Atelier 15: les graphiques

Pascal Brissette (U. McGill)

9/25/22

Les graphiques sont l'un des plus puissants leviers d'explication des données. Il n'y a guère de meilleur moyen de parler des données et de les faire "parler" que de les donner à voir. L'utilisation des graphiques ne date pas d'hier. On peut admirer ci-dessous celui produit en 1869 par l'ingénieur Charles Joseph Mignard. Celui-ci illustre les pertes colossales subies par la Grande Armée (450 000 hommes) de Napoléon au cours de la Campagne de Russie. L'épaisseur du trait indique la quantité d'hommes, la couleur, la direction de la marche. Bien que fondé sur de solides statistiques, ce graphique donne à voir un phénomène qui ne requiert aucune connaissance des statistiques. En quelques traits, il synthétise toute une histoire.

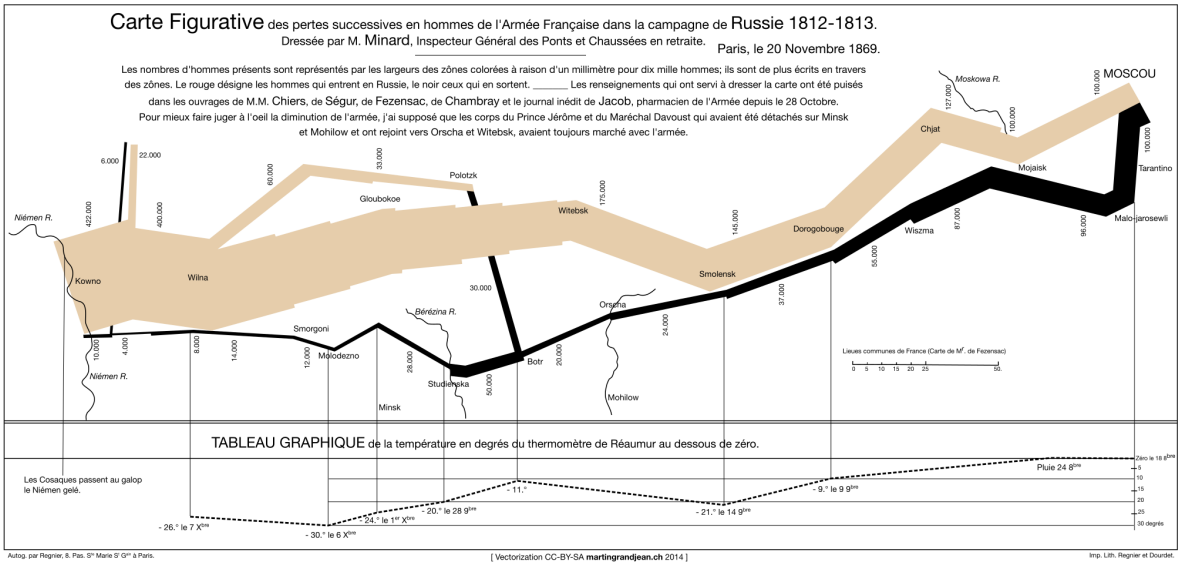


Figure 1: Charles Joseph Minard, "Carte figurative des pertes successives en hommes de l'armée française dans la campagne de Russie 1812-1813", 1869

Programme de l'atelier

Dans le présent atelier, vous aurez l'occasion de vous initier à la grammaire des graphiques et à l'extension `ggplot2`, ainsi qu'aux graphiques à barres et de dispersion. Avant d'en venir aux graphiques, il faut cependant introduire un type de données que vous n'avez pas encore croisé dans les précédents ateliers, le facteur (ou donnée catégorique).

Les facteurs

Dans R, les facteurs servent à emmagasiner des variables catégoriques, c'est-à-dire des variables dont les modalités forment un ensemble fini et dénombrable. Ce type de données est utile pour, par exemple, ordonner des valeurs nominales d'une manière qui nous convienne. Prenons un exemple simple. Si je demande à R d'ordonner les douze mois du calendrier grégorien, il ne comprendra pas ce que je veux: il prendra les chaînes de caractères que je lui donne et ordonnera la série par ordre alphabétique, en prenant la première lettre du mois comme point de comparaison. Voyons ce que cela donne:

```
mois_c <- c("février", "janvier", "novembre","mars", "octobre", "avril", "mai", "juin",
            "juillet", "août","septembre","août", "mars", "février", "décembre", "juin")

sort(mois_c)
```

```
 [1] "août"      "août"      "avril"     "décembre"  "février"   "février"
 [7] "janvier"   "juillet"   "juin"      "juin"      "mai"       "mars"
[13] "mars"      "novembre"  "octobre"   "septembre"
```

Les mois d'août et d'avril arrivent en tête de vecteur, suivis par décembre. C'est ici que la fonction `factor()`, qui transforme une donnée en donnée catégorique, peut être utile. Avec l'argument `levels =`, nous pourrons imposer un ordre. Et comme les mois sont en nombre finis, on ne risque pas de croiser dans nos données des modalités autres que les douze que nous allons définir.

```
# On peut créer un vecteur qui indiquera à la fonction `factor()` quel ordre je souhaite d
ordre_mois <- c("janvier", "février", "mars", "avril", "mai", "juin", "juillet", "août",
                "septembre", "octobre", "novembre", "décembre")

# On transforme ainsi le vecteur mois_c, de type `character`, en vecteur catégorique (fact
mois_f <- factor(x = mois_c,
       levels = ordre_mois)

# On peut vérifier le type de données
```

```r
class(mois_f)
```

```
[1] "factor"
```

```r
# On peut trier les valeurs selon l'ordre fourni à l'argument `levels=`
sort(mois_f)
```

```
 [1] janvier   février   février   mars      mars      avril     mai
 [8] juin      juin      juillet   août      août      septembre octobre
[15] novembre  décembre
12 Levels: janvier février mars avril mai juin juillet août ... décembre
```

```r
# Ou selon l'ordre inversé
sort(mois_f, decreasing = TRUE)
```

```
 [1] décembre  novembre  octobre   septembre août      août      juillet
 [8] juin      juin      mai       avril     mars      mars      février
[15] février   janvier
12 Levels: janvier février mars avril mai juin juillet août ... décembre
```

Un autre intérêt des facteurs est qu'ils peuvent aider à repérer d'eventuelles erreurs typographiques dans les valeurs. Supposons qu'il y ait une telle erreur dans mon vecteur initial. Je vais ordonner ce vecteur de charactères selon un ordre défini dans l'argument `levels=` de la fonction `factor()`, comme je l'ai fait ci-dessus. Voyez le résultat lorsque j'appelle le vecteur catégorique:

```r
# Création d'un vecteur comprenant 16 éléments (répétition de certains noms de mois, dont
mois_c <- c("février", "janvier", "movembre","mars", "octobre", "avril", "mai", "juin",
            "juillet", "août","septembre","août", "mars", "février", "décembre", "juin")

# Le vecteur de caractères est transformé en facteur. L'ordre des mois est précisé avec le
mois_f <- factor(x = mois_c,
       levels = c("janvier", "février", "mars", "avril", "mai", "juin", "juillet", "août",
                  "septembre", "octobre", "novembre", "décembre"))

mois_f
```

```
 [1] février   janvier   <NA>      mars      octobre   avril     mai
 [8] juin      juillet   août      septembre août      mars      février
[15] décembre  juin
12 Levels: janvier février mars avril mai juin juillet août ... décembre
```

Le mois qui a été mal orthographié dans le vecteur des valeurs ne correspond pas à l'une des catégories indiquées dans l'argument `levels=`. L'appel du vecteur catégorique renvoit donc `<NA>` à sa position.

