

# Atelier 15: les graphiques

Pascal Brissette (U. McGill)

2022-09-25

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 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.

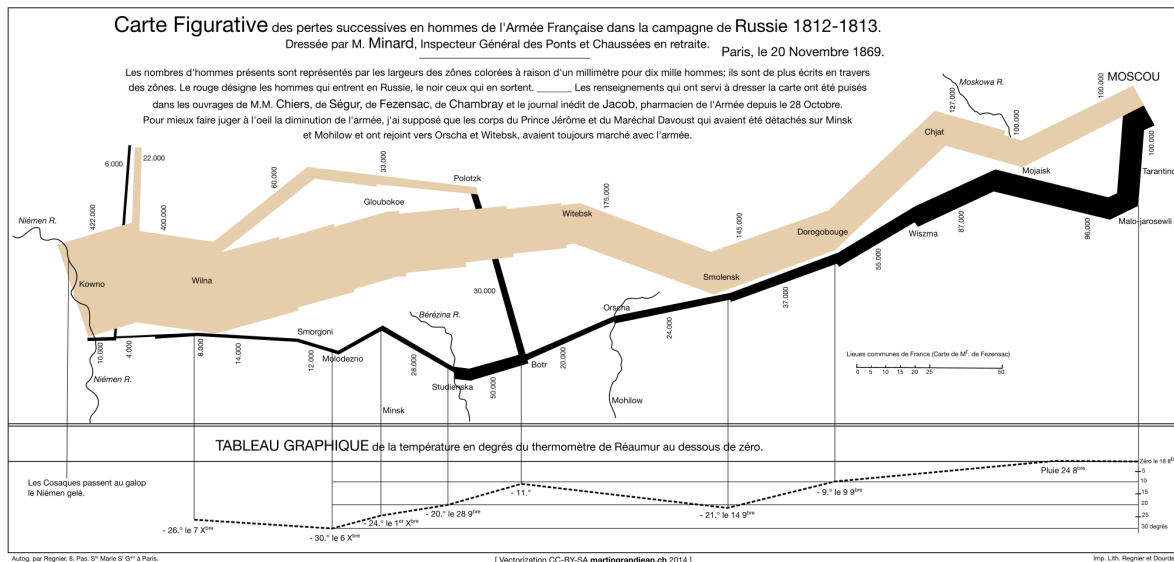


Figure 1: Charles Joseph Minard, “Carte figurative des pertes successives en hommes de l’armée française dans la campagne de Russie 1812-1813”, 1869

## 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 diagrammes à barres et de dispersion. Avant d'en discuter, il faut cependant introduire un type de données que vous n'avez pas encore croisé dans les précédents ateliers, le facteur (ou donnée catégorielle).

### Les facteurs

Pour expliquer ce qu'est un facteur ou une donnée catégorielle, rien de mieux que d'utiliser un exemple simple. Soit le tableau de données suivant, indiquant de manière aléatoire les profits mensuel de vente de vêtements d'occasion.

```
calendrier_profit_df <- data.frame(mois = sample(month.name, size = 50, replace = T),
                                   profit = round(sample(
                                     1:500, size = 50, replace = T
                                   )))

head(calendrier_profit_df)
```

	mois	profit
1	September	397
2	October	125
3	July	25
4	July	288
5	July	31
6	May	327

Le tableau comporte 50 lignes, mais même s'il en avait 5000, il n'y aurait jamais d'autre modalité dans la première colonne que ceux du calendrier grégorien. À moins que la personne ayant entré les données ait fait des erreurs dans l'orthographe des noms de mois, bien sûr, mais on y reviendra ci-dessous. Le fait est que si le vendeur a bien fait son travail, le nombre maximal de modalités dans la première colonne est de 12. On peut en observer la distribution comme suit:

```
table(calendrier_profit_df$mois)
```

April	August	December	February	January	July	June	March
2	3	4	3	3	8	4	2
May	November	October	September				
3	6	3	9				

Le vecteur de type **character**, quand il comporte un nombre **fini** et **limité** de modalités, gagne souvent à être converti en objet **factor**. Les objets **factors** ne sont pas stockés en mémoire comme des chaînes de caractères, mais comme des nombres correspondant à des niveaux distincts. Cela réduit donc de manière importante la taille du vecteur en mémoire et améliore les performances des analyses statistiques. De plus, les modalités distinctes d'un objet factor peuvent être ordonnées, ce qui est très utile lorsque, dans un graphique, on souhaite présenter les données d'une manière très précise.

