

Note book GGplot_3

Utilisation de GGplot_2

GGplot2 est une extension du tidyverse qui permet de générer des graphiques avec une syntaxe cohérente et puissante. Elle nécessite l'apprentissage d'un mini-langage supplémentaire, mais permet la construction de graphiques complexes de manière efficace.

Une des particularités de GGplot2 est que l'extension part du principe que les données relatives à un graphique sont stockées dans un tableau de données (data frame, tibble ou autre).

Les prérequis :

Il faut tout d'abord avoir un data frame du type

```
library(ggplot2)
head(mpg)
```

| manufacturer <chr> | model <chr> | displ <dbl> | year <int> | cyl <int> | trans <chr> | drv <chr> | cty <int> | hwy <int> | fl <chr> |
|-----------------------|----------------|----------------|---------------|--------------|----------------|--------------|--------------|--------------|-------------|
| audi | a4 | 1.8 | 1999 | 4 | auto(l5) | f | 18 | 29 | p |
| audi | a4 | 1.8 | 1999 | 4 | manual(m5) | f | 21 | 29 | p |
| audi | a4 | 2.0 | 2008 | 4 | manual(m6) | f | 20 | 31 | p |
| audi | a4 | 2.0 | 2008 | 4 | auto(av) | f | 21 | 30 | p |
| audi | a4 | 2.8 | 1999 | 6 | auto(l5) | f | 16 | 26 | p |
| audi | a4 | 2.8 | 1999 | 6 | manual(m5) | f | 18 | 26 | p |

6 rows | 1-10 of 11 columns

Une fois le tableau initialisé, il est temps de jouer avec les données. La manipulation passe par deux axes clés :

- la définition des variables sur leur axe respectifs
- le type de graphique à utiliser.

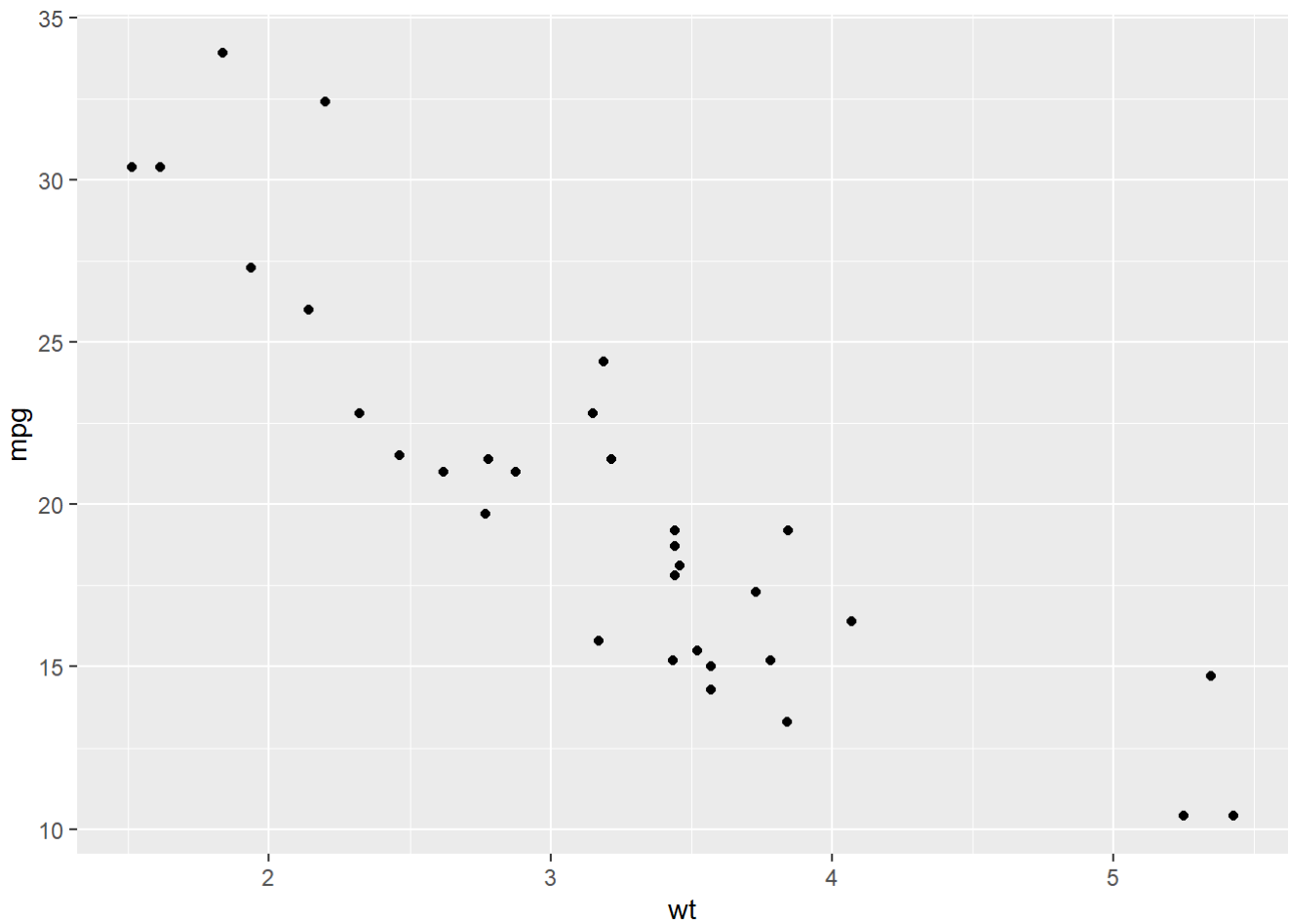
Il existe une multitude de graphiques qui ont chacun une fonction bien précise.

