 <- data.frame(cbind(x, x^2, fact1, fact2)) # data frame
> names(dataf1) <- c("x", "x.2", "fact1", "fact2")
```

```
> plot(x)
> plot(x, x^2)
> plot(mat1)
> plot(fact1)
> plot(fact1, fact2)
> plot(table(fact1, fact2))
> plot(dataf1$x, dataf1$x.2)
> plot(dataf1$x, dataf1$fact1)
```

Arguments Communs de ces Fonctions (1)

Argument	Action
`xlim= ylim=`	Limites inférieure et supérieure de l'échelle sur l'axe
`xlab= ylab=`	Légende de l'axe
`main`	Titre du graphique
`sub`	Sous-titre du graphique (en bas de la fenêtre, sous la légende de l'axe des abscisses
`col`	Couleur de remplissage
`cex`	Valeur numérique indiquant de combien le texte et les symboles doivent être agrandies par rapport à la valeur par défaut
`font`	Fonte des échelles (non employé pour le titre, légendes des axes, sous-titre)
`bty`	Type de cadre entourant la région du tracé
`pch`	Type de points tracés

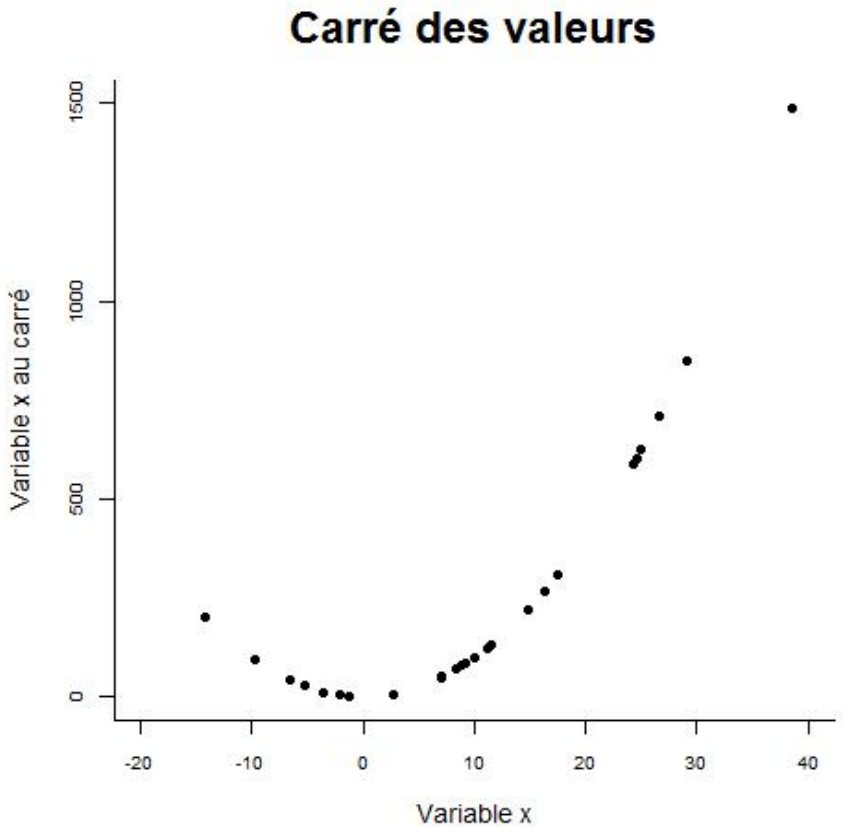
Arguments Communs de ces Fonctions (2)

Argument	Action
`cex.axis`	Taille des valeurs pour les échelles des abscisses et des ordonnées (1 par défaut, la valeur définie étant à multiplier par la valeur de `cex`)
`cex.lab`	Taille des caractères pour les légendes des abscisses et des ordonnées (idem que `cex.axis` pour le multiple)
`cex.main`	Taille des caractères du titre (idem que `cex.axis` pour le multiple)
`cex.sub`	Taille des caractères du sous-titre (idem que `cex.axis` pour le multiple)
`col.axis`	Couleur des échelles des abscisses et des ordonnées
`col.lab`	Couleur des légendes des abscisses et des ordonnées
`col.main`	Couleur du titre
`col.sub`	Couleur du sous-titre
`font.axis,…`	Fonte des échelles (`.axis`), des légendes (`.lab`), du titre (`.main`), du sous-titre (`.sub`)

Fonction Graphique et Arguments (1)

- Comment obtenir ?

```
> set.seed(245) # Pour rendre l'exemple reproductible
> x <- rnorm(n=25, mean=10, sd=10)
```



Fonction Graphique et Arguments (2)

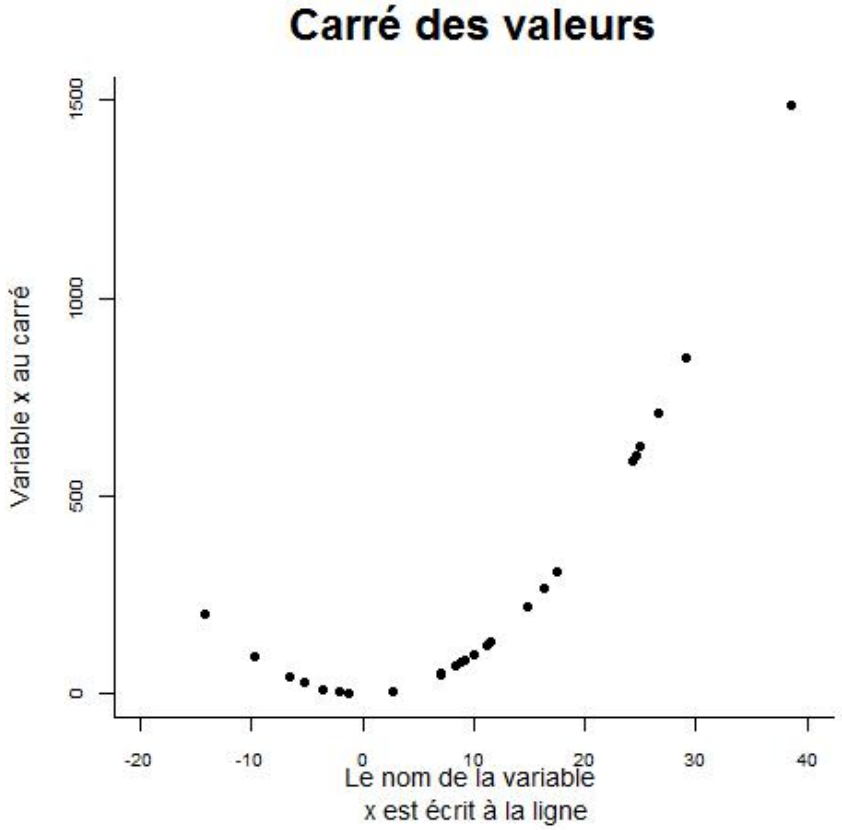
```
> set.seed(245) # Pour rendre l'exemple reproductible
> x <- rnorm(n=25, mean=10, sd=10)
> plot(x, x^2, xlim=c(-20, 40), ylim=c(0, 1500),
+  xlab="Variable x", ylab="Variable x au carré",
+  main="Carré des valeurs",
+  cex.axis=0.8, cex.lab=1.2, cex.main=2,
+  bty="l", pch=16)
>
```

- Remarque : valeurs pour l'argument `bty`
  - ✓ "o" (défaut), "l", "7", "c", "u", or "]" "n"
  - ✓ La boîte résultante ressemble à la lettre majuscule correspondante

Fonction Graphique et Arguments (3)

- Comment obtenir ?

```
> set.seed(245) # Pour rendre l'exemple reproductible
> x <- rnorm(n=25, mean=10, sd=10)
```



Fonction Graphique et Arguments (4)

