



Formation-R-perfectionnement

Module Création de graphiques avec ggplot2



0.1 Sommaire

- | | |
|--|------------------------------|
| 1 - Le package ggplot2 | 8 - Les courbes |
| 2 - Initialisation d'un graphique | 9 - Les camemberts |
| 3 - Les différentes géométries de graphiques | 10 - Les boîtes à moustaches |
| 4 - Croisement avec une autre variable | 11 - Ajout de libellés |
| 5 - Les diagrammes en barres | 12 - Habillage du graphique |
| 6 - Les histogrammes | 13 - Sauvegarde du graphique |
| 7 - Les nuages de points | 14 - Liens |



0.2 Avant-propos

Ce diaporama de formation a été rédigé dans le but d'être le support visuel des formations dispensées au [MASA](#).
Ces formations s'adressent à des agents qui ont suivi la **formation R initiation**.



0.3 Avant-propos

Elles sont données en présentiel sur une durée **de trois journée**, les modules de cette formation sont ajustables suivants le choix des agents.

Champ couvert par cette formation

Ce support couvre les rappels et compléments sur R et l'environnement du Minsitère.

Pour information, Les Modules de la formation R-perfectionnement sont:

- 01 - Module Rappels
- 02 - Module Fonctions
- 03 - Module Cartes statiques et interactives
- 04 - Module Création de graphiques avec ggplot2
- 05 - Module Quarto
- 06 - Module Parquet
- 07 - Module Initiation à l'écriture d'applications Shiny

Ils sont orientés pour être utiles aux agents du SSM MASA et se concentrent sur une utilisation de R via [RStudio](#) qui est mise à disposition des agents sur la plateforme interne Cerise basée sur RStudio Workbench.



1 Le package ggplot2

:



1.1 Le package ggplot2

Le package ggplot2 :

- Créé par Hadley Wickham
- Construit autour de la « grammaire des graphiques » (gg) qui décompose chaque graphique en un ensemble d'éléments sur lesquels le package permet de jouer indépendamment
- **Objectif** : réaliser des graphiques complexes et personnalisés à partir d'un fichier détail, d'un fichier détail pondéré ou de données agrégées
- **Site officiel** : <http://ggplot2.tidyverse.org/reference/>

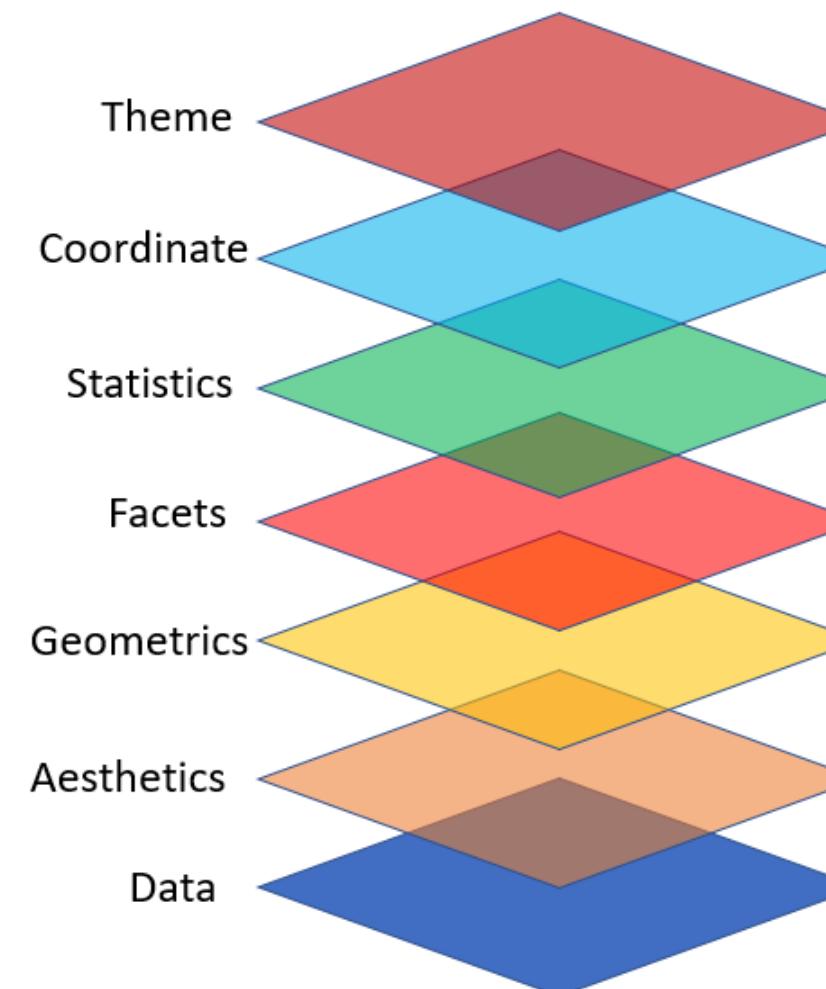


2 Initialisation d'un graphique





2.1 Initialisation d'un graphique



Un graphique est composé:

- des données
- des paramètres esthétiques : les variables à représenter ⇒ `aes()`
- une géométrie ⇒ `geom_*`
- des vignettes éventuelles ⇒ `facet_*`
- des éléments d'échelle ⇒ `scale_*`
- un système de coordonnées ⇒ `coord_*`
- des éléments d'habillage du graphique (apparence, titre, ajout de texte, etc.) ⇒ `theme()`

Un graphique s'initialise avec la fonction `ggplot()` puis chaque élément est ajouté aux autres avec un +

2.2 Initialisation d'un graphique

Initialisation du graphique avec la fonction `ggplot()`.

On déclare :

- `data` : la table contenant les données à représenter
- `mapping` : éventuellement, la ou les variables à représenter L'argument `mapping` requiert l'utilisation de la fonction `aes()` (pour aesthetic mappings). Les variables à représenter sont indiquées dans cette fonction par les paramètres `x` et `y`.

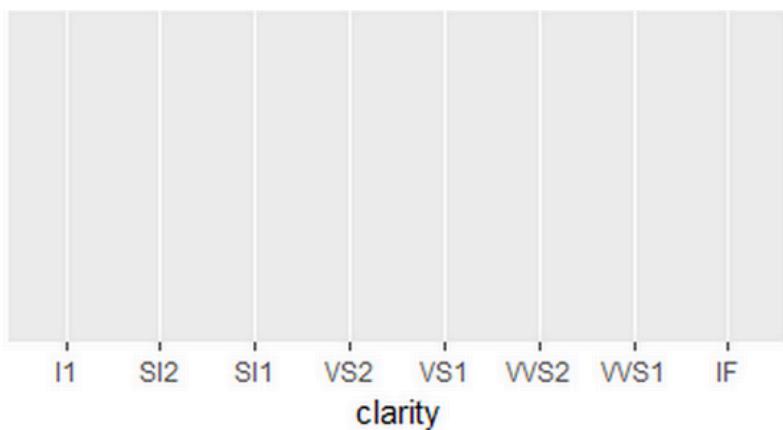
⇒ ce sont les bases du graphique, sur lesquelles on va afficher ensuite les données

2.3 Initialisation d'un graphique

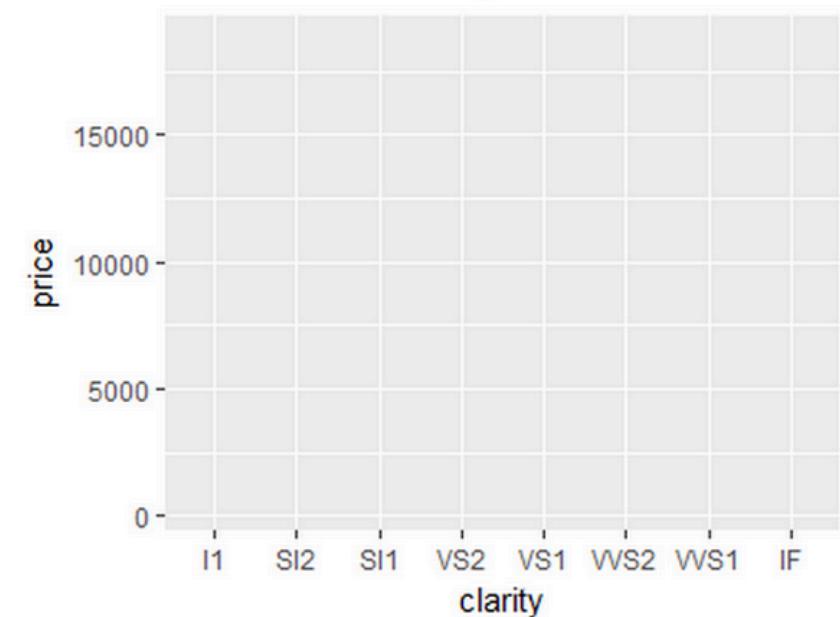
Initialisation d'un graphique, les axes x et y sont créés mais **aucune géométrie** n'est utilisée pour représenter les données.

Exemples :

```
ggplot(data = diamonds,  
       mapping = aes(x = clarity))
```



```
ggplot(data = diamonds,  
       mapping = aes(x = clarity, y = price))
```





2.4 Utilisation de la fonction `aes()`

L'appel de la fonction `aes()` peut se faire :

- Dans l'initialisation du graphique avec `ggplot()`

```
1 ggplot(data = diamonds, mapping = aes(x = clarity, y = price)) +  
2   geom_bar(stat = "identity")
```



3 Les différentes géométries de graphiques

3.1 Les différentes géométries de graphiques

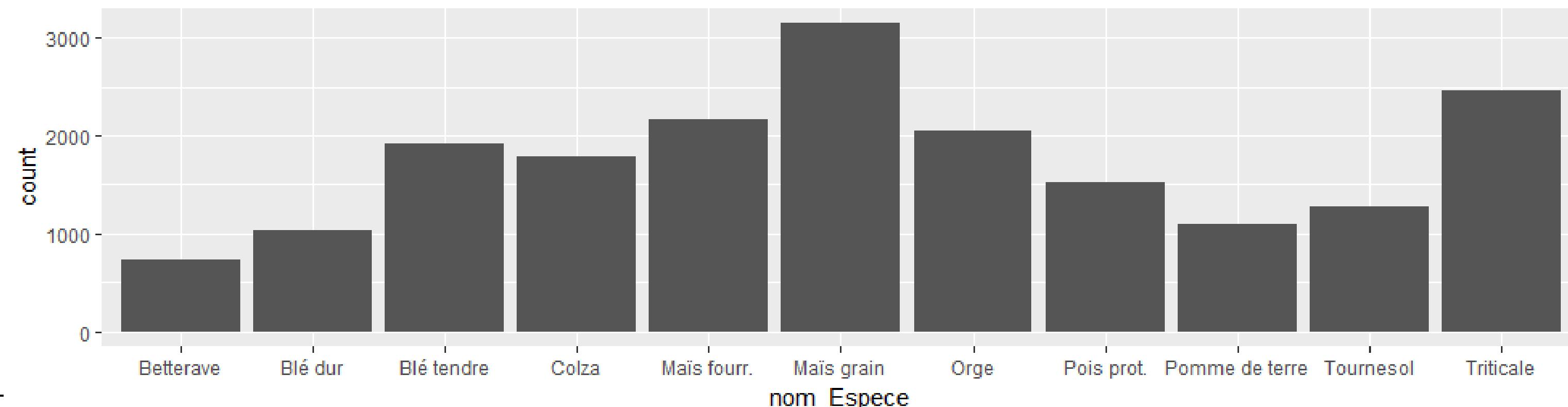
Déclaration de la géométrie du graphique avec les fonctions de la forme `geom_XXX()`.

Les géométries les plus courantes sont :

- `geom_bar()` et `geom_col()` → graphiques en barres (équivalent de `barplot()`)
 - `geom_histogram()` → histogramme (équivalent de `hist()`)
 - `geom_point()` → nuage de points (équivalent de `plot()`)
 - `geom_line()` → points reliés par un segment
 - `geom_boxplot()` → boxplot (équivalent de `boxplot()`)
- ⋮
⋮
⋮
⋮

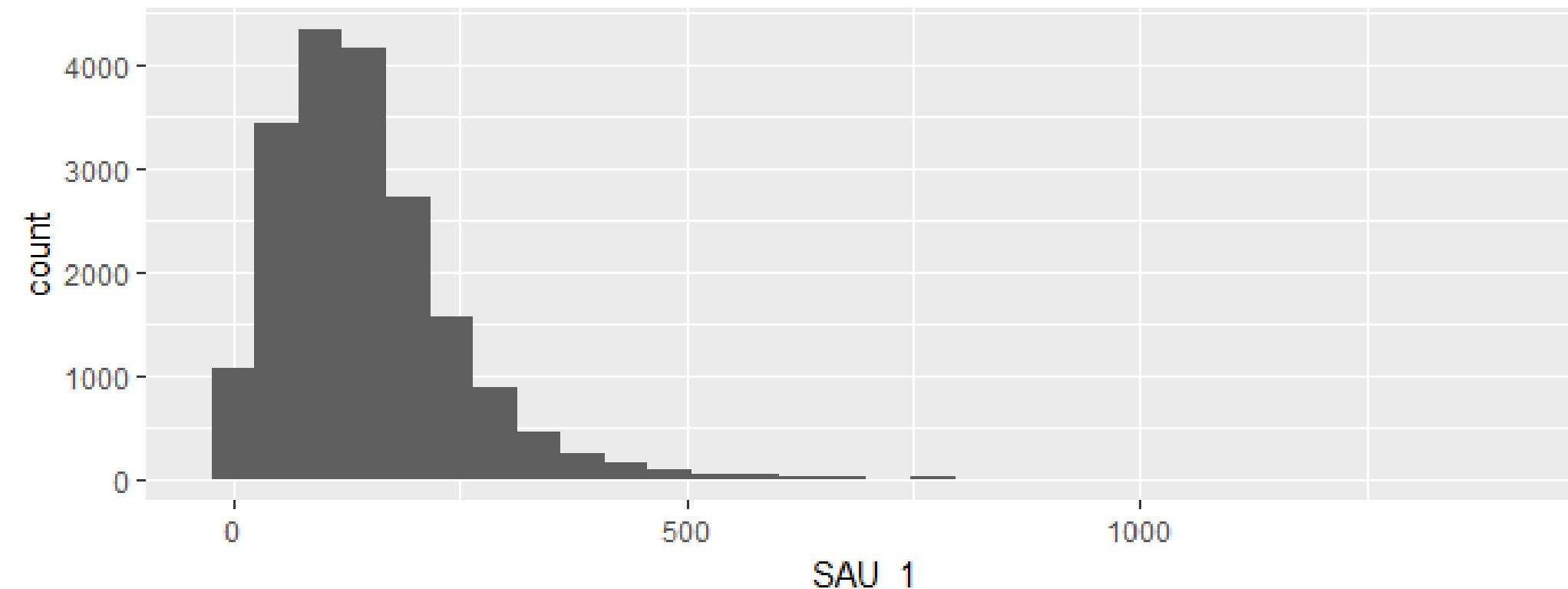
3.2 Les diagrammes en barre

```
### Un premier exemple : le diagramme en barre
### Nombre d'exploitations enquêtées pour PKGC17 par espèce (historiques)
ggplot(data = donnees_gc2, mapping = aes(x = nom_Especie)) + geom_bar()
# ou
ggplot(data = donnees_gc2) + geom_bar(mapping = aes(x = nom_Especie))
# ou
ggplot() + geom_bar(data = donnees_gc2, mapping = aes(x = nom_Especie))
```