Si vous demandez à R de vous fournir la structure du vecteur catégorique, vous verrez une série de chiffres qui pourrait attirer votre attention:

```
str(mois_f)
```

```
 Factor w/ 12 levels "janvier","février",..: 2 1 NA 3 10 4 5 6 7 8 ...
```

La fonction `str()` confirme d'abord que le vecteur est de type catégorique (`factor`) et qu'il comporte 12 catégories (`levels`). Les deux premiers niveaux vous sont indiqués ("janvier", "février"), puis suit un vecteur numérique. À quoi cela correspond-il?

C'est une particularité de la variable catégorique de se présenter sous la forme de caractères, mais d'être emmagasinée dans R comme une suite de nombres entiers. La variable catégorique est une variable discrète et comme il n'y a pas d'intervalle entre les catégories, R peut associer à chacune, selon l'ordre que l'on a indiqué dans `levels=`, un nombre entier positif. De sorte que si on demande à R à quoi correspondent les données de notre vecteur avec `typeof()`, il nous dira qu'il s'agit d'*integer*:

```
class(mois_f) # la structure est de type factor
```

```
[1] "factor"
```

```
typeof(mois_f) # les données elles-mêmes sont de type numérique (integer)
```

```
[1] "integer"
```

La grammaire des graphiques (GG)

L'extension de base `graphics`, activée dès que l'on ouvre RStudio, permet de créer rapidement des graphiques avec la fonction `plot()`. Les utilisateurs de R utilisent plutôt `ggplot2`, qui permet de créer des graphiques de plus grande qualité et de mieux contrôler les paramètres de chaque élément. Cette extension implémente dans R la grammaire des graphiques proposée par Wilkinson en 2005.

Les graphiques sont composés de plusieurs éléments: des mesures, des formes, des titres, des ensembles de couleurs, des coordonnées, des axes et, bien entendu, des données. Il s'agit donc d'objets complexes que la grammaire des graphiques permet de décomposer et de régler séparément. Outre l'ouvrage de Wilkinson, vous lirez avec profit l'article de Hadley Wickham, auteur des extensions ggplot et ggplot2, ainsi que son ouvrage, *ggplot2: Elegant Graphics for Data Analysis*.

Dans la grammaire des graphiques, une couche est composée des éléments suivants:

1. Des **données**;
2. Des composantes **esthétiques**;
3. Une opération **statistique**;
4. Un **objet géométrique** (points, lignes, rectanges, cercles, carte, etc.);
5. Des **ajustements** pour permettre, par exemple, la superposition de points sans nuire à la lisibilité du graphique.

Le graphique est le résultat de la superposition de ces couches. Certaines sont facultatives ou sont pourvues de valeurs par défaut, d'autres sont nécessaires et doivent être précisées par l'utilisateur:

Élément	Fonction	Explications
Données	`ggplot()`	Fonction d'initialisation du graphique. On y insère généralement le **tableau de données** dont les variables serviront à définir les éléments esthétiques. Ces éléments esthétiques, `aes()`, peuvent être insérés dans cette fonction s'ils sont identiques pour toutes les couches du graphique.

Élément	Fonction	Explications
Éléments esthétiques	`aes()`	Éléments esthétiques, précisés comme argument `mapping=` de la fonction d'initialisation s'ils sont identitques pour toutes les couches du graphique, ou à l'intérieur des fonctions commençant par `geom_` s'ils sont différents. Parmi les éléments esthétiques, on trouve: • `x` et `y` ==> variables qui définissent respectivement les axes x et y du graphique; • `fill` ==> variable qui définit la couleur de remplissage des formes géométriques; • `colour` ==> variable qui définit la couleur des contours des formes géométriques; • `size` ==> variable qui définit la taille des points ou des lignes; • `alpha` ==> le degré de transparence des formes géométriques (entre 0 et 1); • `shape` ==> variable qui définit des formes géométriques en complément des points, dans un graphique à points.
Éléments géométriques	`geom_point()`	diagramme de dispersion (à points)
	`geom_bar()`	diagramme à barres. Prend une variable catégorique en x. Par défaut, compte le nombre de valeurs par catégorie (x).
	`geom_col()`	diagramme à barres. Prend une variable catégorique en x et une variable numérique en y. Équivalent de geom_bar(stat="identity") avec définition d'une variable continue y.
	`geom_histogram()`	diagramme à barres. Prend des variables continues en x et en y. Par défaut, bins=30.
Facettes	`facet_wrap()` ou `facet_grid()`	distribue les modalités d'une variable catégorique en plusieurs graphiques de formats réduits.
Statistiques	`stat_identity()`, `stat_summary()`, etc.	Précise les opérations statistiques faites sur les données avant de les afficher dans le graphique.
Coordonnées	`coord_cartesian()`	Permet de fixer des limites aux axes x et y, ce qui a pour effet d'aggrandir une portions du graqphique.
	`coord_map()`	Projette une portion de la géographie terrestre sur une carte en 2 dimensions.

Élément	Fonction	Explications
Thèmes	theme_grey(), theme_light(), theme_dard(), etc.	Série de fonctions permettant de préciser les éléments esthétiques du graphique qui ne concernent pas les données (couleur et opacité du fond, police de caractères, etc.)

Les quelques éléments indiqués dans le tableau ci-dessus donnent une faible idée de la richesse de l'extension `ggplot2`, dont le développement est assuré par une équipe de programmeurs. À peu près toutes les composantes d'un graphique peuvent être contrôlés, pour peu qu'on ait la patience de lire la documentation, abondante, ou de chercher de l'aide en ligne (ex: Stack Overflow).

On notera également que des extensions prennent appui sur `ggplot2` pour projeter l'art du graphique à un tout autre niveau. Par exemple, l'extension `plotly` permet d'interagir avec le graphique à l'aide d'un menu directement accessible avec la souris. D'autres extensions améliorent le rendu de graphiques spéciaux (cartes thermiques, cartes géographiques, réseaux, etc.).

