Produire un diagramme en bâtons avec R et avec le package ggplot2

Bachar Cheaib et Katherine Tanaka

2015-04-13

Table des matières

Introduction – Les diagrammes en bâtons

Le diagramme à bâtons est une représentation visuellement informative et descriptive d'un jeu de données de variables catégoriques [1]. Ce graphique permet de visualiser, pour chacune de ces variables, une fréquence absolue ou relative qui correspond à la hauteur du bâton. Les barres d'un diagramme en bâtons, contrairement aux barres de l'histogramme, sont disjointes, puisque l'axe d'où elles partent ne représente pas un continuum de valeurs [2].

R, par le biais de la fonction `barplot`, ainsi que le populaire *package* `ggplot2` permettent tous les deux de produire des diagrammes en bâtons. Voici la marche à suivre pour produire ces graphiques.

Note :

Les exemples du document ont été produits à l'aide des données *hour.csv* du *Bike sharing dataset*, disponibles au *UCI Machine Learning Repository*. Ces données représentent les conditions et le nombre de locations par heure dans un système de prêt de bicyclettes [3].

Le jeu de données à l'étude : conditions météorologiques horaires

Le jeu de données *hour.csv*, une fois chargé, a la forme suivante :

```
head(hour.dataset)
```

```
##   instant     dteday season yr mnth hr holiday weekday workingday
## 1       1 2011-01-01      1  0    1  0       0       6          0
## 2       2 2011-01-01      1  0    1  1       0       6          0
## 3       3 2011-01-01      1  0    1  2       0       6          0
## 4       4 2011-01-01      1  0    1  3       0       6          0
## 5       5 2011-01-01      1  0    1  4       0       6          0
## 6       6 2011-01-01      1  0    1  5       0       6          0
##   weathersit temp  atemp  hum windspeed casual registered cnt
## 1          1 0.24 0.2879 0.81    0.0000      3         13  16
## 2          1 0.22 0.2727 0.80    0.0000      8         32  40
## 3          1 0.22 0.2727 0.80    0.0000      5         27  32
## 4          1 0.24 0.2879 0.75    0.0000      3         10  13
## 5          1 0.24 0.2879 0.75    0.0000      0          1   1
## 6          2 0.24 0.2576 0.75    0.0896      0          1   1
```

Ce sont les colonnes `season` (*saison*), `yr` (*année*) et `weathersit` (*température au site de location*) qui seront utilisées pour produire les graphiques. L'information qui en découle est codée sous forme de chiffres, qui peuvent être interprétés par le fichier Lisez-moi (*Readme.txt*) disponible avec les données téléchargées. En voici un résumé.

Tableau 1. Signification des colonnes du jeu de données

Nom de la colonne	Valeur numérique	Descriptif
`season`	2	Printemps
	3	Été
	4	Automne
	1*	Hiver
`year`	0	2011
	1	2012
`weathersit`	1	Dégagé à partiellement nuageux
	2	Nuageux et brumeux
	3	Pluie ou neige légère
	4	Orage et fortes précipitations

Source : Fichier *Readme.txt* disponible au téléchargement du *Bike sharing dataset* * Il y a une erreur dans ce ficher, toutes les valeurs de saison sont décalées de 1, selon les dates auxquelles les saisons sont enregistrées.

Pour nous simplifier la tâche, nous pourrions déjà convertir ces trois colonnes en facteurs, dont les niveaux seront donnés par les descriptifs.

```
# Pour la saison
hour.dataset$season <- factor(hour.dataset$season, labels = c("Hiver",
                                                              "Printemps", "Été",
```

```
                                                          "Automne"))
# Pour l'année
hour.dataset$yr <- factor(hour.dataset$yr, labels = c("2011", "2012"))

# Pour la température
hour.dataset$weathersit <- factor(hour.dataset$weathersit,
                                  labels = c("Dégagé",
                                             "Nuage, brume",
                                             "Pluie ou neige",
                                             "Orage"))
# Les trois colonnes concernées contiennent maintenant des facteurs
with(hour.dataset, {
  str(season)
  str(yr)
  str(weathersit)
})
```

```
##  Factor w/ 4 levels "Hiver","Printemps",..: 1 1 1 1 1 1 1 1 1 1 ...
##  Factor w/ 2 levels "2011","2012": 1 1 1 1 1 1 1 1 1 1 ...
##  Factor w/ 4 levels "Dégagé","Nuage, brume",..: 1 1 1 1 1 2 1 1 1 1 ...
```

Produire un diagramme à bâtons avec la fonction `barplot`

```
barplot(height, ...)
```

Fonction de base barplot, selon les fiches d'aide R

Cette fonction permet de créer des diagrammes en bâtons, horizontaux ou verticaux. Faisant partie de la famille des fonctions du package `graphics`, `barplot` peut utiliser certains arguments communs à toutes les fonctions graphiques (comme `main`, `xlim` ou `ylab`), en plus de certains arguments spécifiques.