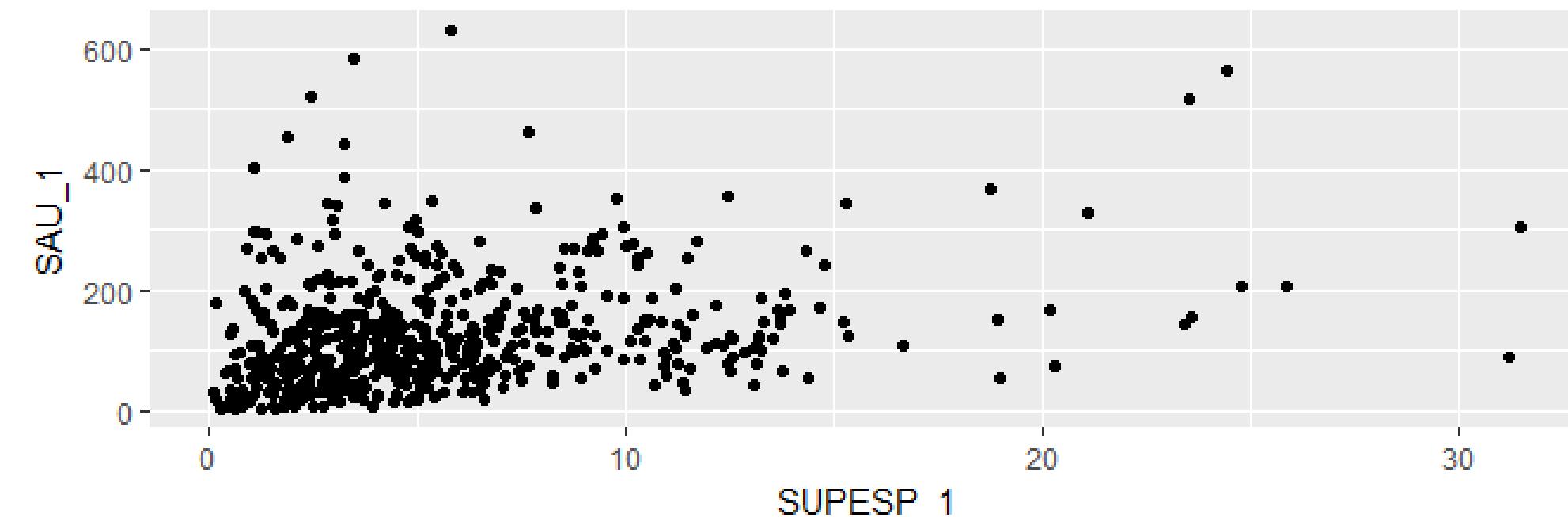
3.3 Les histogrammes

```
### Un deuxième exemple : l'histogramme
### Répartition des exploitations enquêtées pour PKGC17 par SAU
ggplot(data = donnees_gc2, mapping = aes(x = SAU_1)) + geom_histogram()
# ou
ggplot(data = donnees_gc2) + geom_histogram(mapping = aes(x = SAU_1))
# ou
ggplot() + geom_histogram(data = donnees_gc2, mapping = aes(x = SAU_1))
```



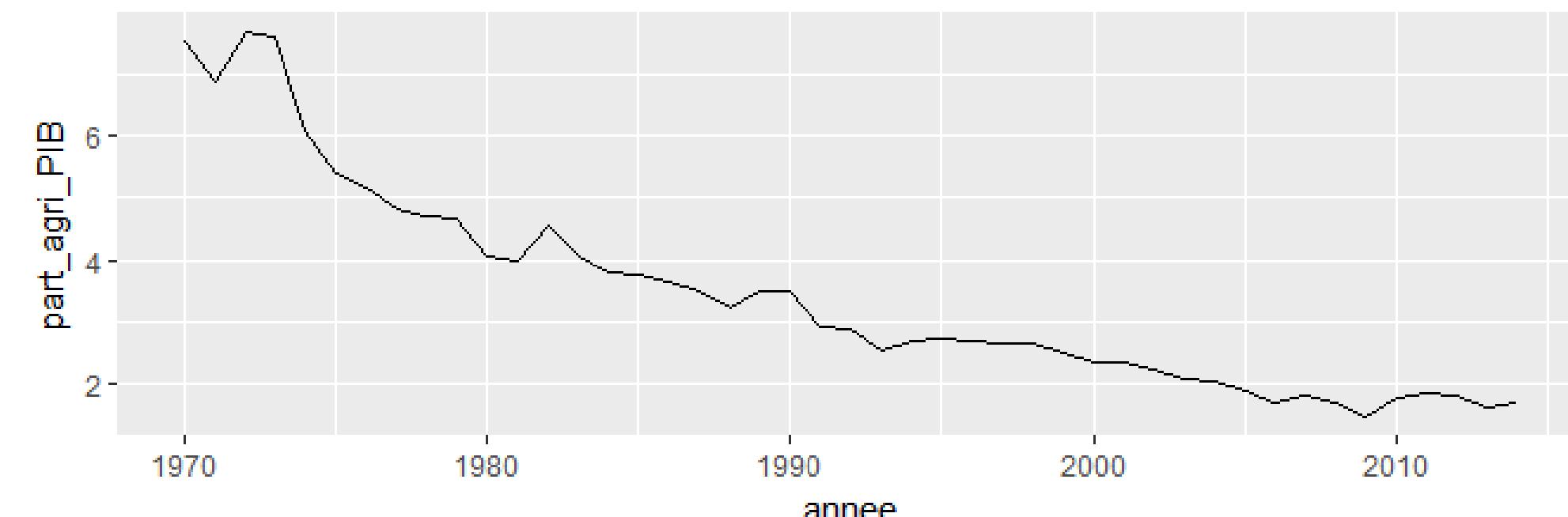
3.4 Les nuages de points

```
### Un troisième exemple : le nuage de points
### Taille du soja et SAU dans les exploitations enquêtées sur le soja pour PKGC17
ggplot(data = donnees_gc_soja, mapping = aes(x = SUPESP_1, y = SAU_1)) +
  geom_point()
# ou
ggplot(data = donnees_gc_soja) +
  geom_point(mapping = aes(x = SUPESP_1, y = SAU_1))
# ou
ggplot() +
  geom_point(data = donnees_gc_soja, mapping = aes(x = SUPESP_1, y = SAU_1))
```



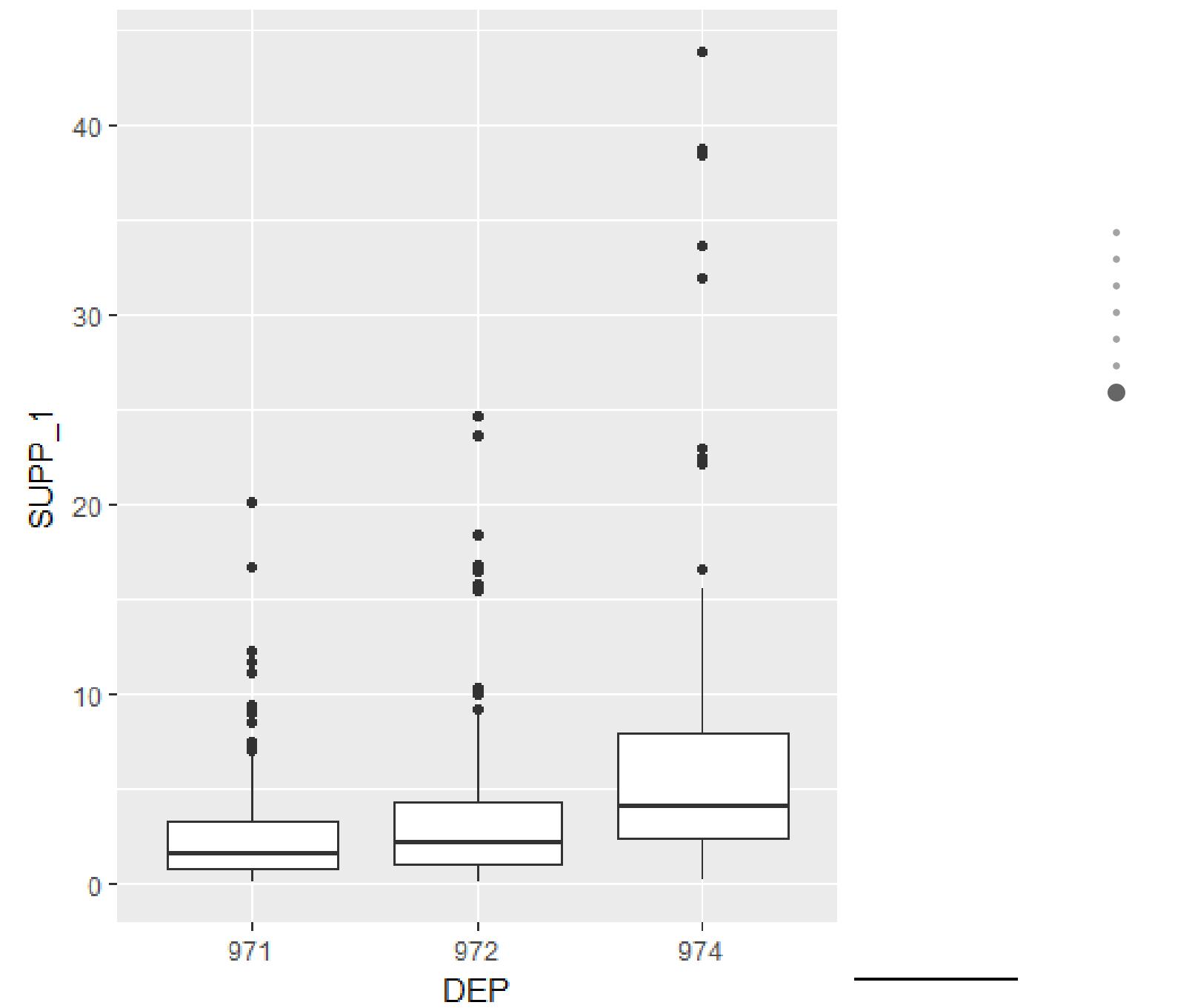
3.5 Les segments

```
### Un quatrième exemple : les points reliés par un segment
### Evolution de la part de l'agriculture dans le PIB français (source Banque Mondiale)
ggplot(data = evol_partagri_pib_fra, mapping = aes(x = annee, y = part_agri_PIB)) +
  geom_line()
# ou
ggplot(data = evol_partagri_pib_fra) + geom_line(mapping = aes(x = annee,
                                                               y = part_agri_PIB))
# ou
ggplot() + geom_line(data = evol_partagri_pib_fra, mapping = aes(x = annee,
                                                               y = part_agri_PIB))
```



3.6 Les boxplots

```
### Un cinquième exemple : le boxplot
### Boxplot sur la surface en canne à sucre des explo.
### enquêtées pour PKGC17 par département
ggplot(data = donnees_gc_CS,
       mapping = aes(x=DEP, y = SUPP_1)) +
  geom_boxplot()
# ou
ggplot(data = donnees_gc_CS) +
  geom_boxplot(mapping = aes(x=DEP, y = SUPP_1))
# ou
ggplot() +
  geom_boxplot(data = donnees_gc,
               mapping = aes(x=DEP, y = SUPP_1))
```





4 Croisement avec une autre variable



4.1 Croisement avec une autre variable

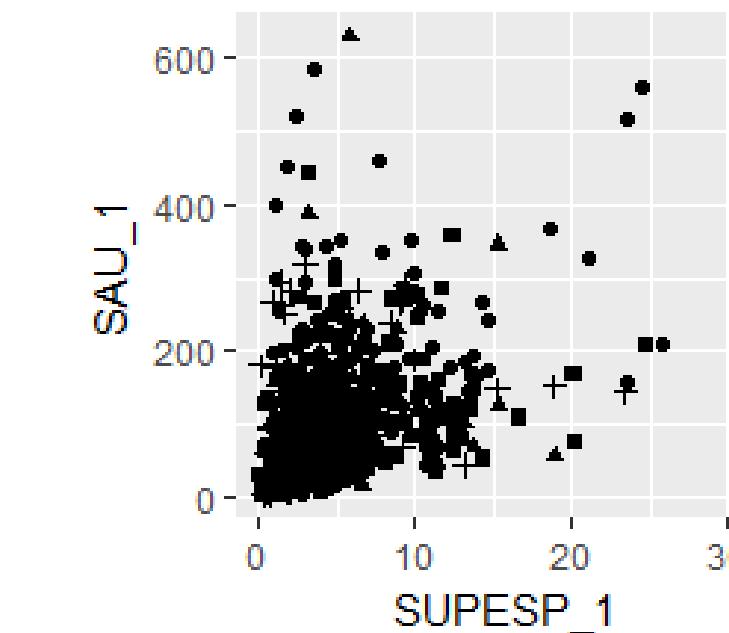
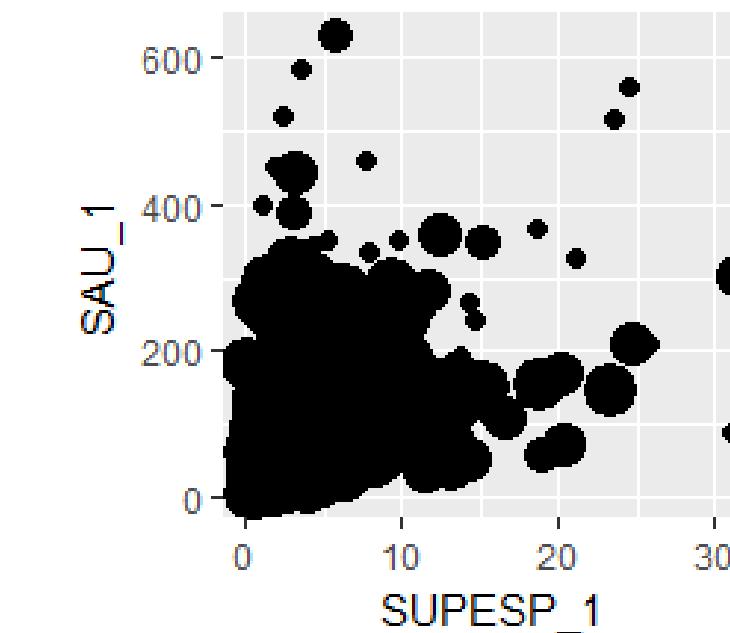
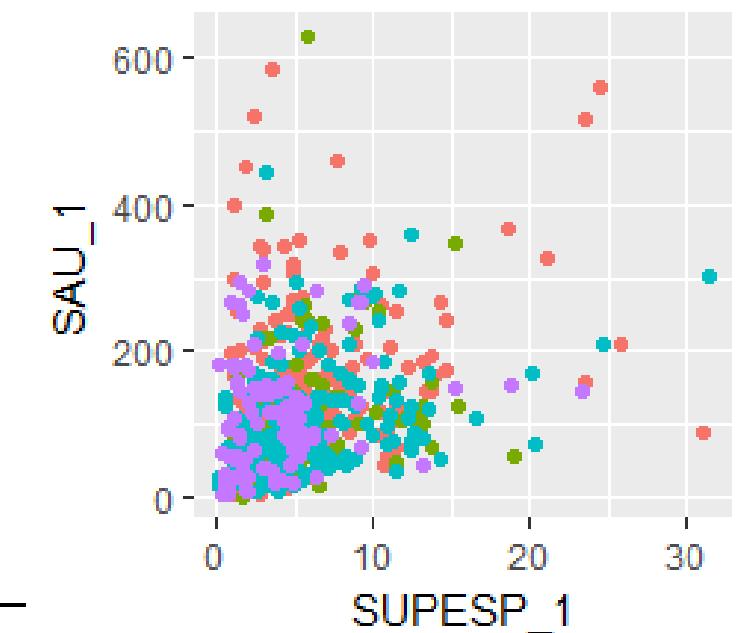
On peut **distinguer les observations appartenant à des classes différentes** au sein d'une distribution. Pour cela, il faut renseigner, selon le cas, les arguments suivants dans la fonction `aes()` :

- **fill** : pour une différenciation par remplissage
- **colour** : pour une différenciation par couleur de bordure
- **shape** : pour une différenciation par forme
- **size** : pour une différenciation par taille
- **alpha** : pour une différenciation par transparence
- **linetype** : type de lignes (continue, pointillées...)
- **linewidth** : taill des lignes



4.2 Croisement avec une autre variable

```
### Identifier les observations par région dans le nuage de points de l'exemple 3
# Différentiation par couleur des points
ggplot(data = donnees_gc_soja) +
  geom_point(mapping = aes(x = SUPESP_1, y = SAU_1, colour=REG))
# Différentiation par taille des points
ggplot(data = donnees_gc_soja) +
  geom_point(mapping = aes(x = SUPESP_1, y = SAU_1, size=REG))
# Différentiation par forme des points
ggplot(data = donnees_gc_soja) +
  geom_point(mapping = aes(x = SUPESP_1, y = SAU_1, shape=REG))
```



4.3 Quand utiliser aes() ? (1/2)

La fonction aes() fait le lien entre les données et le rendu visuel du graphique.