Le nuage de point

Definition: en statistiques, un nuage de points est une représentation de données dépendant de plusieurs variables. Il permet de mettre en évidence le degré de corrélation entre au moins deux variables liées. Les différentes observations des nuages de points permettent de déterminer : Des tendances Des dépendances. Utilisable avec la fonction : `geom_point` (qui permet de modifier : taille, couleur et forme des points).

Une fois l'initialisation terminée, il faut utiliser la fonction suivante :

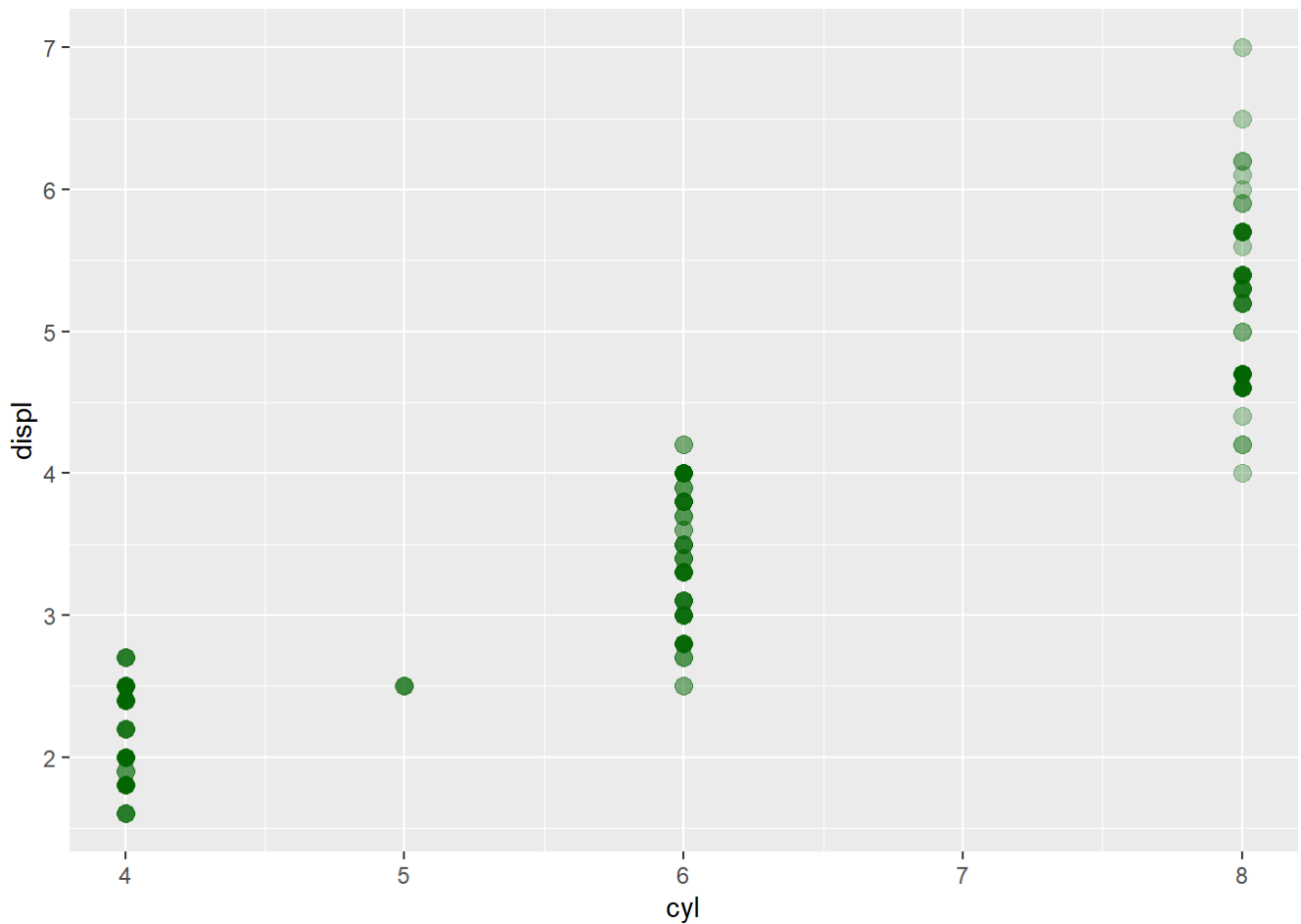
```
ggplot(mtcars, aes(x=wt, y=mpg))+ geom_point()
```



```
#(definition du DataFrame, definition des axes : aes (x = , y = )+ geom_point()
```

Pour la question des points, tout est dans la definition du "geom_point", personnalisable à l'infini

```
ggplot(mpg) +  
  geom_point(aes(x = cyl, y = displ),  
             color = "darkgreen", size = 3, alpha = 0.3)
```