Dans son cas de figure le plus simple, `barplot` accepte comme paramètre `height` soit un vecteur, soit une matrice. Dans le cas présent, nous utiliserons un **vecteur**.

Si `height` est un vecteur, `barplot` tracera un diagramme où chaque valeur (numérique) sera représenté par une barre. De plus, si les valeurs sont nommées, `barplot` ajoutera sous les bâtons le nom des valeurs comme axe des x.

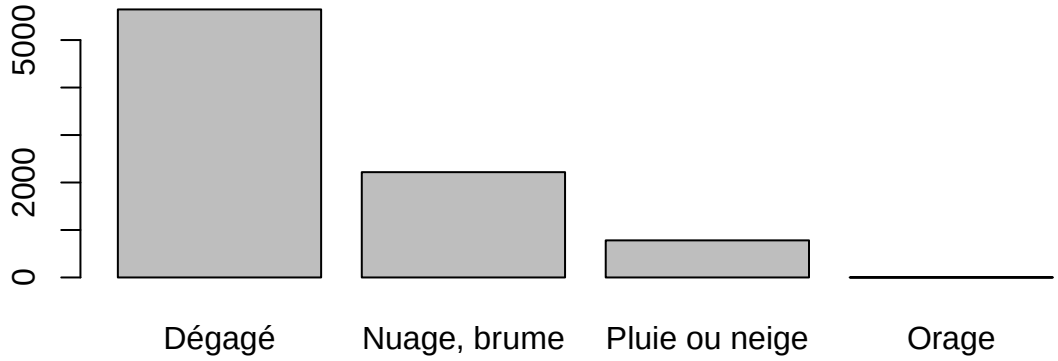
Cependant, les données dans l'état actuel sont inutilisables. En effet, si on avait produit un diagramme à bâtons sur la température du fichier original, on aurait obtenu ceci, un nombre de barres égalisant le nombre d'heures dans notre jeu de données. C'est là que notre conversion en facteurs s'avère utile...

En effet, avec les fonctions `table` ou `xtabs`, nous pouvons désormais créer une table de fréquence de la température et extraire ainsi le nombre d'observations par catégorie. À des fins d'exemple, seules les données de 2011 seront utilisées.

```
t.temperature <- xtabs(~ weathersit, data = hour.dataset, subset = yr == 2011)
t.temperature
```

```
## weathersit
##         Dégagé   Nuage, brume Pluie ou neige          Orage
##           5645           2218            781              1
```

```
barplot(t.temperature)
```



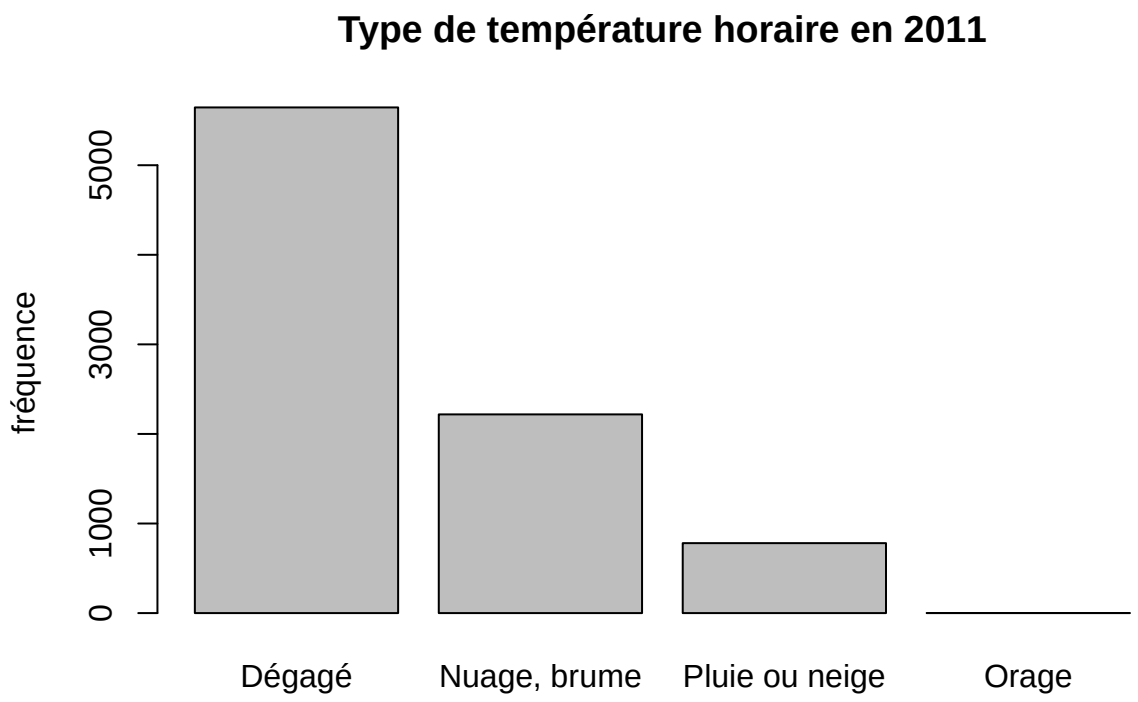
L'avantage d'avoir converti la colonne `weathersit` en facteur est que les valeurs des niveaux sont reportées dans la table de fréquence, puis dans le graphique pour devenir le nom des barres. Bien sûr, nous aurions pu rajouter le nom de ces barres avec l'argument `names.arg` de la façon suivante :

```
# Exemple
barplot(t.temperature, names.arg = c("Dégagé", "Nuage, brume",
                                     "Pluie ou neige", "Orage"))
```