```
> set.seed(245) # Pour rendre l'exemple reproductible
> x <- rnorm(n=25, mean=10, sd=10)
> plot(x, x^2, xlim=c(-20, 40), ylim=c(0, 1500),
+  xlab="Le nom de la variable \n x est écrit à la ligne",
+  ylab="Variable x au carré",
+  main="Carré des valeurs",
+  cex.axis=0.8, cex.lab=1.2, cex.main=2,
+  bty="l", pch=16)
>
```

- Remarque : utilisation de `\n` équivalent de la touche « Entrée »

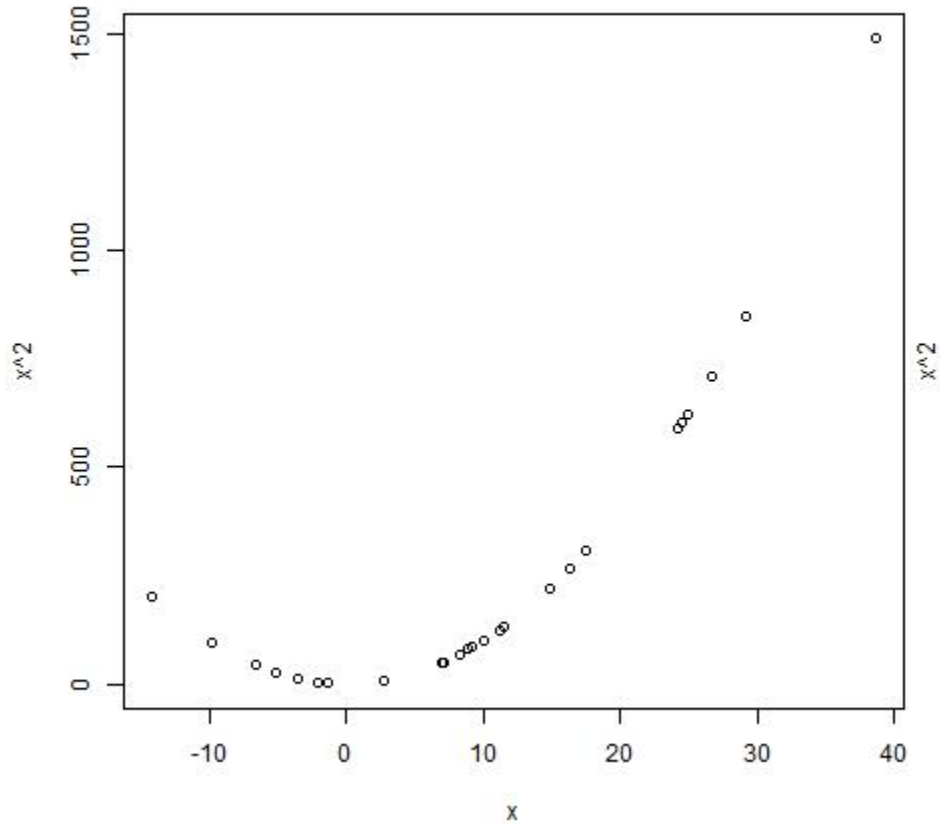
Ajout d'Eléments sur un Graphique (1)

Fonction	Action : Ajoute sur le graphique déjà tracé
`points()`	des points
`lines()`	des points reliés par une ligne
`abline()`	une ligne droite sans limite
`segments()`	un segment de droite
`arrows()`	une flèche
`rect()`	un rectangle
`polygon()`	un polygone
`legend()`	une légende
`rug()`	des marques secondaires sur les axes
`axis()`	une échelle sur l'un des côtés du graphique (en cas de suppression)
`text()`	du texte
`mtext()`	du texte dans les marges de la figure ou de la fenêtre graphique
`title()`	du texte dans le titre, le sous-titre, les légendes des axes,…

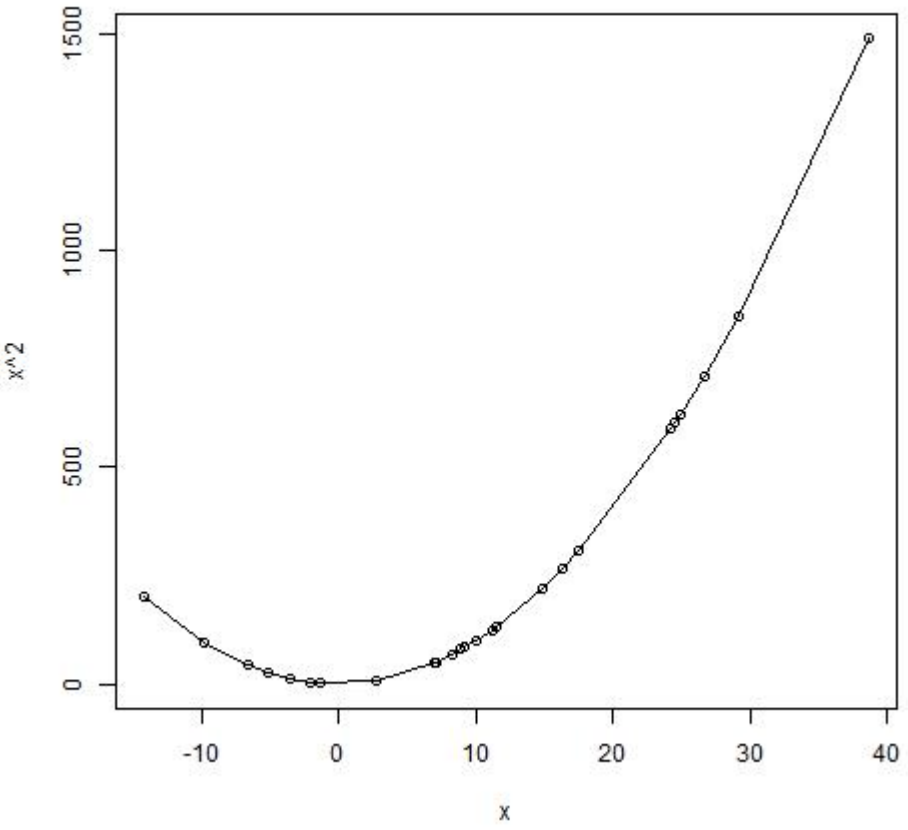
Ajout d'Eléments sur un Graphique (2)

- A partir de ① comment obtenir ② ?

①



②



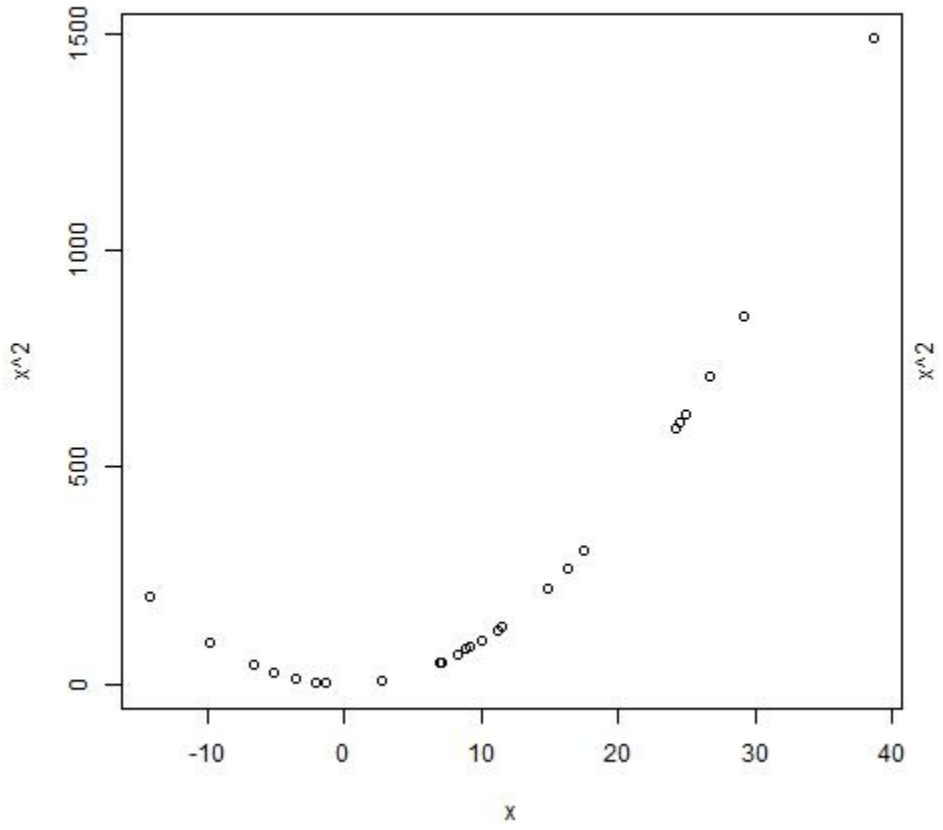
Ajout d'Eléments sur un Graphique (3)

```
> set.seed(245) # Pour rendre l'exemple reproductible
> x <- rnorm(n=25, mean=10, sd=10)
>
① > plot(x, x^2)
>
② > lines(x[order(x)], (x^2)[order(x)])
>
```

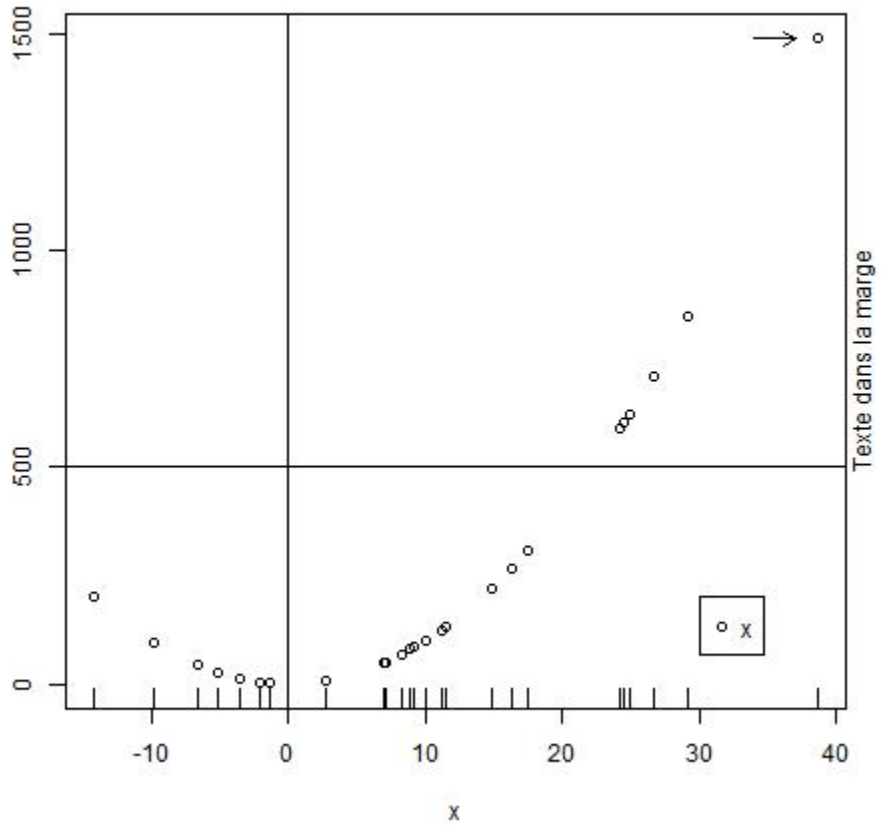
Ajout d'Eléments sur un Graphique (4)

- A partir de ① comment obtenir ② ?

①



②



Ajout d'Eléments sur un Graphique (5)

