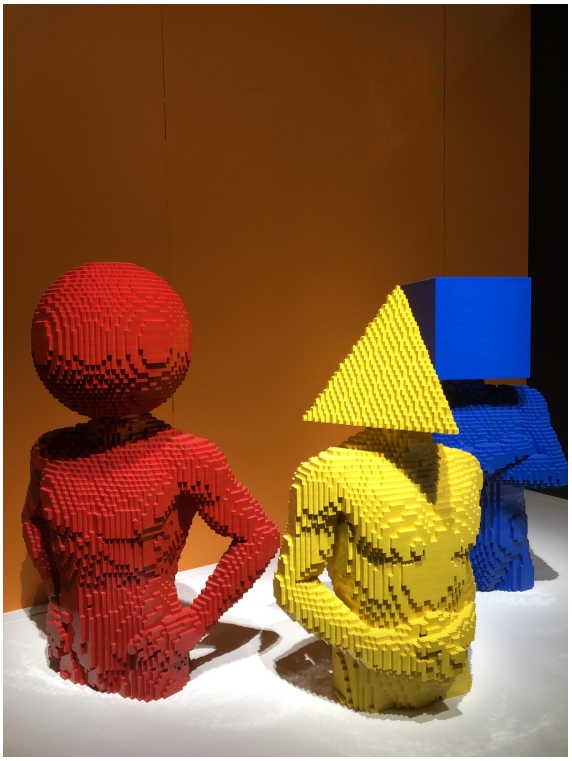
SCIENCES SORBONNE UNIVERSITÉ

Graphiques avec **ggplot2**

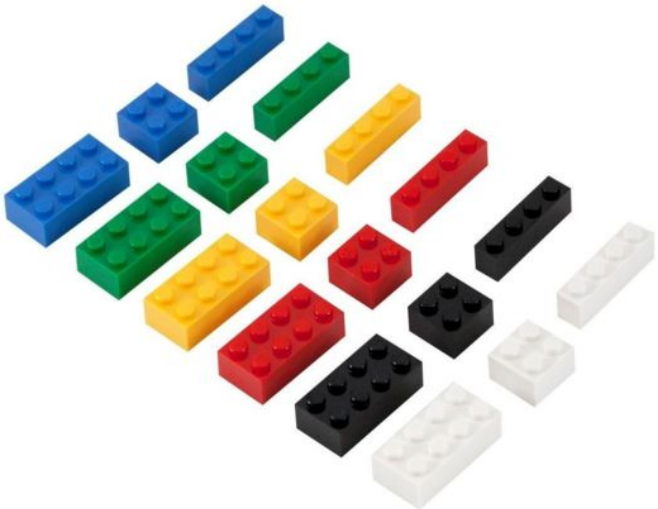
Xavier Raynaud

Présentation générale

Vos attentes



Mes objectifs



Mes objectifs

Qu'est ce que **ggplot2**?

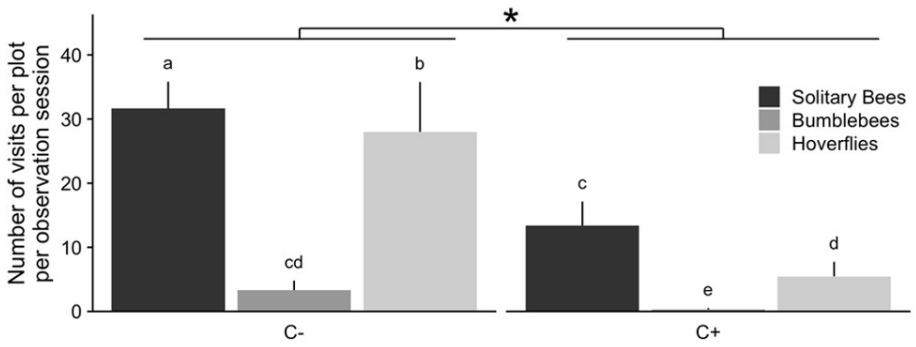
ggplot2: *Grammar of Graphics* plots (2ème version)

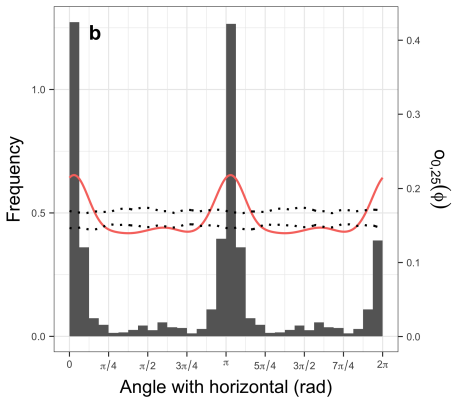
```
library(ggplot2)
```

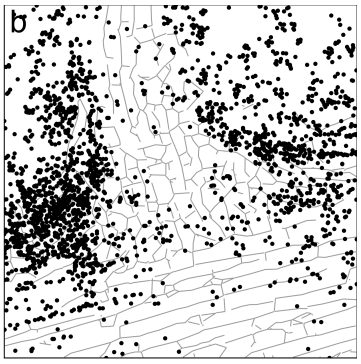
- appartient au `tidyverse` (developpé par H Wickham),
- existe sur `R` depuis plus de 10 ans,
- développement dynamique, version 3.0 sortie le 4 Juillet 2018 (aujourd'hui: 3.3.3),
- compatible avec les graphiques **grid**, incompatible avec les graphiques `base`,
- *Extensible* (possiblilité d'étendre les capacités de visualisation),
- *Themable*

L'idée principale est de séparer le fond (type de visualisation des variables à observer) de la forme (axes, couleurs...).