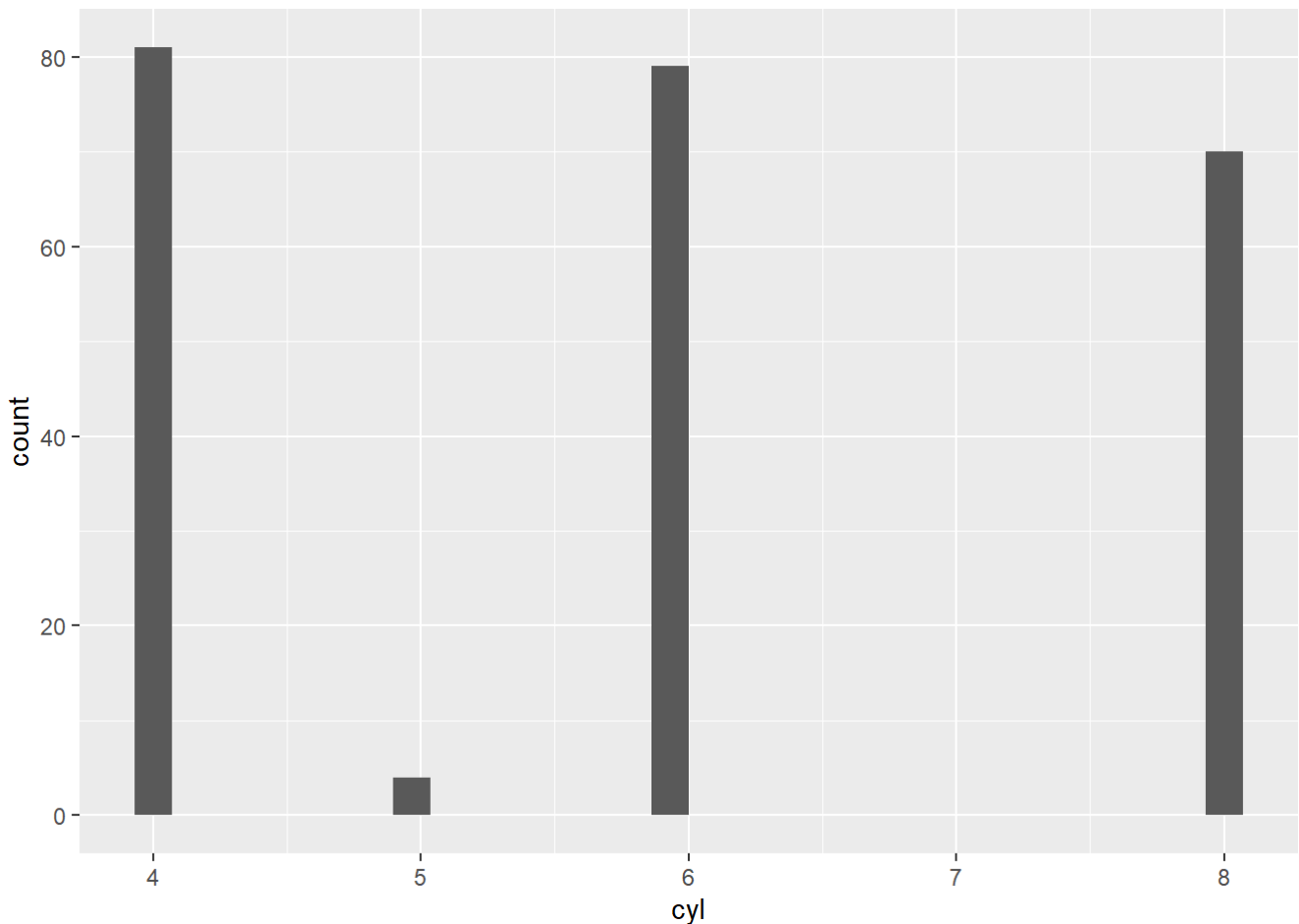


Les Histogrammes:

Definition : Représentation graphique des fréquences ou effectifs relatifs à un caractère quantitatif continu à l'aide d'une série de rectangles dont la base constitue un intervalle de variation des valeurs du caractère et la surface l'effectif correspondant. Pour cela, il faut utiliser la commande : `geom_histogram` qui se configure de la manière suivante :

```
ggplot(mpg) + geom_histogram(aes(x = cyl))
```

```
## `stat_bin()` using `bins = 30`. Pick better value with `binwidth`.
```



Définir X permet d'afficher la donnée que l'on souhaite représenter.

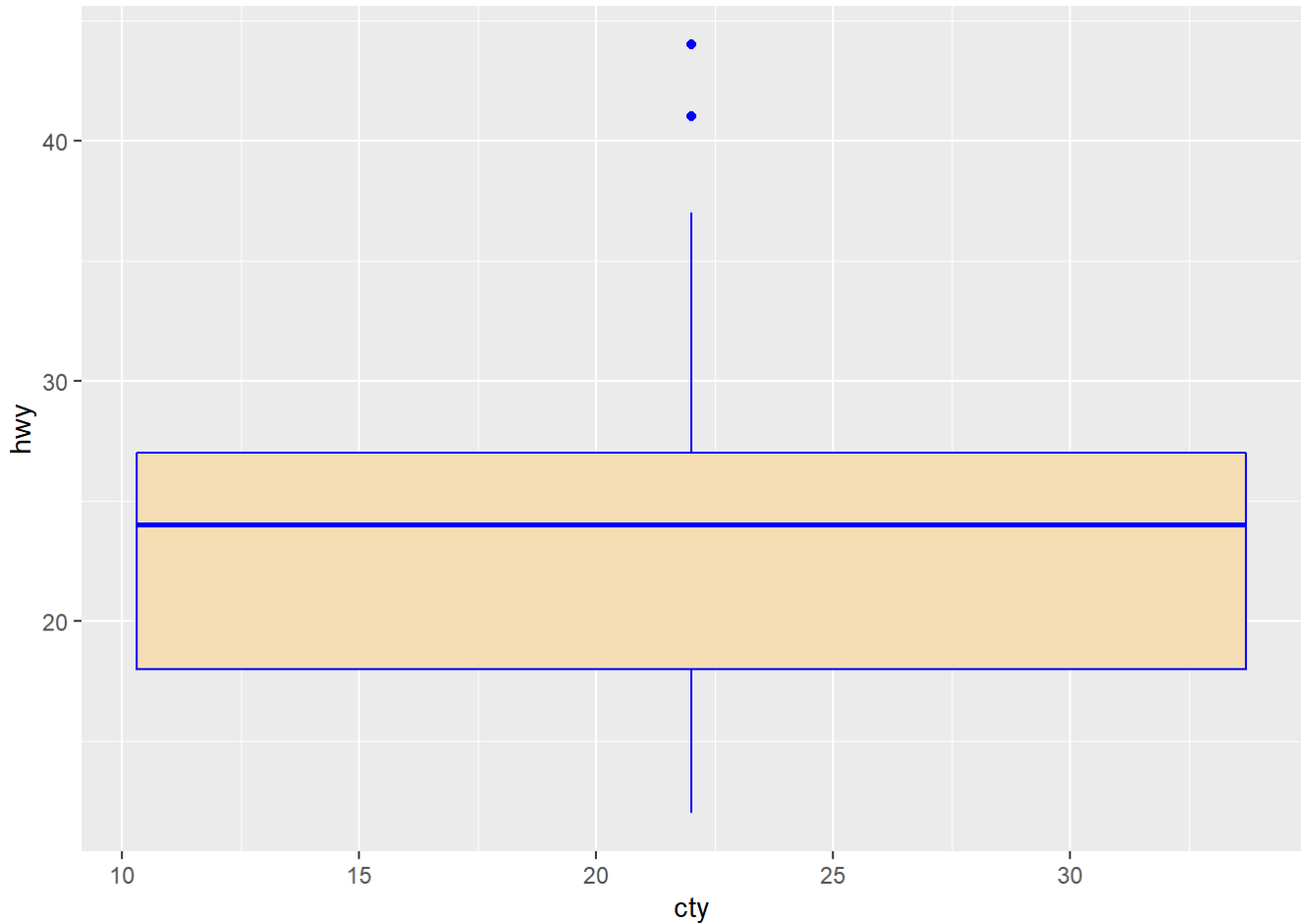
LES BOITES A MOUSTACHE :