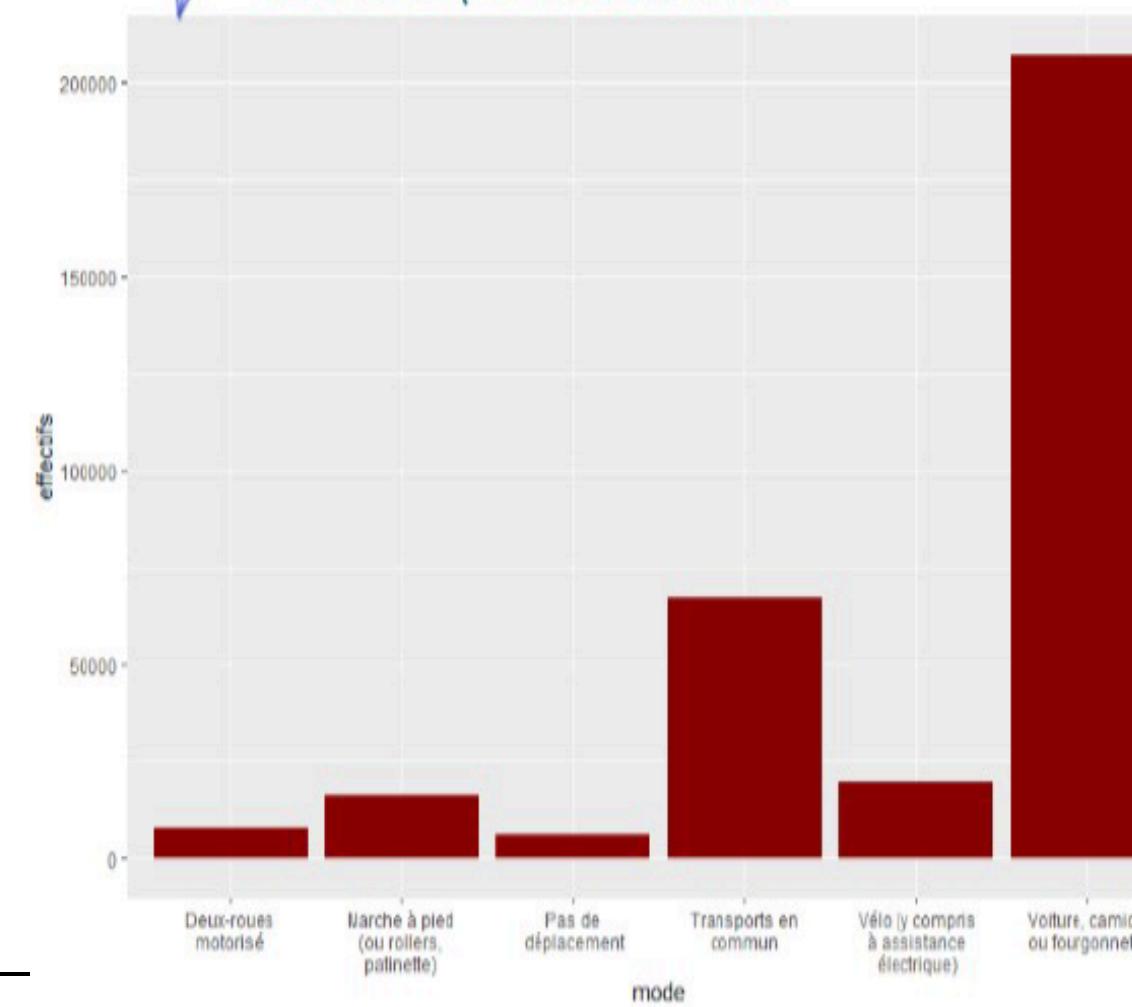
Pour modifier un attribut, on utilise les mêmes noms de paramètres, mais on peut les placer **à l'intérieur ou à l'extérieur** de la fonction aes() selon l'effet souhaité :

- Pour modifier un attribut sans le relier aux valeurs d'une variable dans le jeu de données → on le définit dans la fonction de géométrie, **à l'extérieur de la fonction aes()** : cela s'appliquera alors à l'ensemble du graphique.
- Pour faire varier un attribut en fonction des valeurs prises par une autre variable → on le définit **à l'intérieur de la fonction aes()**

4.4 Quand utiliser aes() ? (2/2)

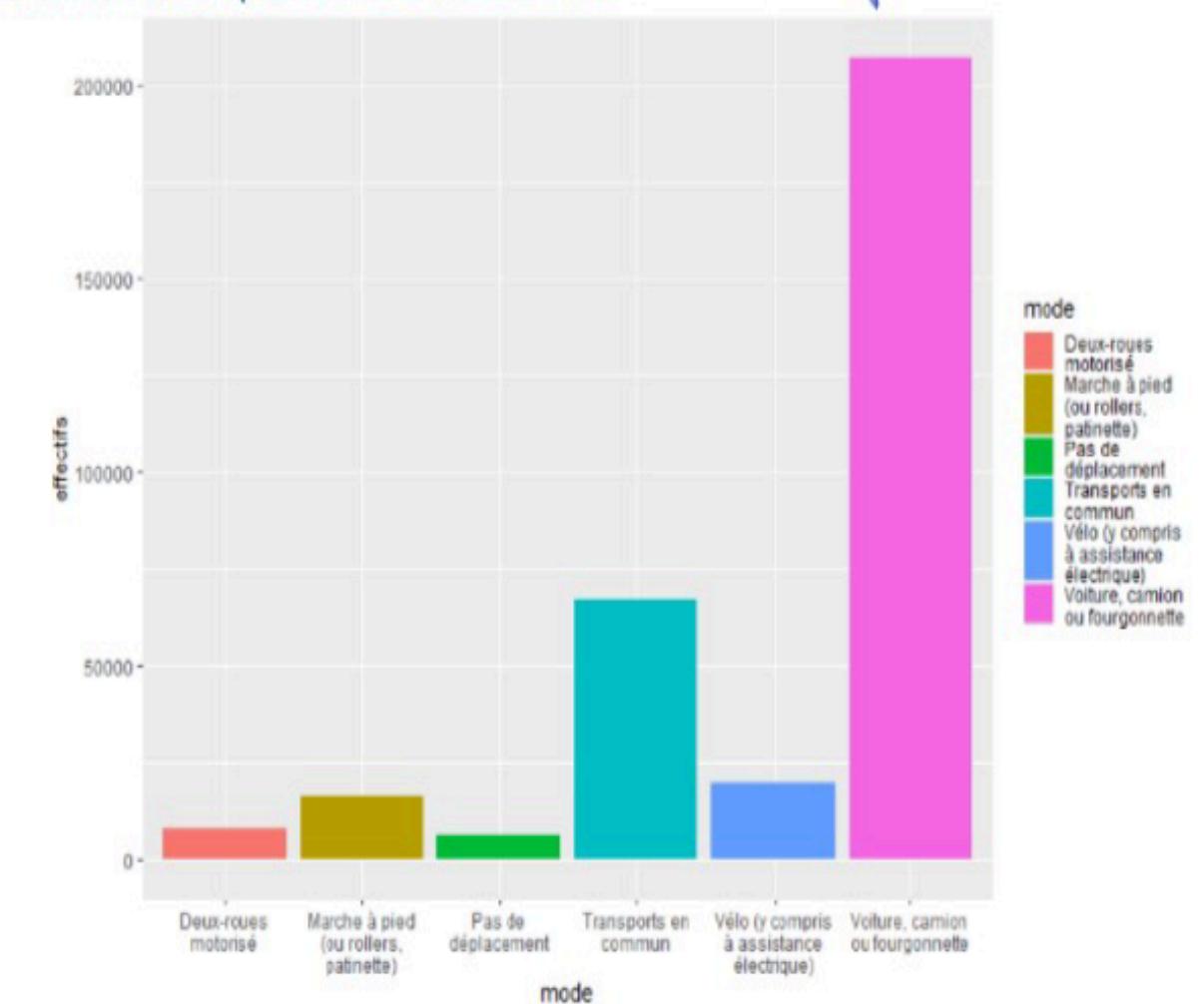
```
mode_transport %>%  
  ggplot(mapping = aes(x = mode, y = effectifs)) +  
  geom_bar(stat = "identity",  
          fill = "darkred")
```

à l'extérieur de la fonction aes()
on précise la couleur souhaitée
même couleur pour toutes les barres



```
mode_transport %>%  
  ggplot(mapping = aes(x = mode, y = effectifs, fill = mode)) +  
  geom_bar(stat = "identity")
```

à l'intérieur de la fonction aes()
on précise la variable selon laquelle le remplissage doit varier
une couleur différente pour chacune des modalités





5 Les diagrammes en barres

5.1 Les diagrammes en barres

Les diagrammes en barres avec `geom_bar()` :

- **Objectif** : compter les occurrences d'une modalité d'une variable qualitative (ou rendue qualitative par la création de classes)

On peut construire un diagramme en barres à partir :

- De données individuelles

```
### Nombre d'exploitations enquêtées pour PKGC17 par espèce (historiques)
ggplot(data = donnees_gc2, mapping = aes(x = nom_Espèce)) + geom_bar()
```

- De données individuelles pondérées : argument `weight` de la fonction `aes()`

```
### Nombre d'exploitations par espèce, extrapolé à partir de l'enquête PKGC17
ggplot(data = donnees_gc2, mapping = aes(x = nom_Espèce, weight = var_pond)) + geom_bar()
```

5.2 Les diagrammes en barres

- De données agrégées : argument `stat = "identity"` de la fonction `geom_bar()`.

Cet argument précise que la hauteur des barres n'est pas obtenue par comptage mais qu'elle est identique à la variable renseignée dans l'argument `y` de la fonction `aes()`

```
### Nombre d'exploitations enquêtées pour PKGC17 par espèce, à partir d'un fichier agrégé
donnees_gc2_agr<-data.frame(table(donnees_gc2$ESPECE))
colnames(donnees_gc2_agr)<-c("ESPECE", "EFFECTIF")
ggplot(data = donnees_gc2_agr, mapping = aes(x = ESPECE, y = EFFECTIF)) +
  geom_bar(stat = "identity")
```

Le paramètre `stat` précise qu'il faut utiliser la valeur de la variable `y` pour calculer la hauteur des barres.

Remarque : `geom_col()` est un raccourci pour utiliser `geom_bar()` avec `stat = identity`.

5.3 Les diagrammes en barres

Pour rendre le graphique plus facile à lire, il est utile de :

- **Réorganiser le graphique**

En réordonnant les différentes modalités dans le jeu de données pour qu'elles soient classées par ordre décroissant

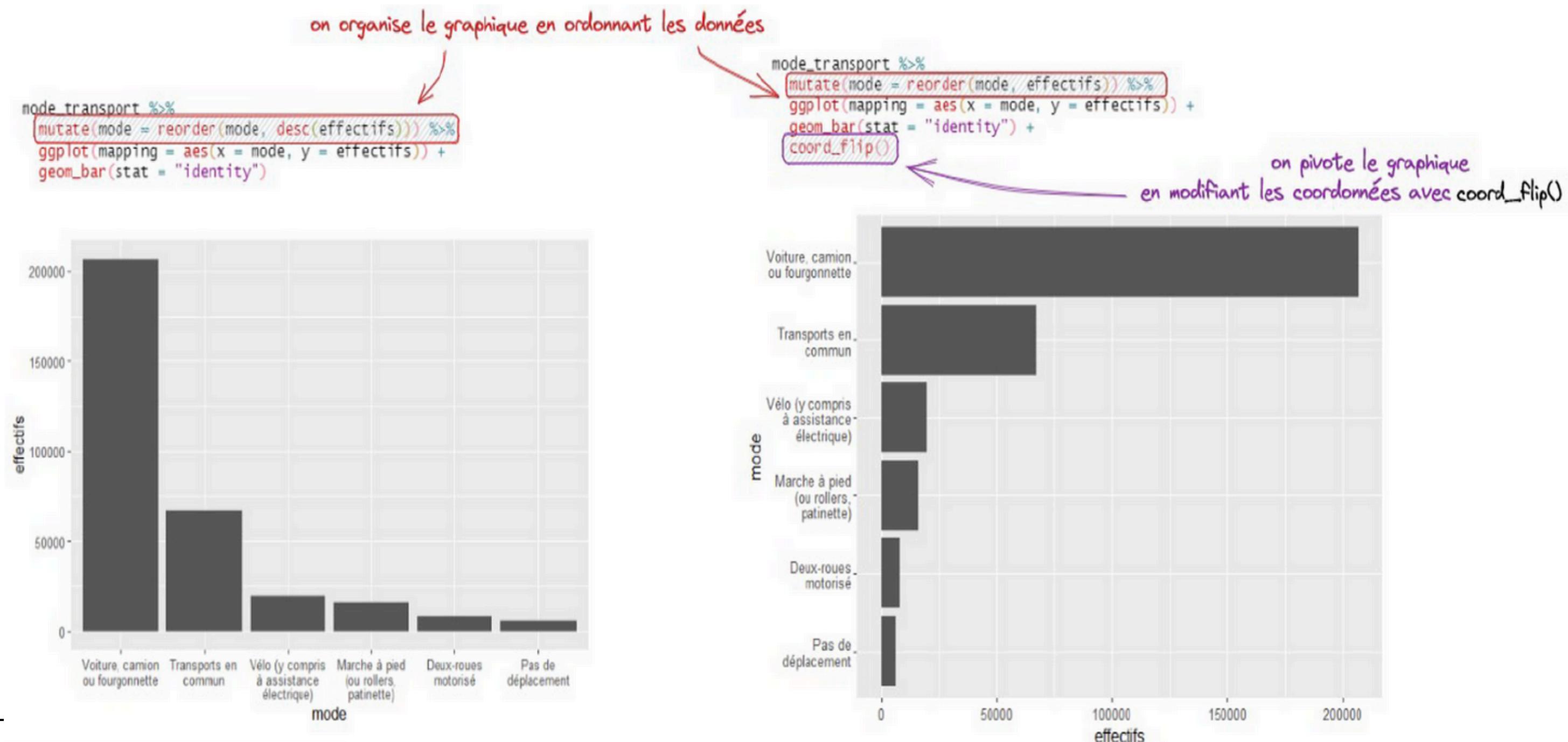
⇒ *fonction reorder() pour réordonner les modalités d'une variable selon les effectifs d'une autre variable*

- **Intervertir l'axe des x et des y**

- en modifiant les variables passées en x et en y dans la fonction aes()

- ou en utilisant la fonction coord_flip() qui permet d'échanger l'axe horizontal et l'axe vertical **sur n'importe quel graphique créé avec ggplot2**

5.4 Les diagrammes en barres



5.5 Les diagrammes en barres

On peut croiser les données avec une seconde variable en renseignant l'argument *fill* dans la fonction `aes()`.