Prenons à nouveau l'exemple des mois du calendrier. Si je demande à R d'ordonner les douze mois du calendrier, 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"
```

Lorsque je demande à ordonner ce vecteur, R place les mois d'août et d'avril en tête de vecteur, suivis par décembre. C'est ici que la fonction **factor()**, qui transforme une donnée en donnée catégorielle, peut être utile. Avec l'argument **levels =**, nous pourrions imposer un ordre. Et comme le nombre de mois est limité et forme un ensemble fini, 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 donner
ordre_mois <- c("janvier", "février", "mars", "avril", "mai", "juin", "juillet", "août", "septembre")

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

# On peut vérifier le type de l'objet
class(mois_f)
```

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

```
# 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
```

```
# 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'éventuelles erreurs typographiques dans les modalités. Supposons qu'il y ait une telle erreur dans mon vecteur initial. Je vais ordonner ce vecteur de caractères 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 catégoriel:

```
# Création d'un vecteur comprenant 16 éléments (répétition de certains noms de mois, dont l'
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 leve
mois_f <- factor(x = mois_c, levels = c("janvier", "février", "mars", "avril", "mai", "juin",
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 catégoriel renvoie donc `<NA>` à sa position.

Si vous demandez à R de vous fournir la structure du vecteur catégoriel, 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 catégoriel (`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 catégorielle de se présenter sous la forme de caractères, mais d’être emmagasinée dans R comme une suite de nombres entiers. La variable catégorielle est dite “discrète” et comme il n’y a pas d’intervalle entre les catégories, R peut associer à chacune, selon l’ordre qu’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, rectangles, 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:

Complete the template below to build a graph.

```

ggplot (data = <DATA> ) +
  <GEOM_FUNCTION>(mapping = aes(<MAPPINGS>),
  stat = <STAT>, position = <POSITION>) +
  <COORDINATE_FUNCTION> +
  <FACET_FUNCTION> +
  <SCALE_FUNCTION> +
  <THEME_FUNCTION>

```

**required**

**Not required, sensible defaults supplied**

**ggplot(data = mpg, aes(x = cty, y = hwy))** Begins a plot that you finish by adding layers to. Add one geom function per layer.

Figure 2: Capture de l'antisèche ggplot2, RStudio

Élément	Fonction	Explications
<b>Données</b>	<b>ggplot()</b>	Fonction d'initialisation du graphique. On y insère généralement le <b>tableau de données</b> dont les variables serviront à définir les éléments esthétiques.

Élément	Fonction	Explications
<b>Éléments esthétiques</b>	<code>aes()</code>	<p>Éléments esthétiques, précisés comme argument <code>mapping=</code> de la fonction d'initialisation s'ils sont identiques pour toutes les couches du graphique, ou à l'intérieur des fonctions commençant par <code>geom_</code> s'ils sont différents.</p> <p>Parmi les éléments esthétiques, on trouve:</p> <ul style="list-style-type: none"> <li>• <code>x</code> et <code>y</code> ==&gt; variables qui définissent respectivement les axes x et y du graphique;</li> <li>• <code>fill</code> ==&gt; variable qui définit la couleur de remplissage des formes géométriques;</li> <li>• <code>colour</code> ==&gt; variable qui définit la couleur des contours des formes géométriques;</li> <li>• <code>size</code> ==&gt; variable qui définit la taille des points ou des lignes;</li> <li>• <code>alpha</code> ==&gt; le degré de transparence des formes géométriques (entre 0 et 1);</li> <li>• <code>shape</code> ==&gt; variable qui définit des formes géométriques en complément des points, dans un graphique à points.</li> </ul>
<b>Éléments géométriques</b>	<code>geom_point()</code>	diagramme de dispersion (à points)
	<code>geom_bar()</code>	diagramme à barres. Prend une variable catégorielle en x. Par défaut, compte le nombre de valeurs par <b>catégorie</b> (x).
	<code>geom_col()</code>	diagramme à barres. Prend une variable catégorielle en x et une variable numérique en y. Équivalent de <code>geom_bar(stat="identity")</code> avec définition d'une variable continue y.
	<code>geom_histogram()</code>	diagramme à barres. Prend des variables continues en x et en y. Par défaut, <code>bins=30</code> .
<b>Facettes</b>	<code>facet_wrap()</code> ou <code>facet_grid()</code>	distribue les modalités d'une variable catégorielle en plusieurs graphiques de format réduit.
<b>Statistiques</b>	<code>stat_identity()</code> <code>stat_summary()</code> , etc.	Précise les opérations statistiques faites sur les données avant de générer le graphique.
<b>Coordonnées</b>	<code>coord_cartesian()</code>	Permet de fixer des limites aux axes x et y, ce qui a pour effet d'agrandir une portion du graphique.
	<code>coord_map()</code>	Projette une portion de la géographie terrestre sur une carte en 2 dimensions.