Definition : Dans les représentations graphiques de données statistiques, la boîte à moustaches est un moyen rapide de figurer le profil essentiel d'une série statistique quantitative. Elle a été inventée en 1977 par John Tukey, mais peut faire l'objet de certains aménagements selon les utilisateurs.

utilisable avec la commande : "geom_boxplot" :

```
ggplot(mpg) + geom_boxplot(aes(x = cty , y = hwy), fill = "wheat", color = "blue")
```

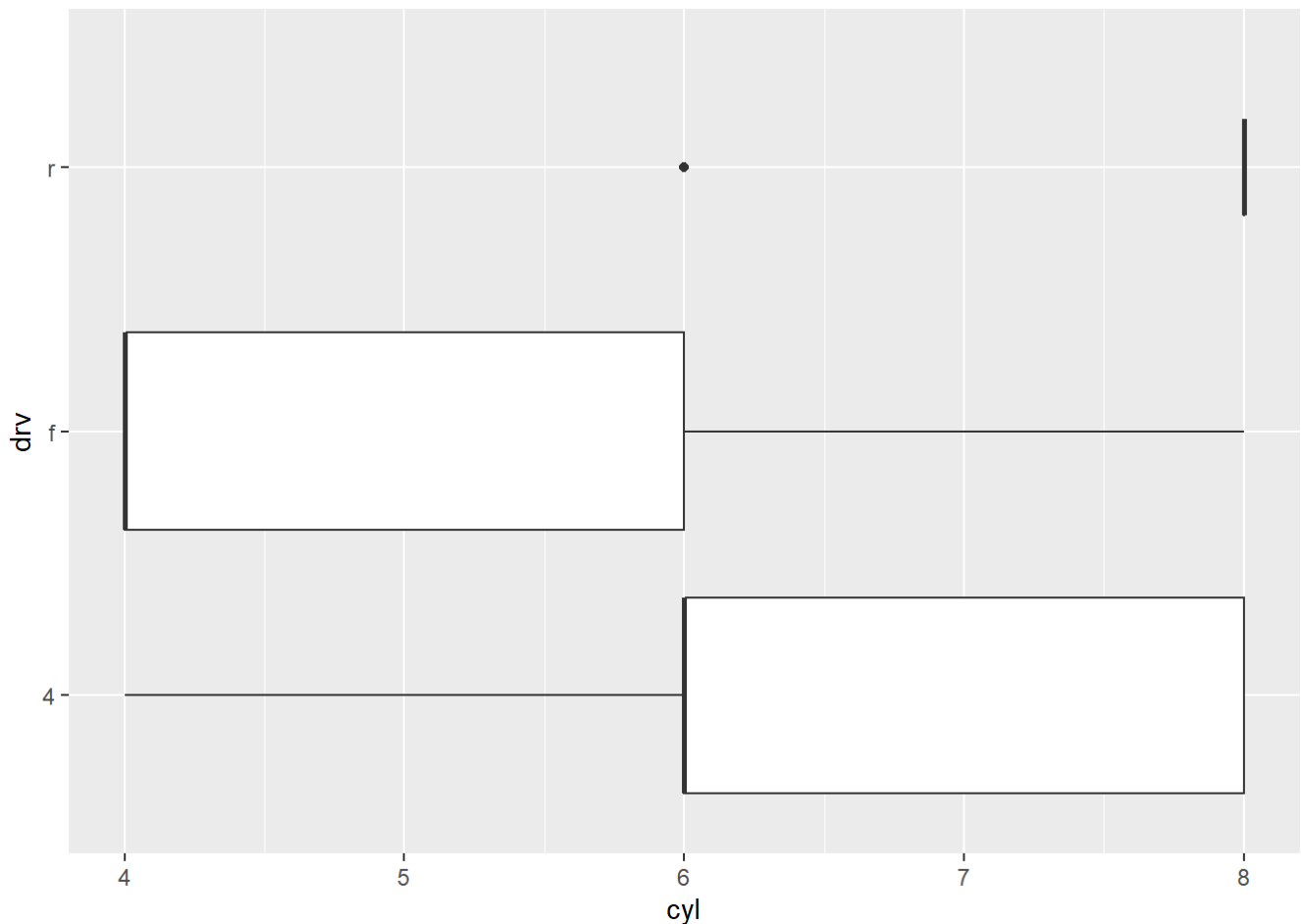
```
## Warning: Continuous x aesthetic -- did you forget aes(group=...)?
```



exemple toujours avec le tableau ci dessus Personnalisable avec les commandes :