Trois dispositions sont alors possibles pour ces données, à choisir dans l'argument *position* de la fonction `geom_bar()`:

- *position* = “stack”: par défaut, empilées en nombre
- *position* = “dodge”: placées côte à côte en nombre
- *position* = “fill”: empilées en fréquence (total de chaque barre=1)

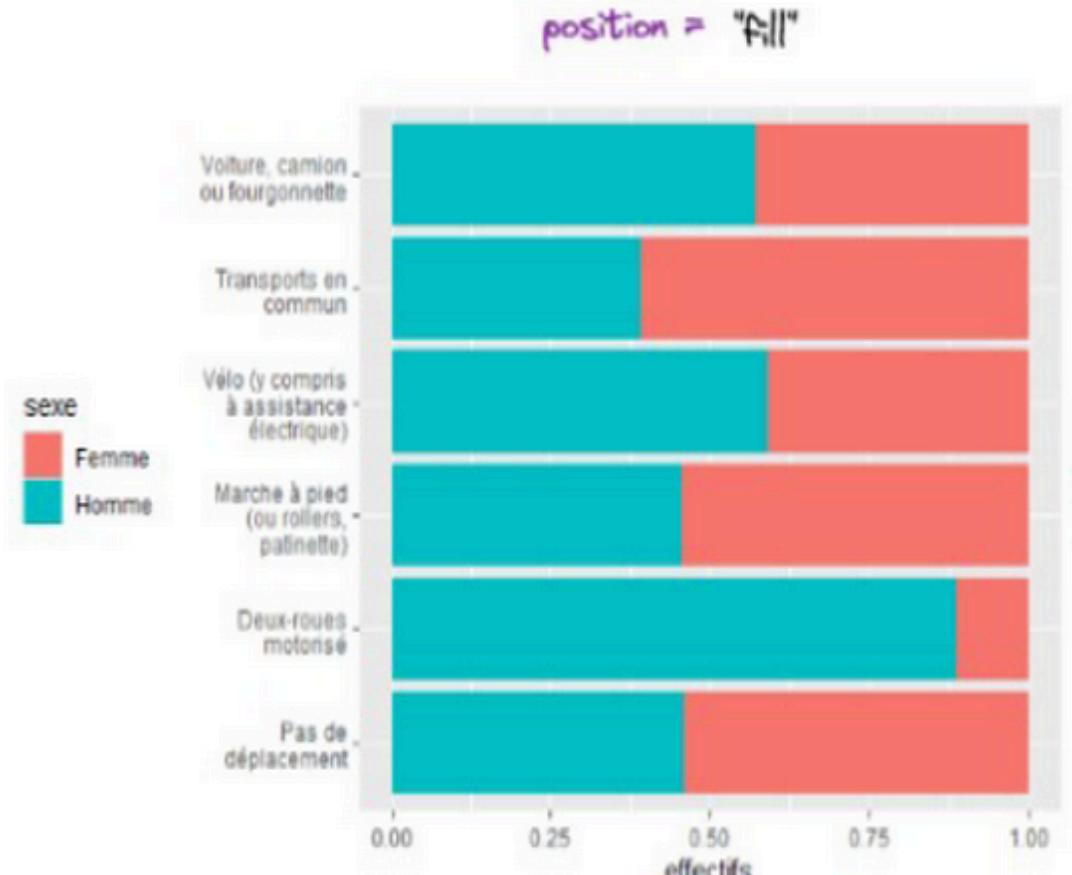
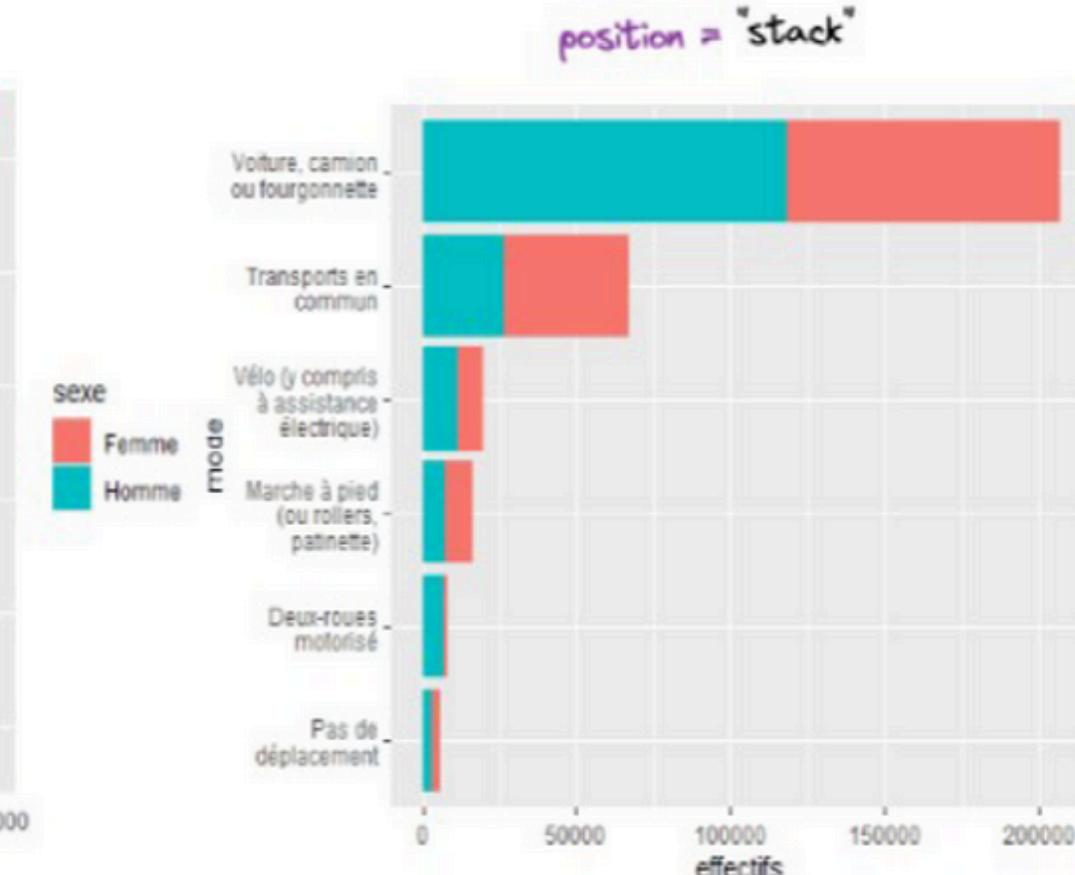
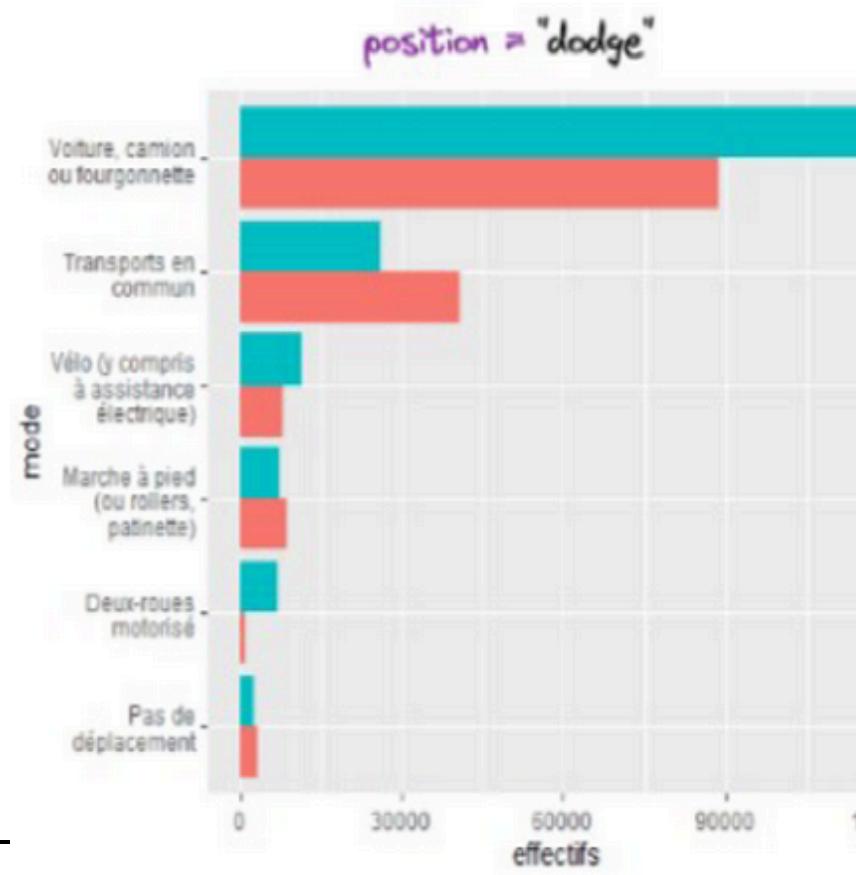


5.6 Les diagrammes en barres

```
mode_transport %>%
  mutate(mode = reorder(mode, effectifs)) %>%
  ggplot(mapping = aes(x = mode, y = effectifs, fill = sexe)) +
  geom_bar(stat = "identity", position = "...") +
  coord_flip()
```

représentation de la variable "sexe" sur le graphique

choix de la position des différentes catégories





6 Les histogrammes

6.1 Les histogrammes

Les histogrammes avec `geom_histogram()` :

Objectif : représenter la distribution d'une variable quantitative continue

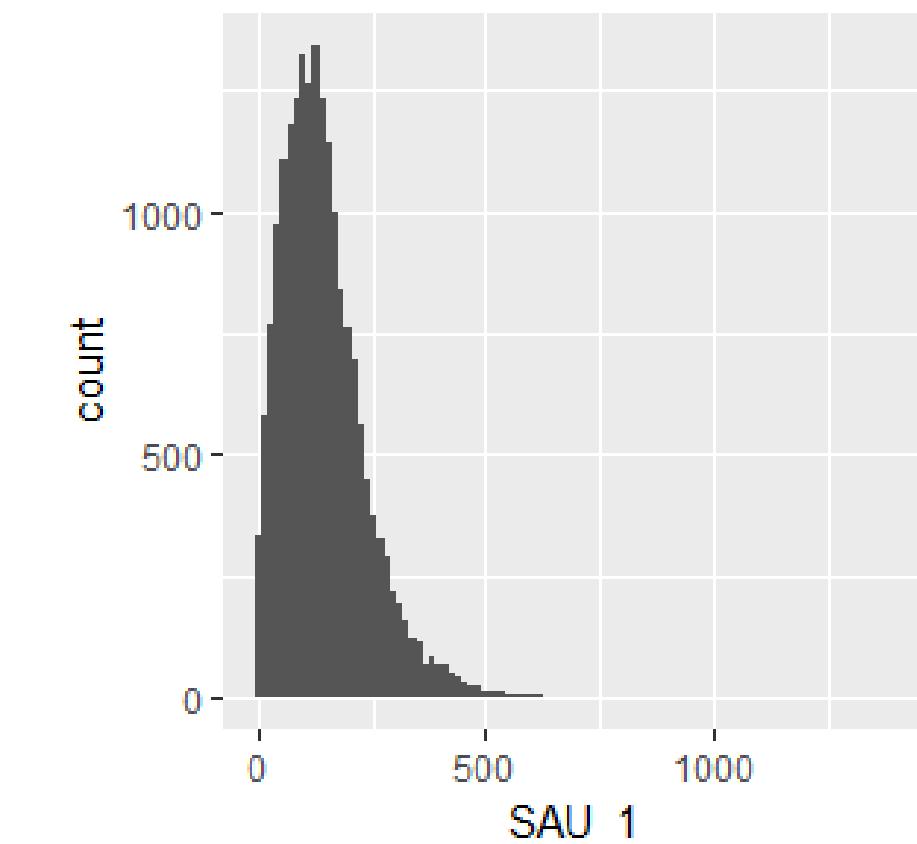
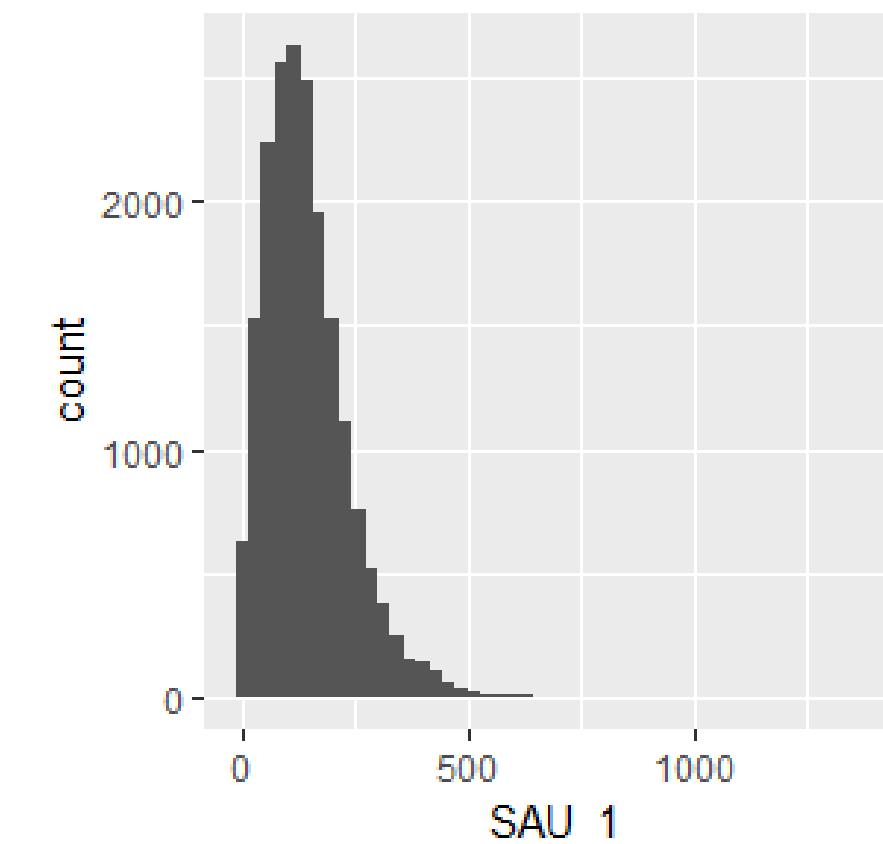
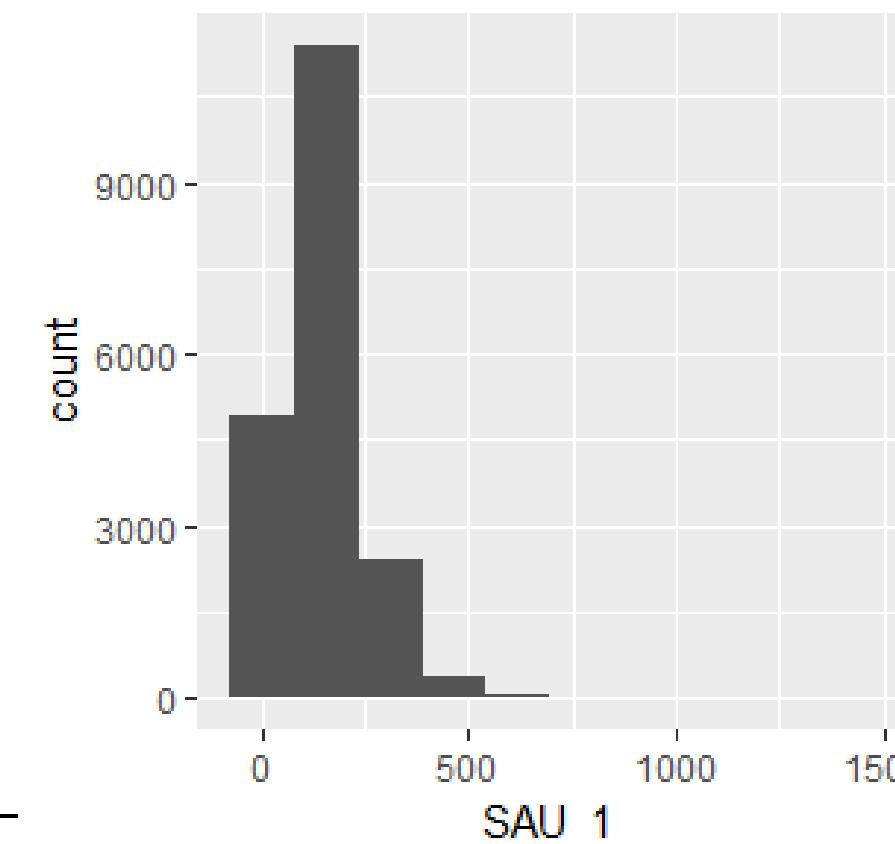
La fonction `geom_histogram()` possède deux arguments pour gérer la largeur des barres :