Certaines informations supplémentaires pourraient être ajoutées pour clarifier ce graphique, à même la commande `barplot`.

- L'argument `ylab` sera utile pour nommer l'axe des y.
- L'argument `main` permettra d'intégrer un titre.

```
barplot(t.temperature, ylab = "fréquence", main = "Type de température horaire en 2011")
```



Voilà qui est beaucoup plus informatif !

Certains arguments facultatifs permettent de modifier l'apparence d'un diagramme en bâtons produit avec `barplot`. Ces arguments sont présentés dans le tableau suivant.

Tableau 2. Arguments facultatifs de barplot

Paramètres	Description
width	Permet de produire des barres de largeur variable
space	L'espace précédant chaque barre. Peut être variable aussi
beside	Permet d'empiler les barres au lieu de les juxtaposer
horiz	Trace un diagramme horizontal ou vertical

L'ensemble des paramètres qui constituent `barplot` peuvent être visionnées avec `help(barplot)`.

Produire un diagramme à bâtons simple avec le package `ggplot2`

Ce package, créé par Hadley Wickham, est basé sur une construction systémique des graphiques décrite par Leland Wilkinson dans son livre *Grammar of Graphics* [4]. Dans cette approche, tous les éléments peuvent être ajoutés au graphique sous forme de couches successives, même les barres ou les points ! Pour produire un graphique avec `ggplot2`, il faut donc faire une **somme** de différents éléments à partir d'un object original, l'objet `ggplot`.

La création d'un diagramme en bâtons de base avec `ggplot2`, puis d'un diagramme avancé nous permettra de visualiser certaines des fonctions de ce package.

L'objet `ggplot`

```
ggplot(data, aes(x, y, <other aesthetics>))
```

Une des trois façons d'appeler `ggplot` selon la documentation associée

Contrairement à la fonction `barplot` de R, la fonction `ggplot` ne nous demandera pas de modifier les données avec une table de fréquence, puisque ce sont les fonctions du package qui s'en occuperont. De plus, `ggplot` prend en entrée un *dataframe* seulement.

Le premier objet à créer est donc un `ggplot` qui contient nos données, ici, un sous-ensemble de hour.dataset à l'année 2011. En assignant l'objet à une variable, nous pourrons rajouter plusieurs ensembles de couches à tout moment.

```
# Initiation d'un objet ggplot pour les observations de 2011 seulement
library(ggplot2)
a <- ggplot(subset(hour.dataset, yr == 2011), aes(x = weathersit, fill = weathersit))
```

À la suite de l'argument `data`, nous avons aussi ajouté des paramètres esthétiques `aes` (pour *aesthetic*). Deux arguments ont été utilisés.

- `x`, qui donne ce qui sera mesuré en axe des x, ici les types de température
- `fill`, qui définit la couleur. Cet argument sera repris plus tard par la fonction qui crée les barres du diagramme (`geom_bar`). L'avantage d'utiliser `fill` dans `aes` est qu'on laisse les différentes fonctions attribuer des couleurs en fonction du nombre de niveaux de weathersit, c'est-à-dire quatre.

L'argument `y` ne sera pas utilisé ici, puisque `geom_bar` va par défaut compter la fréquence absolue pour ce qu'on lui a défini en axe des x.