Le diagramme à barres

Revenons aux bases et voyons comment se combinent concrètement les éléments du graphique. Nous allons utiliser un jeu de données construit à partir du roman *Maria Chapdelaine*, de Louis Hémon. Ce roman a été récupéré depuis le Projet Gutenberg. Le péritexte de ce roman a été supprimé et n'ont été conservés que les titres de chapitre ("CHAPITRE I", "CHAPITRE II", etc.) et les mots formant le texte. Ces mots ont été mis en minuscules, puis séparés les uns des autres à partir des espaces et des ponctuations (le motif utilisé pour séparer les chaînes est l'expression régulière: "\W"). On a d'abord calculé la fréquence brute de chaque mot et on s'est intéressé aux prénoms des trois prétendants de Maria: François Paradis, Lorenzo Surprenant et Eutrope Gagnon. Dans premier tableau ci-dessous, très simple, il n'y a que deux variables et trois observations. La variable `nom` est discrète et contient les prénoms des trois prétendants. La variable `freq_brute` est également discrète, puisqu'elle contient des nombres entiers, mais elle sera traitée dans les graphiques comme continue. Pour une explication sur les types de variables (discrète, continue, etc.), voir Statistiques Canada.

```
# Installation et activation des librairies requises
if(!"data.table" %in% rownames(installed.packages())) {install.packages("data.table")}
if(!"ggplot2" %in% rownames(installed.packages())) {install.packages("ggplot2")}
library(ggplot2)
library(data.table)
```

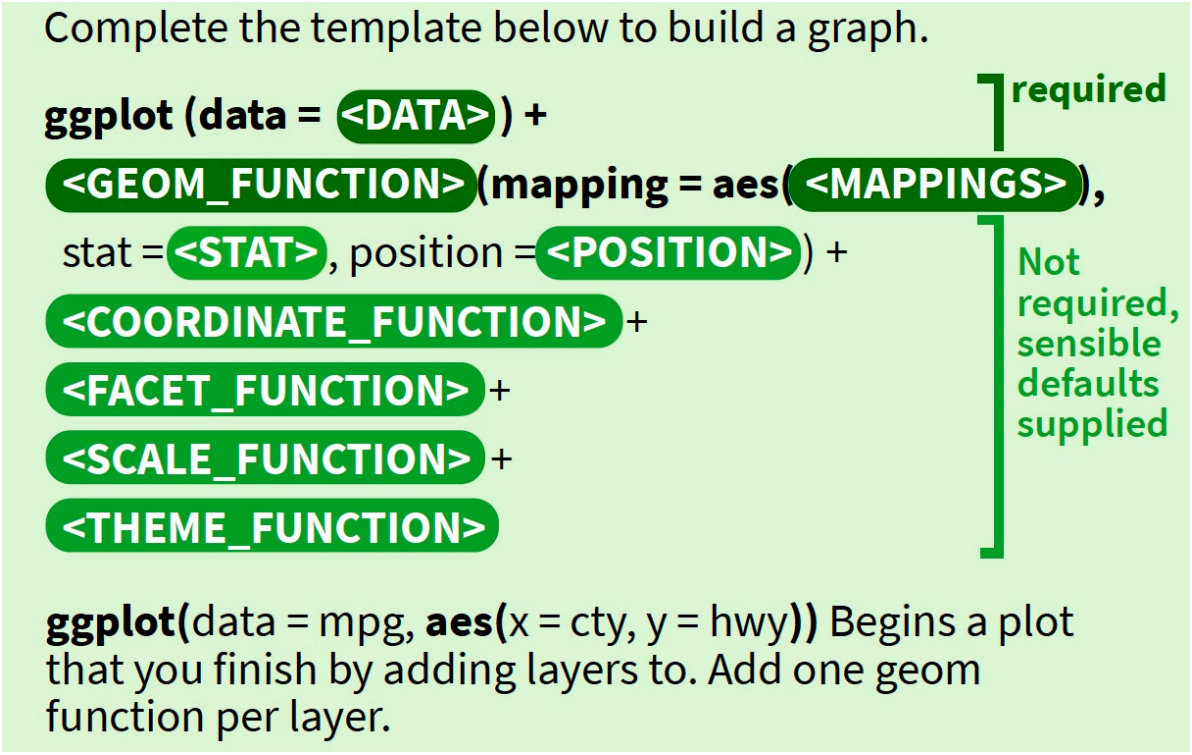
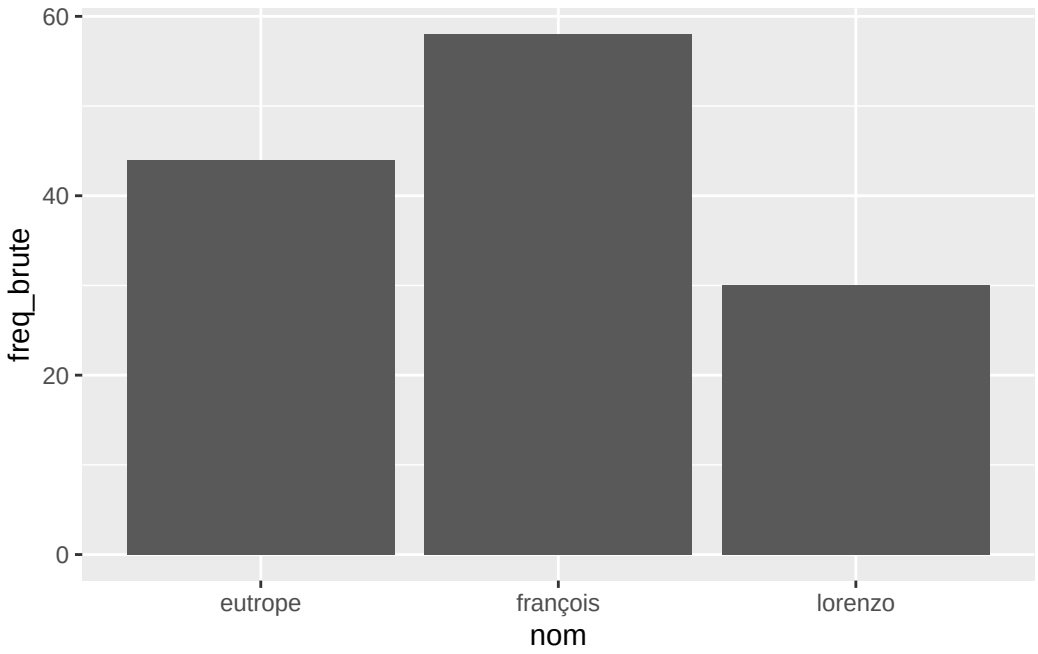


Figure 2: Capture de l'antisèche ggplot2, RStudio

```
# Importation de la structure de données
pretendants_freq_brute <- fread("../donnees/maria_freq_brute_pretendants.csv")
```

Un simple graphique à barres nous permettra de visualiser ce décompte brut des occurrences des trois prénoms:

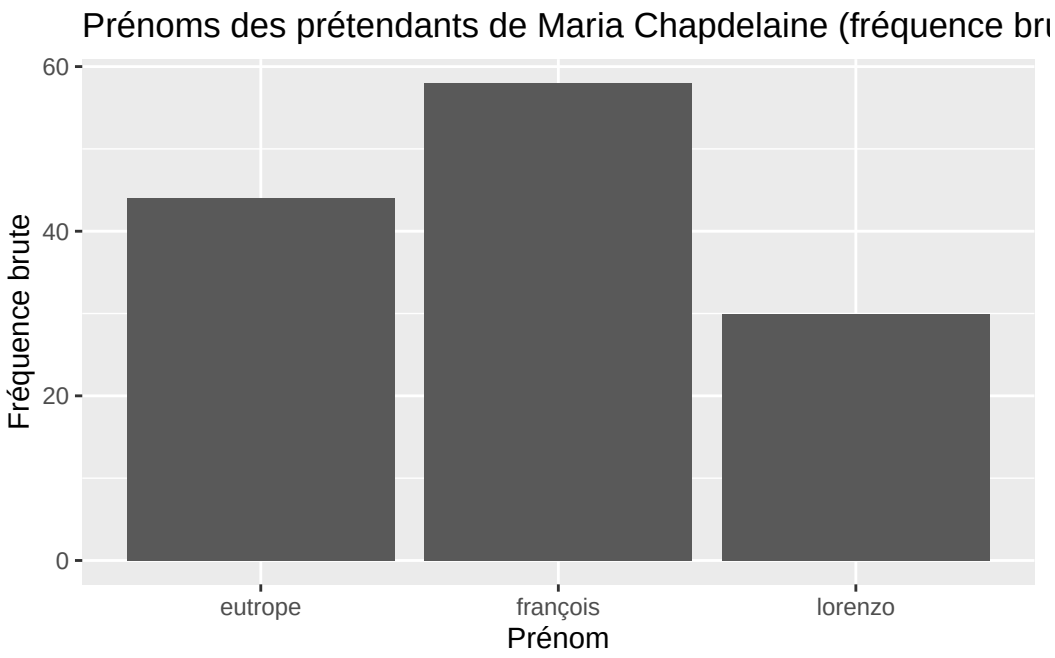
```
# On crée une première couche contenant les données et leur projection en composantes esth
p <- ggplot(pretendants_freq_brute, aes(x=nom, y=freq_brute))

# On ajoute à cette première couche (avec l'opérateur + ) les formes géométriques.
p2 <- p + geom_bar(stat = "identity")

p2
```



Modifions les titres d'axes et ajoutons un titre général au diagramme:

```
p3 <- p2 +
  ggtitle("Prénoms des prétendants de Maria Chapdelaine (fréquence brute)")+
  xlab("Prénom") +
  ylab("Fréquence brute")
```

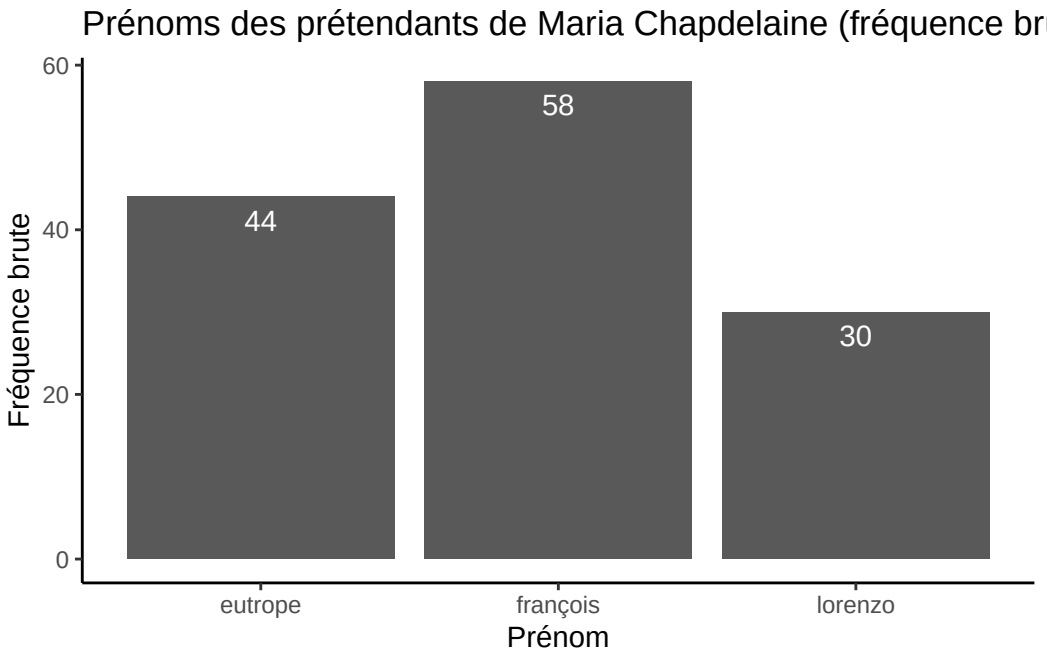
```
p3
```



On notera l'utilisation de l'opérateur `+` pour l'ajout de couches.

La hauteur de chaque colonne est déterminée par la fréquence (valeur numérique) associée à chaque prénom. On peut indiquer ces valeurs directement sur les colonnes respectives. Profitons-en pour alléger modifier le thème en utilisant l'une des fonctions commençant par `theme_` :

```
p3 +
  geom_text(aes(label=freq_brute), vjust=1.6, color="white") +
  theme_classic()
```



Le diagramme à barres est très efficace pour représenter des jeux de données où les modalités des variables discrètes sont en nombre relativement limité. Lorsque ces modalités sont très nombreuses, le graphique peut devenir confus.

Voyons cela avec un autre jeu de données.

Celui que nous allons importer maintenant a été constitué à partir du même roman, *Maria Chapdelaine*. Les mêmes opérations de base ont été faites (bas de casse, séparation des mots, création d'une table lexicale), mais on a calculé la fréquence des prénoms pour chacun des 16 chapitres du roman. Le tableau contient ainsi trois colonnes:

- Numéro du chapitre (variable discrète);
- Le prénom du prétendant de Maria (variable discrète);
- La fréquence des mentions de chaque prénom (variable continue) dans chaque chapitre.

La variable `chapitre` a été transformée en facteur, de manière à pouvoir définir l'ordre des chapitres.