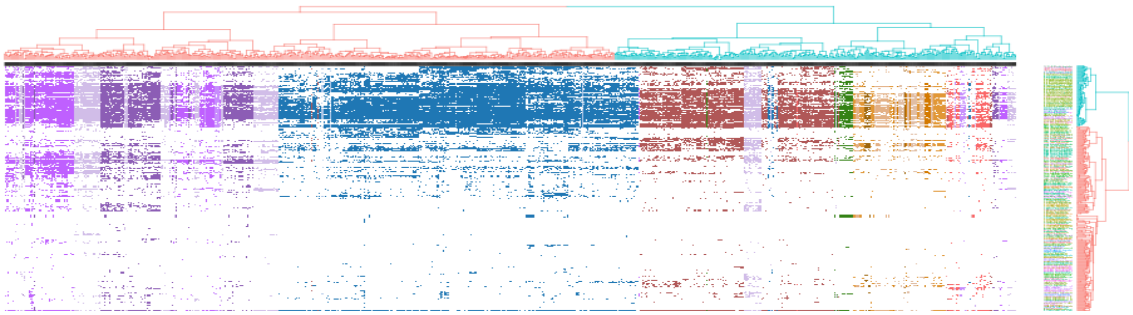
Exemples



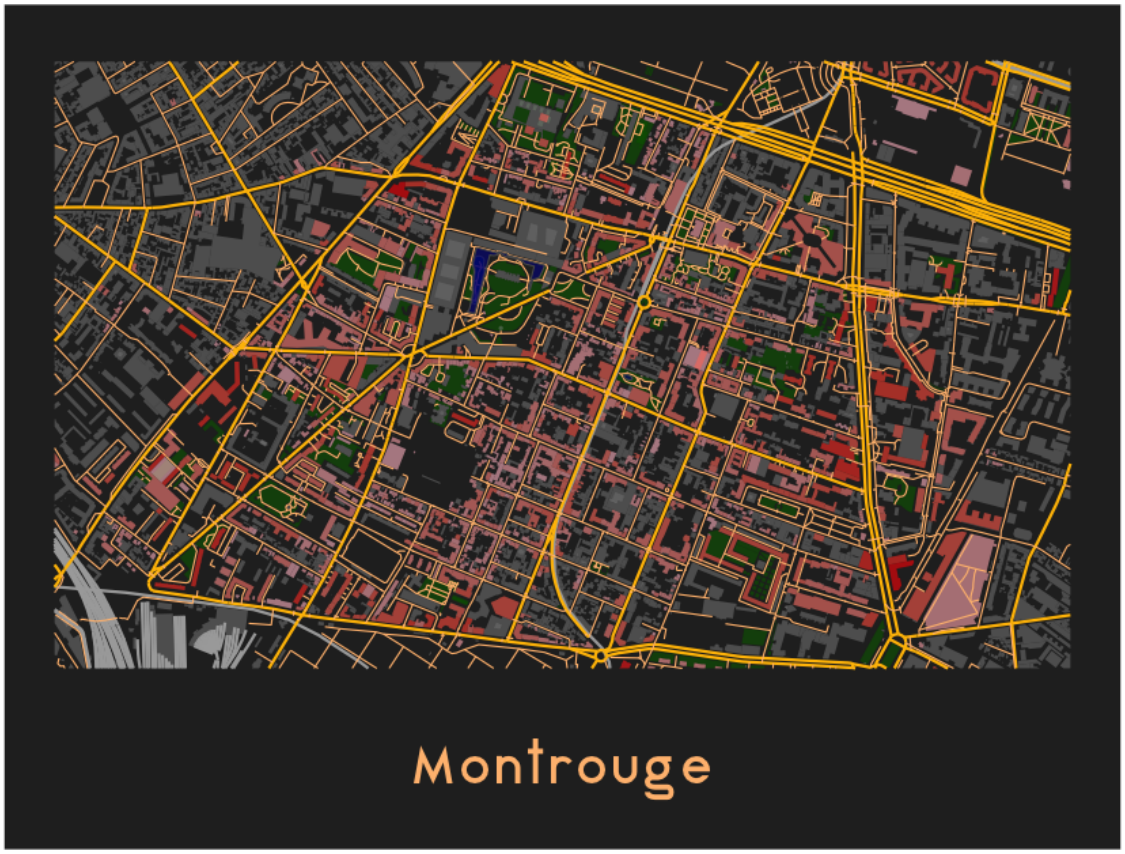




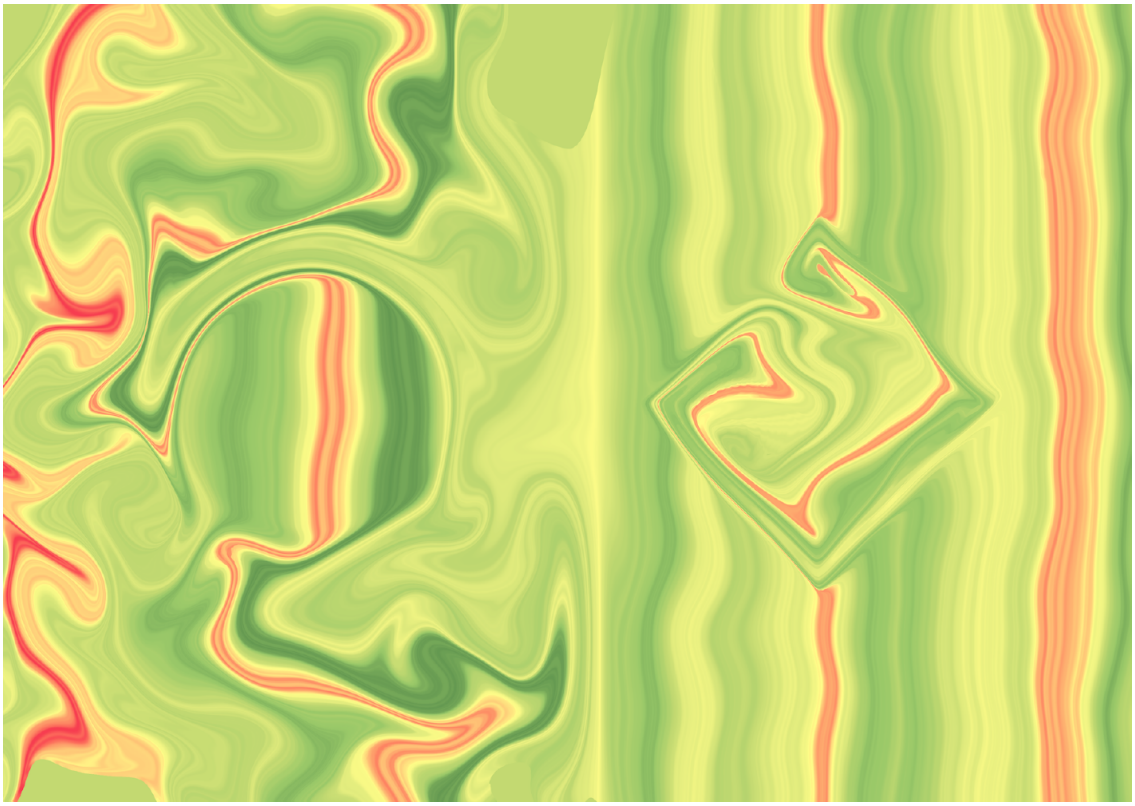
Exemples avancés



Exemples graphiques

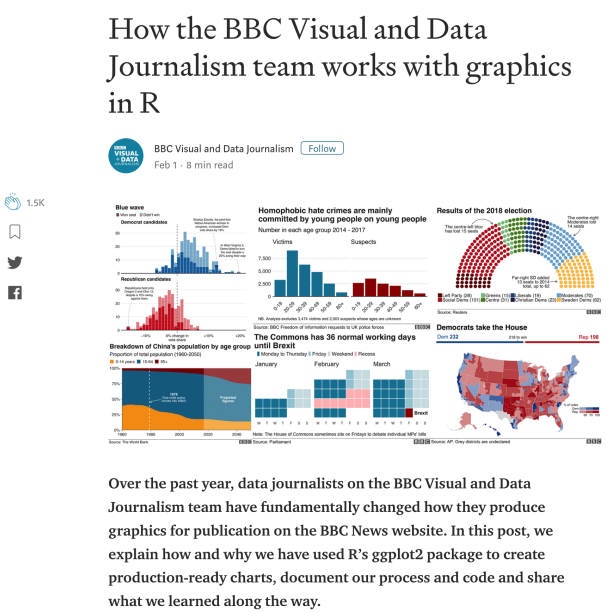


Exemples graphiques



Thomas Lin Pedersen

A la pointe du progrès



How the BBC Visual and Data Journalism team works with graphics in R

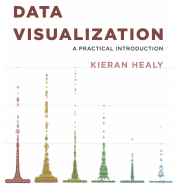
BBC Visual and Data Journalism · Follow
Feb 1 · 8 min read

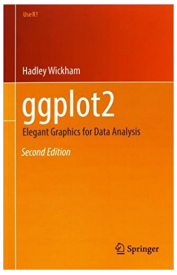
1.5K

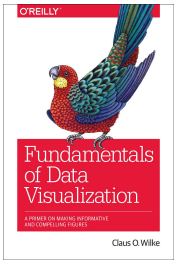
Over the past year, data journalists on the BBC Visual and Data Journalism team have fundamentally changed how they produce graphics for publication on the BBC News website. In this post, we explain how and why we have used R's ggplot2 package to create production-ready charts, document our process and code and share what we learned along the way.

https://medium.com/bbc-visual-and-data-journalism/how-the-bbc-visual-and-data-journalism-team-works-with-graphics-in-r-ed0b35693535
https://github.com/bbc/bbplot

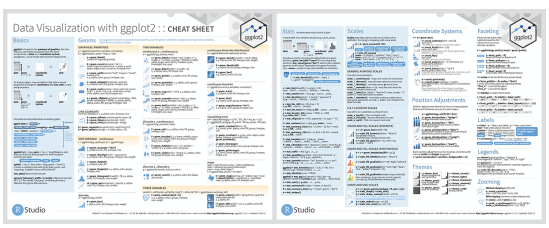
Ressources







- **Data Visualization: A practical introduction**, *K Healy*, Princeton Univ Press, http://socviz.co/
- **Site web de ggplot2**, https://ggplot2.tidyverse.org/
- **ggplot2: Elegant Graphics for Data Analysis**, *H Wickham*, Springer, https://github.com/hadley/ggplot2-book
- **Fundamentals of Data Visualization**, *C Wilke*, O'Reilly, https://serialmentor.com/dataviz/
- **StackOverflow**, www.stackoverflow.com
- **RStudio**, www.rstudio.com



Principes

On *additionne* des composants graphiques pour le fond (`ggplot()`, `geom_...()`) et la forme (`stats_...()`, `scale_...()`, `theme_...()`):

> **ggplot**(data=*données*, mapping=aes(x=*variable_x*, y=*variable_y*)) +
> **geom_...**(mapping=aes(colour=...,shape=...)) +
> **stats_...**() +
> **scale_..._...**() +
> **theme**()

L'objet graphique est créé par la fonction `ggplot()` qui prend 2 arguments (facultatifs) `data=` et `aes=`.

Les autres éléments ajoutent des visualisations (**geom_**, nuages de points, barres...) ou modifient l'allure graphique générale (**scale_**, **theme**).

Les **geom_** peuvent aussi prendre les arguments `data=` et `mapping=`.

Données

`ggplot()` prend comme argument un tableau de donnée (`data.frame` ou assimilé) avec *une variable par colonne* (tableau large). On passe ce tableau à `ggplot` par l'argument **data=**.

L'argument **mapping=aes()** sert à associer une variable du tableau à un élément graphique (coordonnée, couleur, forme, etc).

Dans la suite, on utilisera les données suivantes:

```
df = data.frame(V1 = runif(99),
                V2 = rnorm(99,mean=rep(5:7,each=33)),
                V3 = rep(LETTERS[1:3],each=33))
```

V1	V2	V3
0.2000780	6.904868	A
0.9121755	4.726743	A
0.9412326	6.025528	A
0.9031244	4.078240	A
0.2535742	6.565076	A
0.3476868	2.381178	A

Les geoms

Les **geom_** sont les fonctions permettant de représenter graphiquement un ou des ensembles de données. On trouve donc des **geoms_** pour

- décrire une variable:

  geom_boxplot(), *geom_histogram()*, *geom_density()*

- décrire les liens entre deux ou plusieurs variables:

  geom_point()/*geom_jitter()*, *geom_line()*/*geom_path()*, *geom_ribbon()*

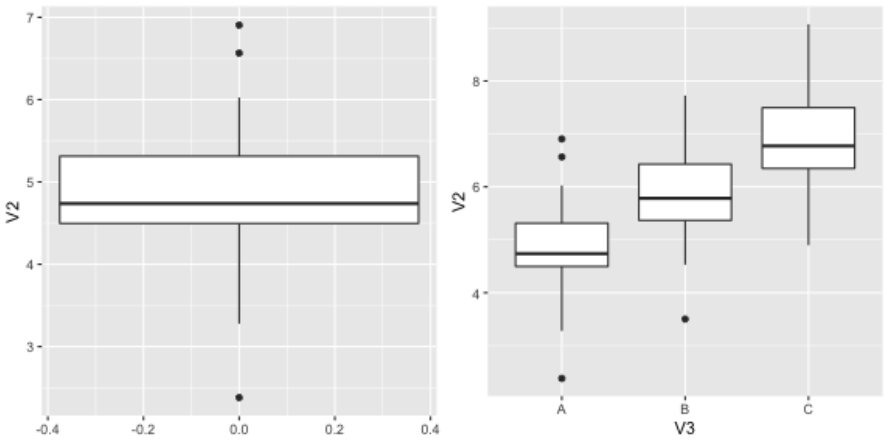
- pour représenter des données complexes (cartes, ...):

  geom_sf(), ...

Les variables à décrire sont passées au **geom_** à travers une fonction esthétique **aes()**. C'est grâce à cette fonction que l'on précise de quelle manière doit être codée la variable (coordonnée dans un plan, couleur, forme, taille des points, *etc*)

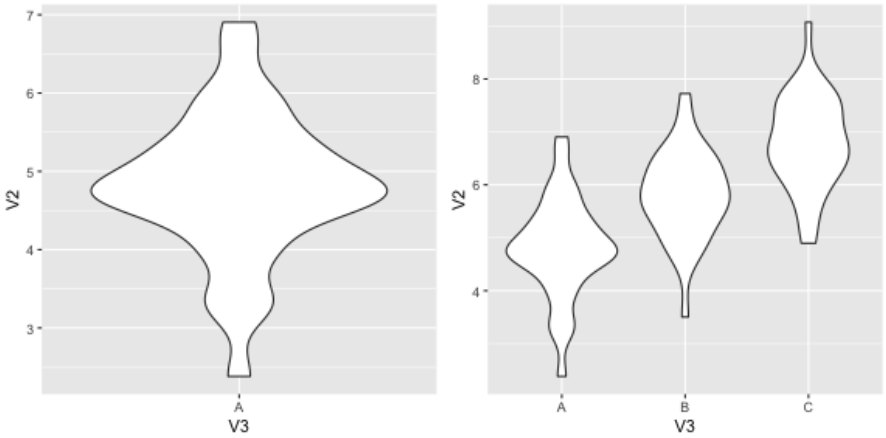
Geoms pour décrire une variable
geom_boxplot()

```
ggplot(data=subset(df,V3=="A"),aes(y=V2)) + geom_boxplot()

ggplot(data=df,aes(x=V3, y=V2)) + geom_boxplot()
```



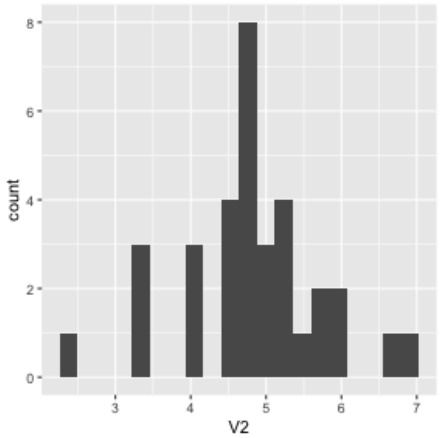
Geoms pour décrire une variable
geom_violin()

```
ggplot(data=subset(df,V3=="A"), aes(x=V3, y = V2)) + geom_violin()
ggplot(data=df,aes(x=V3, y=V2)) + geom_violin()
```



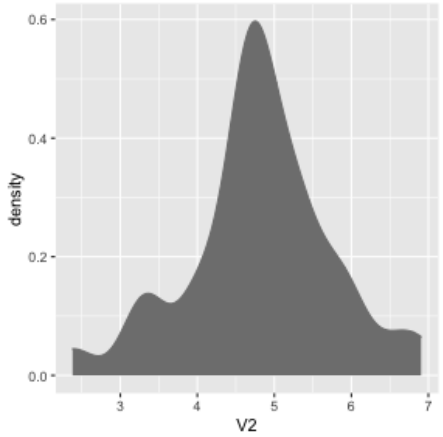
Geoms pour décrire une variable
geom_histogram()

```
ggplot(data=subset(df,V3=="A"),aes(x=V2)) + geom_histogram(bins=20)
```



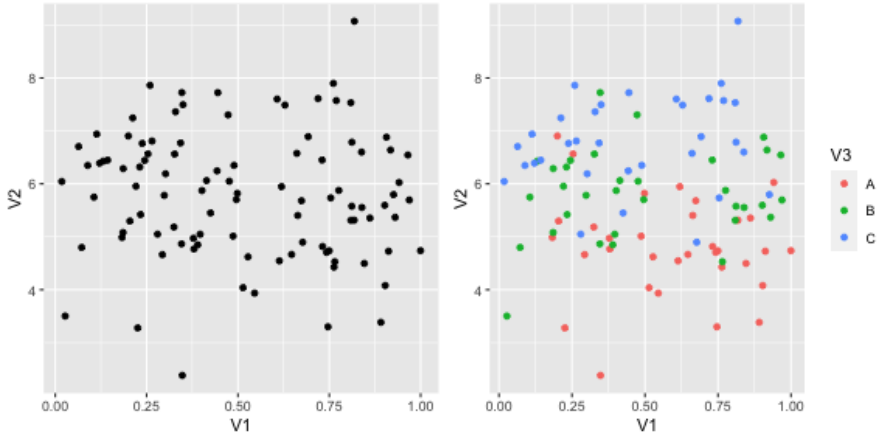
Geoms pour décrire une variable
geom_density()

```
ggplot(data=subset(df,V3=="A"),aes(x=V2)) +
  geom_density(fill="gray50", colour="gray50")
```