Rajouter des couches

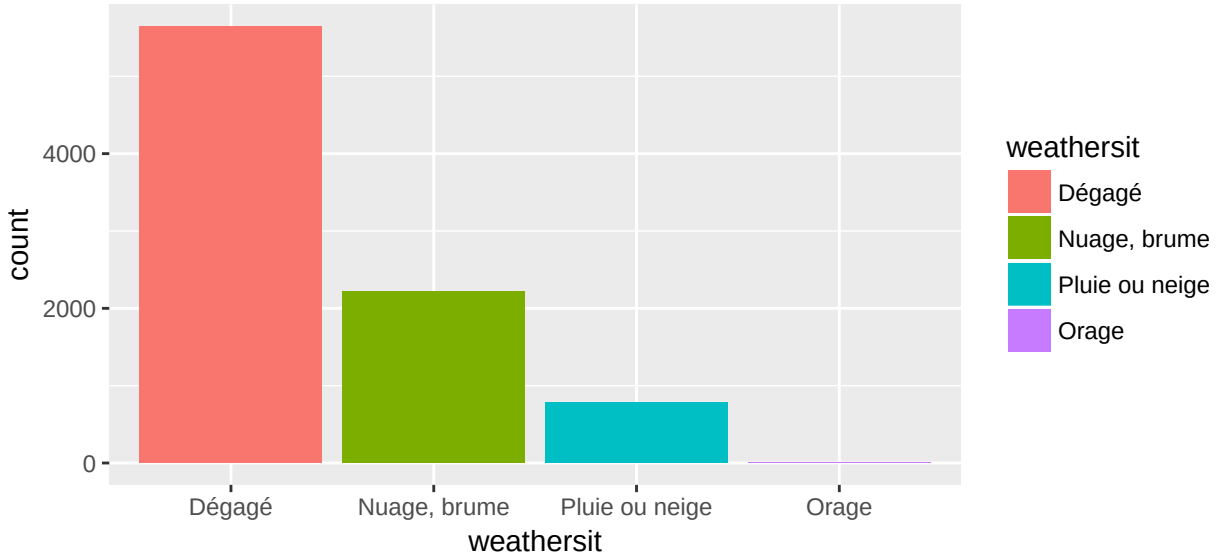
La fonction utilisée ici, `geom_bar`, fait partie des objets géométriques. Ce sont eux qui tracent le coeur du graphique, c'est-à-dire, les lignes, les barres ou les points que nous voulons représenter.

`geom_bar` va utiliser directement les arguments de `aes` qui ont été ajoutés à l'objet `ggplot`. De plus, cette fonction utilise par défaut l'argument `stat = "count"` qui **compte** le nombre d'éléments par `x` pour produire les bâtons.

Les éléments sont simplement ajoutés par sommation `+`.

```
a + geom_bar(stat = "count")
```

Nous obtenons déjà un graphique semblable au graphique de base produit en R. Cependant, un peu de peaufinage doit être fait. Les noms des axes peuvent être changés, et un titre pourrait être ajouté. De plus, dans ce cas-ci, la légende et le nom de l'axe des x ne fournissent pas d'information pertinente. Nous pourrions donc les enlever. Tous ces aspects seront modifiés par différentes fonctions couches de `ggplot`.