```
> set.seed(245) # Pour rendre l'exemple reproductible
> x <- rnorm(n=25, mean=10, sd=10)
>
① > plot(x, x^2)
>
② > arrows(x0=34, y0=1490, x1=37, y1=1490, length=0.1)
> abline(h=500)
> abline(v=0)
> legend(x=30, y=200, legend="x", pch=1)
> mtext("Texte dans la marge", side=4)
> rug(x)
>
```

Paramètres Graphiques (1)

- Fenêtre graphique définie par 72 paramètres
  - ✓ `par()`
- 66 étant modifiables
  - ✓ Amélioration considérable du tracé
  - ✓ `cex`, `cex.axis`, `cex.lab`, `cex.main`, `col.axis`,…
  - ✓ Voir `?par`

Paramètres Graphiques (2)

- Remarque 1
  - ✓ Modification des paramètres s'applique à la fenêtre active
  - ✓ Doit être spécifiée avant le tracé du graphique
  - ✓ S'appliquera à tout les graphiques exécutés par la suite dans cette fenêtre

- Remarque 2
  - ✓ Certains arguments de `par()` sont acceptés par les fonctions graphiques
    - `cex` est accepté par `plot()`
  - ✓ Le changement s'appliquera alors qu'au tracé

Paramètres Graphiques (3)

- Sauvegarde / réinitialisation des ou d'une valeur initiale des paramètres

```
> # Fermer les fenêtres graphiques
> # Sauvegarde DES paramètres graphiques initiaux
> par.ini <- par(no.readonly=TRUE)
> # Réinitialisation DES paramètres graphique dans la fenêtre
> par(par.ini)
```

```
> # Fermer les fenêtres graphiques
> # Sauvegarde D'UN paramètre graphique initial
> par.ini <- par(cex=3)
> # Réinitialisation DU paramètre graphique dans la fenêtre
> par(par.ini)
```

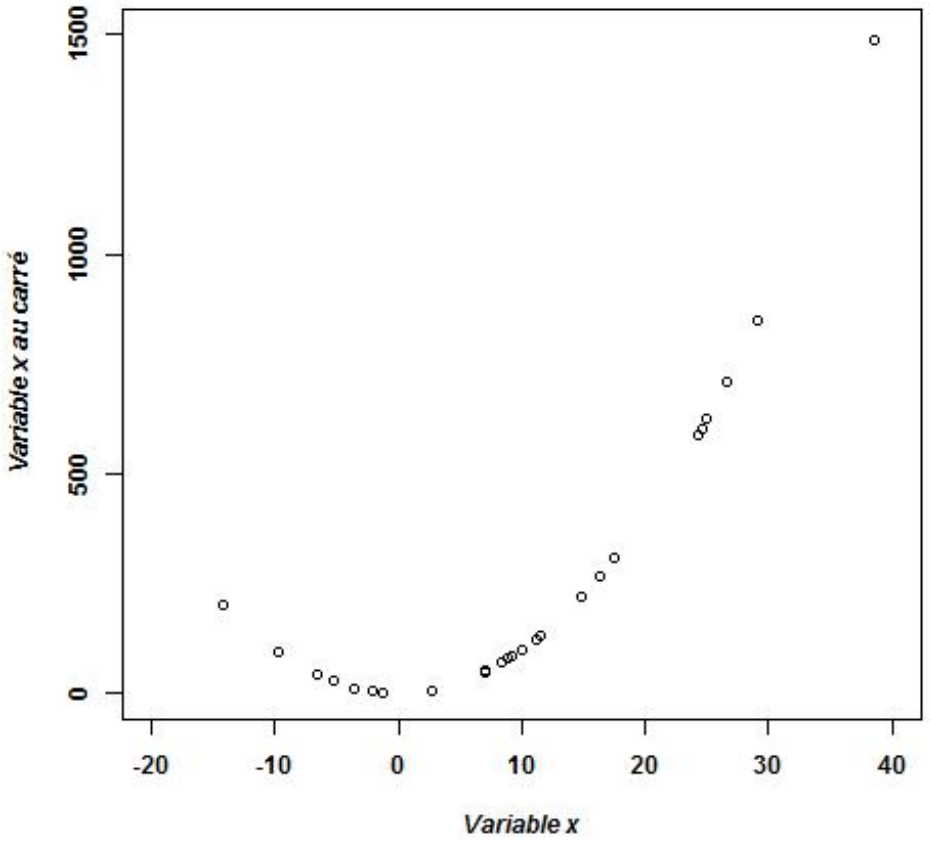
Paramètres Graphiques (4)

- Réinitialisation des valeurs initiales des paramètres
  - ✓ Fermeture de la fenêtre graphique
  - ✓ Sauvegarde des paramètres initiaux avant exécution

Polices de Caractères (1)

- Définitions
  - ✓ Police de caractères : style graphique personnalisé
    - Times, `Courier`, Arial,…
  - ✓ Fonte : regroupe les caractéristiques communes appliquées sur les polices
    - Normal, *italique*, **gras**,…
- Dans R
  - ✓ 19 fontes
  - ✓ constituées de 7 polices (Arial, Symbol, Times New Roman, Courier New, Century Gothic, Matisse ITC, Wingdings)
  - ✓ et styles associés (normal, gras, italique, gras & italique)

Polices de Caractères (2)