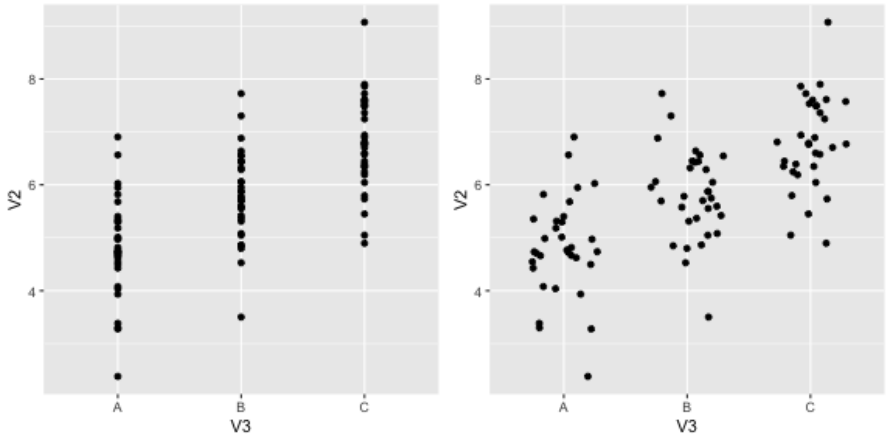
Geoms pour décrire les liens entre variables
geom_point()

```
ggplot(data=df,aes(x = V1, y=V2)) + geom_point()
ggplot(data=df,aes(x = V1, y=V2)) + geom_point(aes(colour = V3))
```



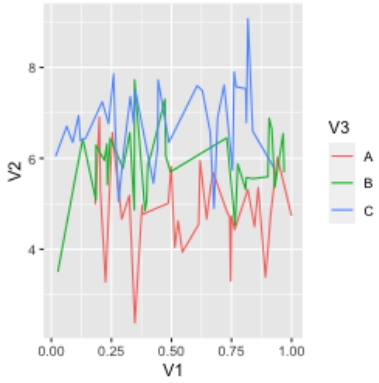
Geoms pour décrire les liens entre variables
geom_jitter()

```
ggplot(data=df,aes(x=V3, y=V2)) + geom_point()

ggplot(data=df,aes(x=V3, y=V2)) + geom_jitter(width=0.30)
```



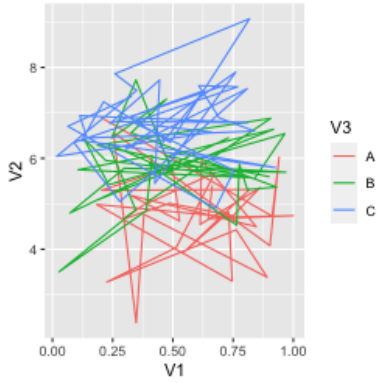
Geoms pour décrire les liens entre variables
geom_line(), geom_path()

```
ggplot(data=df,aes(x = V1, y=V2))
  geom_line(aes(colour = V3))
```



Points reliés par ordre croissant de x

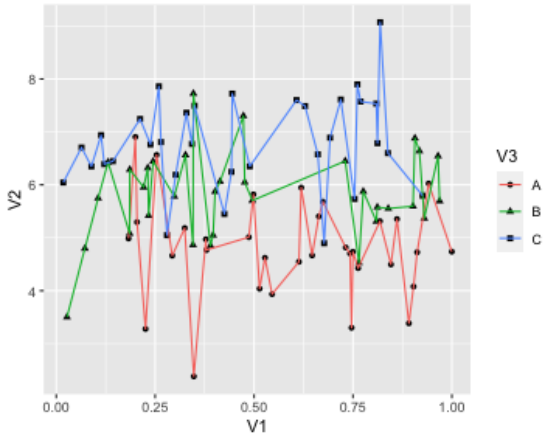
```
ggplot(data=df,aes(x = V1, y=V2))
  geom_path(aes(colour = V3))
```



Points reliés par ordre d'apparition dans le tableau de données

Combiner les geoms

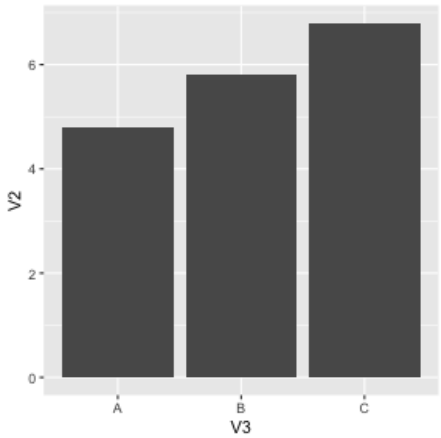
```
ggplot(data=df,aes(x = V1, y=V2)) +
  geom_point(aes(shape= V3)) +
  geom_line(aes(colour = V3))
```



Geoms pour décrire les liens entre variables

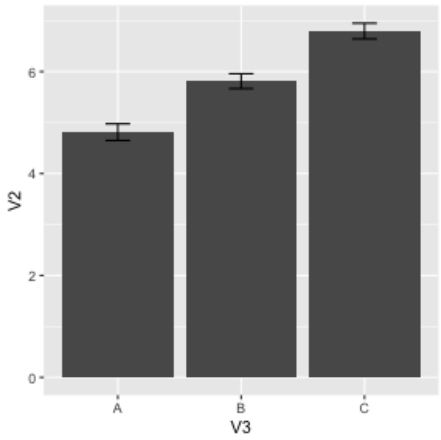
geom_bar(), geom_col()

```
dfaggr = aggregate(df[,1:2], by=list(V3 = df$V3), mean)
ggplot(data=dfaggr,aes(x = V3,y=V2)) +
  geom_col() # ou geom_bar(stat='identity')
```



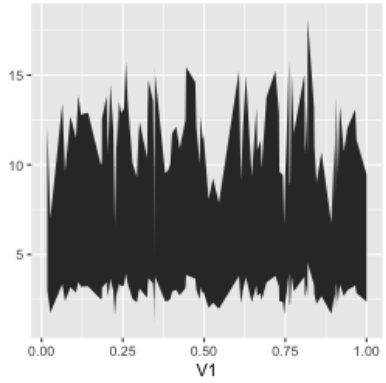
Geoms pour décrire les liens entre variables
geom_linerange(), geom_errorbar()

```
dfaggr$sd = aggregate(df[,2], by=list(V3 = df$V3), sd)$x
ggplot(data=dfaggr,aes(x = V3,y=V2)) + geom_col() +
  geom_errorbar(aes(ymin=V2-sd/sqrt(33), ymax = V2+sd/sqrt(33),widt
```



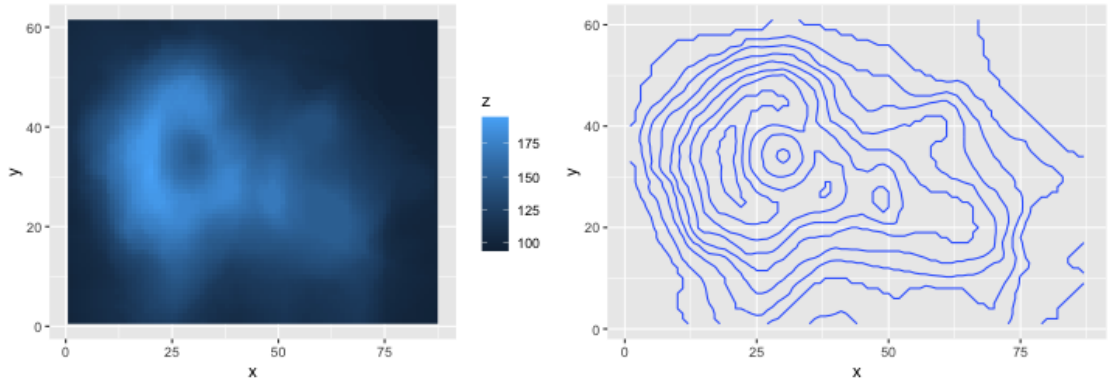
Geoms pour représenter des surfaces
geom_ribbon()

```
ggplot(data=df,aes(x = V1, ymin=V2/2, ymax=2*V2)) +
  geom_ribbon()
```



Geoms pour les surfaces
geom_tile(), geom_rect() et geom_raster()

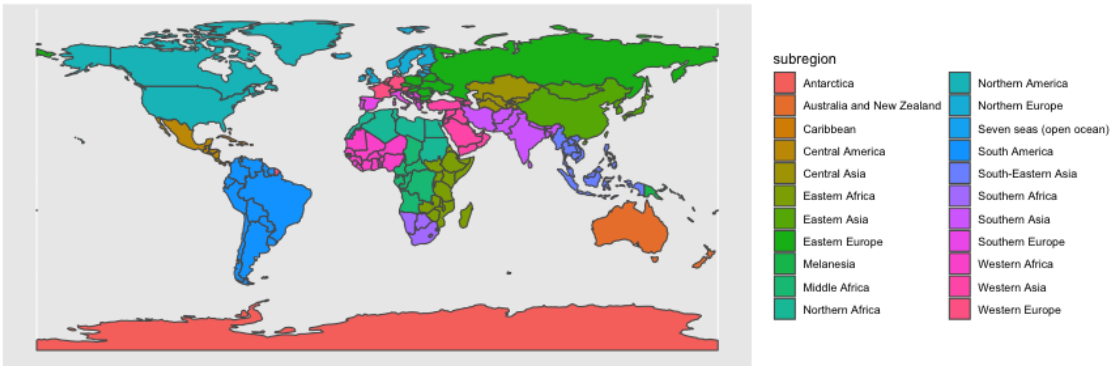
```
ggplot(data=cbind(expand.grid(x=1:dim(volcano)[1], y=1:dim(volcano)
                z=as.vector(volcano)),aes(x=x, y=y)) +
  geom_raster(aes(fill=z))
ggplot(data=cbind(expand.grid(x=1:dim(volcano)[1],  y=1:dim(volcanc
                  z=as.vector(volcano)),aes(x=x, y=y)) +
  geom_contour(aes(z=z))
```



geom_tile() est synonyme de *geom_raster()* mais permet de gérer des dalles non carrées.

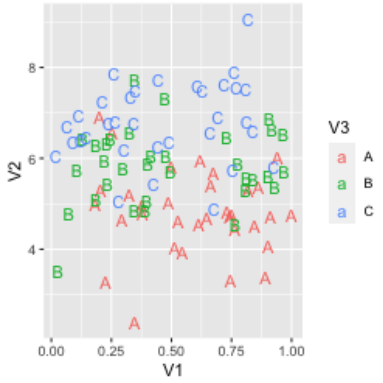
Geoms pour les cartes
geom_sf()

```
library(rnaturalearth)
wrld = ne_countries(returnclass="sf")
ggplot(wrld) + geom_sf(aes(fill=subregion))
```



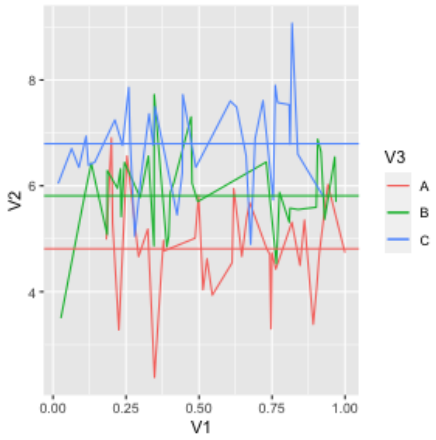
Geoms pour écrire du texte
geom_text()

```
ggplot(data=df,aes(x = V1, y=V2)) +
  geom_text(aes(label=V3,colour=V3))
```



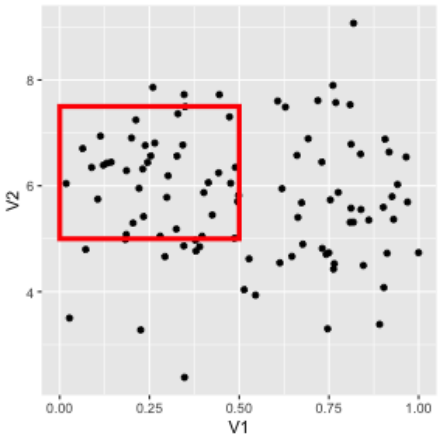
Pour annoter des graphiques
geom_hline(), geom_vline(), geom_abline()

```
ggplot(data=df,aes(x = V1, y=V2)) +
  geom_line(aes(colour= V3)) +
  geom_hline(data = aggregate(df$V2,by=list(V3=df$V3),mean),
             aes(yintercept=x,colour=V3))
```