Toujours avec les boites à moustaches, la commande “varwidth” permet de varier la longueur des boites en fonctions des effectifs de la catégorie. Pour l’appliquer, on fait :

```
ggplot(mpg) + geom_boxplot(aes(x = cyl, y = drv), varwidth = TRUE)
```



Pour cela, il faut utiliser la commande : `geom_histogram` qui se configure de la manière suivante

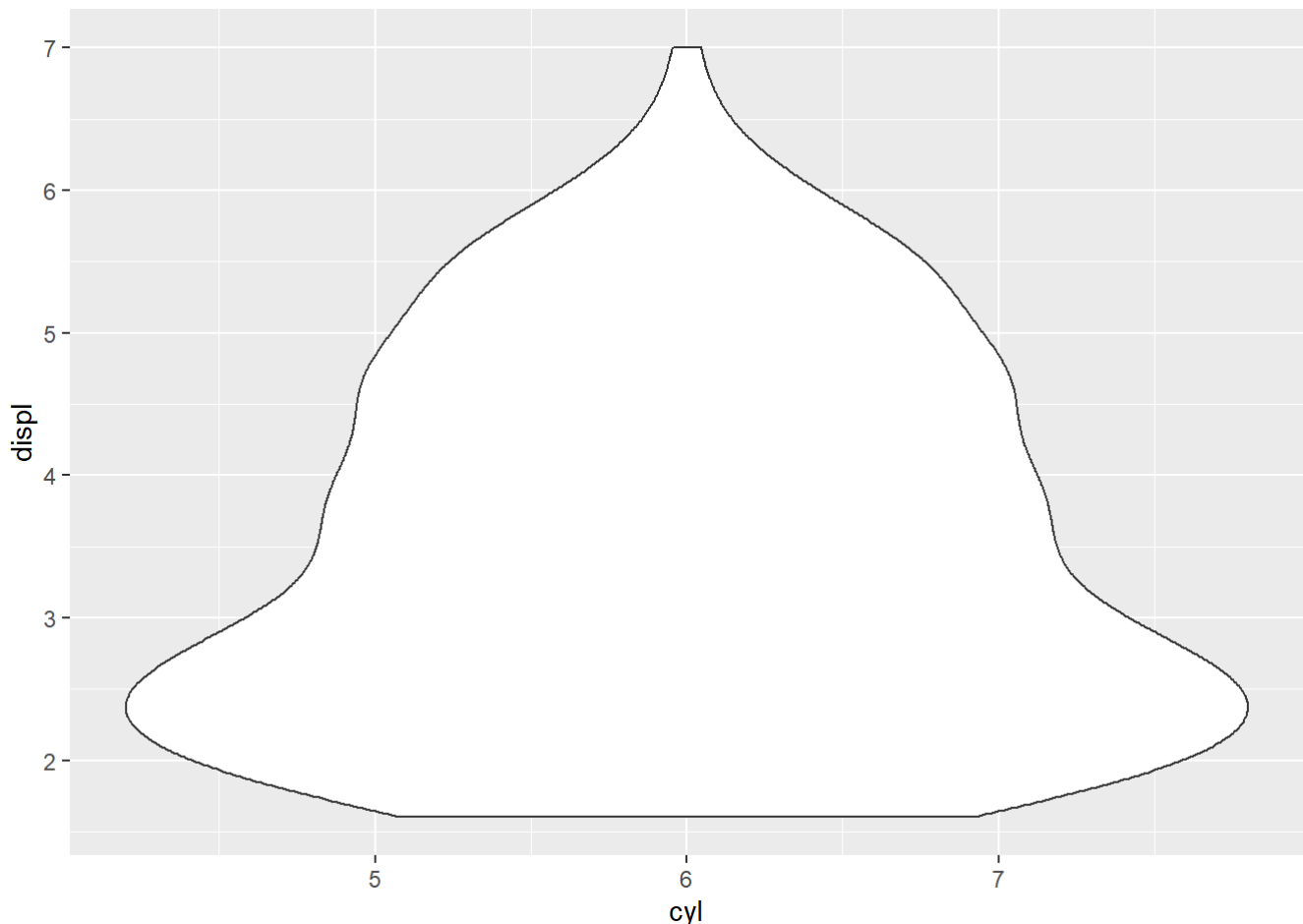
```
geom_histogram(aes(x = mpg))
```

Définir X permet d'afficher la donnée que l'on souhaite représenter.