```
> plot(x, x^2, xlim=c(-20, 40), ylim=c(0, 1500),
>  xlab="Variable x", ylab="Variable x au carré",
>  font.axis=2, font.lab=4)
>
```

Couleurs (1)

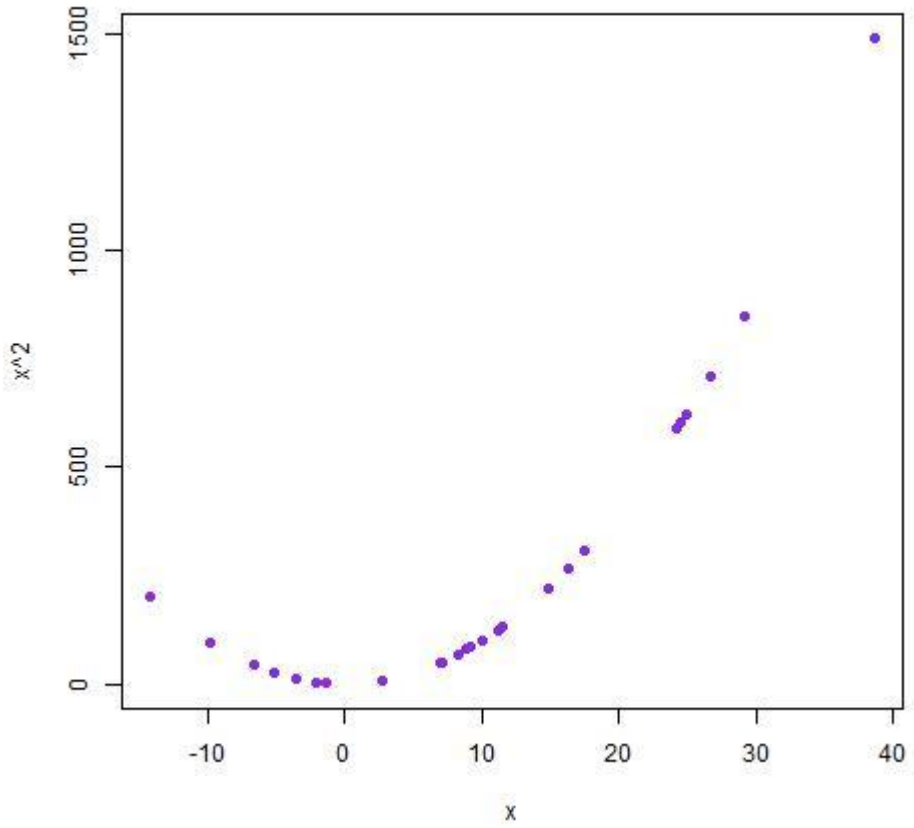
Attribut	Description
`col`	Paramètre graphique modifiant la couleur des éléments tracés
`colors()`	Liste de 657 mots (couleurs) utilisables par le paramètre `col`
`palette()`	Associe des chiffres (1 à 8) à des couleurs utilisables par le paramètre `col`
`rgb()`	Fixe la couleur à l'aide des couleurs primaires red, green et blue
`gray()`	Fixe un niveau de gris
`grey()`	Idem
`hsv()`	Fixe la couleur suivant sa teinte, sa saturation et sa valeur (Hue, Saturation, Value)
`rainbow()`	Permet de choisir n couleurs équidistantes sur le cercle de la représentation HSV (couleurs de l'arc en ciel), ou sur un arc de cercle

Couleurs (2)

- Selon la valeur numérique de l'argument `col`

Couleurs (3)

- Comment obtenir ?



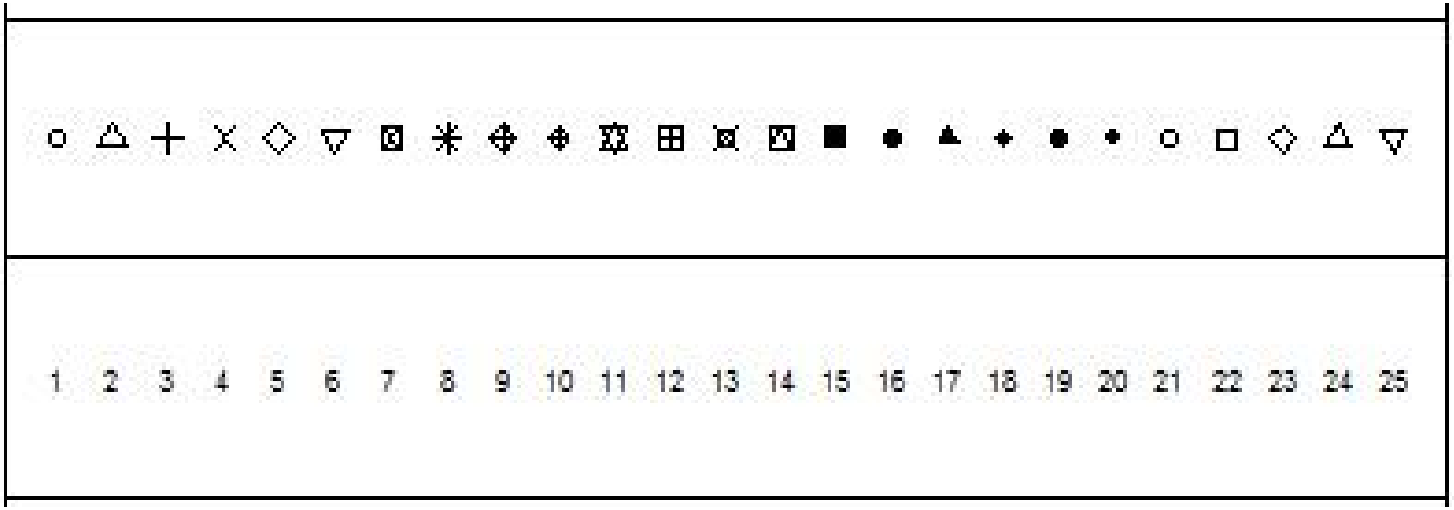
Couleurs (4)

```
> # mot choisi pour l'argument col
> plot(x, x^2, pch=16, col="blueviolet")
>
> # valeurs RGB
> plot(x, x^2, pch=16, col="#8A2BE2")
>
> # numéro de couleur
> palette(colors())
> plot(x, x^2, pch=16, col=31)
>
```

Voir : https://www.uvm.edu/~ngotelli/Rscripts/ColorChart.pdf

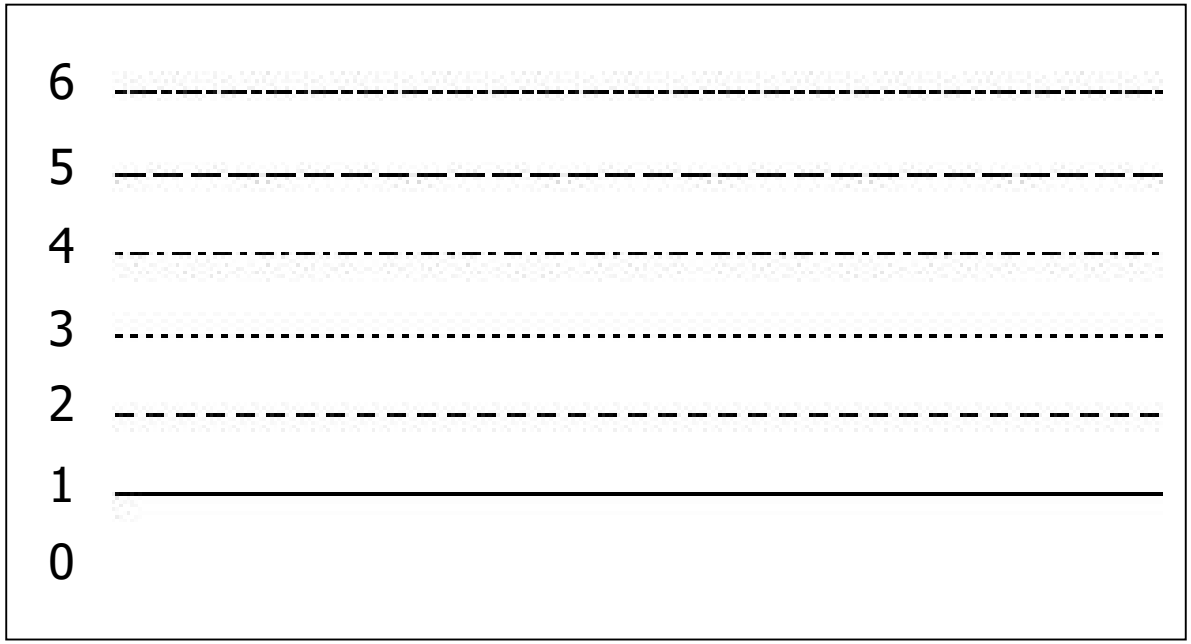
Symboles

- Selon la valeur numérique de l'argument `pch`



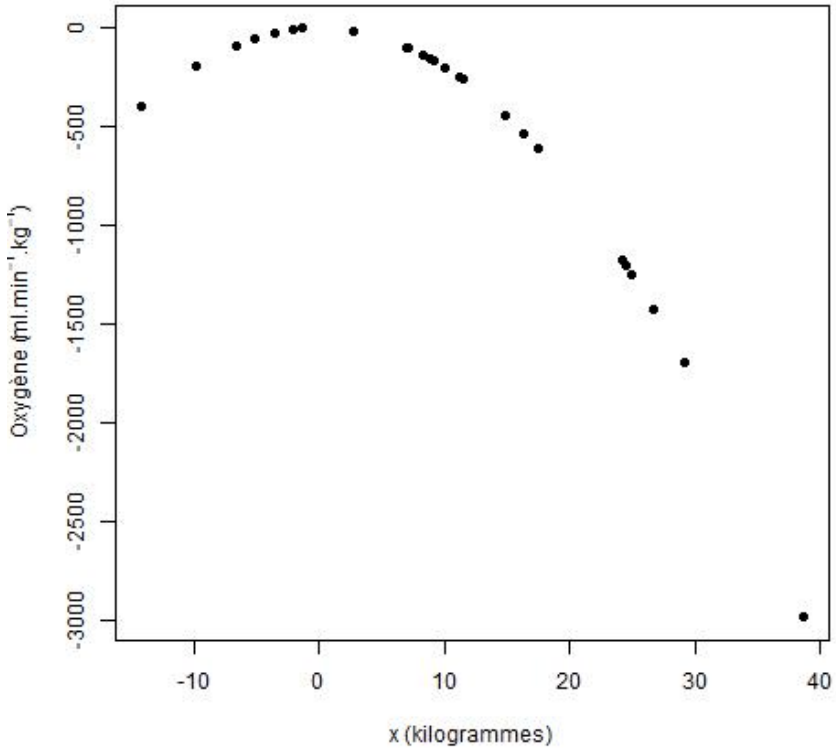
Lignes

- Selon la valeur numérique de l'argument `lty`



Texte et Equation (1)

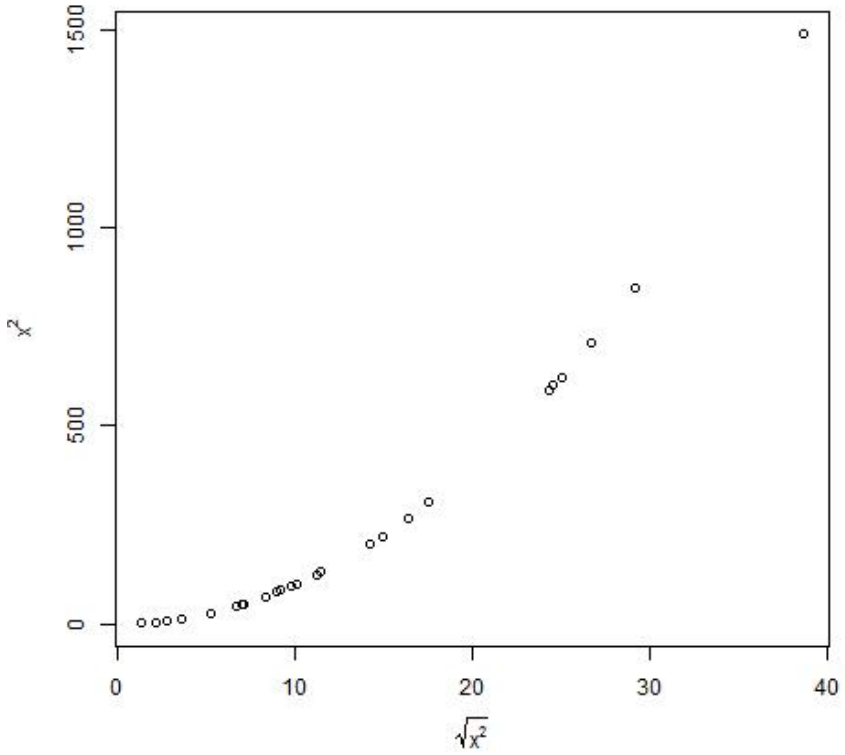
- Avec la fonction `expression`