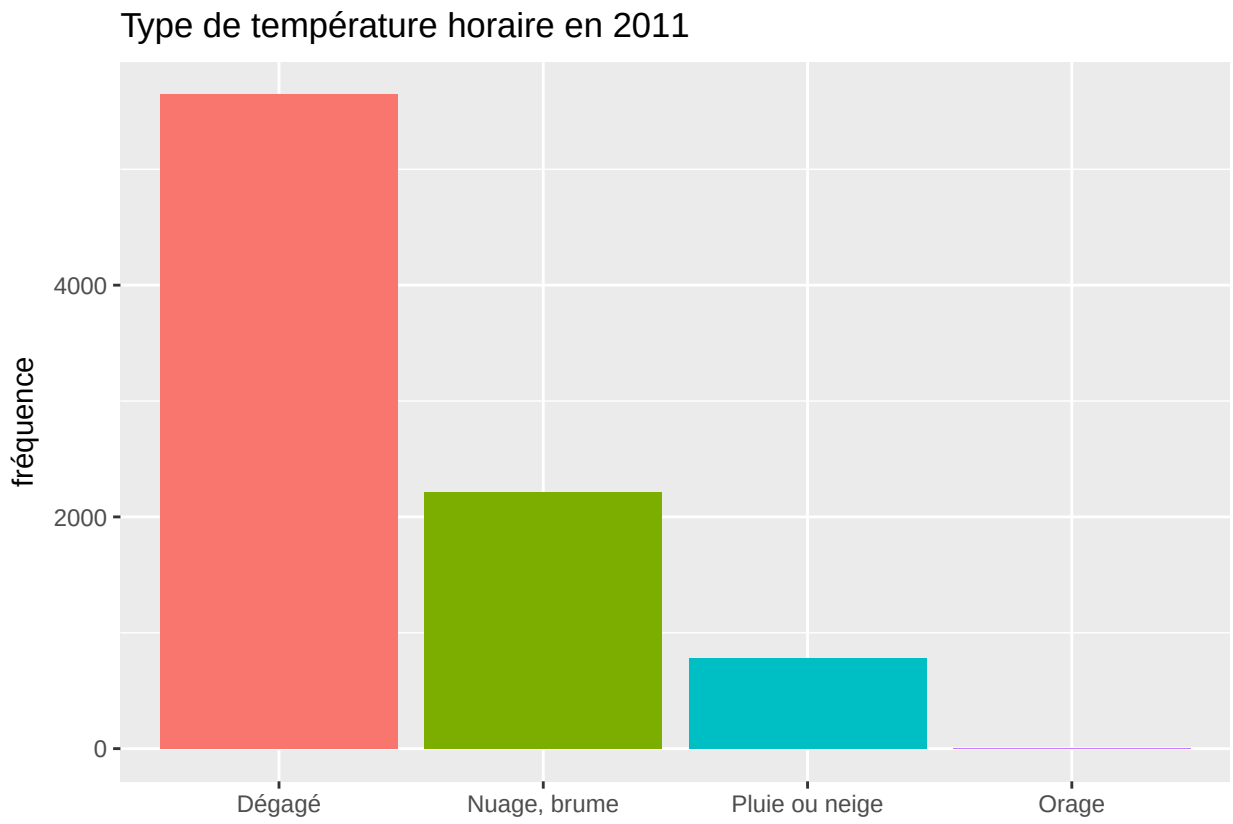


Les autres couches

Les fonctions suivantes permettront de compléter le diagramme avec l'information pertinente.

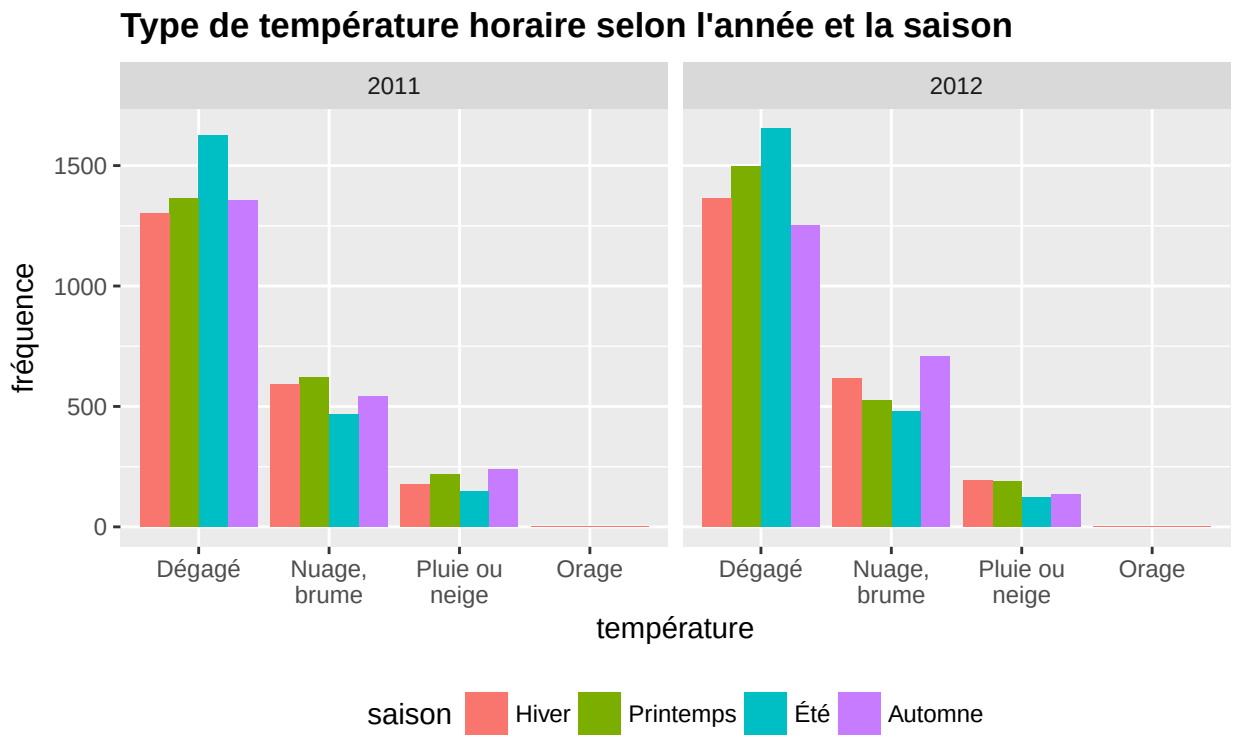
- La fonction `theme` gouverne plusieurs aspects graphiques (le **contenant**), dont la position de la légende. En quelque sorte, elle ressemble à la fonction `par` des graphiques de base.
- `xlab` et `ylab` sont des fonctions qui prennent en entrée le titre des axes respectifs.
- `ggtitle` prend, de la même façon, le titre du graphique

```
a +
  geom_bar(stat = "count") +                        # Bâtons
  theme(legend.position = "none") +                 # Enlève la légende
  xlab("") +                                        # Axe des x, non pertinent
  ylab("fréquence") +                               # Axe des y
  ggtitle("Type de température horaire en 2011")    # Titre
```



Produire un diagramme à bâtons complexe avec le package `ggplot2`

Jetons d'abord un coup d'oeil au graphique suivant, qui donne une vue d'ensemble sur la température à la location en fonction des saisons et des années.