- *bins* : fixe le nombre de barres de l'histogramme
- *binwidth* : fixe la largeur des barres dans l'unité de la variable représentée



6.2 Les histogrammes

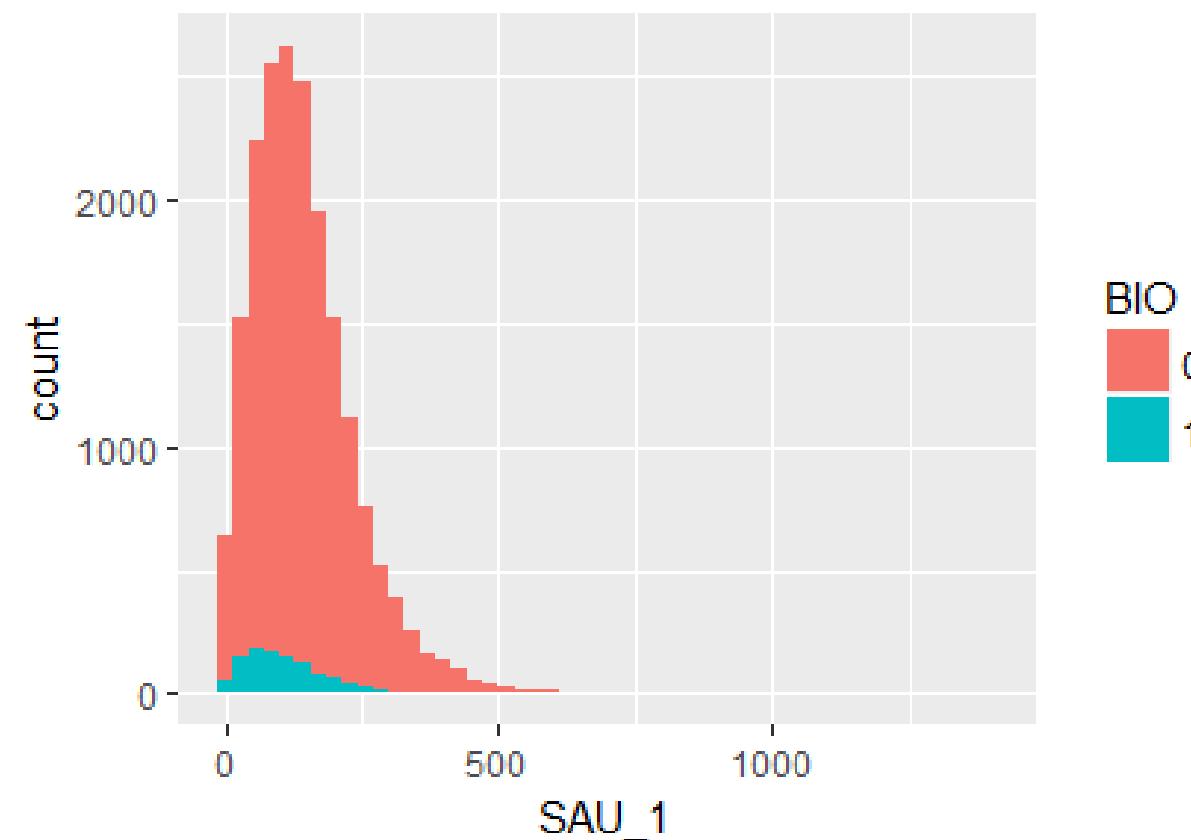
```
# Répartition des exploitations enquêtées pour PKGC17 selon la SAU
# 10 barres
ggplot(data = donnees_gc2) + geom_histogram(mapping = aes(x = SAU_1),bins = 10)
# 50 barres
ggplot(data = donnees_gc2) + geom_histogram(mapping = aes(x = SAU_1),bins = 50)
# 100 barres
ggplot(data = donnees_gc2) + geom_histogram(mapping = aes(x = SAU_1),bins = 100)
⋮
```



6.3 Les histogrammes

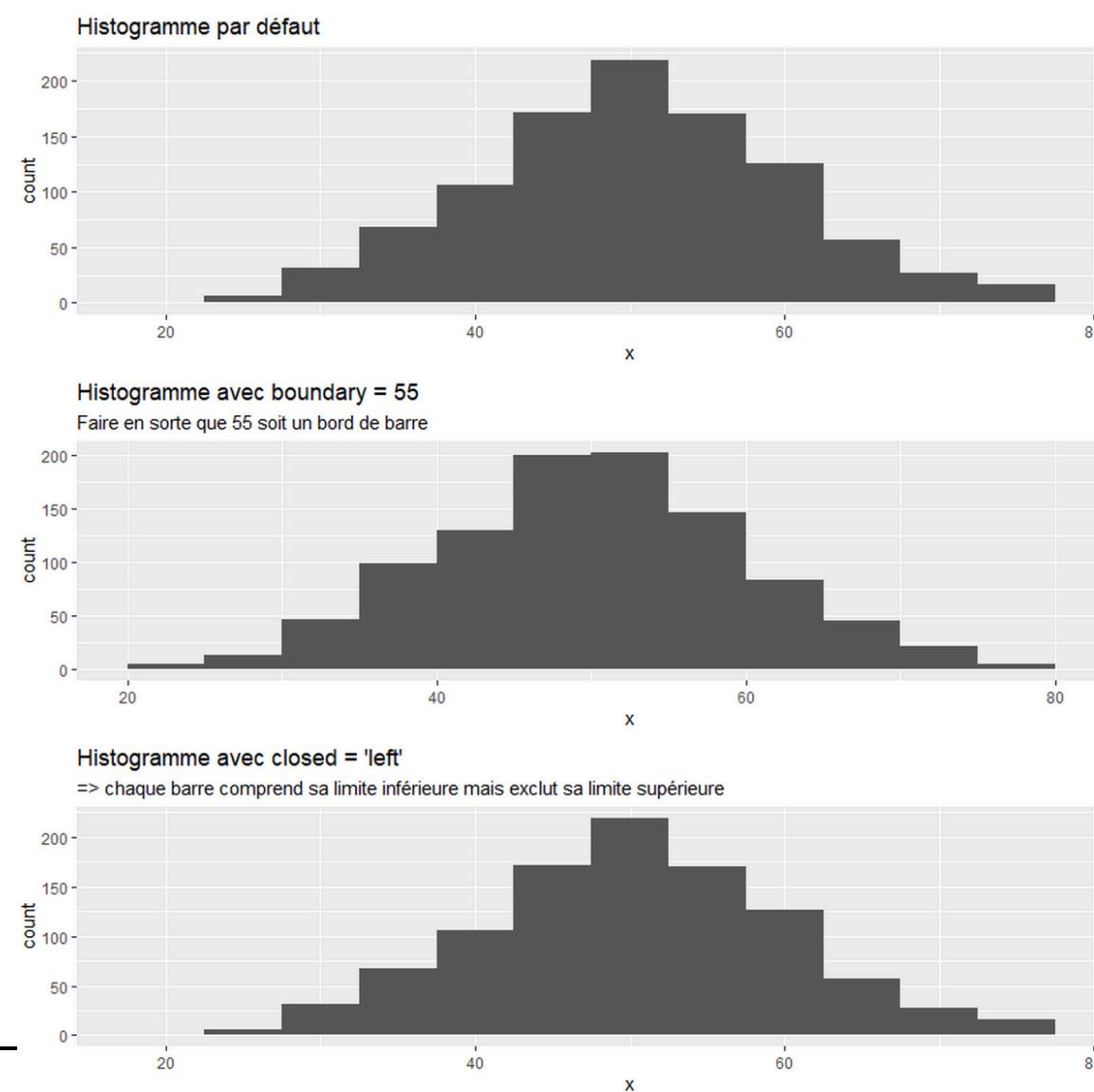
On peut croiser les données avec une variable qualitative en renseignant l'argument *fill* dans la fonction *aes()*.

```
# Répartition des exploitations enquêtées pour PKGC17
# selon la SAU et croisé avec la variable BIO
ggplot(data = donnees_gc2) +
  geom_histogram(mapping = aes(x = SAU_1, fill=BIO),bins = 50)
```



6.4 Les histogrammes

Les arguments *boundary* et *closed* de la fonction *geom_histogram()* peuvent être utiles dans certains cas d'usage.

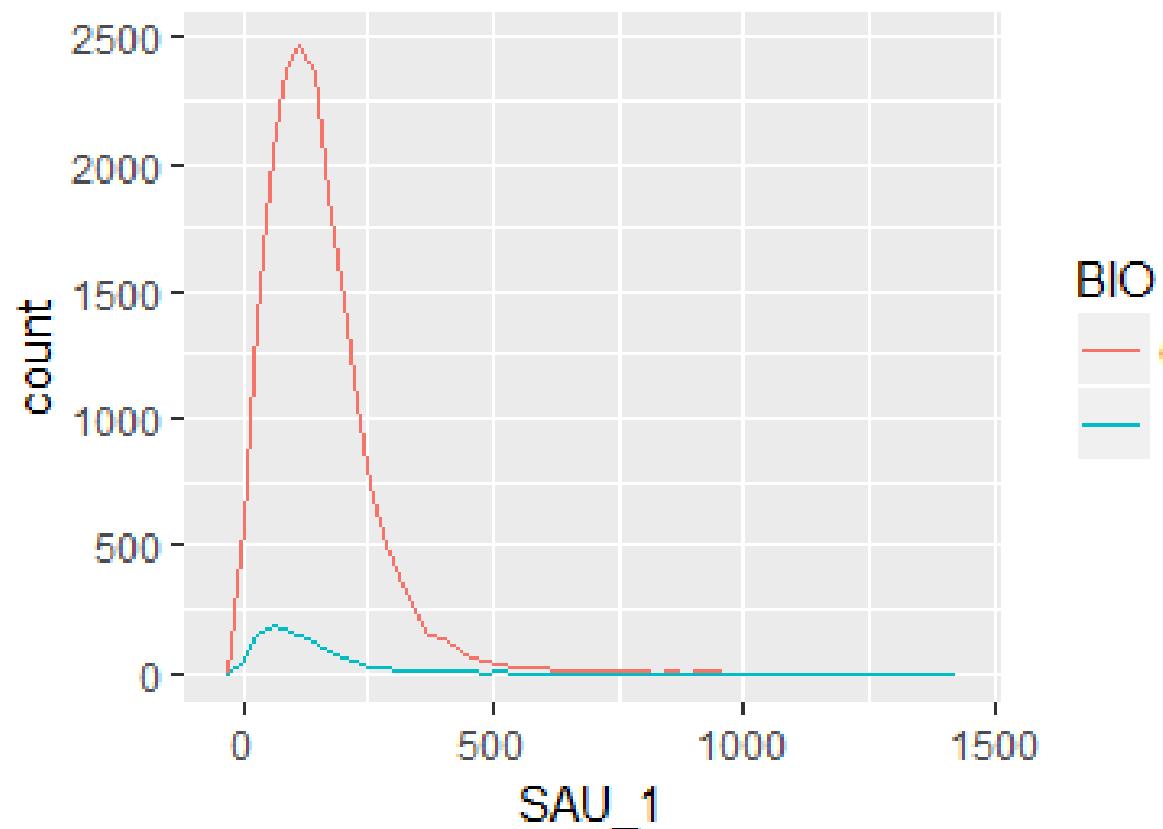




6.5 Les histogrammes

La fonction `geom_freqpoly()` permet de construire un polygone de fréquence pour chaque classe de la variable de croisement

```
# Répartition des exploitations enquêtées pour PKGC17
# selon la SAU pour les expl bio et non bio
ggplot(data = donnees_gc2) +
  geom_freqpoly(mapping = aes(x = SAU_1, colour=BIO),bins = 50)
```





7 Les nuages de points



7.1 Les nuages de points

Les nuages de points avec `geom_point()` :

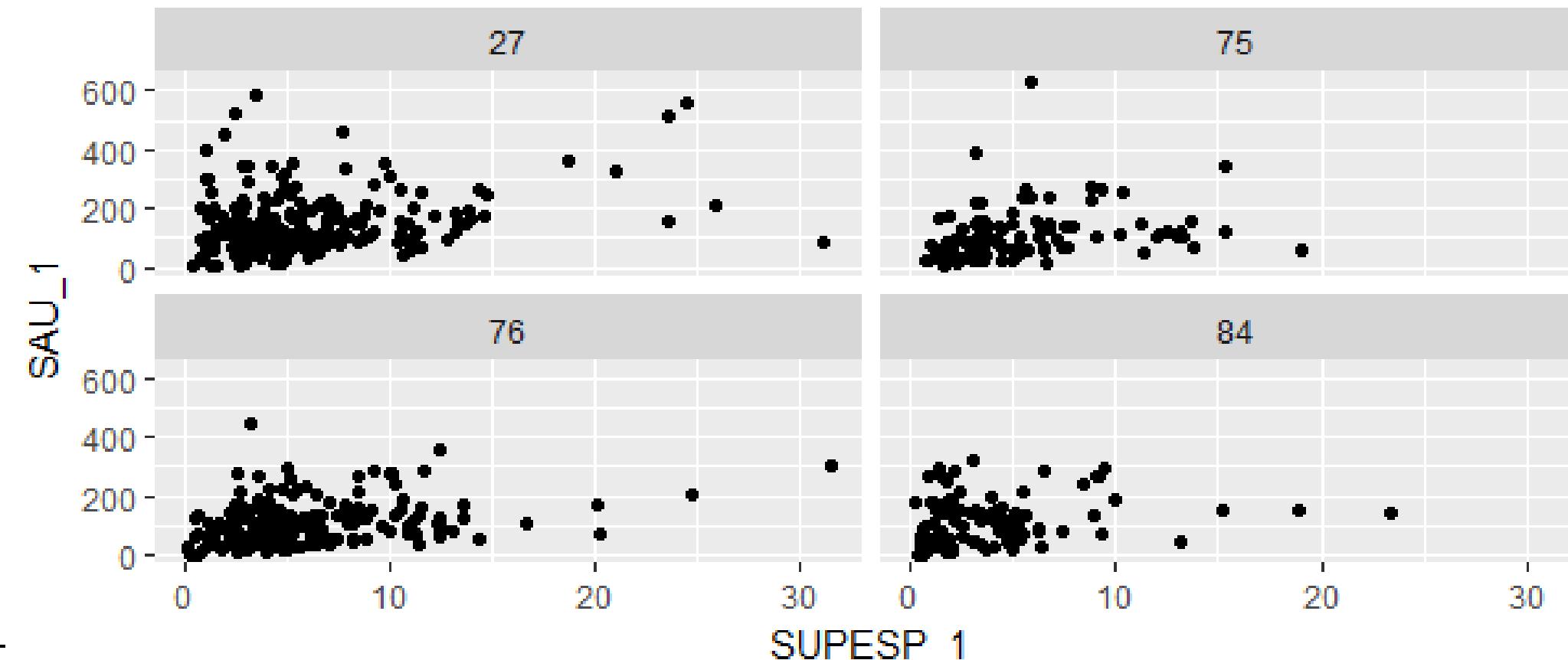
- **Objectif** : croiser deux variables dans un graphique de coordonnées cartésiennes
- On peut croiser les observations avec :
 - **une variable qualitative** en utilisant davantage l'argument `colour` ou `shape` de la fonction `aes()` (ou `fill` pour **certaines formes de points**)
 - **une variable quantitative** en utilisant l'argument `size` de la fonction `aes()`