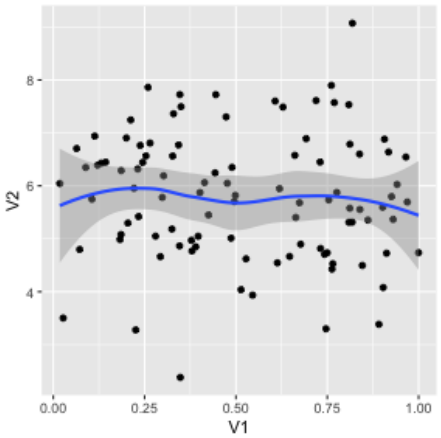
Pour annoter des graphiques
geom_rect(), geom_tile()

```
ggplot(data=df,aes(x = V1, y=V2)) + geom_point()+
  geom_rect(data=data.frame(V1=0, x1=0.5,V2=5,y1=7.5),
            aes(xmin=V1,ymin=V2, xmax=x1,ymax=y1),colour="red",fill
```

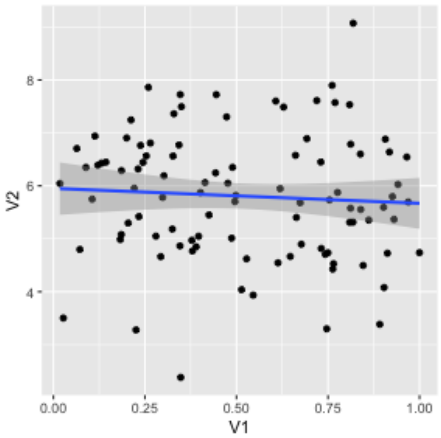


Pour annoter des graphiques
les fonctions stat_...()

```
ggplot(data=df,aes(x = V1, y=V2)
  geom_point() +
  stat_smooth()
```

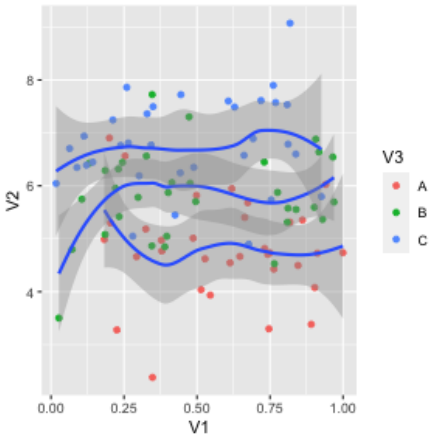



```
ggplot(data=df,aes(x = V1, y=V2)
  geom_point() +
  stat_smooth(method="lm")
```



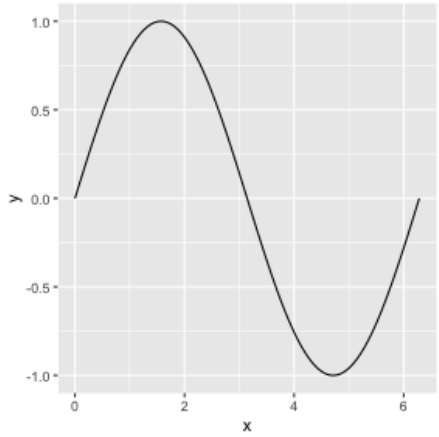
Pour annoter des graphiques
les fonctions stat_...()

```
ggplot(data=df,aes(x = V1, y=V2, group=V3)) +
  geom_point(aes(colour=V3)) +
  stat_smooth()
```



Pour annoter des graphiques
les fonctions stat_...()

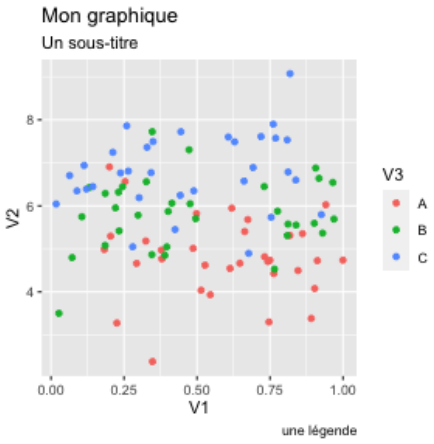
```
ggplot(data.frame(x=c(0,2*pi)),aes(x)) +
  stat_function(fun=sin)
```



Pour annoter des graphiques

Mettre un titre

```
ggplot(data=df,aes(x = V1, y=V2)) + geom_point(aes(colour = V3))+
  labs(title="Mon graphique",
       subtitle = "Un sous-titre",
       caption = "une légende")
```



les fonctions `annotate`

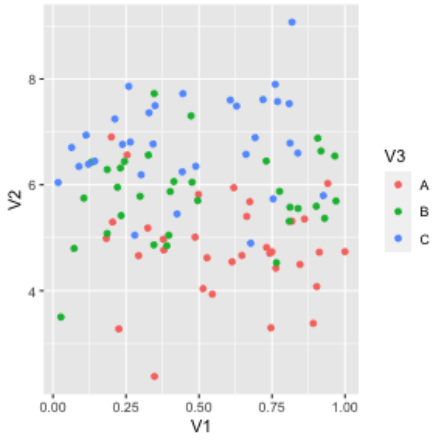
```
ggplot(data=df,aes(x = V1, y=V2)) + geom_point(aes(colour = V3))+
  annotate("text",x=0.5,y=6,
           label="Secret",size=20,col="red",angle=30)
```

Mise en forme

Graphique:

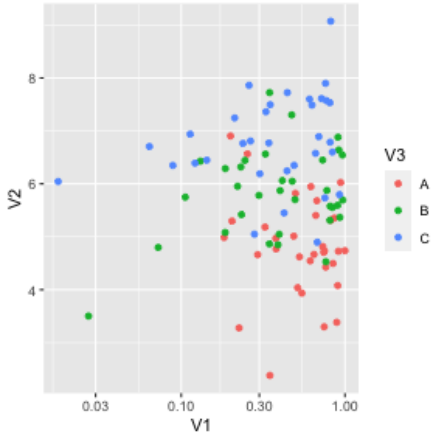
Dans la suite, on va travailler à partir du graph suivant:

```
g = ggplot(data=df,aes(x = V1, y=V2)) + geom_point(aes(colour = V3)
```

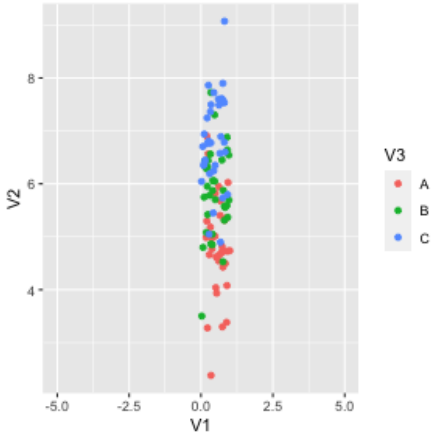


Les échelles
Les axes: scale*x,y*...()

```
g +
  scale_x_log10()
```



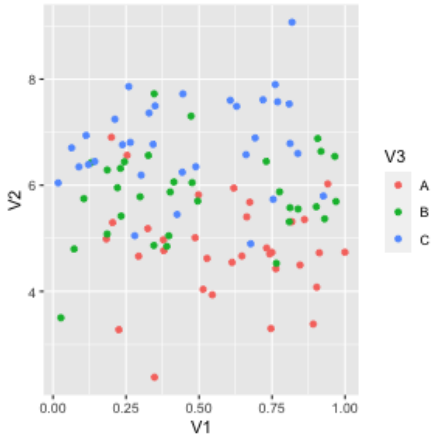

```
g + scale_x_continuous(
  limits = c(-5,5))
```



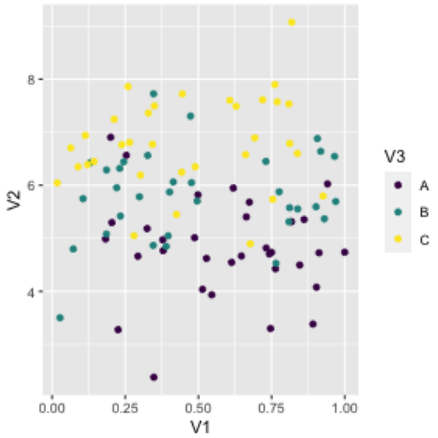
Attention: Cette manière de redéfinir les limites des axes *supprime* les données situées hors de cette gamme ! Il faut utiliser les arguments **`xlim`** et **`ylim`** de **`coord_cartesian()`** pour redéfinir les axes sans perte de données.

Les échelles
couleurs : scale*colour*...()

```
g
```

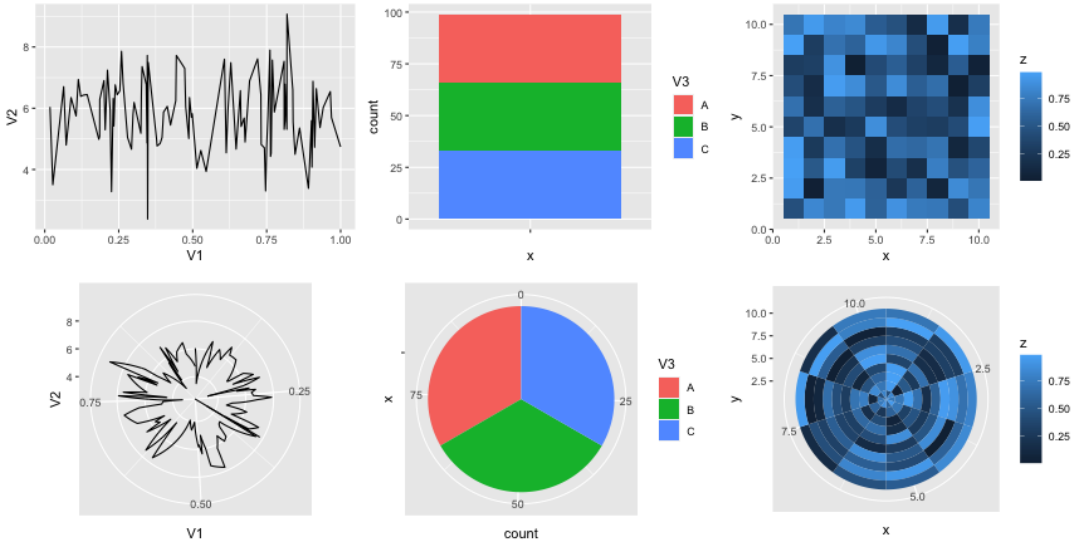



```
g +  scale_colour_viridis_d()
```



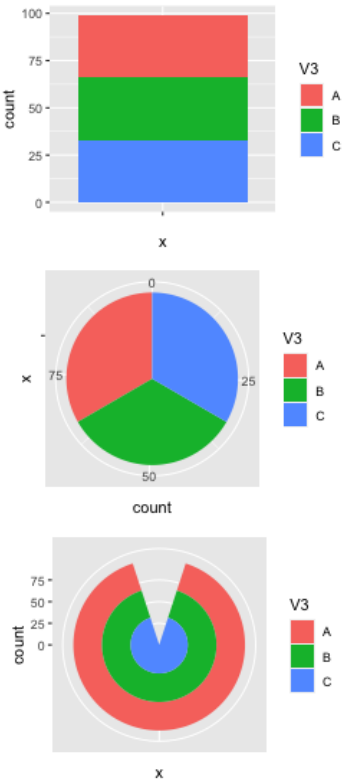
Les axes: coord_*()

```
p                    p                      p
p+ coord_polar()     p + coord_polar("y")   p + coord_polar()
```