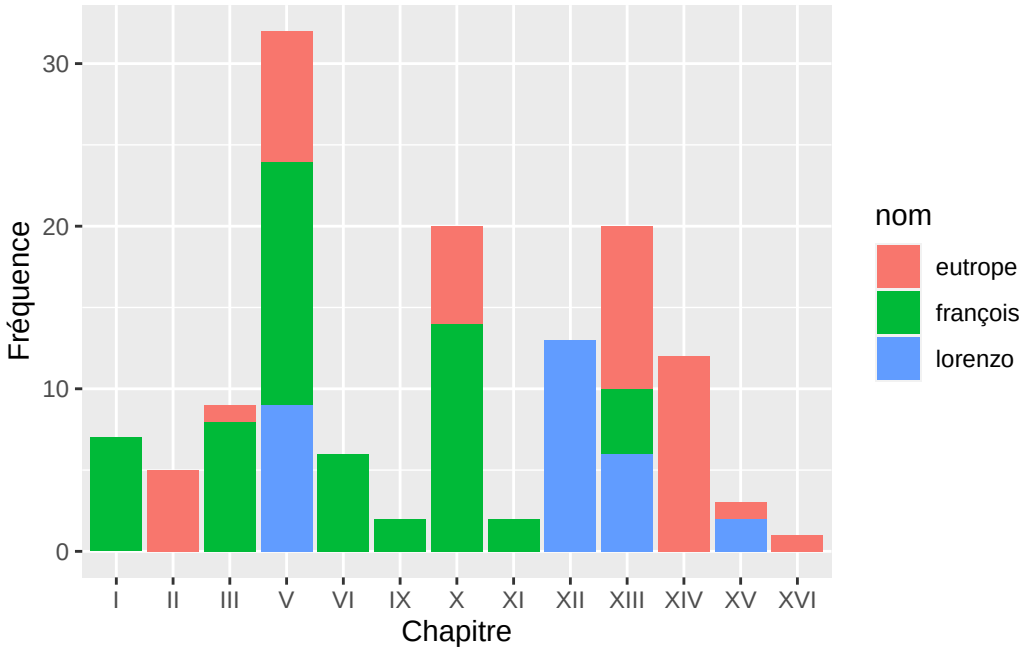
```
# Importation de la table
pretendants_freq_chap <- fread("../donnees/maria_freq_pretendants_chapitre.csv")

# On raccourci les noms donnés aux chapitres pour alléger les titres d'axes des graphiques
```

```r
library(stringr)
pretendants_freq_chap[, chapitre:=str_extract(chapitre, pattern = "(?<=\\s)[IVX]+$")]

# On transforme les titres de chapitres en données catégoriques (facteurs) et on ordonne l
pretendants_freq_chap[, chapitre:=factor(chapitre, levels = c("I", "II", "III", "V", "VI",

pretendants_freq_chap
```

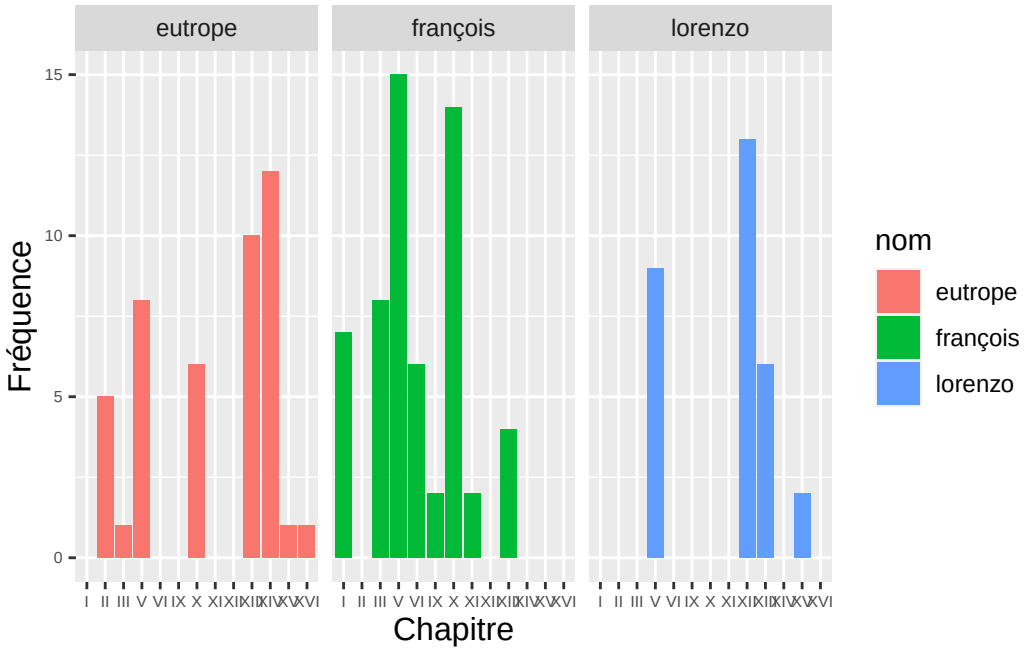
```
    chapitre      nom freq
 1:        I françois    7
 2:      III françois    8
 3:       IX françois    2
 4:        V françois   15
 5:       VI françois    6
 6:        X françois   14
 7:       XI françois    2
 8:     XIII françois    4
 9:       II  eutrope    5
10:      III  eutrope    1
11:        V  eutrope    8
12:        X  eutrope    6
13:     XIII  eutrope   10
14:      XIV  eutrope   12
15:       XV  eutrope    1
16:      XVI  eutrope    1
17:        V  lorenzo    9
18:      XII  lorenzo   13
19:     XIII  lorenzo    6
20:       XV  lorenzo    2
```

On ne peut séparer la valeur numérique (`freq`) des deux autres variables dont elle est la mesure, cela n'aurait pas de sens. La valeur de la première ligne, par exemple, fournit la fréquence du mot "françois" dans le segment du roman composé du chapitre I. Les trois variables sont solidaires. Pour transposer ces trois variables dans un graphique, on peut soit utiliser un élément esthétique supplémentaire (une couleur pour le remplissage des formes, par exemple) ou décliner en plusieurs facettes les observations du tableau en fonction de l'une des variables catégoriques. Dans le premier cas, toutes les observations du tableau seront insérées dans le même graphique; dans le deuxième cas, les observations seront distribuées en autant de petits graphiques que la variable catégorique possède de modalités:

```r
# La première couche constituée des données et des éléments esthétiques sera la même pour
p <- ggplot(pretendants_freq_chap, aes(x=chapitre, y=freq, fill=nom))

# Option 1: le diagramme utilise les couleurs pour indiquer la distribution des valeurs se
p + geom_bar(stat = "identity", position = "stack")+
  xlab("Chapitre")+
  ylab("Fréquence")
```