Comparé au diagramme de base, ce graphique comporte plusieurs différences :

- Les **deux** années du jeu de données sont présentées, et chaque année a son sous-graphique
- L'axe des x est toujours divisé en température, cependant, les bâtons correspondent maintenant à la fréquence par **saison**
- Il y a maintenant une **légende**, accompagnée d'un titre approprié
- Le titre de la figure est en gras

Produire un graphique de ce type, à partir du graphique de base, va nous demander plusieurs types d'actions.

- Changer les paramètres de l'objet `ggplot` et de la fonction géométrique `geom_bar`
- Diviser le graphique en 2 panneaux
- Ajouter de l'information à la légende
- S'occuper de divers paramètres de mise en forme : positionnement de la légende, mise en forme des étiquettes de l'axe des x, police du titre.

Modifier le coeur du graphique avec un nouvel objet ggplot et des arguments pour geom_bar

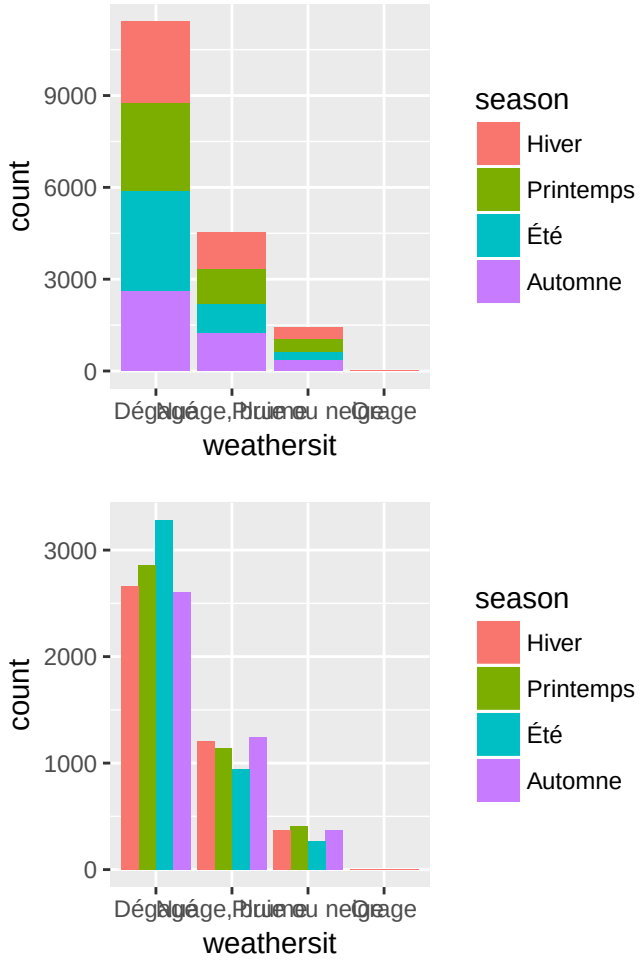
Puisque le jeu de données est différent, un nouvel objet `ggplot` doit être créé. C'est aussi l'occasion de modifier les paramètres esthétiques qui seront pris en compte par `geom_bar`.

```
b <- ggplot(hour.dataset, aes(x = weathersit, fill = season))
```

L'axe des x est toujours occupé par la température, mais ce sont les saisons qui créent les barres. C'est pourquoi la colonne `season` du jeu de données est maintenant passée à l'argument `fill`.

Comparons maintenant les exemples suivants :

```r
# Diagramme de gauche, superposé
b + geom_bar()
# Diagramme de droite, juxtaposé
b + geom_bar(position = "dodge")
```