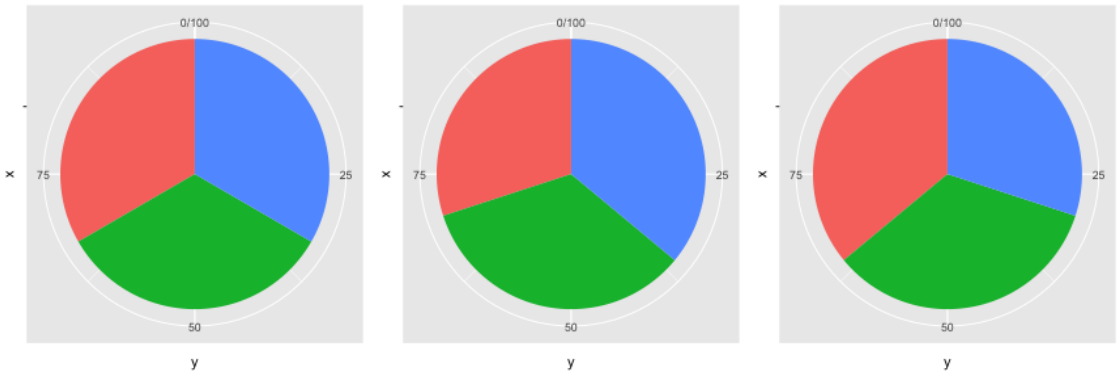


Les axes

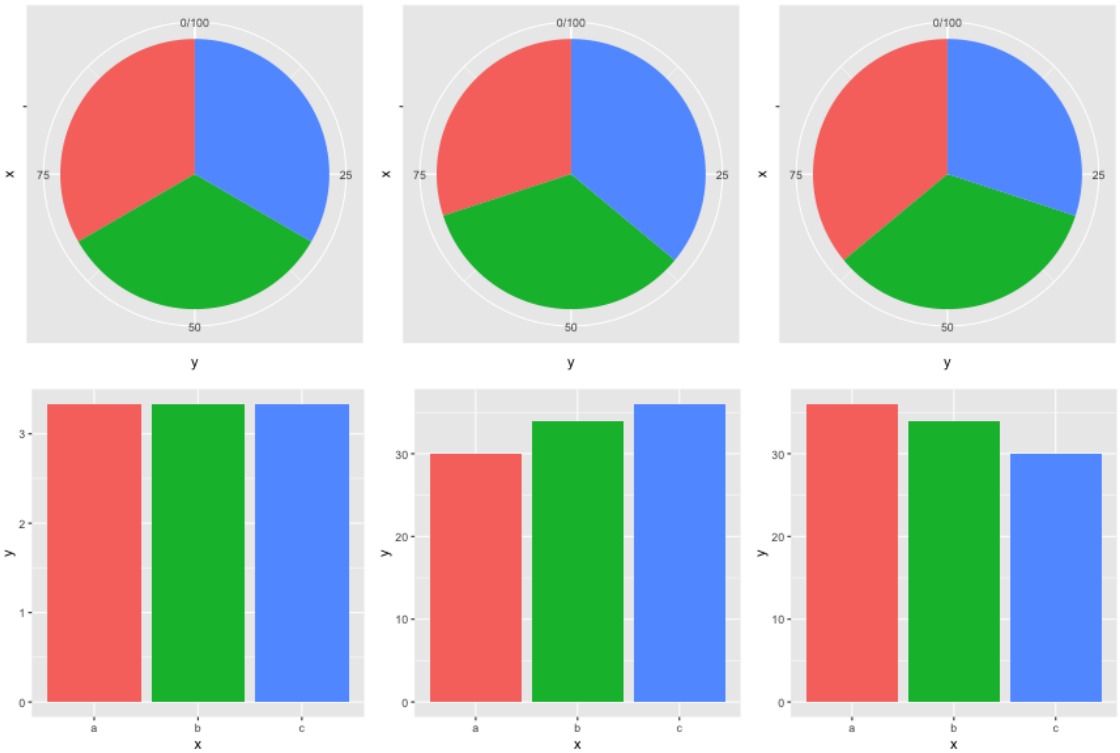
```
ggplot(df,aes(x="")) +
  geom_bar(aes(fill=V3))

ggplot(df,aes(x="")) +
  geom_bar(aes(fill=V3)) +
  coord_polar("y",clip = "off",start=0)

ggplot(df,aes(x="")) +
  geom_bar(aes(fill=V3)) +
  coord_polar("x",clip = "off",start=0)
```



Les axes
Les camemberts: une bonne idée ?

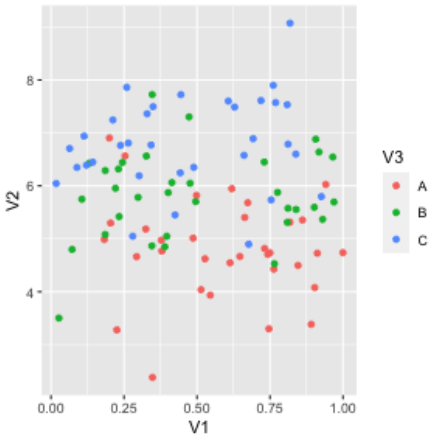


Les axes
Les camemberts: une bonne idée ?



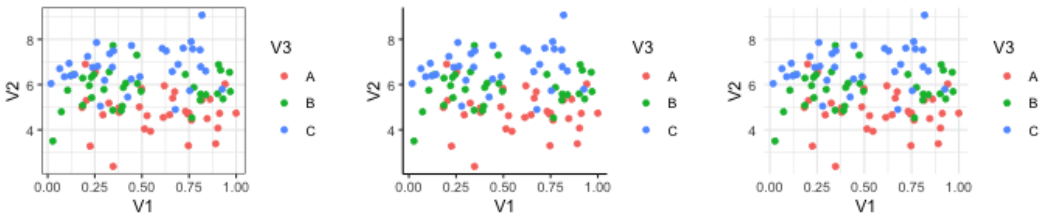
Les themes par defaut

```
(g = ggplot(data=df,aes(x = V1, y=V2)) + geom_point(aes(colour = V3
```



Les themes par defaut

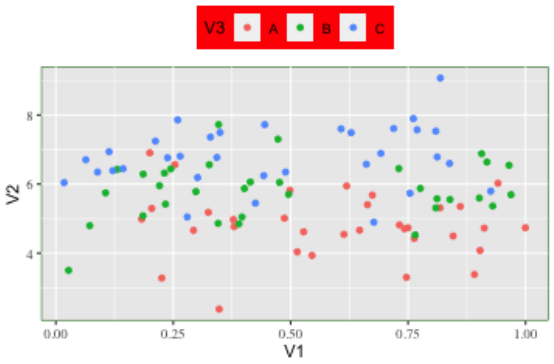
```
g + theme_bw()        g + theme_classic()   g + theme_minimal()
```



Les thèmes **`theme_...()`** ne modifient que l'allure générale du graphique et ne modifient pas la couleur des points (définis via un **`aes()`**). Il faut également modifier l'échelle de couleurs avec **`scale_colour_discrete()`** si l'on veut un graph en noir & blanc, par exemple.

Les mains dans le cambouis avec theme()

```
g + theme(aspect.ratio=0.5, legend.position = "top",
          legend.background = element_rect(fill="red"),
          panel.background = element_rect(colour="darkgreen"),
          axis.text=element_text(family="Times"))
```

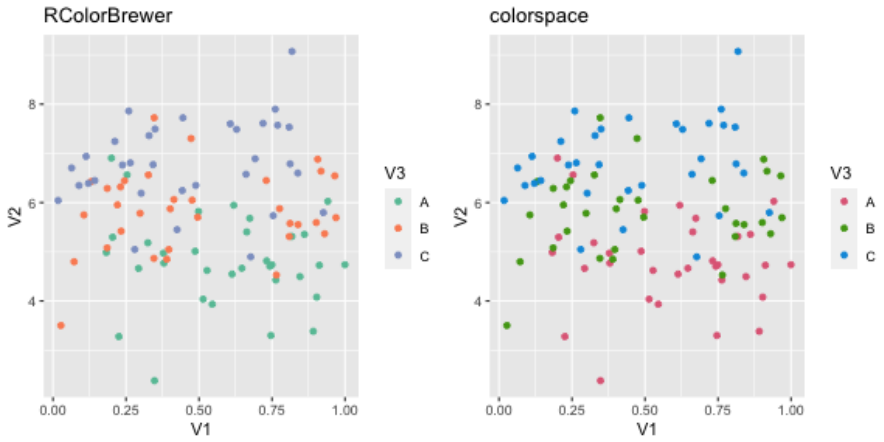


Pour aller plus loin: les extensions

ggplot2 possède une architecture permettant d'étendre assez facilement ses capacités via les extensions. Ces extensions ajoutent souvent des nouvelles fonctions **scale_*** ou **geom_***. La liste des extensions supportées officiellement est disponible à https://exts.ggplot2.tidyverse.org/gallery/

Changer les couleurs
colorspace, RColorBrewer...

```
library(RColorBrewer) #http://colorbrewer2.org
g + scale_color_brewer(type='qual',palette="Set2") + labs(title="RC
library(colorspace)
g +  scale_colour_discrete_qualitative()+ labs(title="colorspace")
```

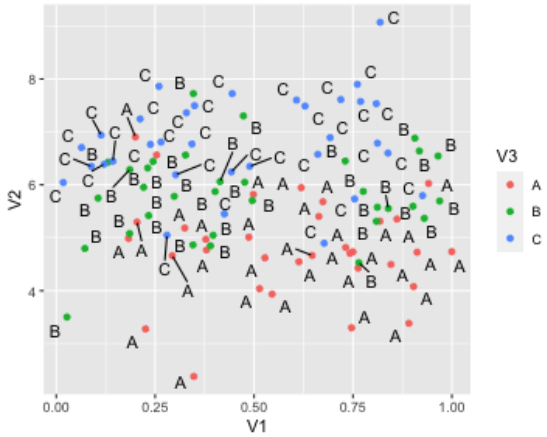


Une liste relativement exhaustive de toutes les palettes disponibles pour R est disponible à https://github.com/EmilHvitfeldt/r-color-palettes

Améliorer les textes
ggrepel

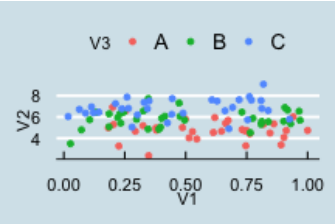
```
library(ggrepel)

ggplot(data=df,aes(x = V1, y=V2)) + geom_point(aes(colour=V3))+
  geom_text_repel(aes(label=V3))
```

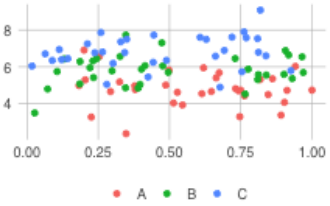
```
## Warning: ggrepel: 1 unlabeled data points (too many overlaps). Consider
## increasing max.overlaps
```

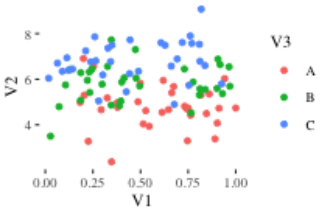


Les themes ggthemes

```
g + theme_economist() g + theme_excel_new() g + theme_tufte()
```



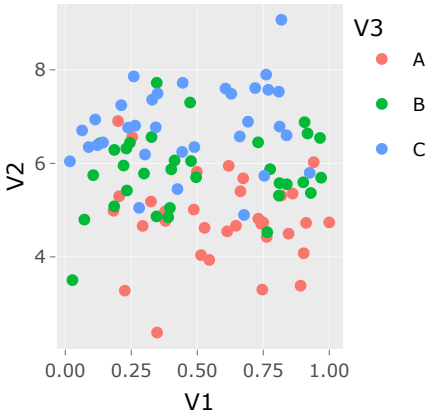




Rendre ses graphs interactifs

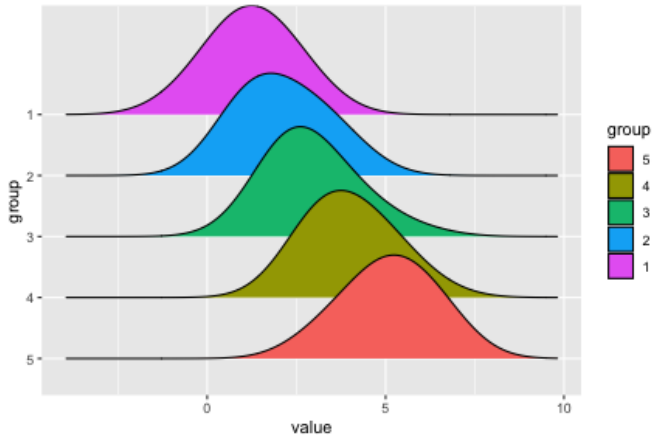
plot.ly

```
library(plotly)
ggplotly(g)
```