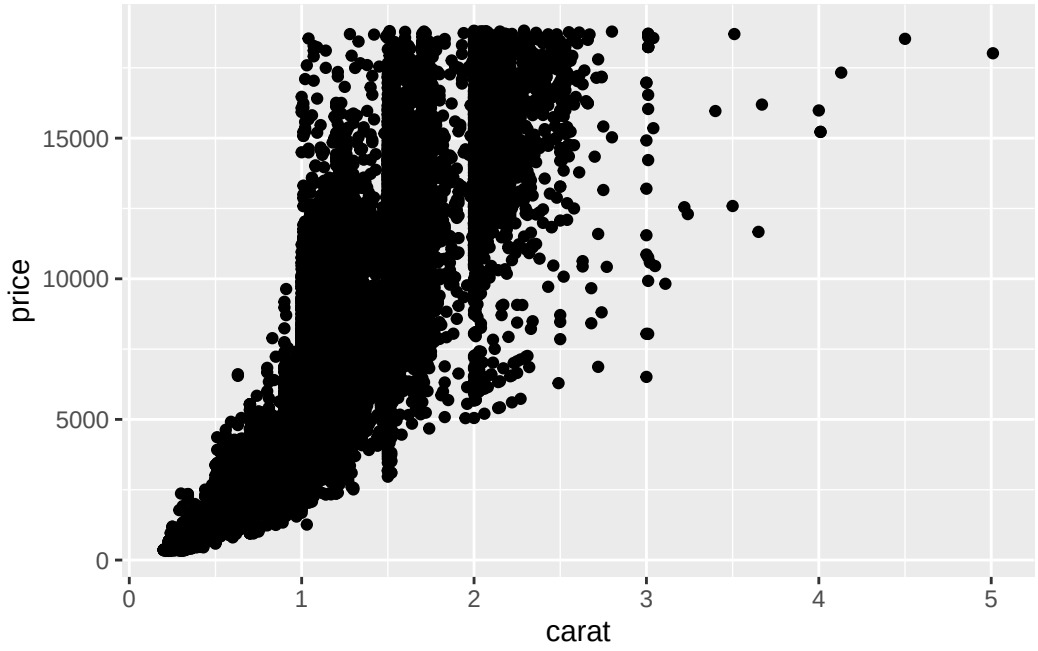

```r
# Option 2: le diagramme distribue les données dans trois facettes
p + geom_bar(stat = "identity") +
  facet_wrap(~ nom) +
  theme(axis.text=element_text(size=6),     #On diminue la taille des titres d'axes pour é
        axis.title=element_text(size=12))+  #les chevauchements
  xlab("Chapitre")+
  ylab("Fréquence")
```



Le diagramme de dispersion

Le diagramme de dispersion, ou nuage de points, transpose chaque donnée d'une distribution en un point. Il est souvent utilisé pour vérifier la corrélation, positive ou négative, entre deux variables (généralement continues) projetées sur l'axe des `x` et des `y`. Prenons le jeu de données `diamonds` proposé par l'extension `ggplot2`. Celui-ci contient une multitude d'informations sur 53 940 diamants. Si on voulait vérifier avec un diagramme de dispersion la corrélation entre les variables `carats` et `price`, deux variables continues, on donnerait à R les instructions suivantes:

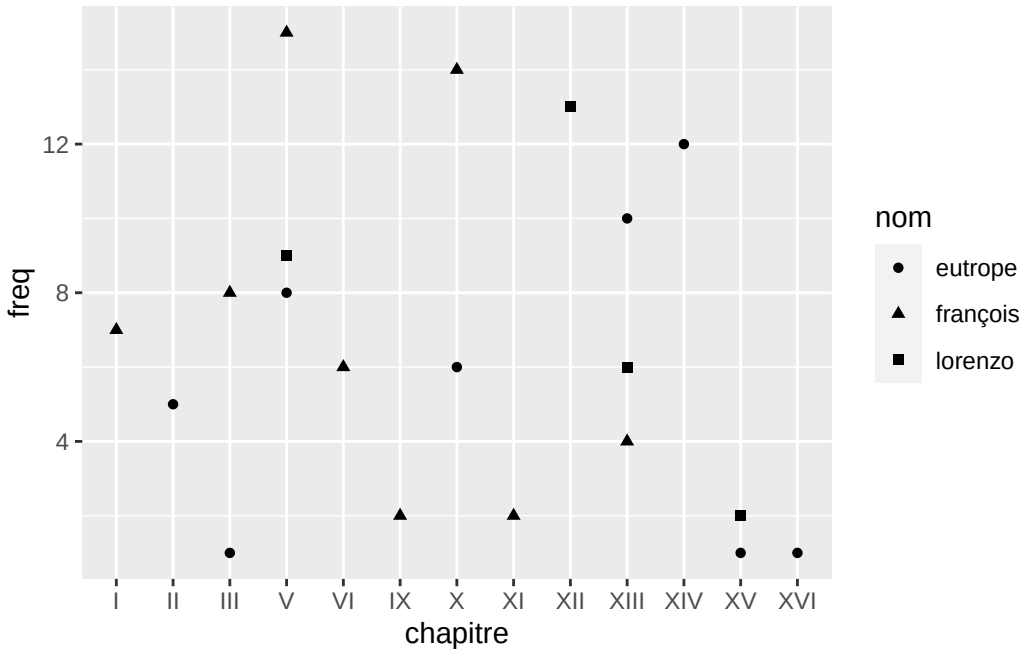
```
ggplot(diamonds, aes(x=carat, y=price))+
  geom_point()
```



Chaque point de ce graphique représente un diamant défini par sa qualité, exprimée en carats, et son prix, exprimé en dollars.

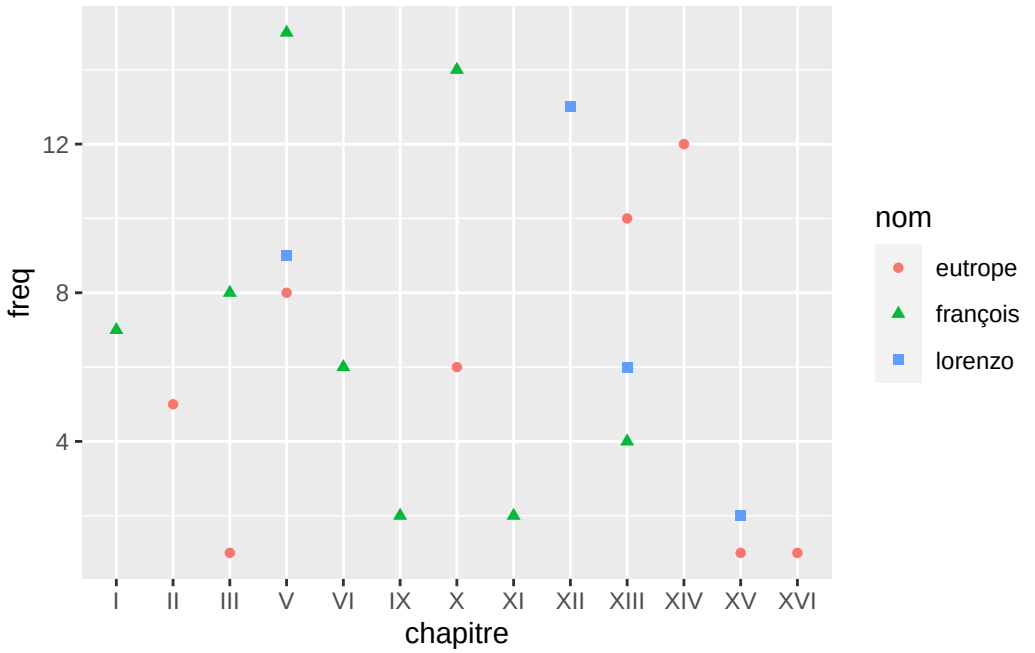
Reprenons maintenant le jeu de données créé à partir du roman *Maria Chapdelaine*. Nous allons utiliser le diagramme de dispersion pour simplement observer, comme on l'a fait avec le diagramme à barres, les mentions de prénoms des prétendants de Maria. Nous avons trois variables à projeter sur la surface en deux dimensions du graphique: les chapitres, les et la fréquence de leur mention. Comme nous n'avons que deux axes (`x` et `y`), nous devrons utiliser un troisième élément esthétique pour représenter l'une des trois variables. En `x`, on mettra la variable indépendante, `chapitre`, en y, `freq`, et on donnera à chaque point du graphique une forme correspondant à la troisième variable, `nom`. Il n'y a que trois noms, donc trois formes distinctes. On utilise, dans les esthétiques, l'argument `shape=` pour indiquer la variable qui doit servir à créer les formes. `ggplot` créera automatiquement une légende qu'il placera, par défaut, à droite du diagramme.

```
p <- ggplot(pretendants_freq_chap, aes(x=chapitre, y=freq, shape = nom))

p + geom_point()
```