```
> plot(x, y, xlab="x (kilogrammes)",
+ ylab=expression("Oxygène ("*ml.min^{-1}*.kg^{-1}*")"))
>
```

Texte et Equation (2)

- Avec la fonction `expression`



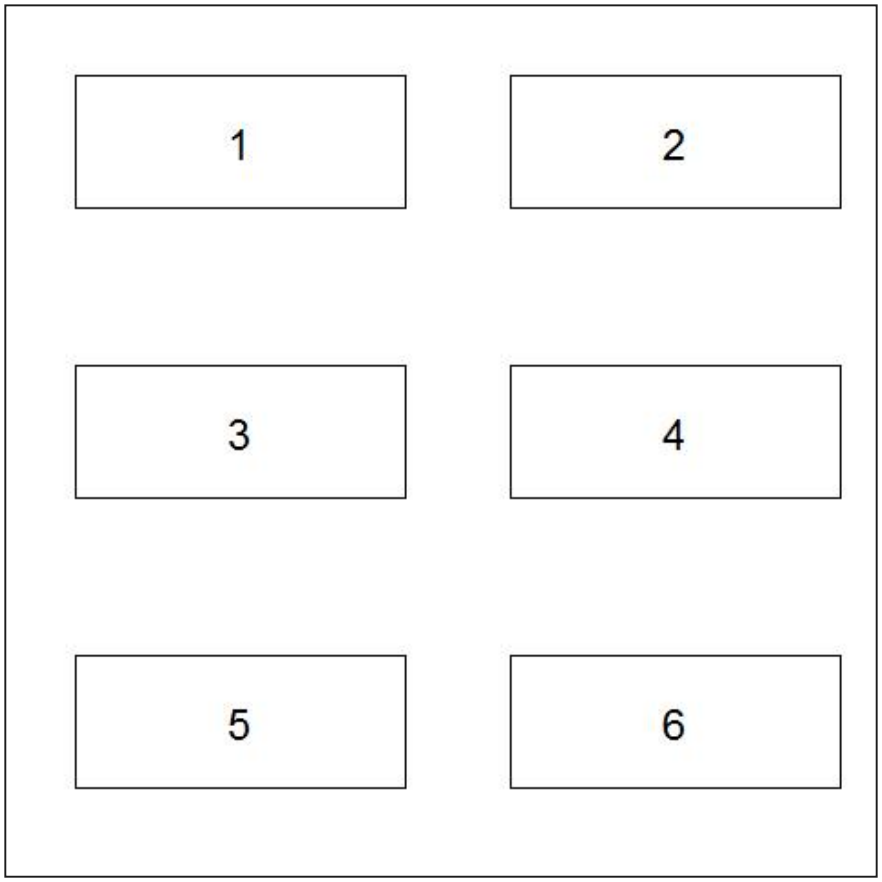

```
> plot(sqrt(x^2), x^2,
+ xlab=expression(sqrt(x^2)), ylab=expression(x^2))
>
```

Plusieurs Graphiques dans une Fenêtre (1)

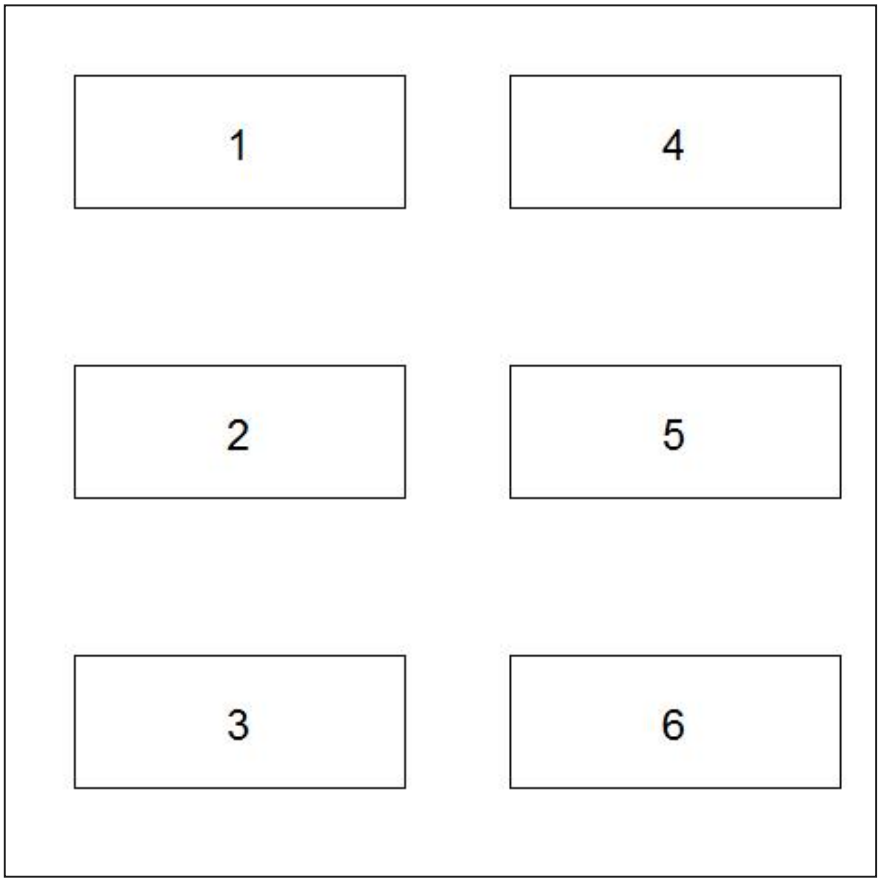
- La fenêtre est alors divisée en plusieurs régions de figure
- Arguments `mfrow` ou `mfcol` de `par()`
  - ✓ Découpent la fenêtre graphique en k lignes et c colonnes
  - ✓ Avec k x c régions de figures identiques (forme, surface)
  - ✓ Graphique tracé en utilisant les régions
    - de gauche à droite et de haut en bas : `mfrow`
    - de haut en bas et de gauche à droite : `mfcol`

Plusieurs Graphiques dans une Fenêtre (2)