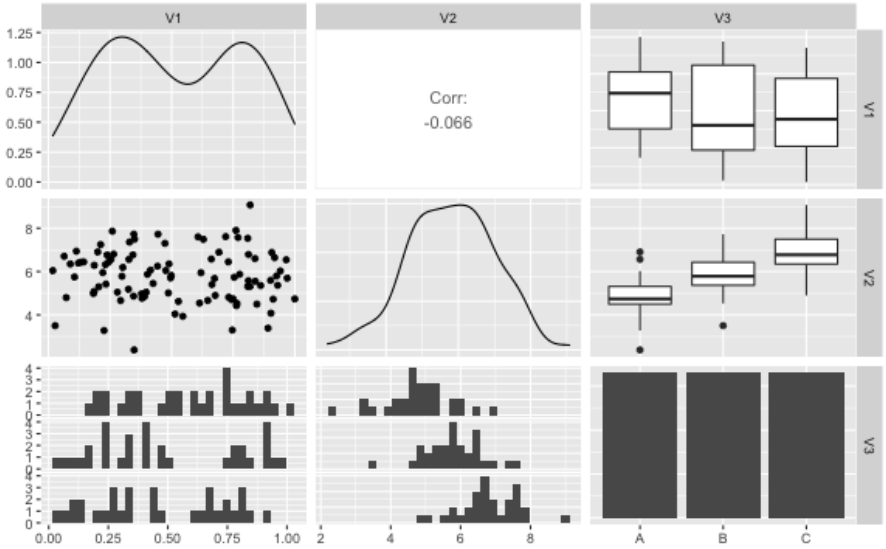
Nouveaux geoms
Joy plots avec ggridges:

```
library(ggridges)
norm_data <- data.frame(var = rep(1:5, each = 30))
norm_data$group <- factor(norm_data$var, levels=5:1)
norm_data$value <- rnorm(nrow(norm_data), norm_data$var,1)

ggplot(norm_data,aes(x=value,y=group,group=group,fill=group)) +
  geom_density_ridges(bandwidth=1)
```



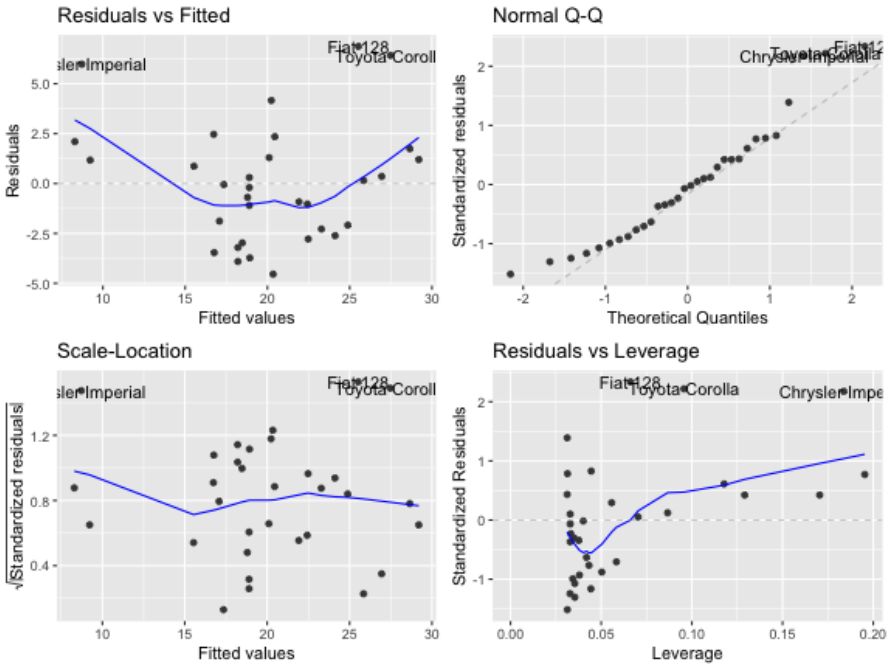
Les plots classiques
GGally pour remplacer pairs()

```
library(GGally)
ggpairs(df)
```



Les plots classiques
ggfortify pour les modèles

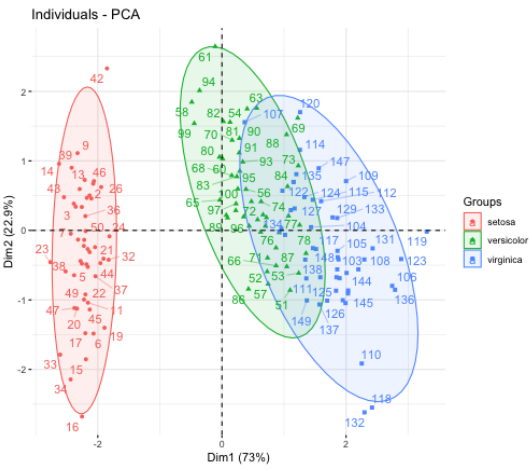
```
library(ggfortify)
autoplot(lm(mpg~wt,data=mtcars))
```



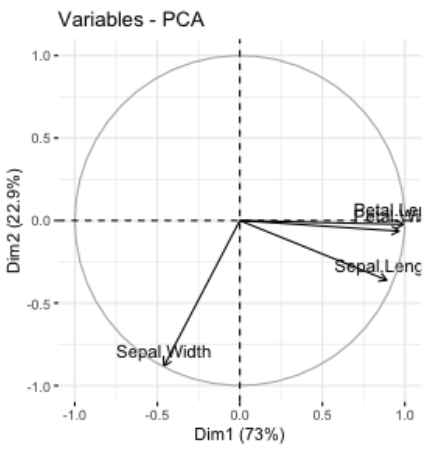
Pour les analyses multivariées
factoextra (compatible avec prcomp, ade4, factominer)

```
library(factoextra)
res.pca <- prcomp(iris[, -5],  scale = TRUE)
```

```
fviz_pca_ind(res.pca, habillage=
```

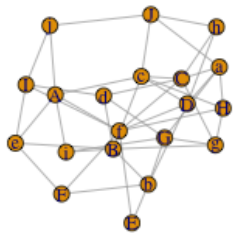



```
fviz_pca_var(res.pca)
```



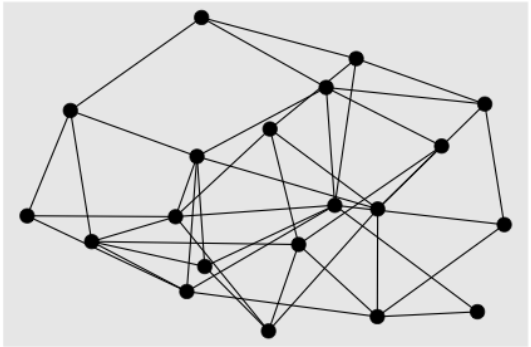
Représenter des réseaux
le paquet igraph

```
library(igraph)
m <- as.data.frame(matrix(rbinom(100,1,.4),10,10))
names(m) = LETTERS[1:10]
row.names(m) = letters[1:10]
reseau = graph_from_incidence_matrix(m)
plot(reseau)
```

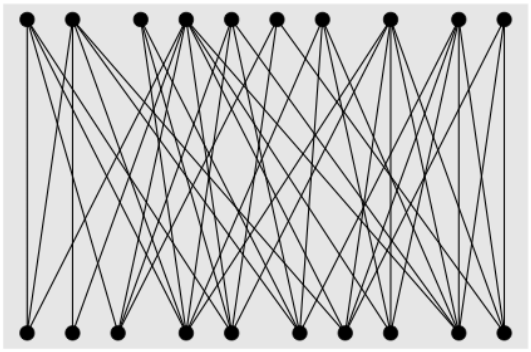


Représenter des réseaux
le paquet ggraph

```
library(ggraph)
ggraph(reseau) +
  geom_edge_fan( show.legend = F
  geom_node_point(size=5)
```



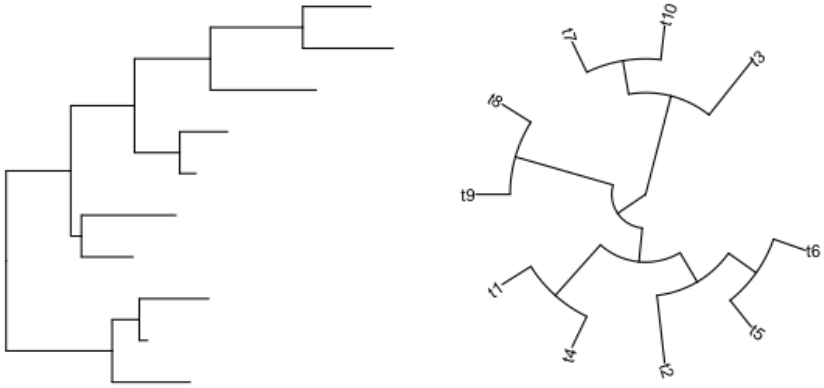
```
ggraph(reseau,layout="bipartite"
  geom_edge_fan( show.legend = F
  geom_node_point(size=5)
```



ggraph peut faire des hiveplots.

Représenter des réseaux

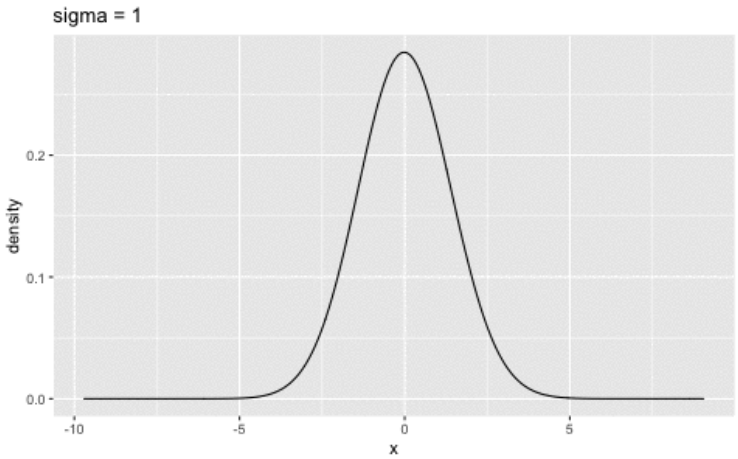
Les arbres phylogénétiques avec ggtree (Bioconductor)



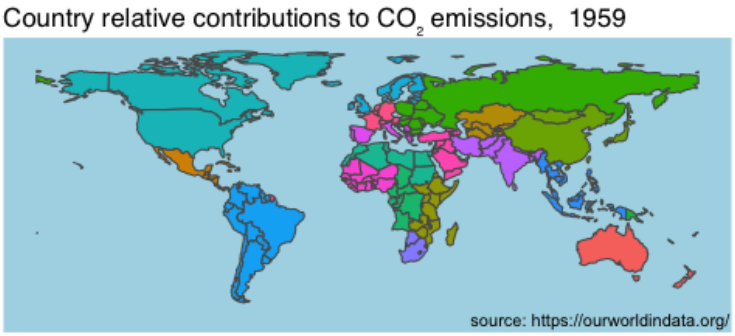
Animer des graphiques

gganimate

```
library(gganimate)
animate(
  ggplot(df2,aes(x=x)) + geom_density(bw=1) +
    transition_states(y) + ease_aes('linear') +
    labs(title = "sigma = {closest_state}"),
  height=300)
```