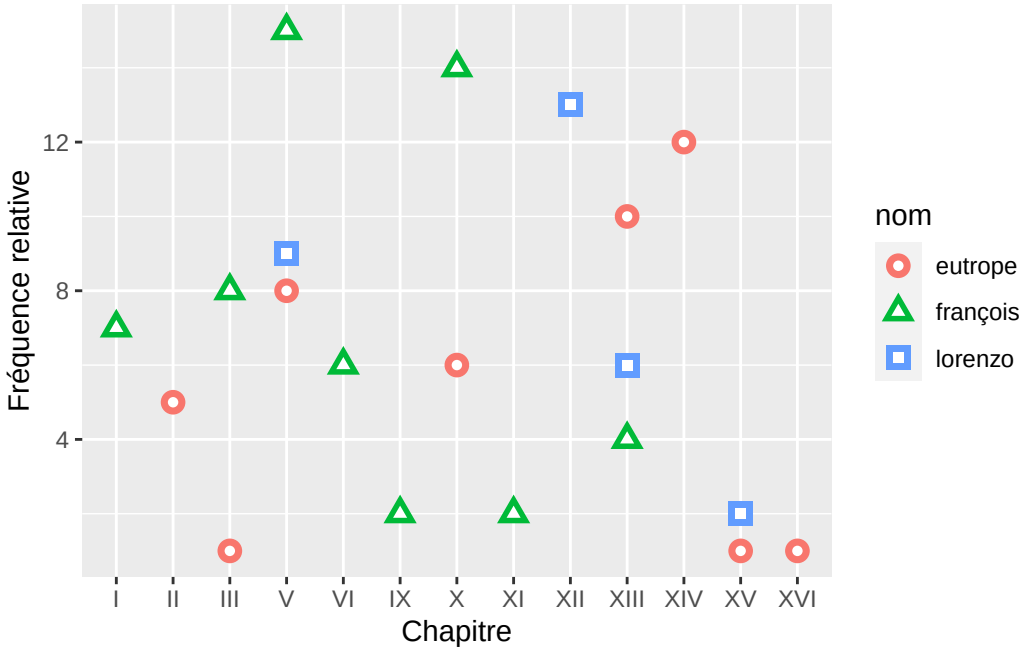
Pour accentuer le contraste entre les points, on pourrait attribuer une couleur unique aux formes. Cela se fait aisément en utilisant l'argument `colour=` (ou `color=`) dans les esthétiques:

```
p <- ggplot(pretendants_freq_chap, aes(x=chapitre, y=freq, shape = nom, colour=nom))

p + geom_point()
```



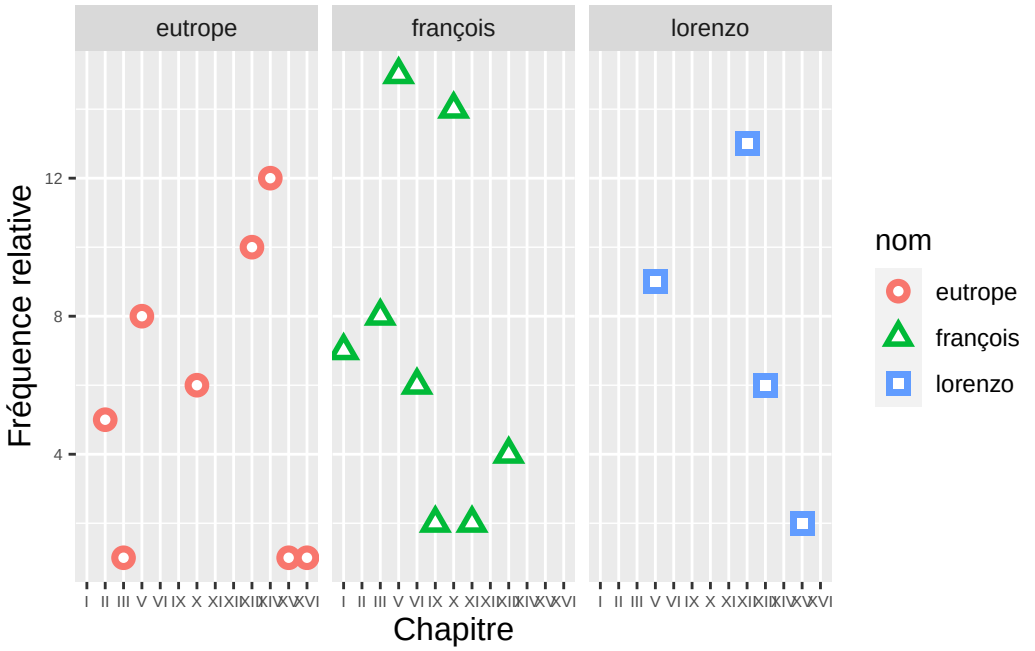
Puisque les formes et les couleurs s'appliquent directement aux points, créés avec la fonction `geom_point()`, on pourrait déplacer les précisions esthétiques dans la parenthèse de cette fonction sans modifier le diagramme. On a l'habitude de définir les éléments esthétiques qui s'appliqueront à chacun des éléments géométriques dans l'instruction initiale introduite par `ggplot()`, et à indiquer dans les arguments des fonctions `geom_***()` ceux qui s'appliquent uniquement à l'élément géométrique défini par la fonction. Par exemple, on pourrait superposer des points (formes) de différentes tailles de manière à faciliter leur repérage dans le diagramme. La forme la plus grande sera d'une couleur donnée, déterminée par la modalité spécifique de `nom`, et la plus petite sera définie par une couleur unique, le blanc. On aura ainsi deux appels de la fonction `geom_point()` qui définiront, chacune, la couleur et la taille des points:

```
p <- ggplot(pretendants_freq_chap, aes(x=chapitre, y=freq, shape = nom))

p + geom_point(aes(colour = nom), size = 4) +
  geom_point(colour = "white", size = 1.5) +
  xlab("Chapitre") +
  ylab("Fréquence relative")
```