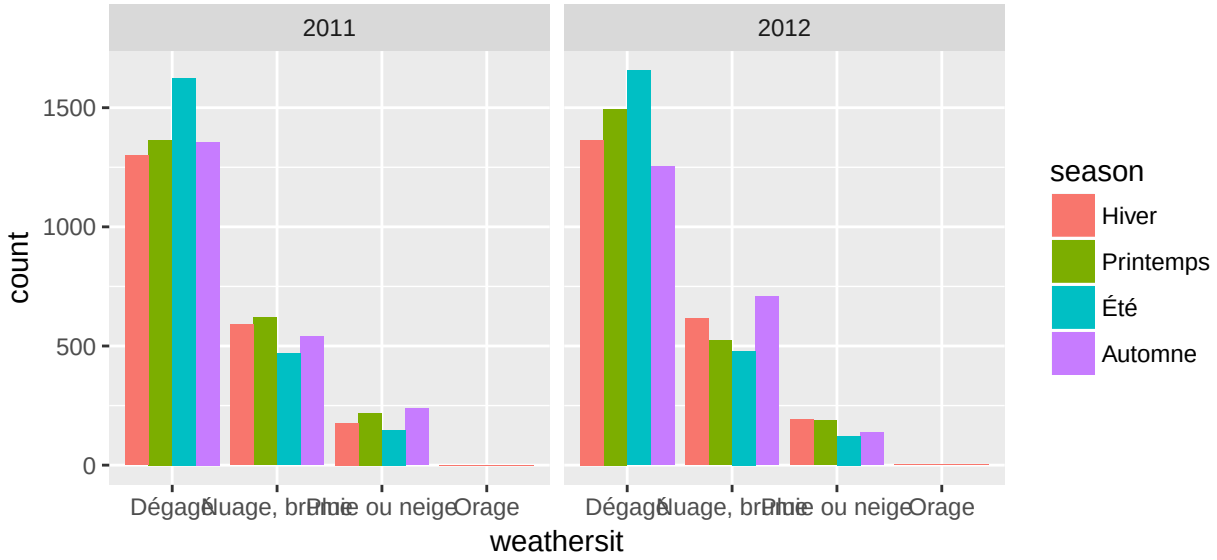


Utiliser l'argument `position` permet de changer l'apparence des barres. C'est le paramètre `"dodge"`, qui supprime l'empilement par variable, qui nous intéresse ici.

Diviser le graphique en 2 panneaux : plus simple que mfrow

Un autre type de fonction, les fonctions de "facettes" (*faceting*) permettent de séparer un graphique en panneaux directement à partir d'une formule.

```r
b + geom_bar(position = "dodge") + facet_grid(~ yr)
```



Ici, c'est la fonction facet_grid qui permet de diviser les données représentées en deux panneaux, selon l'année.

Les derniers ajouts : `guide`, `theme` et `scale_x_discrete`

On réutilise la fonction `theme` pour modifier des éléments de **contenant** :

- L'argument `legend.position` permet cette fois-ci de déplacer la légende en bas du graphique
- L'argument `plot.title` demande d'appeler une autre fonction : `element_text`, qui contrôle les différents paramètres d'un texte (police, taille, couleur, etc.). En utilisant cette combinaison, on demande d'appliquer une police en caractère **gras** au titre

La fonction `guide` se situe à l'interface entre l'esthétique, le **contenant** et l'information, le **contenu**. Tout comme `theme`, elle permet d'appeler d'autres fonctions directement dans ses arguments. Nous l'utilisons ici pour appeler la fonction `guide_legend`, qui va nous permettre de modifier le titre de la légende.