Élément	Fonction	Explications
Thèmes	<code>theme_grey()</code> , <code>theme_light()</code> , <code>theme_dark()</code> , 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 d'un graphique. Nous allons utiliser un jeu de données construit à partir du roman *Maria Chapdelaine*, de Louis Hémon. Plus précisément, nous allons nous intéresser à la présence dans le roman des trois prétendants de Maria Chapdelaine: François Paradis, Lorenzo Surprenant et Eutrope Gagnon.

Dans le premier tableau ci-dessous, très simple, il n'y a que deux colonnes et trois lignes. La colonne `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
inst_pack_f <- function(x) {
  if(!x %in% rownames(installed.packages())) {
    install.packages(x)
  }
  require(x, character.only = TRUE)
}

extensions_v <- c("data.table",
```



```
      "ggplot2",
      "stringr"
    )

lapply(extensions_v, inst_pack_f)
```

Loading required package: data.table

Loading required package: ggplot2

Loading required package: stringr

```
[[1]]
[1] TRUE
```

```
[[2]]
[1] TRUE
```

```
[[3]]
[1] TRUE
```

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

pretendants_freq_brute
```

```
      nom freq_brute
    <char>    <int>
1: françois      58
2: lorenzo       30
3: eutrope       44
```

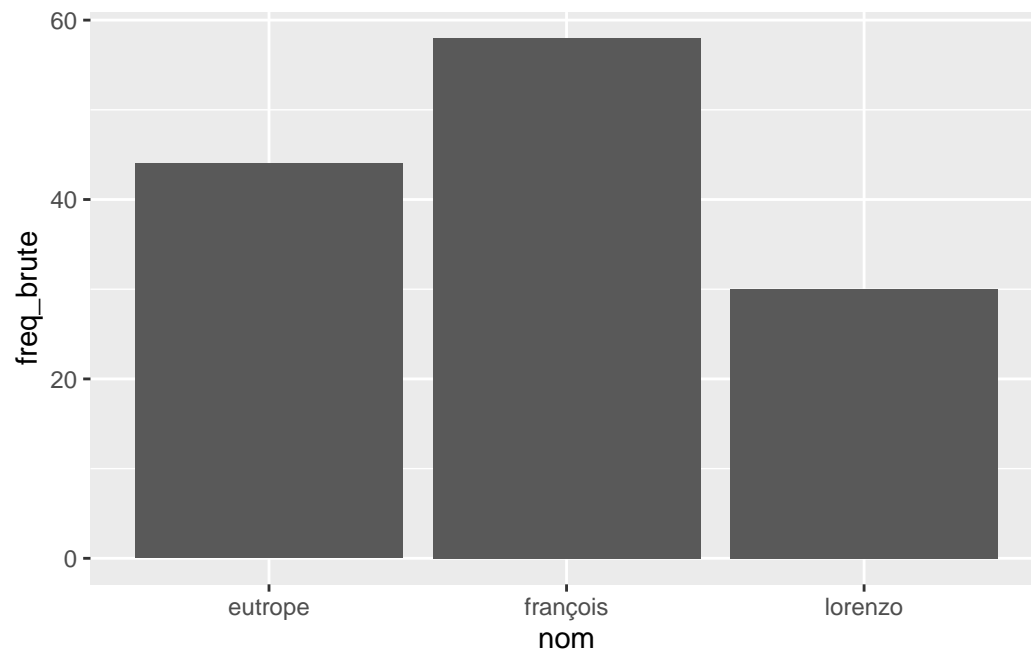
Un simple diagramme à barres nous permettra de visualiser ce décompte brut des occurrences des trois prénoms:

```
# On crée une première couche contenant les données et leur projection en composantes esthét.
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")
```