```
> par(mfrow=c(3, 2))
> plot(…)
```




```
> par(mfcol=c(3, 2))
> plot(…)
```

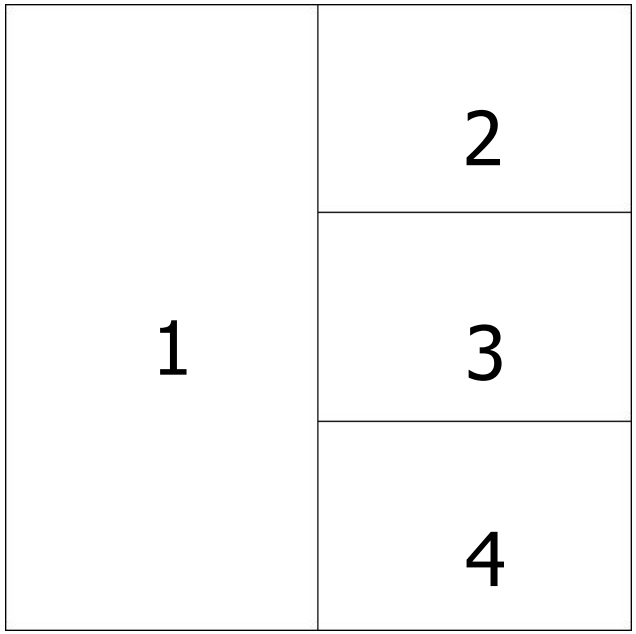


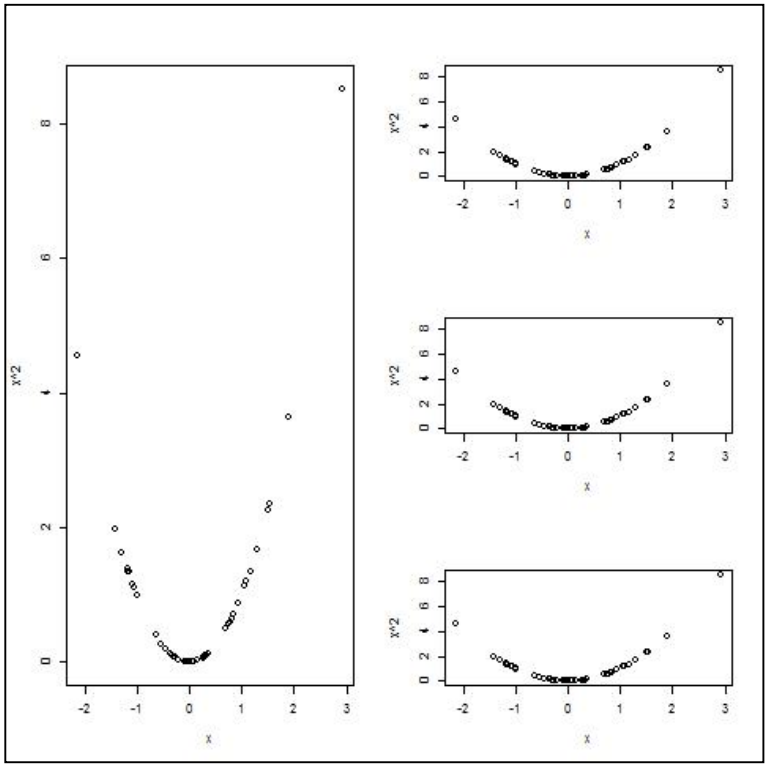
Plusieurs Graphiques dans une Fenêtre (3)

- Fonction `layout()`
  - ✓ Découpent la fenêtre graphique en k lignes et c colonnes
  - ✓ Avec k x c régions de figures pouvant être différentes (forme, surface)
  - ✓ Graphique tracé en utilisant les chiffres indiqués dans la matrice

Plusieurs Graphiques dans une Fenêtre (4)

```
> zones <- matrix(c(1, 1, 1, 2, 3, 4), ncol=2)
> layout(zones)
> plot(x, x^2);plot(x, x^2);plot(x, x^2);plot(x, x^2)
>
```

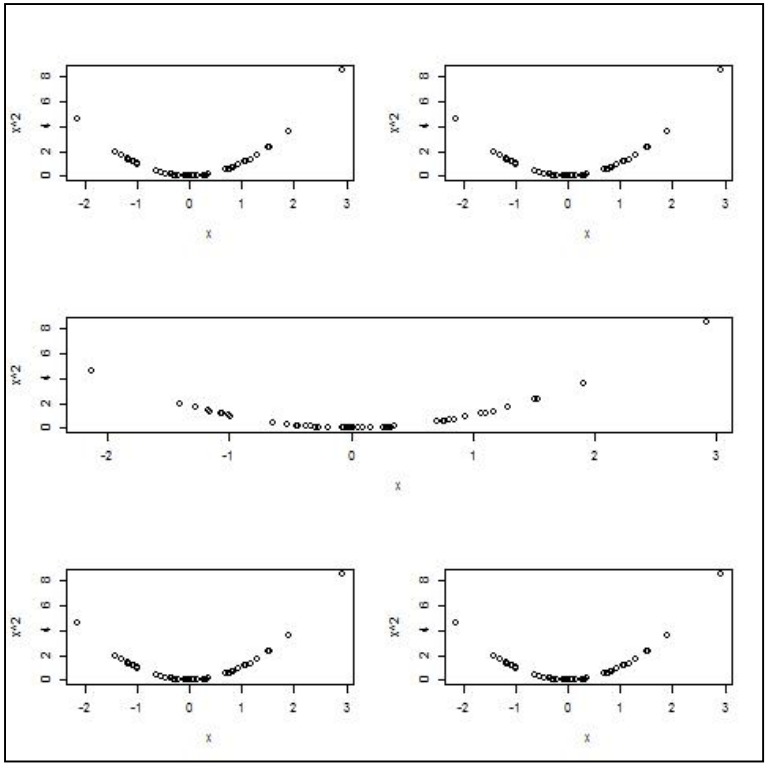




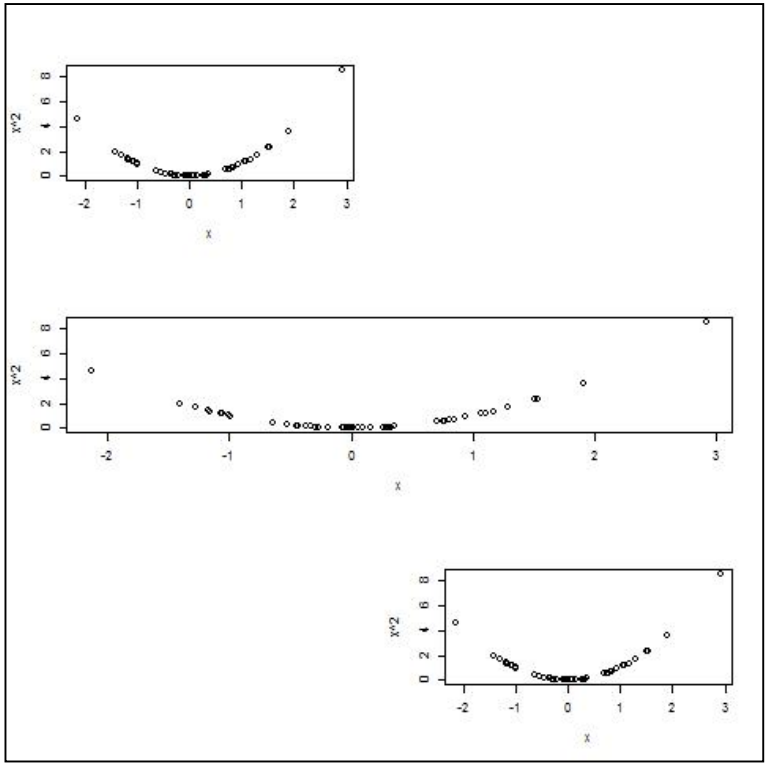
Plusieurs Graphiques dans une Fenêtre (5)

- Comment obtenir ?

①



②



Plusieurs Graphiques dans une Fenêtre (6)

①

```
> zones <- matrix(c(1, 2, 3, 3, 4, 5), ncol=2, byrow=TRUE)
> layout(zones)
> plot(x, x^2);plot(x, x^2);plot(x, x^2)
> plot(x, x^2);plot(x, x^2)
>
```

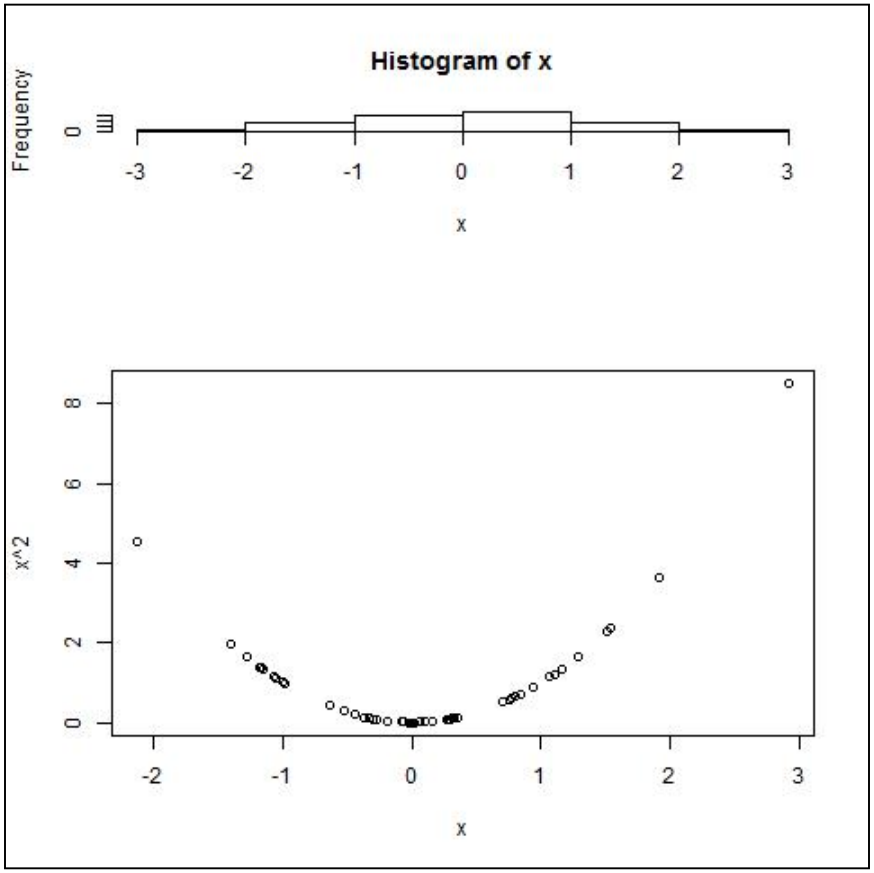
②