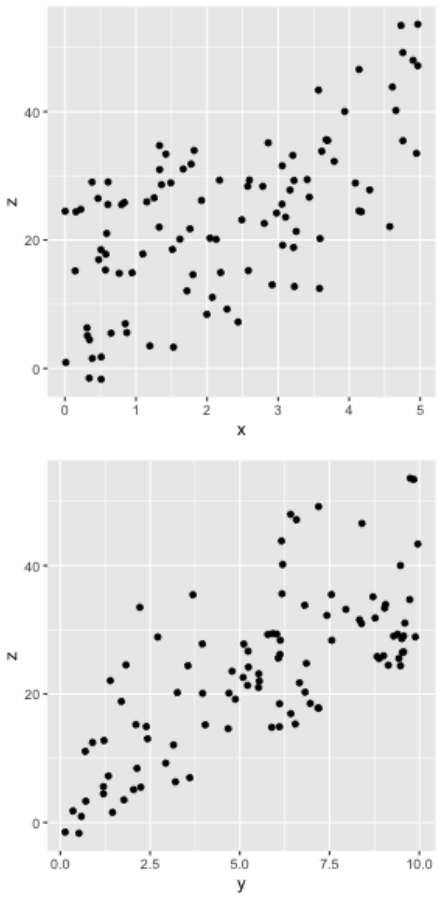
Animer des graphiques
gganimate



Visualisations 3D

Supposons

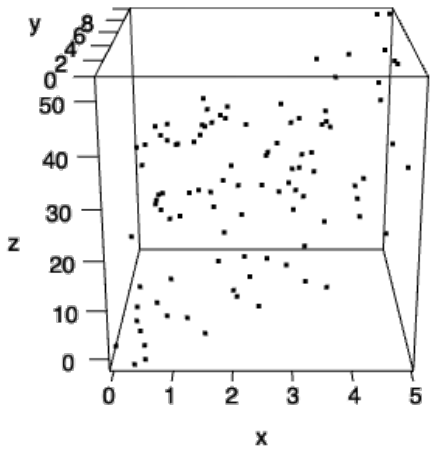
$$z = x^2 + 3 \times y + \epsilon$$



Visualisations 3D

rgl

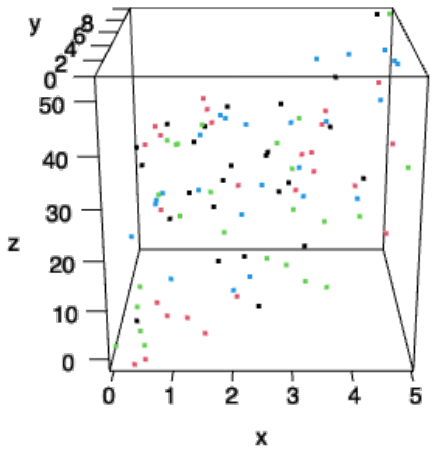
```
library(rgl)
plot3d(x,y,z)
rglwidget(width=400,height=400)
```



Visualisations 3D

rgl

```
plot3d(x,y,z,col=rep(1:4,each=25))
rglwidget(width=400,height=400)
```



Visualisations 3D

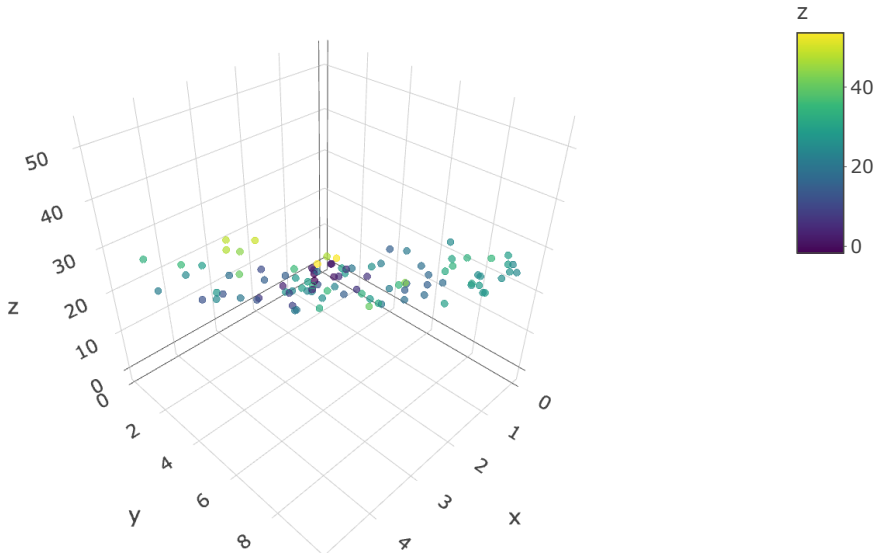
rgl

```
xseq = 0:5;yseq = 0:10
surf=expand.grid(x=xseq,y=yseq)
surfmat = matrix( surf$x^2 + 3*surf$y ,nrow=length(xseq))
plot3d(x,y,z,col=rep(1:4,each=25))
persp3d(xseq,yseq,surfmat,col="gray",add=T)
rglwidget(width=400,height=400)
```


Visualisations 3D

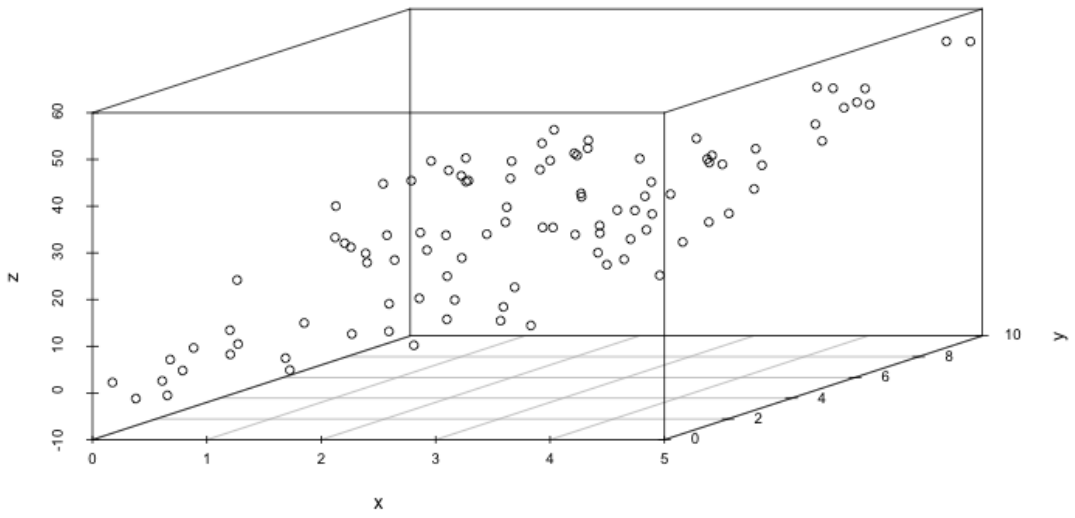
plotly

```
library(plotly)
plot_ly(data =data.frame(x=x, y=y, z=z), x=~x, y=~y, z=~z, color =
  add_markers(size=1) %>%
  layout(scene= list(xaxis = list(title = 'x'), yaxis = list(title
```



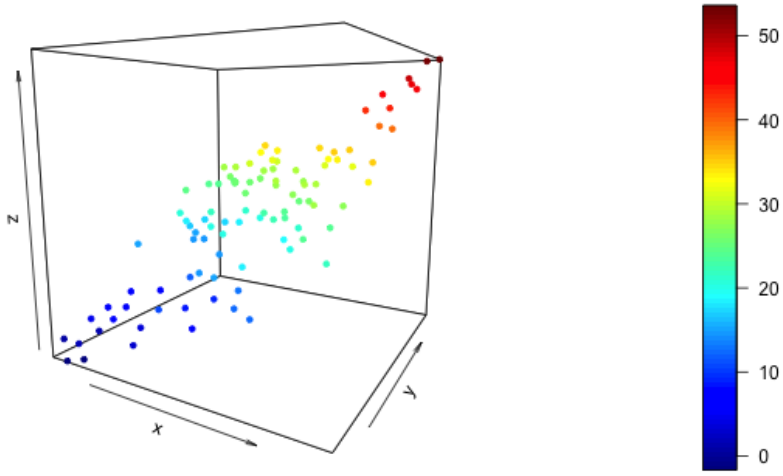
Visualisations 3D
scatterplot3d

```
library(scatterplot3d)
scatterplot3d(x,y,z)
```



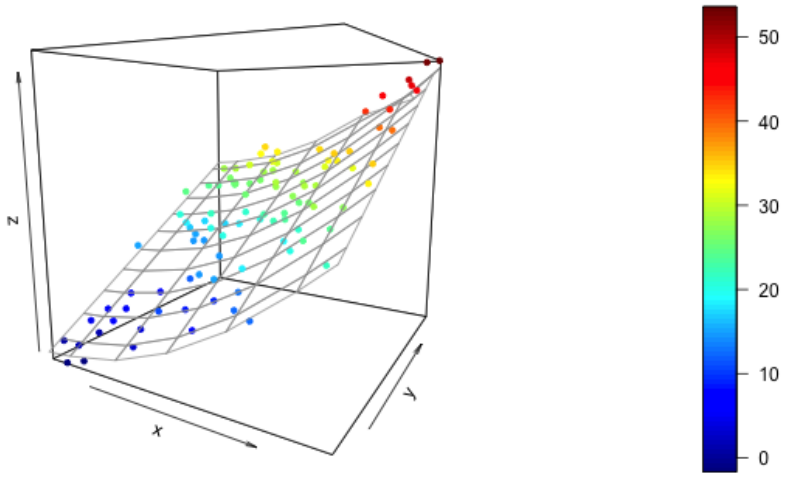
Visualisations 3D
plot3D

```
library(plot3D)

scatter3D(x,y,z,pch=20,phi=10,theta=30)
```



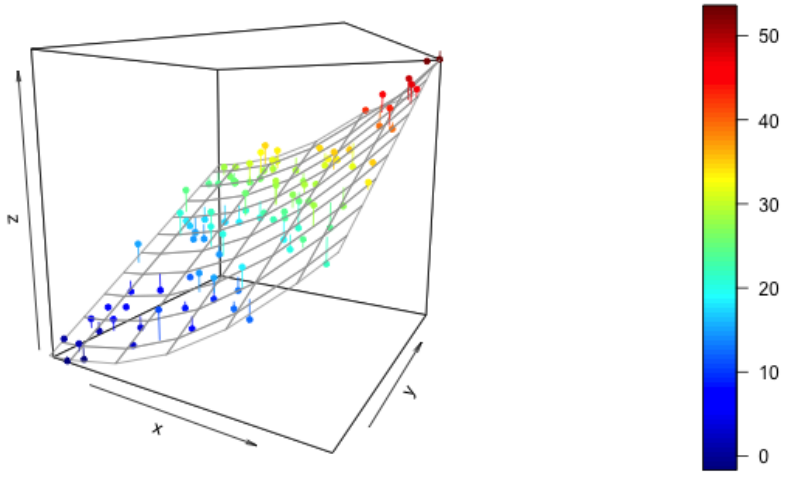
Visualisations 3D
plot3D

```
scatter3D(x,y,z,pch=20,phi=10,theta=30,
          surf=list(x=xseq,y=yseq,z=surfmat,col=8,facets=NA,fill=F)
```



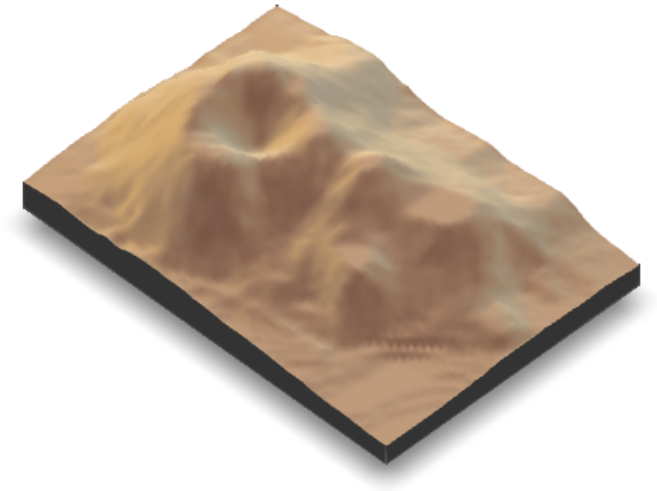
Visualisations 3D
plot3D

```
model = lm(z~I(x^2)+y)
surfmat2 = matrix(coef(model)[1] + coef(model)[2]*surf$x^2 + coef(m
scatter3D(x,y,z,surf=list(x=xseq,y=yseq,z=surfmat2,col=8,facets=NA,
```