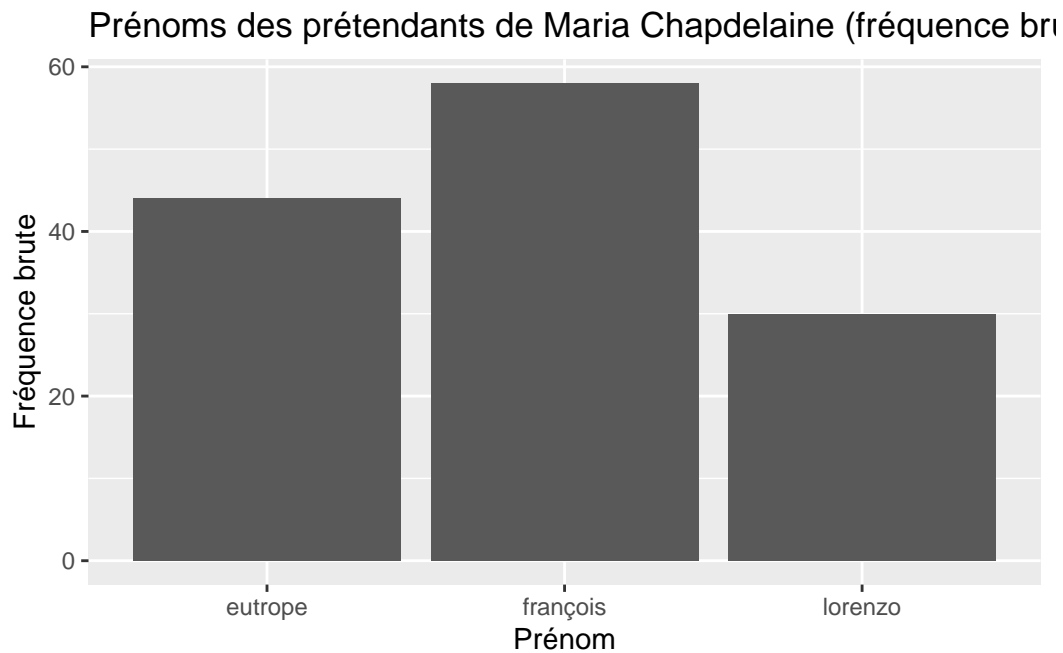
p2



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")
```

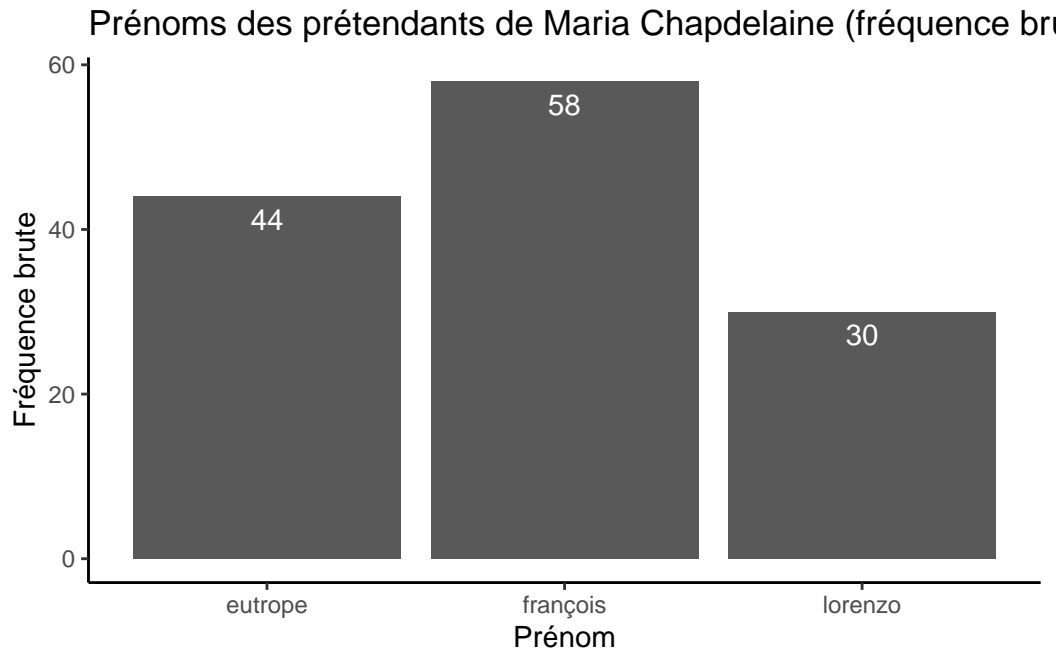
p3



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*. Cette fois, au lieu de calculer la fréquence brute des noms de prétendants pour tout le roman, on a calculé la fréquence des prénoms pour chacun des 16 chapitres du roman. Le tableau contient ainsi trois colonnes:

- Numéro du chapitre (**chapitre**);
- Le prénom du prétendant de Maria (**nom**);
- La fréquence des mentions de chaque prénom dans chaque chapitre (**freq**).