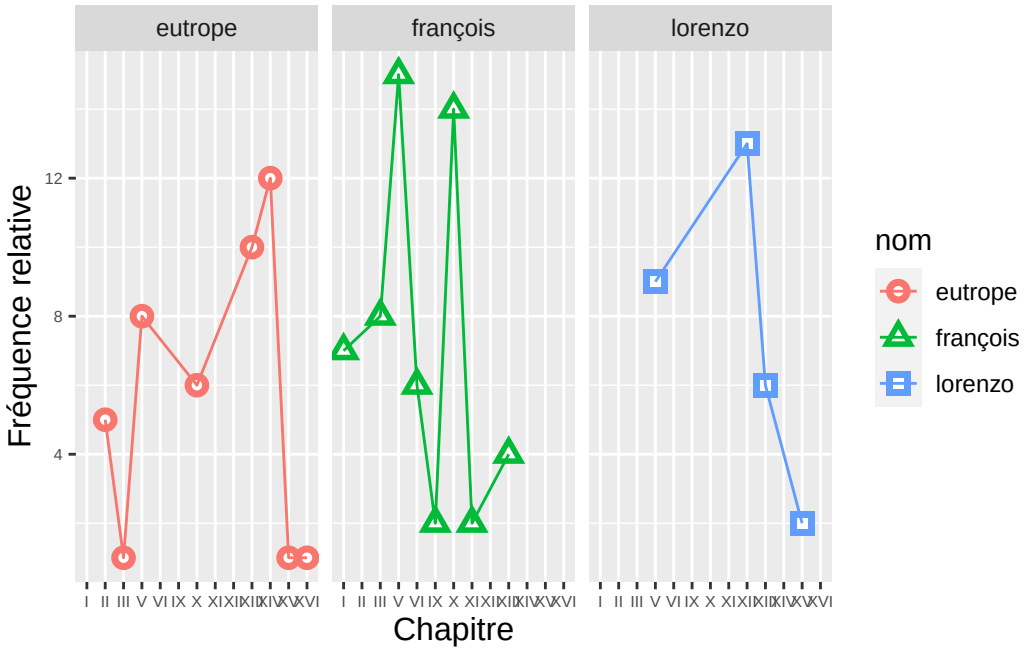
Le diagramme gagne en lisibilité, mais il est encore difficile d'en tirer une information pertinente, mis à part que "françois" atteint des sommets en fait de mention, ce qui traduit l'importance qu'il prend dans les chapitres centraux du roman. Comme ici, il est parfois souhaitable de subdiviser le diagramme en autant de facettes qu'il y a de modalités dans la variable catégorique d'intérêt. On peut faire une telle chose en utilisant la fonction `facet()` et en lui donnant, comme argument, la variable qui doit déterminer, par le nombre de ses modalités, le nombre de diagrammes à créer. On utilise l'opérateur ~ pour indiquer cette variable déterminante.

```
# Ajout d'une couche geom_line()
p <- ggplot(pretendants_freq_chap, aes(x=chapitre, y=freq, shape = nom))

p + geom_point(aes(colour = nom), size = 4)+
  geom_point(colour = "white", size = 1.5)+
  facet_wrap(~nom)+
  theme(axis.text=element_text(size=6),  # La fonction theme() permet de réduire la taille
        axis.title=element_text(size=12))+ # des caractères d'axes
  xlab("Chapitre")+
  ylab("Fréquence relative")
```



De mieux en mieux, non? On pourrait ajouter une autre couche géométrique au graphique, soit une ligne qui relie chacun des points de chaque graphique. Cette ligne sera noire si on ne définit aucune couleur, ou prendra la couleur des modalités de nom si on le précise dans ses éléments esthétiques:

```
# Division du graphique en facettes

p + geom_point(aes(colour = nom), size = 4)+
  geom_point(colour = "white", size = 1.5)+
  geom_line(aes(group = nom, colour = nom))+
  facet_wrap(~nom)+
  theme(axis.text=element_text(size=6),
        axis.title=element_text(size=12))+
  xlab("Chapitre")+
  ylab("Fréquence relative")
```



Trois facettes, trois prétendants. Pour aller plus loin, il faudrait travailler les données en amont, vérifier que les personnages ne sont pas mentionnés par des surnoms ou leur patronyme, se demander si on doit également prendre en compte leur évocation ou ce qu'ils représentent (*l'amour* pour François, *la vie facile* pour Lorenzo, *le devoir et la famille* pour Eutrope). En l'état, le graphique traduit tout de même des faits narratifs: François est celui dont le nom est le plus souvent convoqué, ce qui traduit la focalisation dont il fait l'objet. Il apparaît tôt dans le roman, mais disparaît dans les bois et s'efface donc comme possible prétendant. Ne restent plus que Lorenzo et Eutrope. Ce dernier n'est jamais le "gagnant" en fait de mentions, mais il est celui qui reste là jusqu'à la fin et qui aura la main de Maria. Quant au chatoyant Lorenzo, il suscite un temps un grand intérêt, mais son étoile palit à la fin du roman, jusqu'à disparaître.

Défi

1. Videz l'environnement de travail avec l'instruction `rm(list=ls())` , puis importez le jeu de données ouvrages dans son format .RDS ou .csv;
2. Créez un diagramme à barres montrant la distribution des ouvrages signés par les femmes et les hommes. Par défaut, la fonction `geom_bar()` a un argument `stat= "count"`. Vous n'avez donc aucun argument à fournir à cette fonction;
3. Créez un nouveau diagramme à barres montrant la distribution des romans selon leur genre littéraire. Même si cela n'est pas nécessaire ici, vous pouvez transformer, avant de créer le diagramme, la variable `genre.litteraire` en variable catégorique;

4. Faites en sorte que ce dernier diagramme prenne également en compte la variable du genre des auteur · e · s.
5. Importez maintenant les données du fichier `filmographies_iconv.csv`.
6. Ce tableau n'est pas pourvu d'identifiant unique; créez un tel identifiant (utilisez par exemple le numéro de ligne);
7. Créez un diagramme de dispersion en utilisant, en x, la variable ANNEE et, en y, la variable doc_id que vous venez de créer;
8. Colorez chaque point en fonction d'une troisième variable, telle GENRE.

Pour aller plus loin

Centre de la science de la biodiversité du Québec, Série d'ateliers R du CSBQ, Chapitre 5. La grammaire des graphiques (GG).

Hadley Wickham et Garrett Grolemund, *R for Data Science. Import, Tidy, Transform, Visualize, and Model Data*, Sebastopol, O'Reilly

Hadley Wickham, "A Layered Grammar of Graphics", *Journal of Computational and Graphical Statistics*, vol. 19, no 1, p. 3-28. DOI: 10.1198/jcgs.2009.07098

Winston Chang, R Graphics Cookbook: Practical Recipes for Visualizing Data, Second Edition, Sebastopol (CA), O'Reilly, 2018.