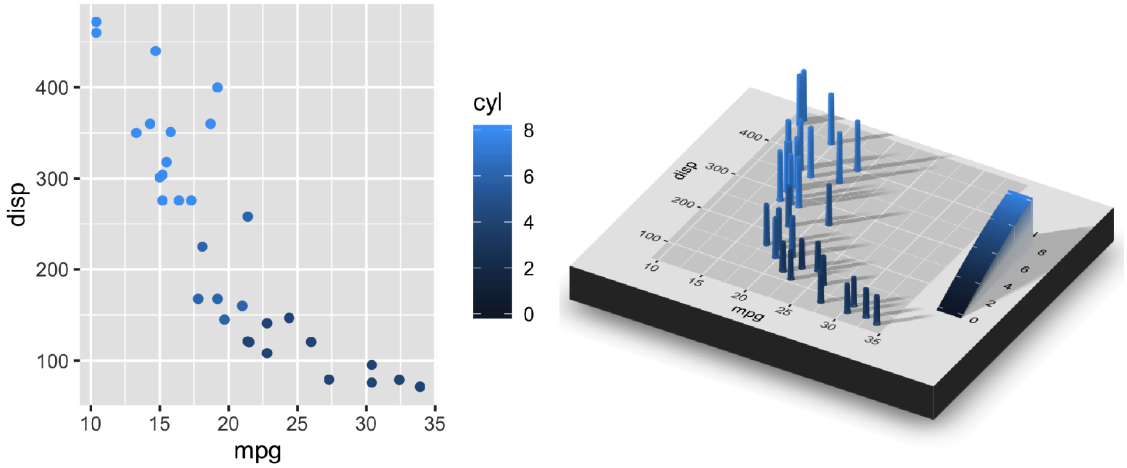


Visualisations 3D
rayshader

```
library(rayshader)
plot_3d(sphere_shade(volcano,texture="desert",progbar=F),volcano,zs
rglwidget()
```

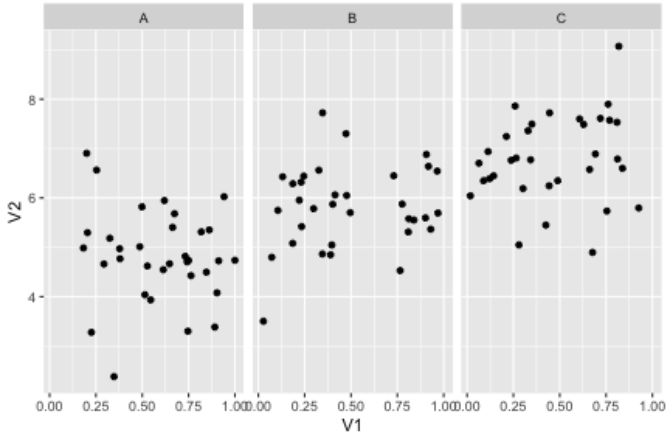


Visualisations 3D
rayshader



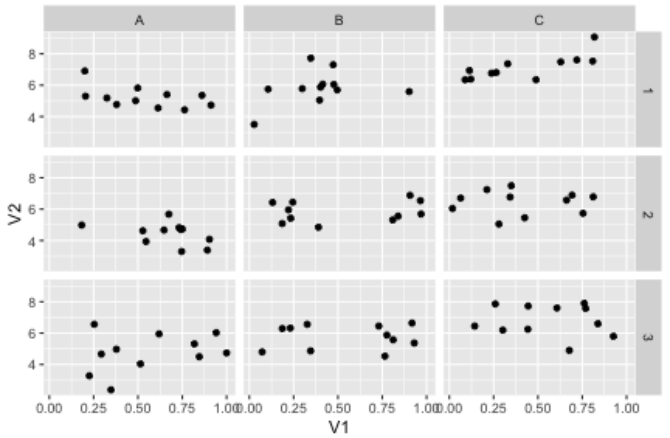
Faire des graphs complexes: les facets
facet_grid(), facet_wrap()

```
ggplot(df,aes(x = V1, y=V2)) + geom_point() +
  facet_grid(cols=vars(V3))
```



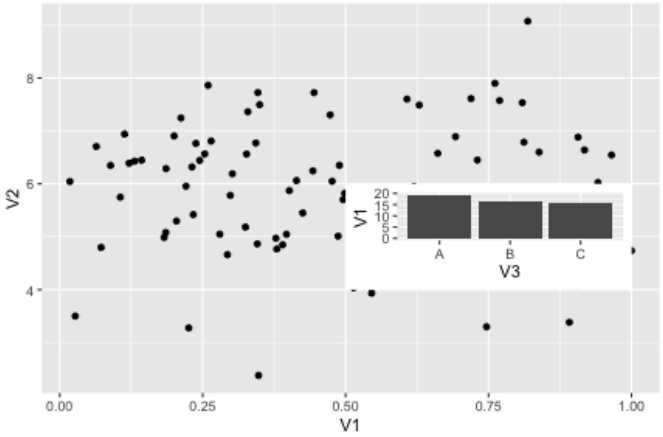
Faire des graphs complexes: les facets
facet_grid(), facet_wrap()

```
df$V4 = sample(rep(1:3, each=11))
ggplot(df,aes(x = V1, y=V2)) + geom_point() +
  facet_grid(cols=vars(V3), rows=vars(V4))
```



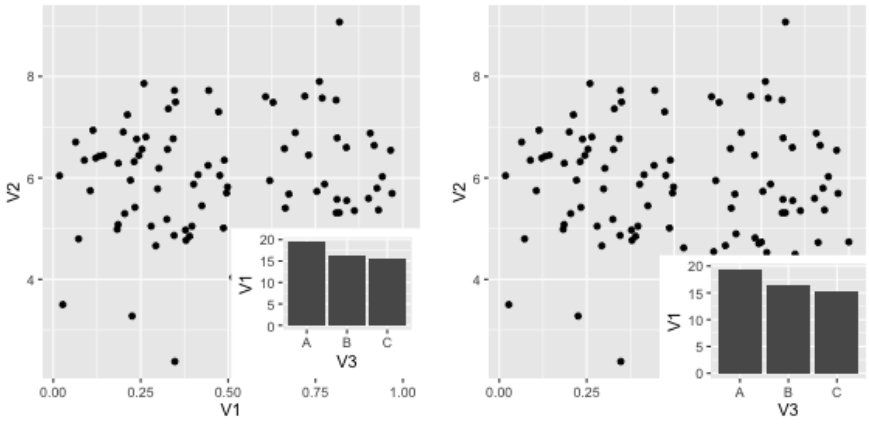
Faire des graphs complexes: encarts
draw_plot()

```
library(cowplot)
inset = ggplot(df, aes(x=V3,y=V1)) +  geom_col()
principal = ggplot(df, aes(x=V1,y=V2)) +  geom_point()

principal + draw_plot(inset,0.5,4,0.5,2)
```



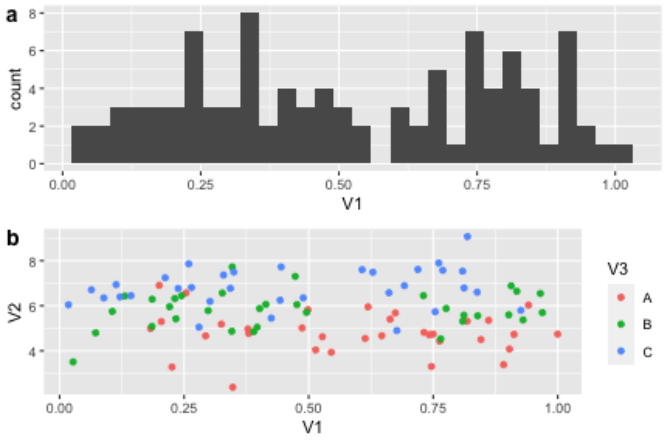
Faire des graphs complexes: encarts
inset_element()

```
library(patchwork)

inset = ggplot(df, aes(x=V3,y=V1)) +  geom_col()
principal = ggplot(df, aes(x=V1,y=V2)) +  geom_point()

principal + inset_element(inset,0.5,0, 1, 0.4) # left, bottom, righ
principal + inset_element(inset,0.5,0, 1, 0.4, align_to="full")
```



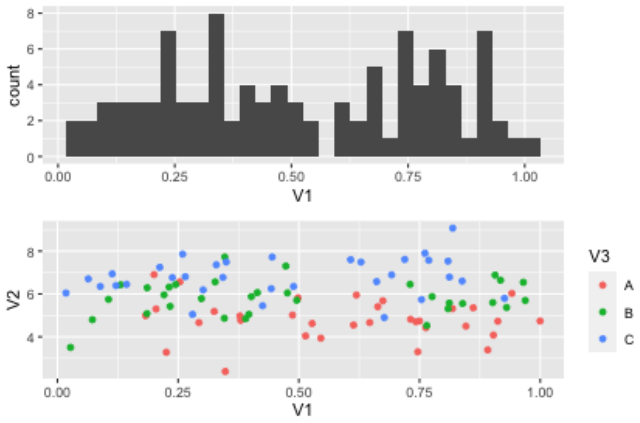
Faire des graphs complexes: assemblages
cowplot

```
library(cowplot)
plot_grid(
  ggplot(data=df,aes(x=V1)) +
  geom_histogram(), g,nrow = 2,labels=letters[1:2])
```



Faire des graphs complexes: assemblages
patchwork

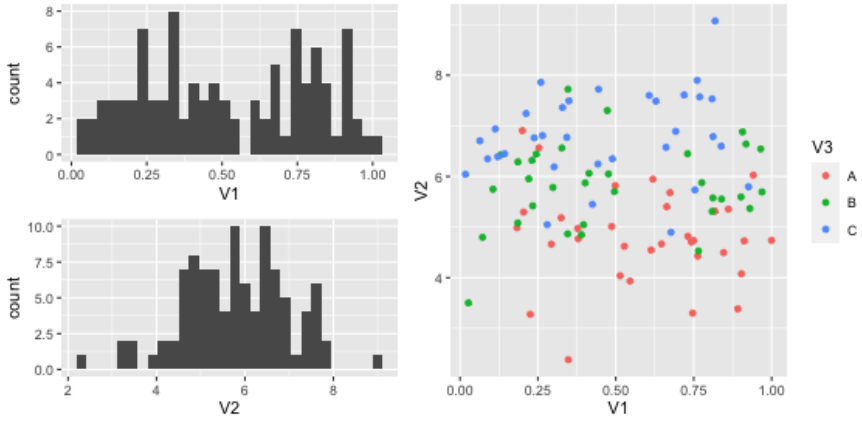
```
library(patchwork)
(ggplot(data=df,aes(x=V1)) + geom_histogram()) / g
```



Faire des graphs complexes: assemblages

patchwork

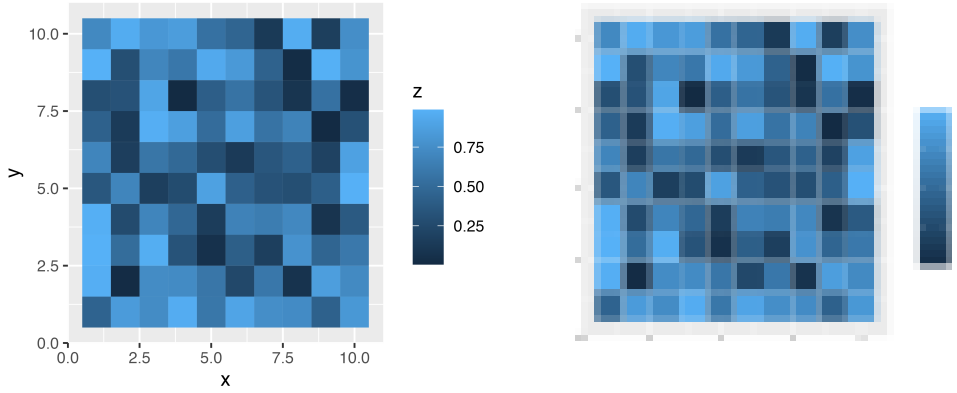
```
library(patchwork)
((ggplot(data=df,aes(x=V1)) + geom_histogram()) / (ggplot(data=df,a
```



Exporter ses graphs

ggsave(), ggsave2()

```
ggsave("highres.png", plot=p, device="png", dpi=600, width=10, heig
ggsave("lowres.png", plot=p, device="png", dpi=20, width=10, height
```



Formats supportés: jpg, png, tiff, pdf...

Fin