Geom_violin :

`geom_violin` est très semblable à `geom_boxplot`, mais utilise des graphes en violon à la place des boîtes à moustache. Les graphes en violon peuvent donner une lecture plus fine des différences de distribution selon les classes.

```
ggplot(mpg) + geom_violin(aes(x = cyl, y = displ))
```



Utilisation de GGplot2 sur des dossiers CSV : l'exemple de notre base de donnée Covid

Les prérequis :

```
library(lubridate)#Utilisation de la fonction {dmy} pour changer le format de date de notre fichier.
library(ggplot2) #Utilisation des fonctions pour réaliser des graphiques. library(dplyr)#Utilisation de la fonction
{filter} pour pour filter certaines données clés que nous souhaitons afficher.
```

Etape 1 : lire le fichier CSV

on demande au programme de lire le fichier CSV qui contient nos données a plat on renomme ce dossier
`data_covid read.table("donneeaplatcovid2.csv" [nom du fichier],header=TRUE [titre des colonnes],sep=';',`
`[permet de determiner quel est le type de separateur],dec='.'` [decimal]
`data_covid<-`
`read.table("donneeaplatcovid2.csv",header=TRUE,sep=';',dec='.')` on a constater que la colonne "JOUR" avait
un nom différent, pour voir ce nom on effectue la commande suivante : `names(data_covid[1])` puis la fonction ci
dessous)

```
colnames(data_covid)[colnames(data_covid) == names(data_covid[1])] <- 'JOUR'
```

```
head(data_covid) str(data_covid)
```

#permet de connaitre le nom de la premiere colonne (`names(data_covid[1])`) #permet de changer le nom de la
premiere colonne (cf ligne 15)

on a le probleme du format de date, pour cela on execute la commande ci dessous pour changer le format :
`(data_covid"JOUR <- dmy(data_covid"JOUR)#transformation du format de la colonne JOUR en date OU`
FONCTION `dplyr::glimpse(data_covid)`

```
ggplot(data_covid, aes(x=JOUR, y=REA)) + geom_line()
```

Phase de filtre : lorsque l'on plot les données covid, on se retrouver avec toutes les données de tous les départements ici on cherche seulement les données du 95 covid_95 <-

`dplyr::filter(data_covid, DEP==95)` # permet de sélectionner le département

pour les deux commandes suivantes, on crée une variable p et q qui montrent des données différentes toujours en fonction des jours p <- `ggplot(covid_95, aes(x=JOUR, y=HOSP)) + geom_line(colour = 'blue')` p

q <- `ggplot(covid_95, aes(x=JOUR, y=REA)) + geom_line(colour = 'green')` q

ici il s'agit de la commande initiale ou l'on voit tous les départements

`ggplot(data_covid, aes(x=JOUR, y=HOSP)) + geom_point()` # s'affiche normalement

Utilisation des graphiques avec ESQUISSE

Cet addin vous permet d'explorer de manière interactive vos données en les visualisant avec le package `ggplot2`. Il vous permet de dessiner des graphiques à barres, des courbes, des nuages de points, des histogrammes, des boîtes à moustaches et des objets `sf`, puis d'exporter le graphique ou de récupérer le code pour reproduire le graphique.

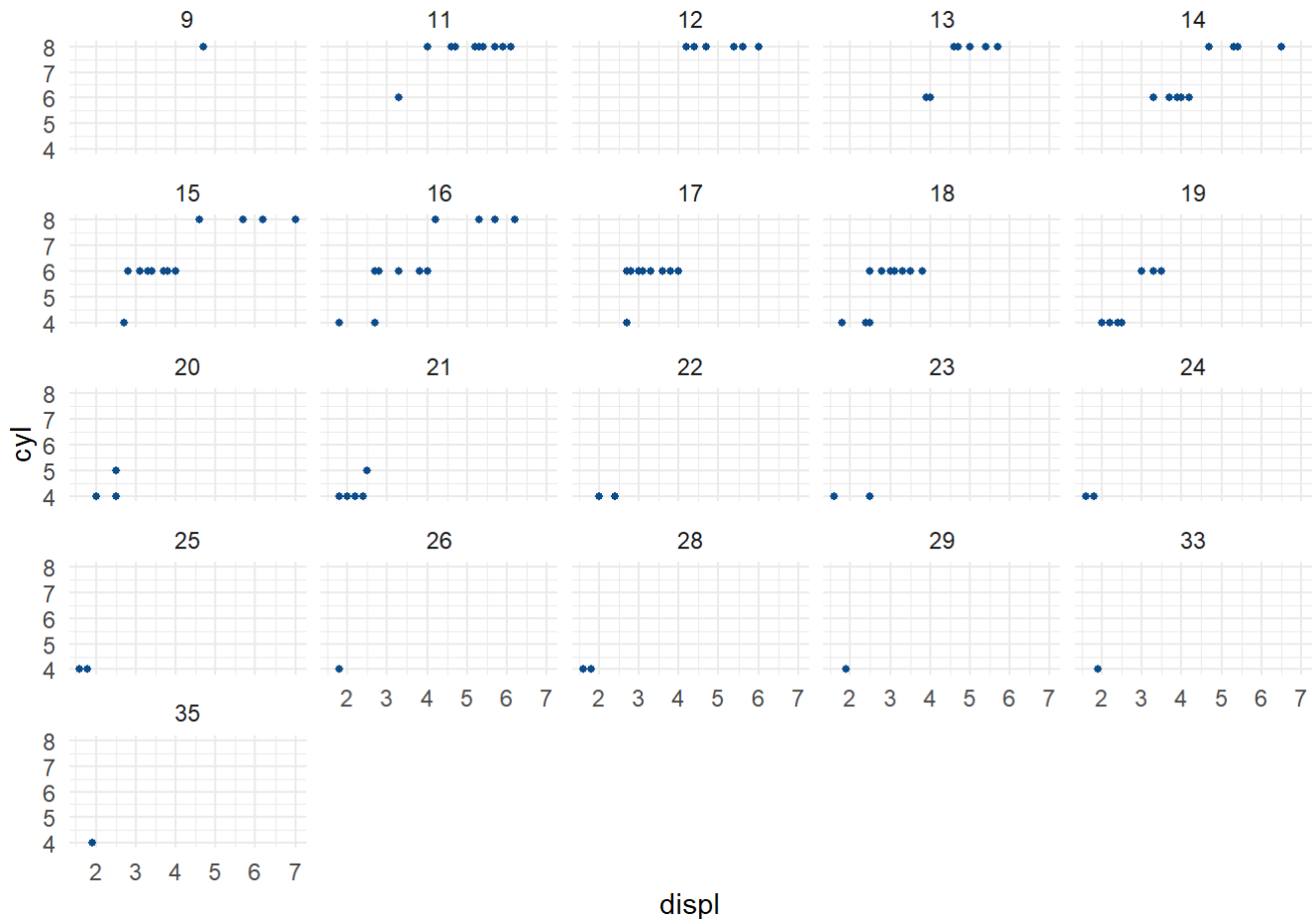
Pour accéder à cette fonction il faut aller dans Rstudio :