7.2 Les nuages de points

On peut aussi créer un graphique pour chaque modalité de la variable de croisement avec la fonction `facet_wrap()`

```
# Reprise de l'exemple 3, croisement par région
ggplot(data = donnees_gc_soja) +
  geom_point(mapping = aes(x = SUPESP_1, y = SAU_1)) +
  facet_wrap(~REG)
```

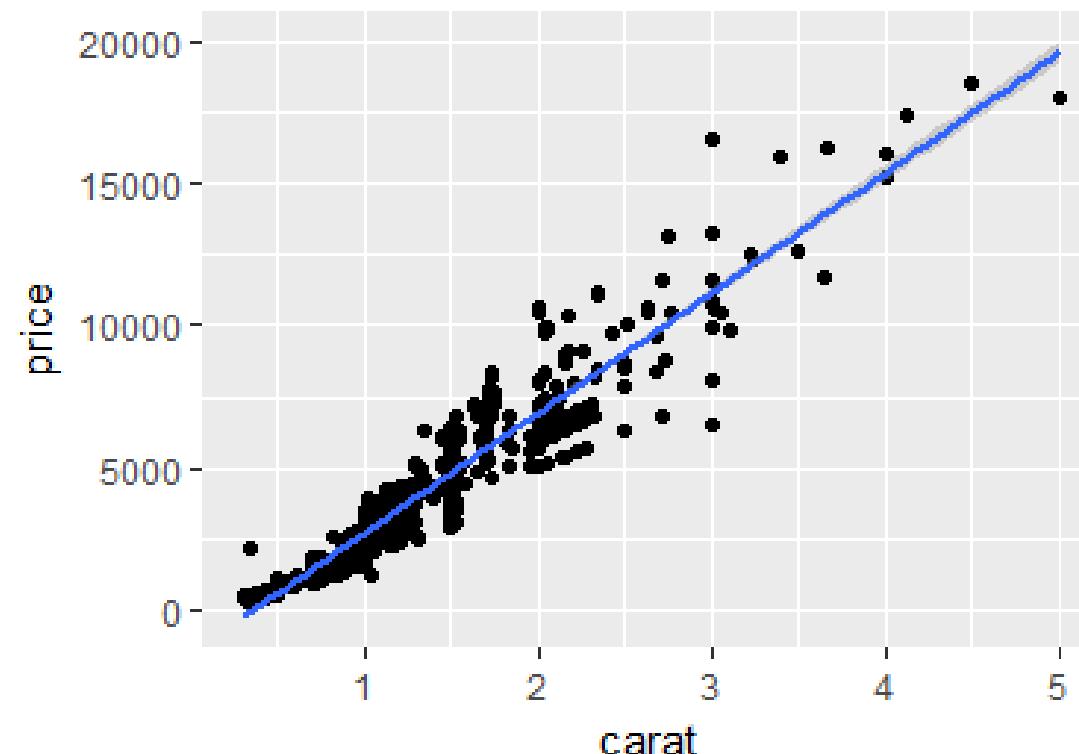


7.3 Les nuages de points

On peut ajouter une droite de régression linéaire sur un graphe construit comme un nuage de points en utilisant la fonction `geom_smooth(method = "lm")`.

- **Attention !** C'est une régression sans pondération

```
ggplot(data=diamonds[diamonds$clarity=="I1",],mapping = aes(x=carat,y=price)) +  
  geom_point() + geom_smooth(method="lm")
```



7.4 Les nuages de points

Pour ajouter une droite de régression linéaire qui tient compte des pondérations, il faut :

- Calculer les coefficients de régression par ailleurs

```
reg <- lm(y~x, data = tab, weights = pond)
```

⋮

- Récupérer ces coefficients

```
a <- reg$coefficients[2]  
b <- reg$coefficients[1]
```

- Ajouter manuellement une droite sur le graphique avec les coordonnées calculés précédemment

```
ggplot(data = tab, mapping = aes(x = x, y = y)) +  
  geom_point() + geom_abline(intercept = b, slope = a)
```



8 Les courbes





8.1 Les courbes

Les courbes avec `geom_line()` :

- **Objectif** : réaliser une courbe, en reliant les points d'un nuage de points par des segments, notamment à partir d'une variable temporelle en abscisse
- Le format de la table en entrée est particulier, il ne doit y avoir qu'une valeur d'ordonnées par valeur d'abscisse

	annee	part_agri_PIB
1	1970	7.52
2	1971	6.87
3	1972	7.68
4	1973	7.62
5	1974	6.09
6	1975	5.40
7	1976	5.15
8	1977	4.83
9	1978	4.69
10	1979	4.68
11	1980	4.06
12	1981	3.97
13	1982	4.55
14	1983	4.04
15	1984	3.78
16	1985	3.76
17	1986	3.63
18	1987	3.47
19	1988	3.21

8.2 Les courbes

On peut croiser les données avec une variable qualitative et obtenir autant de courbes que de modalités de la variable.

Exemple : réaliser sur un même graphique les courbes d'évolution de la part de l'agriculture dans le PIB pour quatre pays de l'Union Européenne.

Voilà les données initiales :

	Année	Allemagne	Espagne	France	RU
1	1995	1.07	4.21	2.73	1.50
2	1996	1.12	4.76	2.70	1.30
3	1997	1.11	4.69	2.63	1.15
4	1998	1.05	4.55	2.65	1.04
5	1999	1.04	4.21	2.51	0.97
6	2000	1.08	4.12	2.34	0.88
7	2001	1.18	4.01	2.34	0.81
8	2002	0.97	3.79	2.23	0.81
9	2003	0.89	3.73	2.06	0.84
10	2004	1.03	3.41	2.03	0.85
11	2005	0.78	3.03	1.87	0.62
12	2006	0.80	2.64	1.70	0.60
13	2007	0.84	2.71	1.80	0.62
14	2008	0.91	2.49	1.69	0.66
15	2009	0.76	2.34	1.46	0.56
16	2010	0.74	2.55	1.78	0.68
17	2011	0.81	2.46	1.84	0.64
18	2012	0.87	2.44	1.82	0.68



8.3 Les courbes

Il faut créer une variable qualitative correspondant au pays

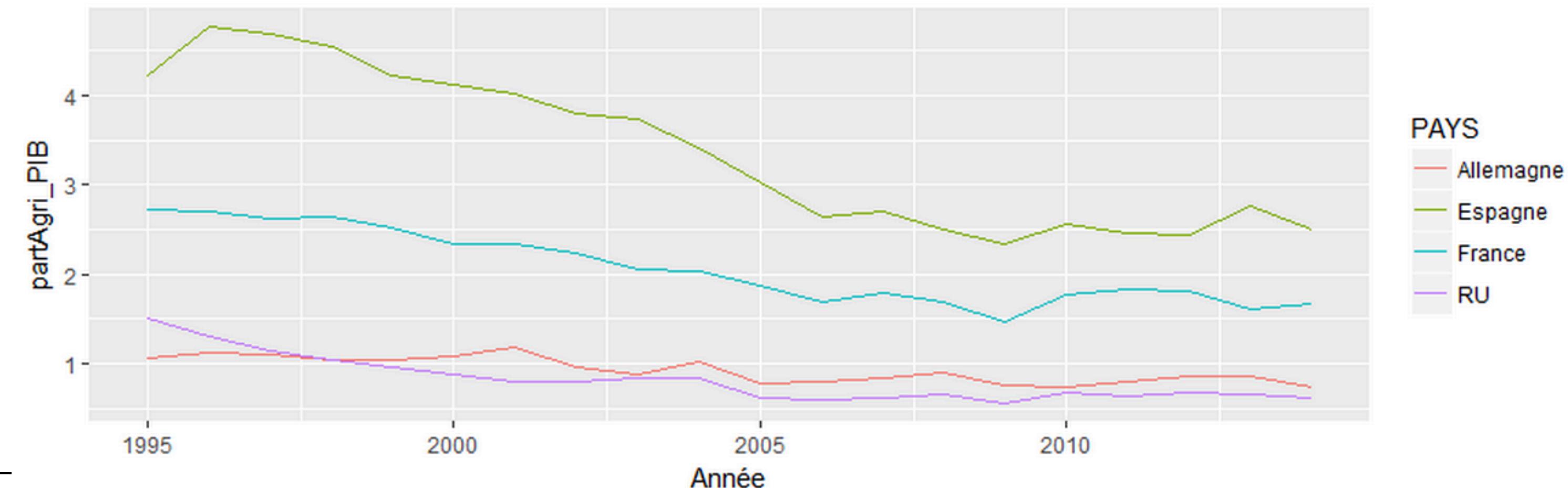
- Fonction *pivot_longer()*

```
1 evol_partagri_pib_UE2 <- evol_partagri_pib_UE %>%
2   pivot_longer(
3     cols = Allemagne:RU,
4     names_to = "PAYS",
5     values_to = "partAgri_PIB")
```



8.4 Les courbes

```
# Evolution de la part de l'agriculture dans le PIB de 4 pays de l'UE
ggplot(data = evol_partagri_pib_UE2,
        mapping = aes(x = Année, y = partAgri_PIB, col = PAYS)) +
        geom_line()
```





9 Les camemberts

9.1 Les camemberts

Les camemberts avec `geom_bar()` + `coord_polar()`

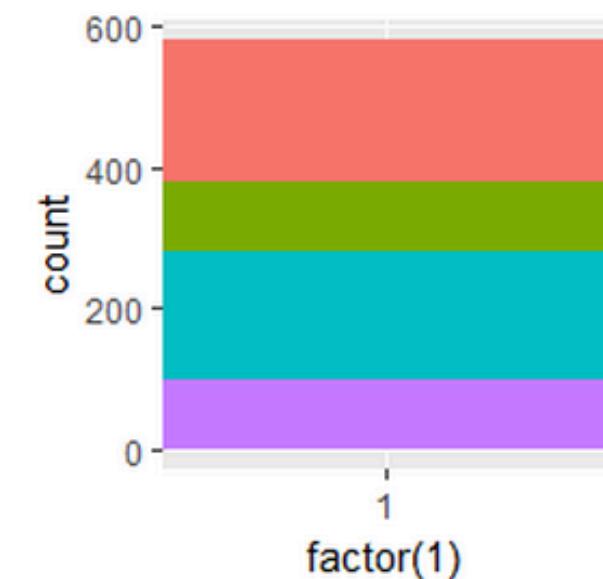
- **Objectif :** représenter la répartition en proportions des modalités d'une variable qualitative sur un ensemble de données

 **Attention**

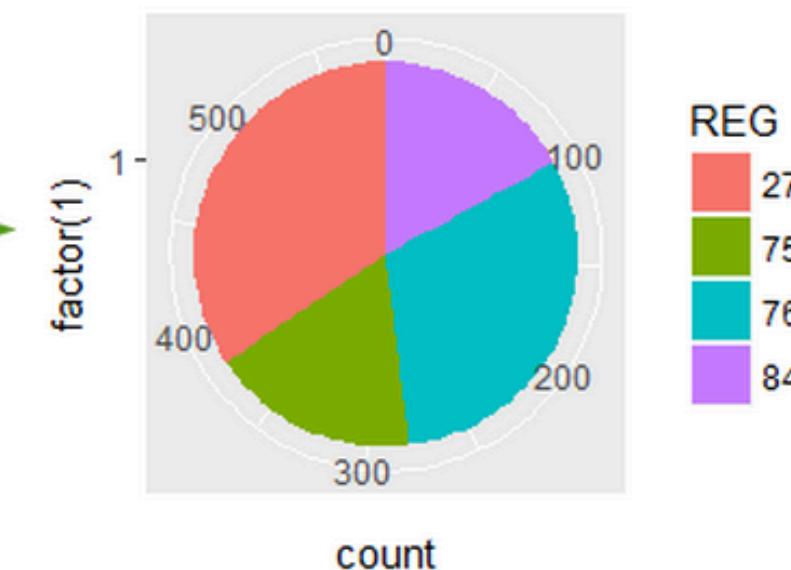
Il n'existe pas de fonction dans le package `ggplot2` permettant de construire directement un camembert. Pour ce faire, il faut utiliser la fonction `geom_bar()` et changer par ailleurs le système de coordonnées en coordonnées polaires

9.2 Les camemberts

```
### Camembert : répartition des exploitations enquêtées pour le soja dans PKGC17 par région
# Pour bien comprendre, construction du graphique à mettre ensuite
# en coordonnées polaires :
ggplot(donnees_gc_soja, aes(x = factor(1), fill = REG)) +
  geom_bar(width = 1)
# Passage du graphique en coordonnées polaires pour obtenir un camembert
ggplot(donnees_gc_soja, aes(x = factor(1), fill = REG)) +
  geom_bar(width = 1) +
  coord_polar(theta = "y")
```