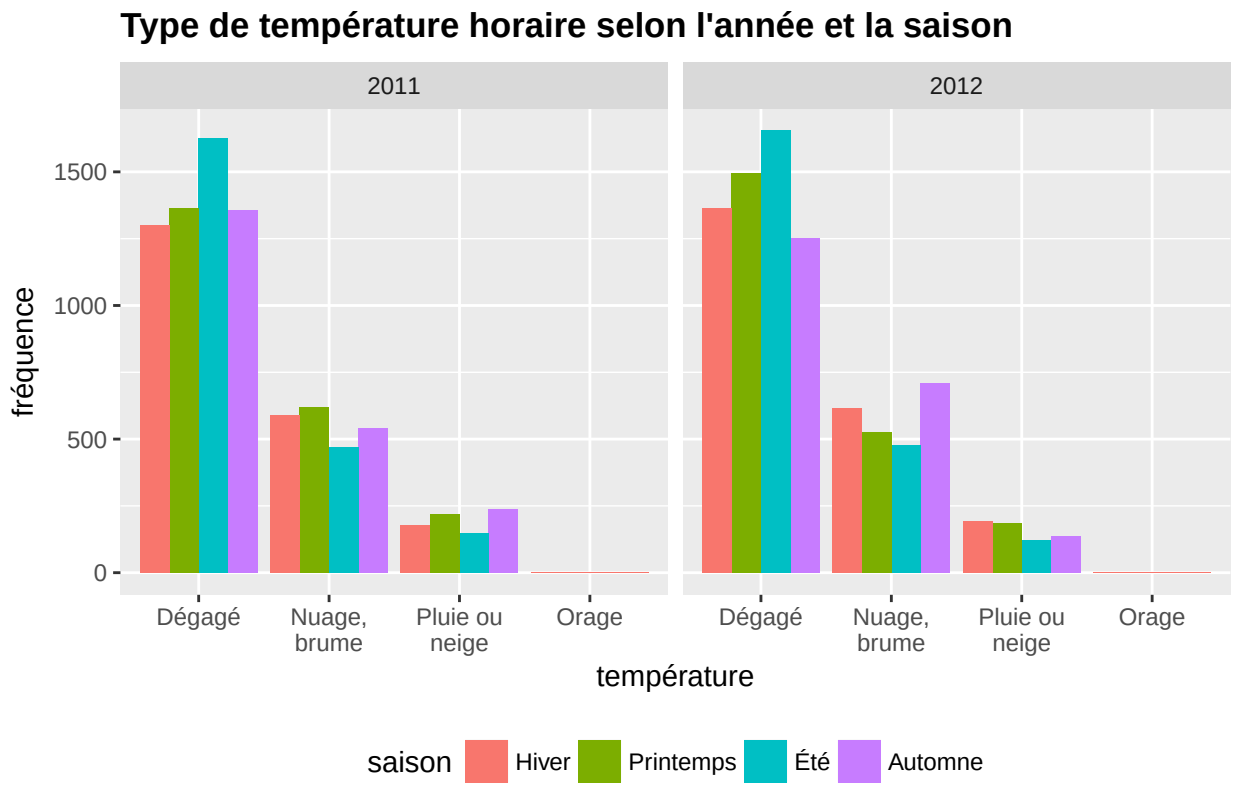
Finalement, la fonction `scale_x_discrete` permet un meilleur contrôle des paramètres de l'axe des x que la fonction `xlabs`. Dans ce cas, elle va nous permettre d'introduire des retours de chariots dans les étiquettes de l'axe des x de façon à mieux équilibrer les lignes.

Combiné aux fonctions précédemment couvertes, cela va nous permettre de produire un graphique complet et informatif.

```
b +
  geom_bar(position = "dodge") + facet_grid(~ yr) +

  guides(fill = guide_legend(title = "saison")) +
  theme(legend.position="bottom", plot.title = element_text(face = "bold")) +

  scale_x_discrete("température", labels = c("Dégagé" = "Dégagé",
                                             "Nuage, brume" = "Nuage,\nbrume",
                                             "Pluie ou neige" = "Pluie ou\nneige",
                                             "Orage" = "Orage")) +
  ylab("fréquence") +
  ggtitle("Type de température horaire selon l'année et la saison")
```



Le package `ggplot2` permet une production de graphiques complexes pour lesquels il serait impossible de couvrir tous les aspects dans ce seul exemple. Pour une aide ponctuelle et pour l'accès au répertoire des fonctions de ce package, veuillez voir la documentation en ligne de `ggplot2` à l'adresse suivante http://docs.ggplot2.org/current/index.html.

Références complètes

Références citées

[1] KOOPMANS, Lambert H. (1987). *Introduction to Contemporary Statistical Methods, Second Edition.* Duxbury Press, Boston, États-Unis, **ISBN** : 0-87150-073-6, 683 pages.

[2] RENY-NOLIN, Emmanuelle, MASSÉ, Jean-Claude, VANDAL, Nathalie, BAILLARGEON, Sophie et TALBOT, Denis (2014). *STT-1100, Statistiques Descriptives, Notions théoriques*, Département de mathématiques et de statistique, Université Laval, Québec, Canada, 91 pages.

[3] https://archive.ics.uci.edu/ml/datasets/Bike+Sharing+Dataset, consulté le 9 mars 2015.

[4] CHANG, Winston (2012). *R Graphics Cookbook.* O'Reilly Media, Sebastopol, États-Unis, **ISBN** : 978-1-449-31695-2, 416 pages.

Références supplémentaires

R et l'ensemble de sa documentation : R Core Team (2014). *R : A Language and Environment for Statistical Computing*, R Foundation for Statistical Computing, Vienna, Austria, http://www.R-project.org

la documentation de ggplot2, disponible à : WICKHAM, Hadley. http://docs.ggplot2.org/current/index.html, consulté le 16 mars 2015.

Les notes du présent cours, pour une introduction rapide à R, et une présentation de ressources avancées : BAILLARGEON, Sophie (2015). *Ressources R*. http://archimede.mat.ulaval.ca/dokuwiki/doku.php?id=r, consulté le 16 mars 2015.

pour une introduction pratique à ggplot2, avec exemple : Pedro M. (5 mars 2015). *Part 3a : Plotting with ggplot2*. http://theanalyticalminds.blogspot.ca/2015/03/part-3a-plotting-with-ggplot2.html, consulté le 14 mars 2015.