ADDINS -> GGLOT BULDER

les facettes :

```
library(ggplot2)
dataframe <- data.frame(mpg) #definition du DataFrame

ggplot(dataframe) +
  aes(x = displ, y = cyl) +
  geom_point(size = 1L, colour = "#0c4c8a") +
  theme_minimal() +
  facet_wrap(vars(cty))
```

Grâce à cette fonctionnalité, il est enfantin et didactique de faire des graphiques en tout genre, il suffit juste d'indiquer à la machine quelle est le DataFrame à utiliser et les paramètres à définir.

Utilisation des facettes

Nous allons utiliser la librairie (mpg) afin d'illustrer les différents exemples

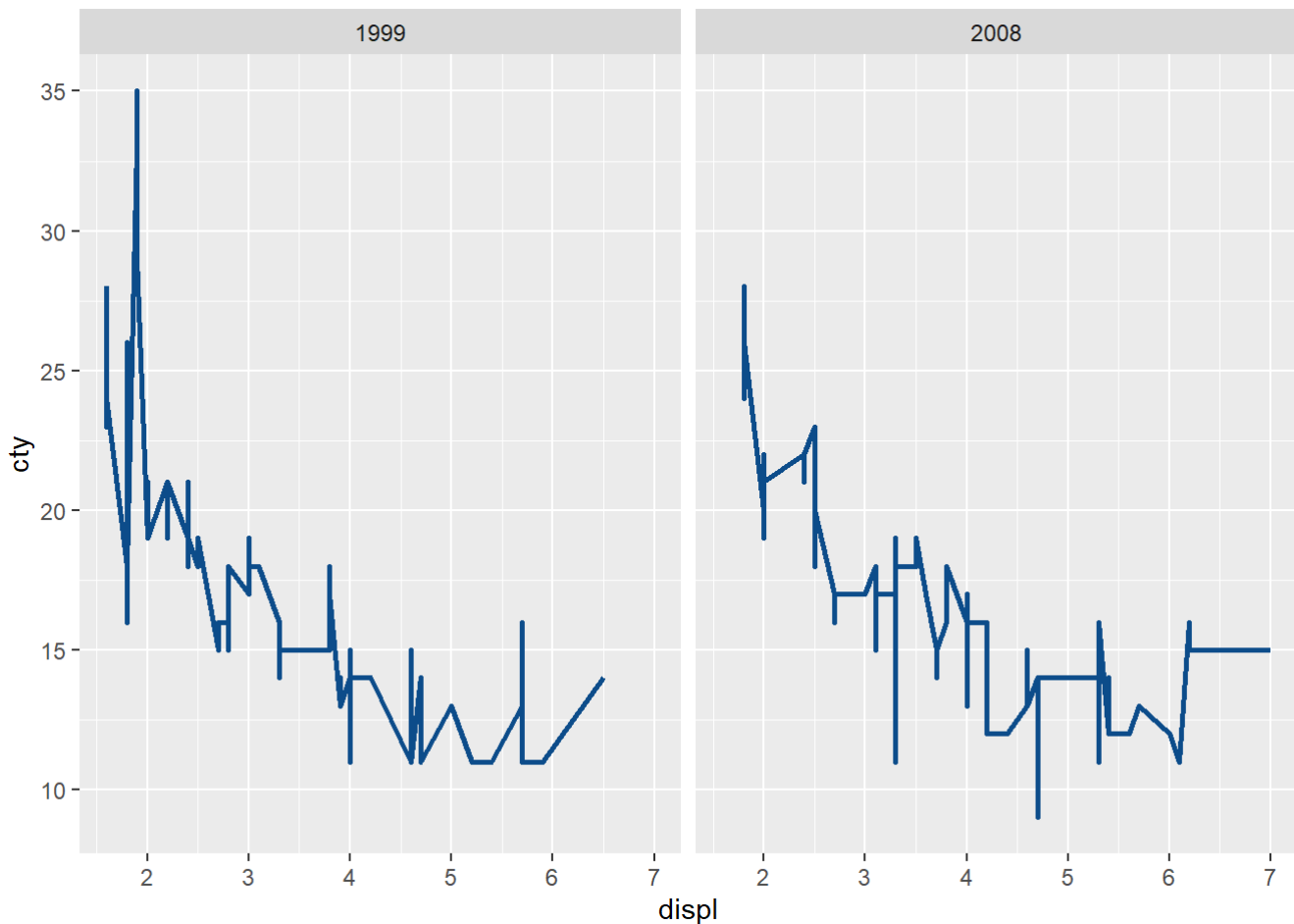
Cette fonction est équivalente à `ggplot2::facet_grid()` en ce qu'elle permet de construire une grille de petits multiples où les lignes et les colonnes correspondent à une valeur de données spécifique.