Passage en
coordonnées
polaires



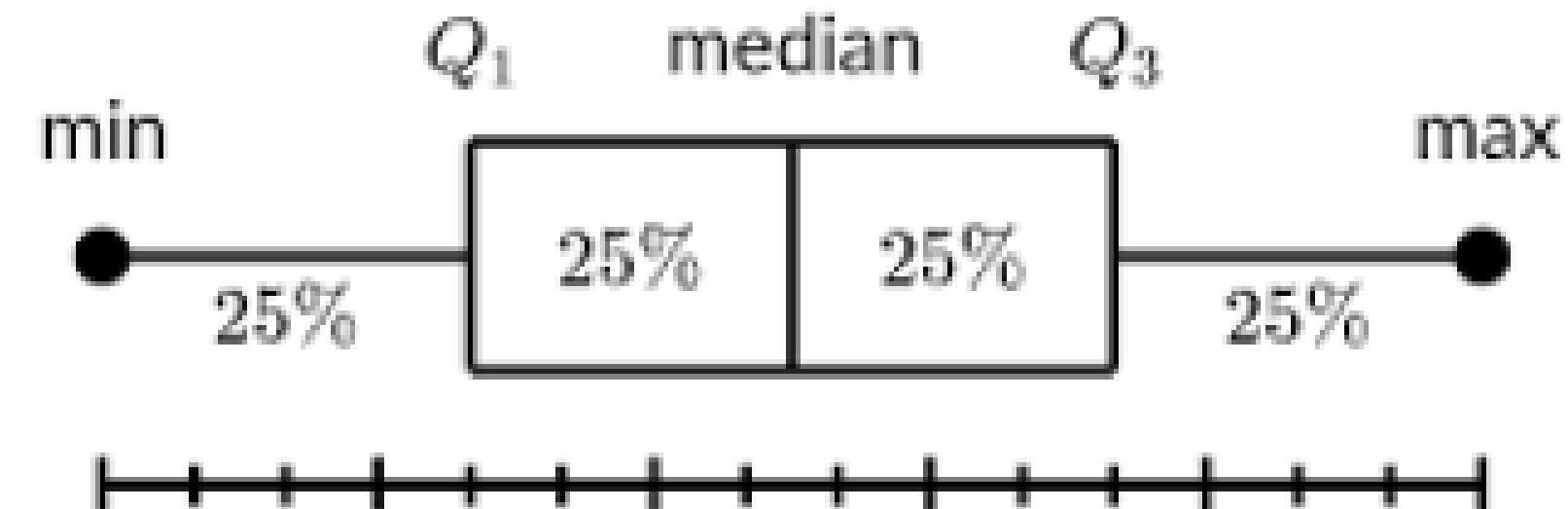


10 Les boîtes à moustaches

10.1 Les boîtes à moustaches

Les boîtes à moustache avec `geom_boxplot()`

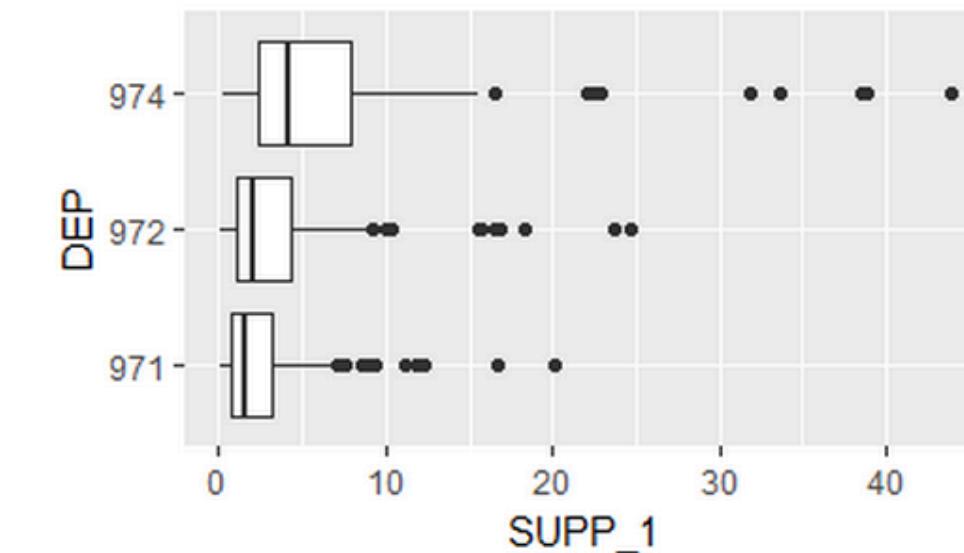
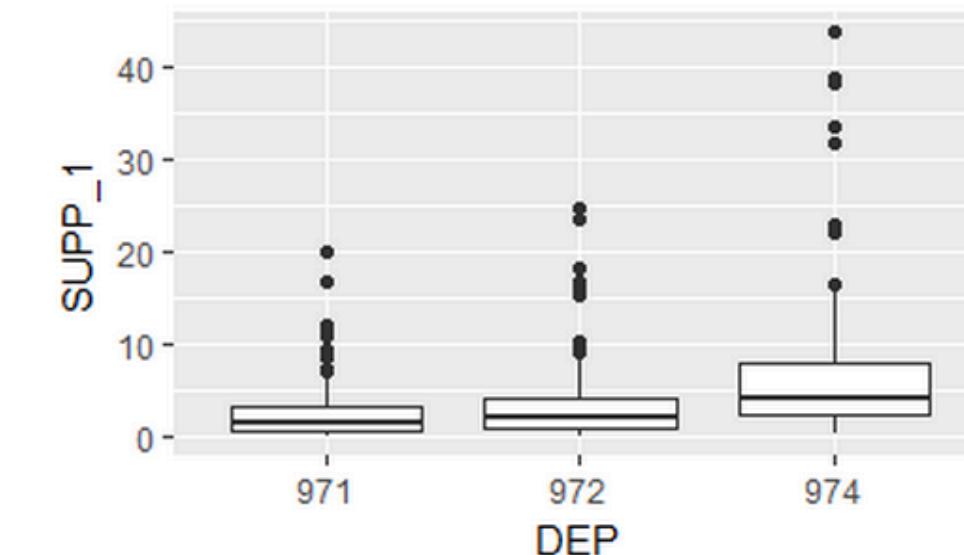
- **Objectif :** représenter la répartition d'une variable quantitative autour des quartiles de sa distribution
- Éventuellement, représenter cette répartition selon les modalités d'une variable qualitative : il y a alors donc deux variables à renseigner en paramètres
 - x : variable qualitative de croisement
 - y : variable quantitative à croiser



10.2 Les boîtes à moustaches

```
### Boxplot sur la surface en canne à sucre
### des exploitations enquêtées pour PKGC17
### par département
ggplot(data = donnees_gc_CS,
       mapping = aes(x=DEP, y = SUPP_1)) +
  geom_boxplot()
```

```
### Obtenir les boxplots à l'horizontale
### -> plus lisible ici
ggplot(data = donnees_gc_CS,
       mapping = aes(x=DEP, y = SUPP_1)) +
  geom_boxplot() +
  coord_flip()
```





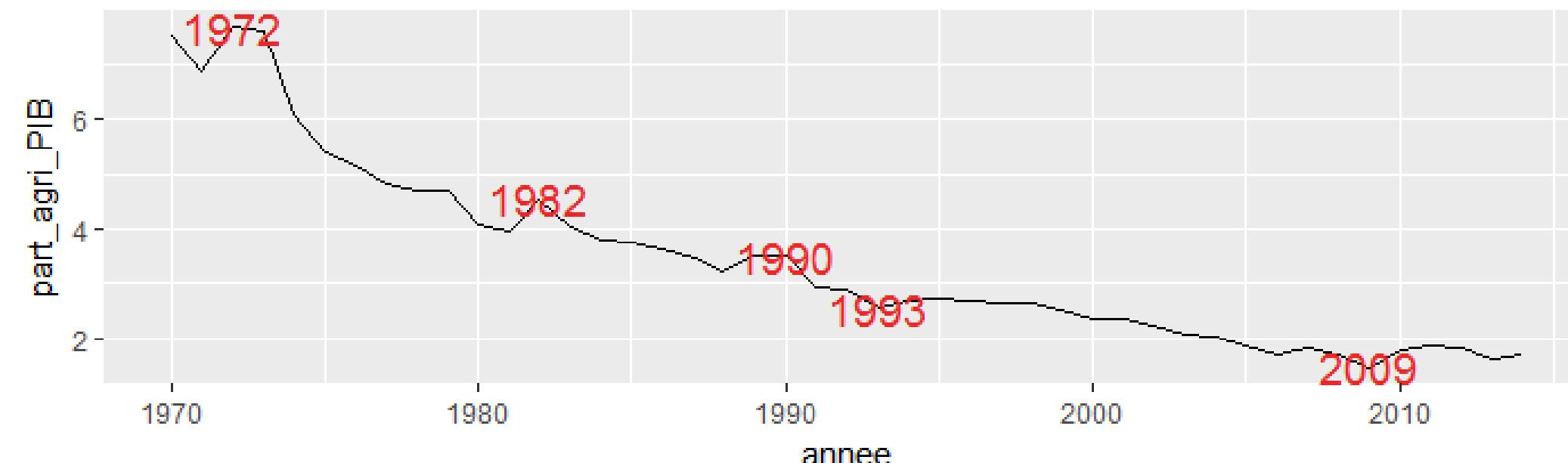
11 Ajout de libellés

:

11.1 Ajout de libellés

La fonction `geom_text()` permet d'ajouter un libellé sur les points souhaités d'un graphique

```
### Evolution de la part de l'agriculture dans le PIB français (source Banque Mondiale)
ggplot(data = evol_partagri_pib_fra, mapping = aes(x = annee, y = part_agri_PIB)) +
  geom_line() +
  geom_text(data=evol_partagri_pib_fra %>% # table des observations pour lesquelles on
            filter(annee%in%c(1972,1982,1990,1993,2009)), # souhaite un affichage
            aes(label = annee), # variable contenant le texte à afficher
            col = "firebrick2", # couleur du texte à afficher
            size = 5) # taille du texte à afficher
```





12 Habillage du graphique

12.1 Habillage du graphique

Habillage du graphique :

- Ajout d'annotation
- Gestion des axes
- Modification du thème
- Ajout de titre, sous-titre, source, libellés de légende...

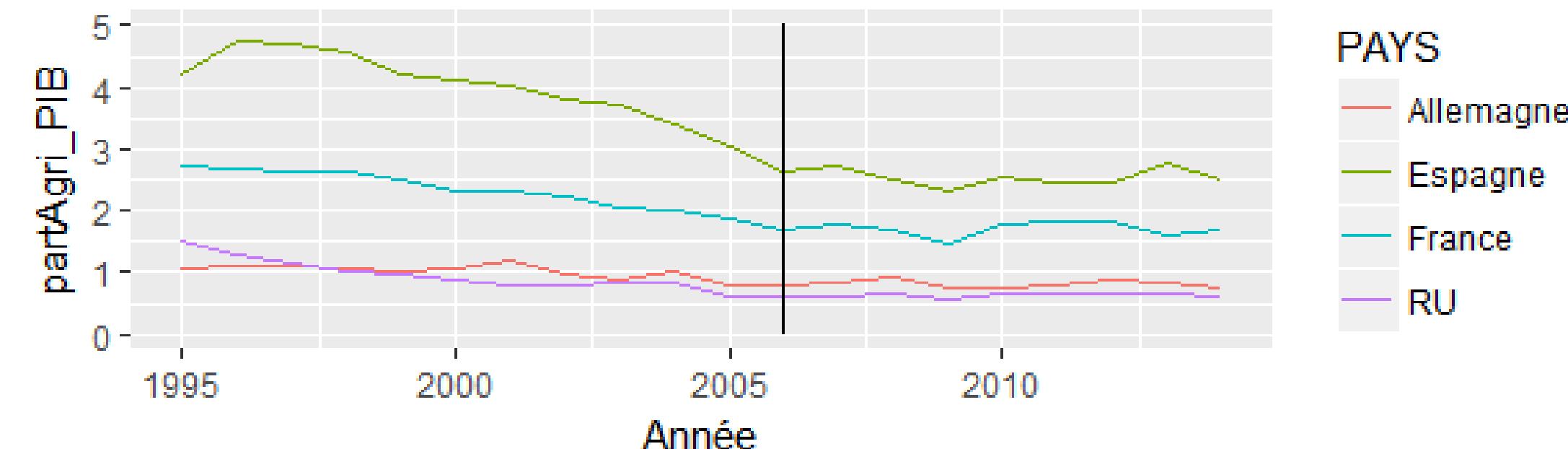
⋮
⋮
⋮
⋮
⋮



12.2 Habillage du graphique

- Ajouter un segment de droite en précisant les coordonnées des extrémités : *geom = "segment"*

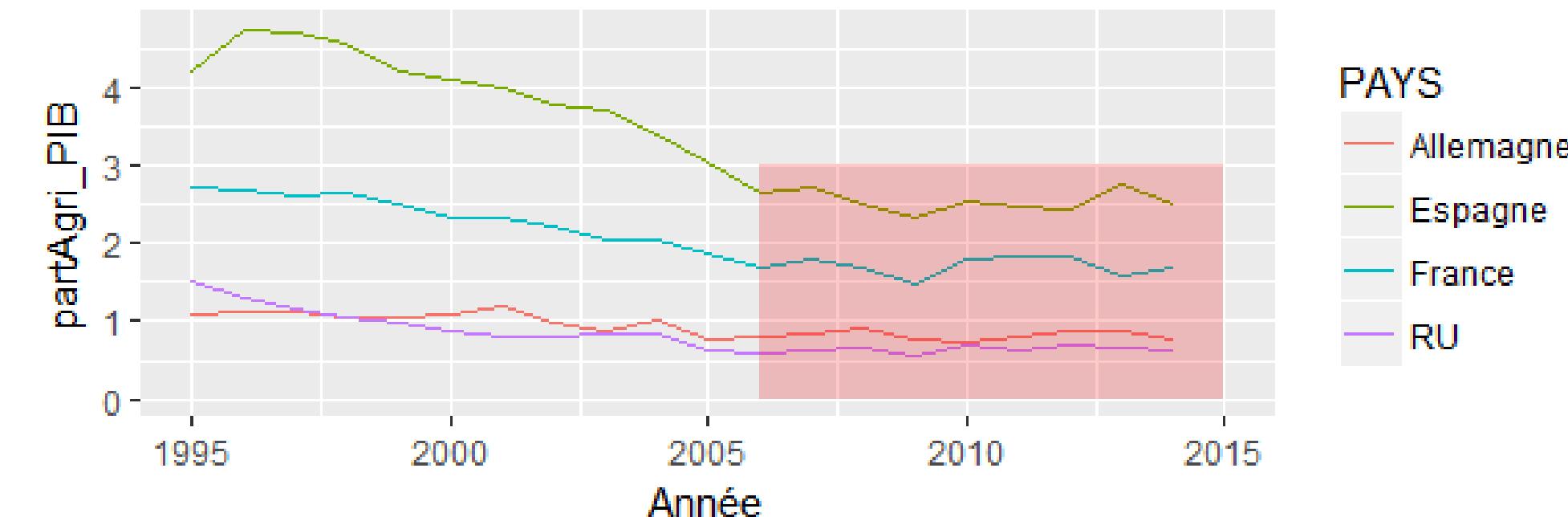
```
# Evolution de la part de l'agriculture dans le PIB de 4 pays de l'UE
ggplot(data = evo1_partagri_pib_UE2,
        mapping = aes(x = Année, y = partAgri_PIB, col = PAYS)) +
  geom_line() + annotate(geom="segment",x=2006,xend=2006,y=0,yend=5)
```



12.3 Habillage du graphique

- Ajouter une figure rectangulaire en précisant les coordonnées des points en diagonale : *geom = "rect"*

```
# Evolution de la part de l'agriculture dans le PIB de 4 pays de l'UE
ggplot(data = evol_partagri_pib_Ue2,
        mapping = aes(x = Année, y = partAgri_PIB, col = PAYS)) +
  geom_line() +
  annotate(geom="rect",xmin=2006,xmax=2015,ymin=0,ymax=3,alpha=0.2,fill="red")
```



12.4 Habillage du graphique

On modifie l'échelle des axes avec les fonctions préfixées `scale_XXX()` et leurs arguments *breaks* :

- `scale_x_discrete()` : gérer l'axe des x d'une **variable qualitative**
- `scale_x_continuous()`* : gérer l'axe des x d'une **variable continue**
- `scale_x_manual()` : gérer manuellement **l'axe des x**
- `scale_fill_manual()` : gérer manuellement la variable de croisement de type **fill**
- `scale_colour_manual()` : gérer manuellement la variable de croisement de type **colour**

12.5 Habillage du graphique

- On modifie l'apparence générale du graphique à l'aide de la fonction `theme()`. De nombreux paramètres à consulter dans l'aide de la fonction permettent de modifier chaque élément d'un graphique.