```
> zones <- matrix(c(1, 0, 2, 2, 0, 3), ncol=2, byrow=TRUE)
> layout(zones)
> plot(x, x^2);plot(x, x^2);plot(x, x^2)
>
```

Voir `?layout` et l'exemple « Create a scatterplot with marginal histograms » pour le paramétrage des marges des régions de figure

Plusieurs Graphiques dans une Fenêtre (7)

- Comment obtenir ?



Plusieurs Graphiques dans une Fenêtre (8)

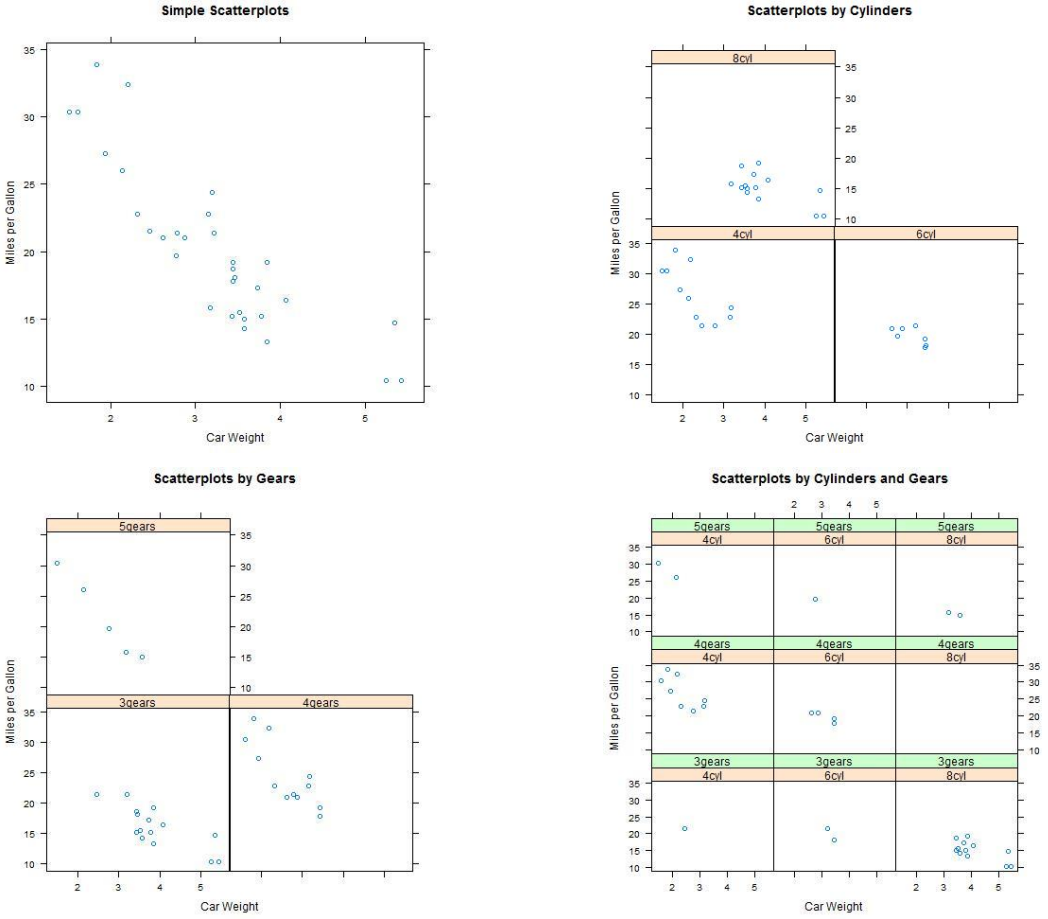
- Argument `fig` de `par()`

```
> par(fig=c(0, 1, 0.70, 1))
> hist(x)
> par(fig=c(0, 1, 0, 0.70), new=TRUE)
> plot(x, x^2)
>
```

où les valeurs de `fig` correspondent à `xleft`, `xright`, `ybottom`, `ytop` (coordonnées normalisées)

Plusieurs Figures dans un Graphique (1)

- Package `lattice` de R : treillis de graphes

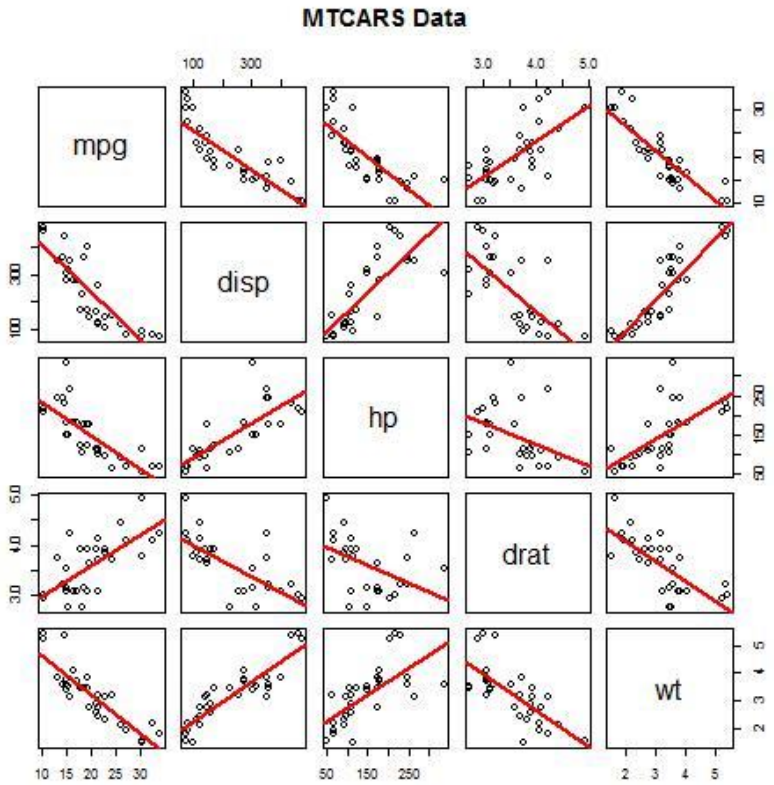


Fonction : `xyplot`

Adapté de : http://www.statmethods.net/advgraphs/trellis.html

Plusieurs Figures dans un Graphique (2)

- Package `lattice` de R : treillis de graphes



Gérer Plusieurs Fenêtres Graphiques

- Ouverture d'une fenêtre graphique automatique lors de l'exécution d'une fonction graphique
- R ne travaille que dans cette fenêtre
  - ✓ Chaque nouvelle exécution d'une fonction graphique écrase le graphique précédent
- Plusieurs fenêtres en même temps
  - ✓ *Voir diapo : Fenêtre Graphique*

Fonction	Description
`x11()`	ouverture d'une nouvelle fenêtre graphique
`windows()`	ouverture d'une nouvelle fenêtre graphique
`graphic.off()`	fermeture de toutes les fenêtres graphiques

Exporter un Graphique (1)

- Menu de la fenêtre graphique
  - ✓ Sauver sous…
- Sur la fenêtre graphique
  - ✓ Clic droit de la souris, Sauver sous…, Copier comme…
- Fonctions
  - ✓ `jpeg()`, `bmp()`, `png()`, `tiff()`, `pdf()`,…
  - ✓ `?device` donne l'ensemble de la liste
  - ✓ Se gère comme tout périphérique dans R
    - Action d'ouverture : `NomFonction(file=…)`
    - Action de fermeture : `dev.off()`

Exporter un Graphique (2)

- Fonctions (suite)
  - ✓ Remarque 1
    - Ces fonctions ouvrent une fenêtre graphique virtuelle
    - D'où la gestion avec `dev.off()` et autres fonctions de gestion graphique
    - Le fichier n'est pas accessible tant que la fonction `dev.off()` n'a pas été exécutée
  - ✓ Remarque 2
    - Création d'un fichier à l'extension souhaité
    - Ecrase sans avertissement tout fichier du même nom, même extension, même adresse

Exporter un Graphique (3)