Bien que cela `ggplot2::facet_grid()` puisse être utilisé, cela entraînerait des résultats inattendus car il n'est pas possible de spécifier si vous faites référence à un nœud ou à un attribut d'arête. De plus `ggplot2::facet_grid()`, dessine les bords dans les panneaux même si le panneau ne contient pas les deux nœuds terminaux.

`Facet_graph` prend en charge tous ces problèmes, vous permettant de définir le type de données auquel les lignes et les colonnes font référence, ainsi que de filtrer les bords en fonction des nœuds de chaque panneau (même lorsque les nœuds ne sont pas dessinés).

```
#install.packages("igraph")

ggplot(mpg) +
  aes(x = displ, y = cty) +
  geom_line(size = 1L, colour = "#0c4c8a") +
  theme_grey() +
  facet_wrap(vars(year))
```



Le fait d'utiliser la fonction `igraph` permet notamment de pouvoir afficher plusieurs graphiques pour notamment comparer l'allure générale des différentes variables.

Utilisation des corrélations

En probabilités et en statistique, la corrélation entre plusieurs variables aléatoires ou statistiques est une notion de liaison qui contredit leur indépendance.

L'objectif va donc consister à mesurer la puissance du lien qu'il y a entre deux (ou plusieurs) variables. Or ce lien peut être plus ou moins complexe. On pense tout d'abord à un lien de type linéaire ... et par extension on va vite se tourner vers la régression linéaire pour trouver ce lien.

Malheureusement toutes les relations de dépendances ne sont pas forcément linéaire, et donc il va falloir pousser plus loin les régressions (polynomiale, etc.). Nous allons donc nous tourner vers la corrélation de Spearman (ρ) qui évalue la relation monotone entre deux variables. Dans une relation monotone, les variables ont tendance à changer ensemble, mais pas nécessairement à un taux régulier.

Pour calculer la corrélation entre deux variables on peut utiliser la fonction : "`cor`", qui fonctionne de la manière suivante :

```
cor(mpg$hwy, mpg$cty)
```

```
## [1] 0.9559159
```

Avec cela on obtient la corrélation entre les variables "`hwy`" et "`cty`". Par conséquent, nous allons utiliser un data frame pour voir toutes les corrélations entre les variables numériques.

Ainsi, nous avons la matrice de corrélation :

```
df <- data.frame(mpg$displ,mpg$cyl,mpg$cty,mpg$hwy)#creation d'un DataFrame avec Les variable
s numériques
mc <- cor(df)#utilisation de ma fonction correlation pour obtenir La matrice de correlation
mm <- as.matrix(mc* mc) # La matrice au carée permet de n'avoir que des valeurs positives.
print(mm)
```

```
##          mpg.displ  mpg.cyl  mpg.cty  mpg.hwy
## mpg.displ 1.0000000 0.8653225 0.6376405 0.5867867
## mpg.cyl   0.8653225 1.0000000 0.6492676 0.5805104
## mpg.cty   0.6376405 0.6492676 1.0000000 0.9137752
## mpg.hwy   0.5867867 0.5805104 0.9137752 1.0000000
```

Ainsi, nous avons la matrice de corrélation, pour faciliter l'interprétation nous allons sortir une matrice nous permettant de retenir toutes les corrélations supérieures 0.6 on fait :

```
mm[mm>0.7] <- 1
mm[mm<=0.7] <- 0
print(mm)
```

```
##          mpg.displ mpg.cyl mpg.cty mpg.hwy
## mpg.displ         1         1         0         0
## mpg.cyl           1         1         0         0
## mpg.cty           0         0         1         1
## mpg.hwy           0         0         1         1
```

Une fois la matrice réalisée, nous allons utiliser la fonction "graph_from_adjacency_matrix" qui nous permettra de voir en un coup d'œil les différentes corrélations.

Pour cela il faut utiliser la librairie igraph. cela nous permet de pouvoir mettre de manière graphique les relations entre les différentes variables

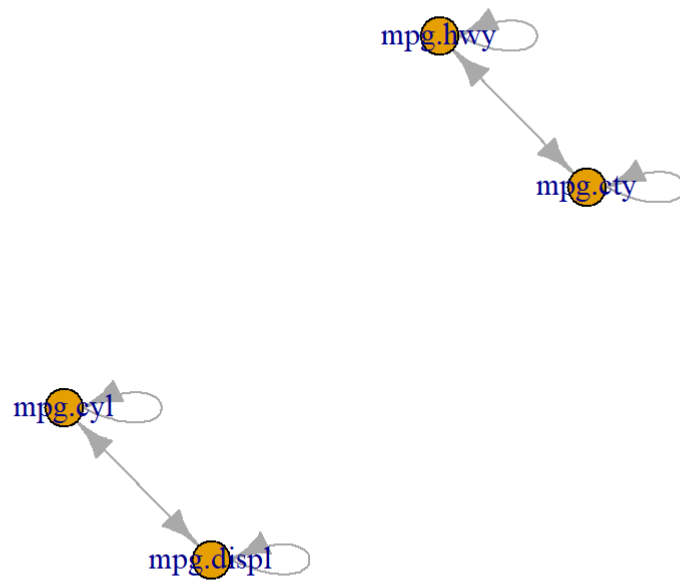
```
library(igraph)
```

```
##
## Attaching package: 'igraph'
```

```
## The following objects are masked from 'package:stats':
##
##      decompose, spectrum
```

```
## The following object is masked from 'package:base':
##
##      union
```

```
network <- graph_from_adjacency_matrix(mm)
plot(network)
```



Pour plus d'information :

<https://juba.github.io/tidyverse/08-ggplot2.html#ressources-1> (<https://juba.github.io/tidyverse/08-ggplot2.html#ressources-1>)