Par exemple, pour supprimer le fond gris d'un graphique et transformer les lignes des axes en lignes grises d'épaisseur 0,2, il faut ajouter le bloc suivant au code du graphique :

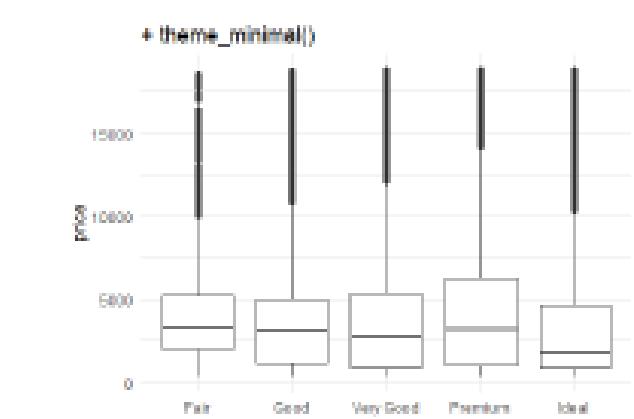
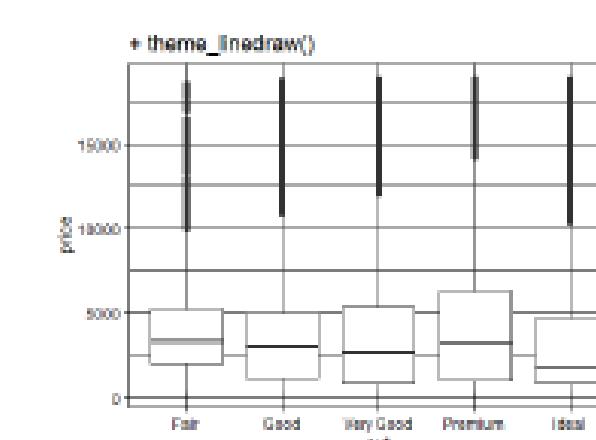
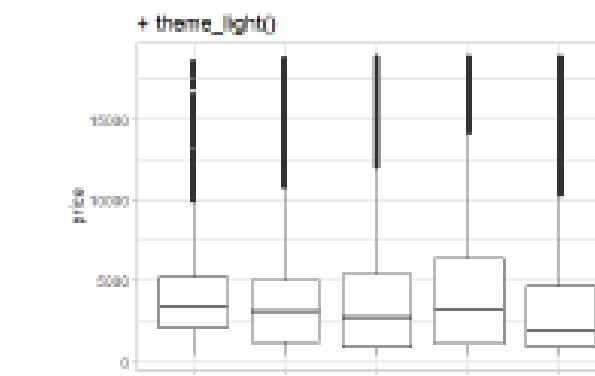
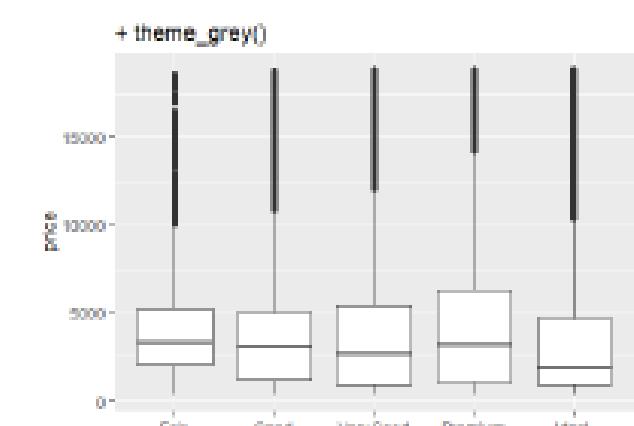
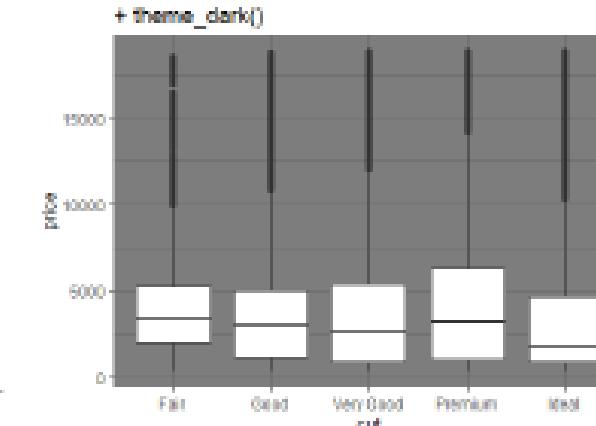
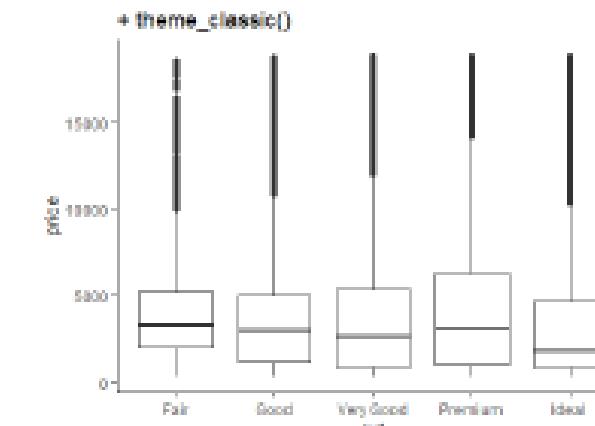
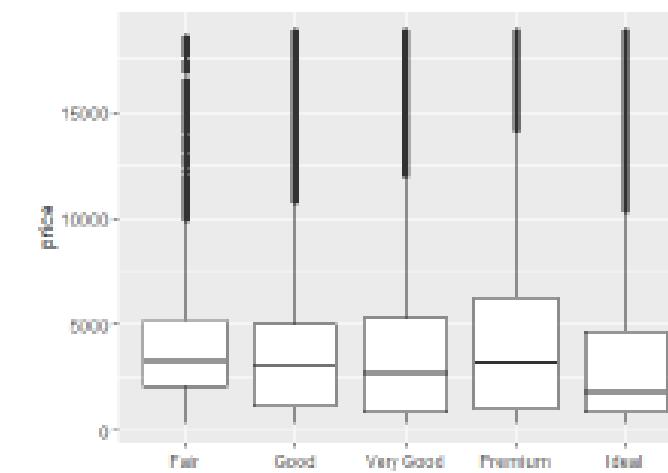
```
+ theme(panel.background = element_blank(),  
       axis.line = element_line(colour = "gray70", size = 0.2))
```



12.6 Habillage du graphique

- Il existe des thèmes prédéfinis que l'on peut ajouter à un graphique

```
ggplot(data = diamonds,
       mapping = aes(x = cut, y = price)) + geom_boxplot()
```

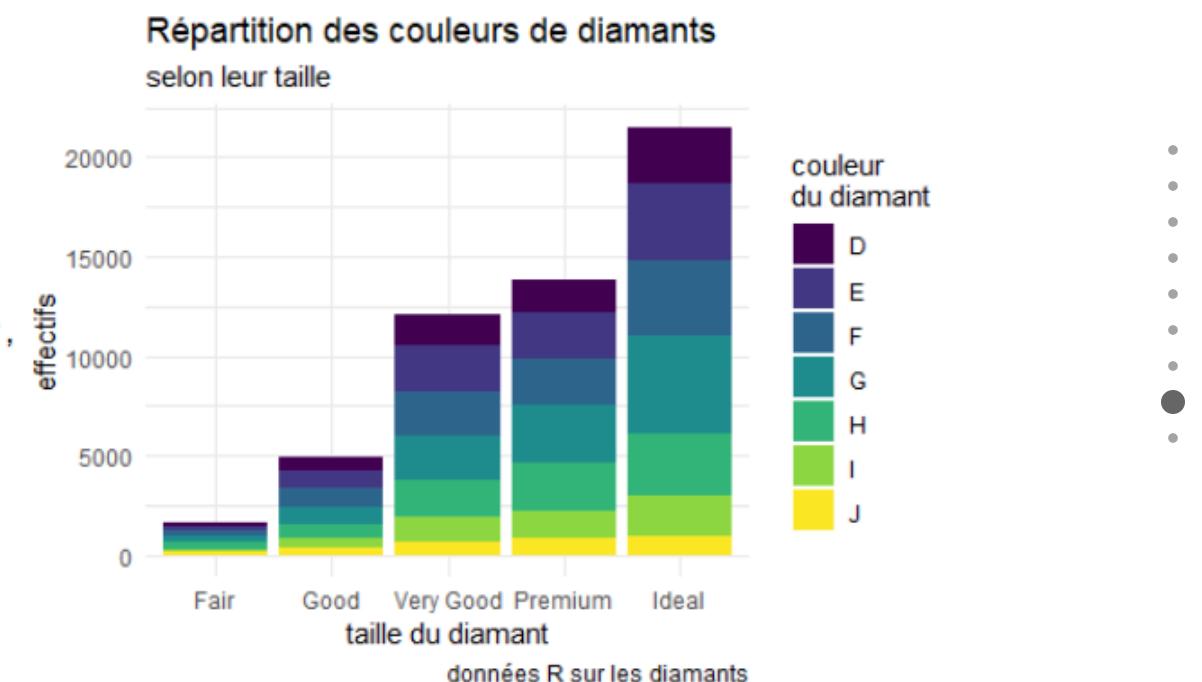


12.7 Habillage du graphique

On peut modifier l'ensemble des libellés avec la fonction **labs()**

- *title* = titre du graphique
- *subtitle* = sous-titre
- *x,y* = noms des axes
- *fill, colour, size, alpha* = intitulé de la légende
- *caption* = source de données

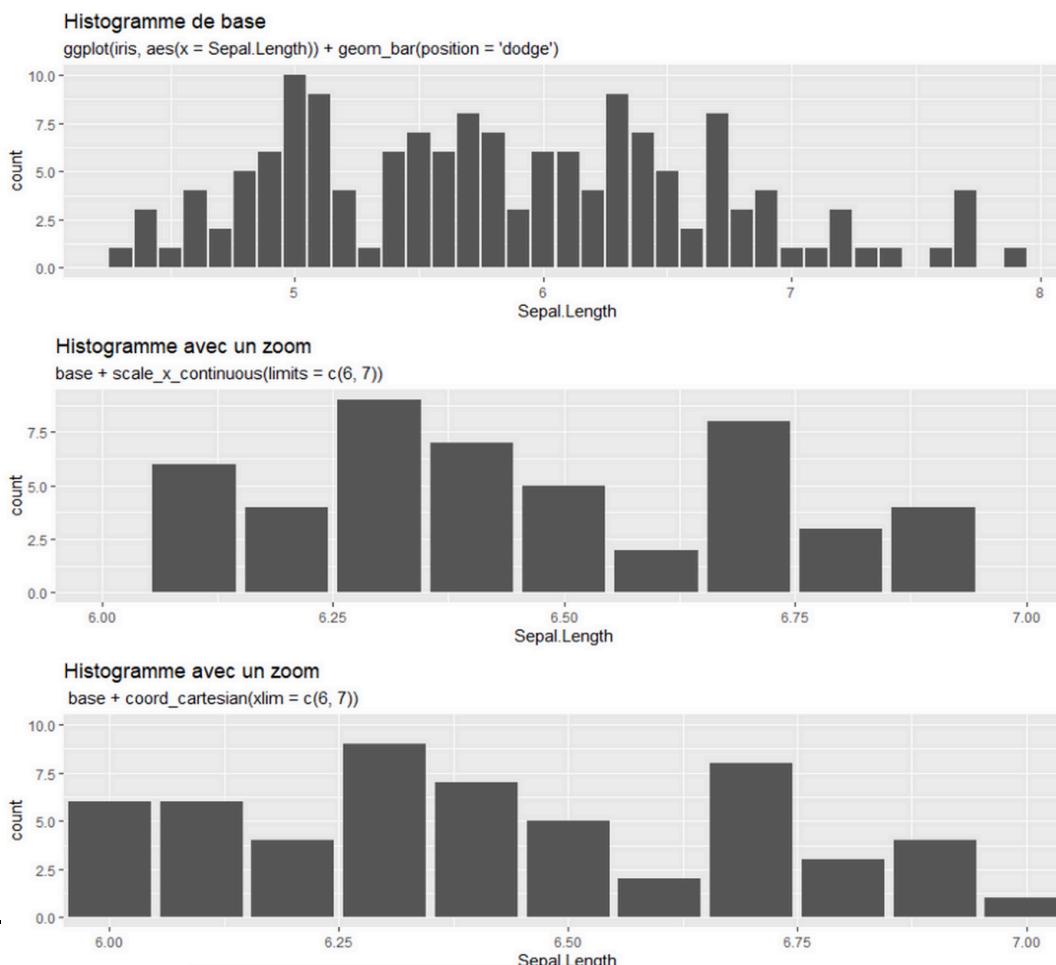
```
ggplot(data = diamonds,
       mapping = aes(x = cut, fill = color)) +
  geom_bar() +
  theme_minimal() +
  labs(title = "Répartition des couleurs de diamants",
       subtitle = "selon leur taille",
       x = "taille du diamant", y = "effectifs",
       fill = "couleur \ndu diamant",
       caption = "données R sur les diamants")
```



12.8 Zoom sur un graphique

Attention à l'argument **limits** de la fonction `scale_x_continuous()` qui enlève les données hors de l'intervalle. Selon que celui-ci est ouvert ou fermé, ggplot2 peut supprimer des barres.

=> Pour zoomer sur un graphique, la bonne pratique est d'utiliser la fonction `coord_cartesian()` et son argument **xlim**.



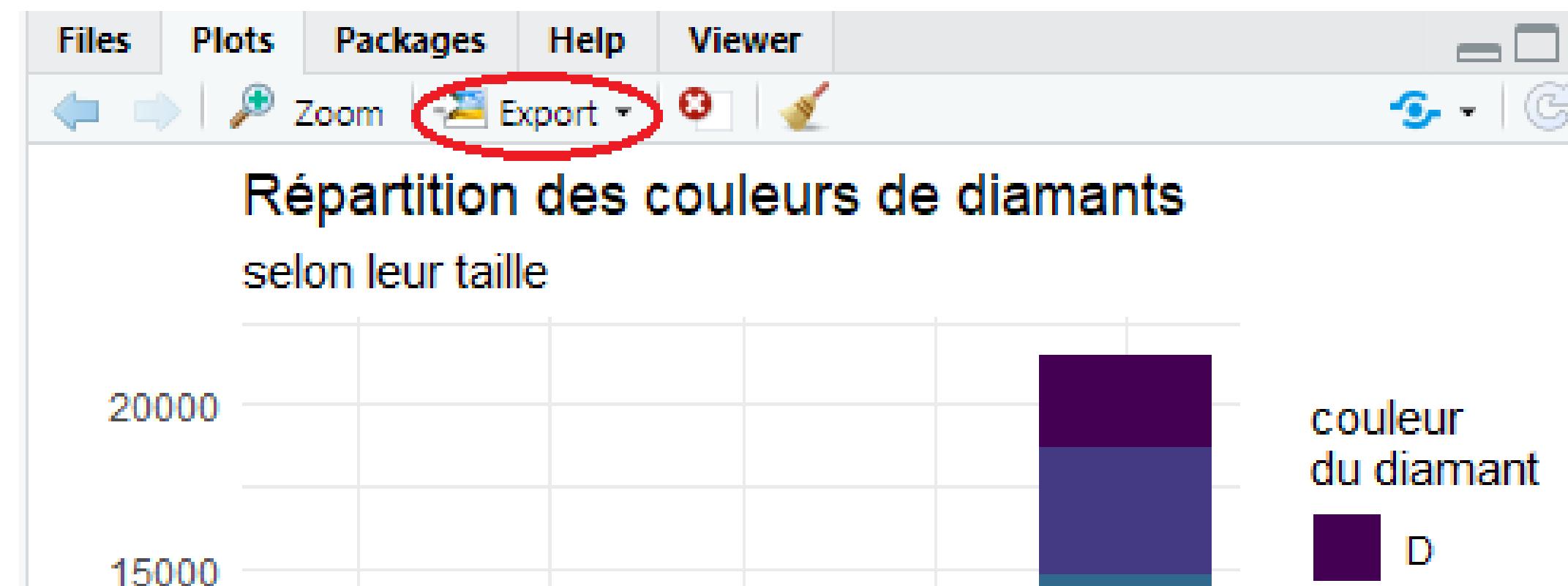


13 Sauvegarde du graphique

:

13.1 Sauvegarde du graphique

- La sauvegarde du graphique peut se faire avec le bouton « export » de l'onglet « Plots »



- On peut également sauvegarder un graphique avec la fonction `ggsave()`
 - possibilité d'exporter au format pdf, svg, png, etc. → il suffit de l'indiquer dans l'extension du nom de fichier en sortie
 - par défaut, la fonction exporte le dernier graphique possibilité de choisir la taille et la résolution du graphique exporté



14 Pour aller plus loin...





14.1 Liens

- Des exemples de graphiques inspirants (à mettre en favoris) <https://www.r-graph-gallery.com/>
- Le livre de recette de ggplot2 (UN MUST) <https://r-graphics.org/>
- Un flipbook pour bien comprendre la superposition des éléments d'un graphique. Par exemple https://evamaerey.github.io/ggplot_flipbook/ggplot_flipbook_xaringan.html#1
- Une liste des différentes extensions de ggplot2 <https://exts.ggplot2.tidyverse.org/gallery/>
- Le package {esquisse} pour construire une ébauche de graphique et générer le code => <https://dreamrs.shinyapps.io/esquisse/>



15 Exercices



15.1 Exercice 1

Exercice 1

À partir des données de la base prenoms :

- Tracer la courbe de l'évolution du nombre de naissances de Camille depuis 2000. Choisir une largeur de ligne de 1 et colorer la courbe en orange. **Attention, la variable correspondant à l'année de naissance doit être numérique.**

Indication : agréger d'abord les données pour l'ensemble des départements pour obtenir le nombre total d'enfants prénommés «Camille» pour chaque année depuis 2000.

- Tracer la courbe de l'évolution du nombre de naissances de « Camille » selon le sexe de l'enfant.
- Pour aller plus loin : ajouter au graphique la courbe représentant le total (F+G) et mettre en pointillés les deux autres courbes.



15.2 Résultat attendu - exercice 1