```
> jpeg(file="c:\\Etude\\Fig\\Fig.x.x2.jpeg")
> plot(x, x^2, xlim=c(-20, 40), ylim=c(0, 1500),
+  xlab="Variable x", ylab="Variable x au carré",
+  main="Carré des valeurs",
+  cex.axis=0.8, cex.lab=1.2, cex.main=2,
+  bty="l", pch=16)
> dev.off()
>
```

Package `ggplot2` (1)

- A sa grammaire graphique
  - ✓ Principe de base
    - Séparer les données de la représentation graphique
    - Diviser la représentation en éléments de base (courbes, axes, labels,…)
  - ✓ Un graphique statistique est donc une représentation, sur un système de coordonnées, de données ayant des objets géométriques (points, lignes, barres) et des attributs esthétiques (couleur, forme, taille)
  - ✓ Il peut aussi contenir des transformations statistiques des données, être partitionné
- Augmente certaines performances graphiques

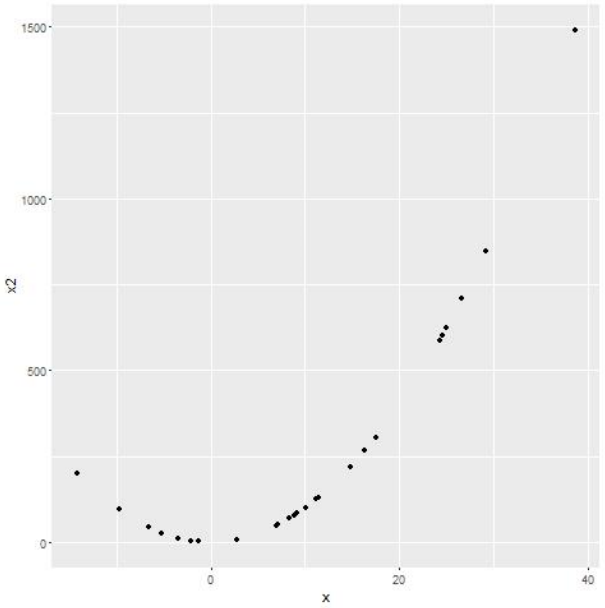
Package ggplot2 (2)

```
> set.seed(245) # Pour rendre l'exemple reproductible
> x <- rnorm(n=25, mean=10, sd=10)
> mdata <- data.frame(x=x, x2=x^2) # Données dans un data.frame
>
> library(ggplot2) # Chargement de la librairie
>
> # Initialisation de l'objet ggplot
> sp <- ggplot(mdata, aes(x=x, y=x2))
>
```

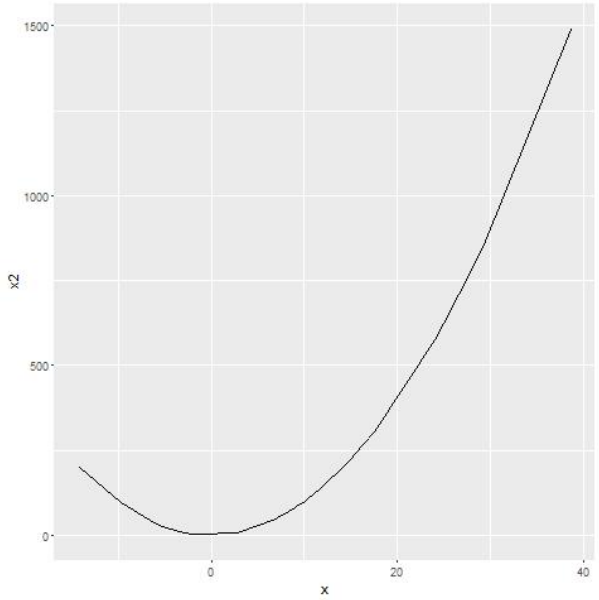
Package ggplot2 (3)

```
sp <- ggplot(mdata, aes(x=x, y=x2))
```

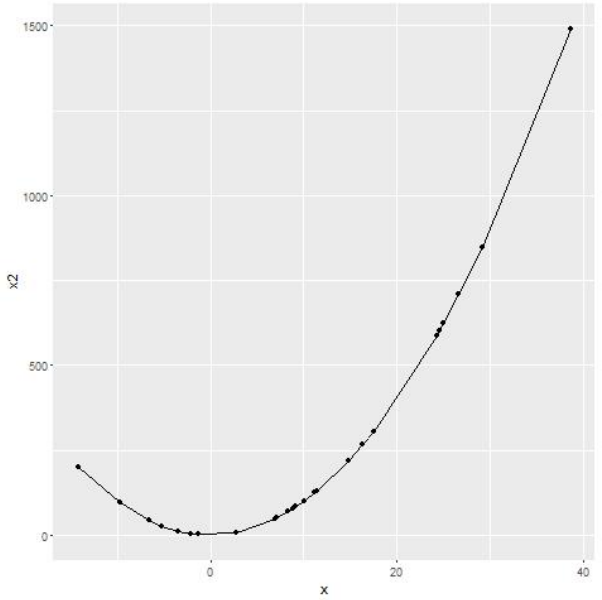
```
> sp+geom_point()
>
```




```
> sp+geom_line()
>
```

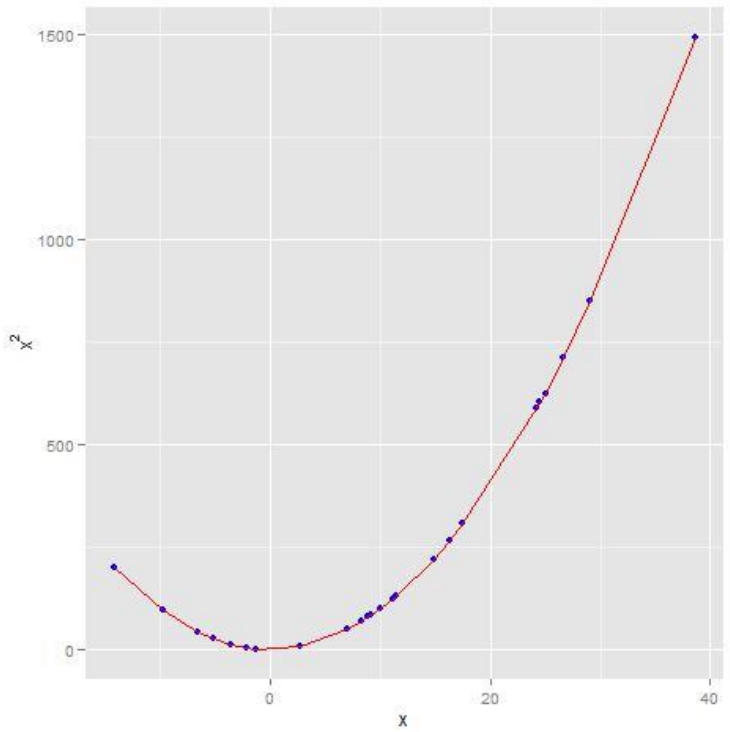



```
> sp+geom_point()+
+ geom_line()
>
```



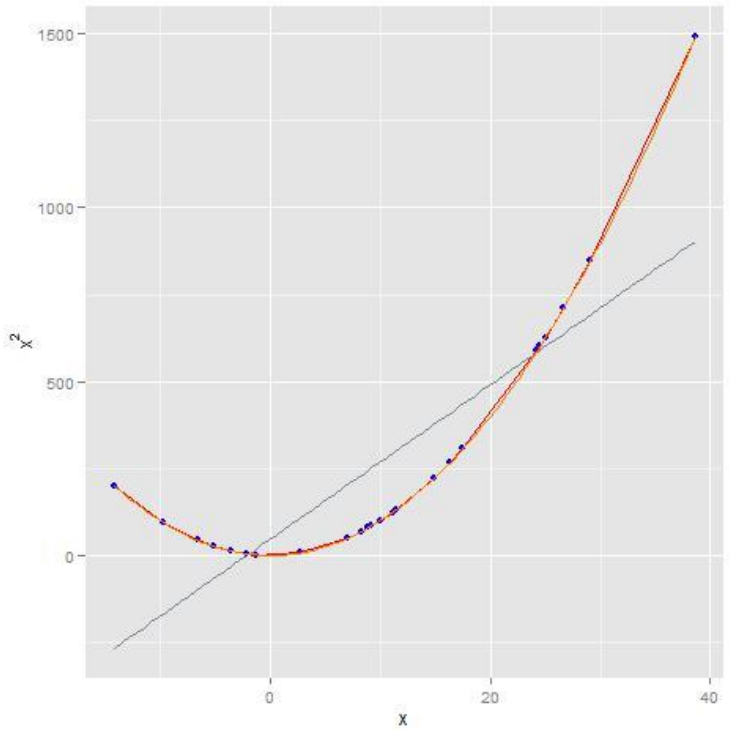
Package ggplot2 (4)

```
> sp + geom_point(colour="blue") +
+  geom_line(colour="red") +
+  xlab("x") + ylab(expression(x^2))
>
```



Package ggplot2 (5)

```
> sp + xlab("x") + ylab(expression(x^2)) +
+  geom_point(colour="blue") + geom_line(colour="red") +
+  stat_smooth(method=lm, se=FALSE, col="lightslategrey") +
+  stat_smooth(method=loess, se=FALSE, col="darkorange1")
>
```



Source

- Introduction à la programmation en R. Vincent Goulet https://cran.r-project.org/doc/contrib/Goulet_introduction_programmation_R.pdf
- Data Analysis and Graphics Using R: An Example-Based Approach. Cambridge Series in Statistical and Probabilistic Mathematics. JH Maindonald, WJ Braun. Third edition.
- R Graphics Cooking. Practical recipes for visualizing data. W Chang. O'Reilly, First Edition.
- ggplot2: Elegant Graphics for Data Analysis. H Wickham. Springer-Verlag NY. 2009,