La colonne **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 raccourcit les noms donnés aux chapitres pour alléger les titres d'axes des graphiques
pretendants_freq_chap[, chapitre:=str_extract(chapitre, pattern = "(?<=\\s)[IVX]+$")]

# On transforme les titres de chapitres en données catégorielles et on ordonne leur présentation
```

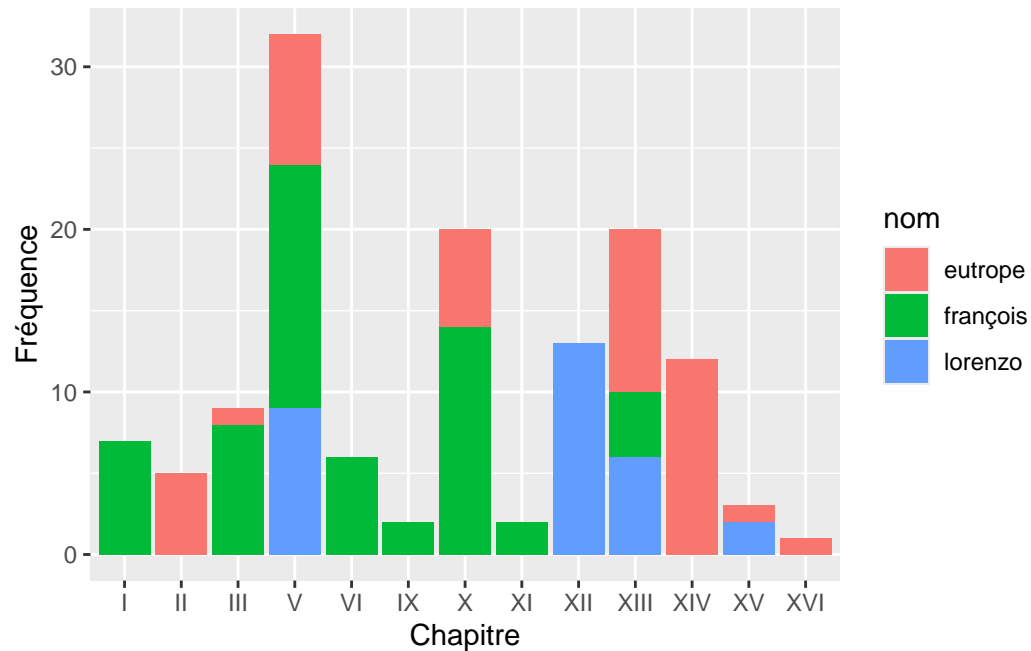
```
pretendants_freq_chap[, chapitre:=factor(chapitre, levels = c("I", "II", "III", "V", "VI", "X", "XI", "XII", "XIII", "XIV", "XV", "XVI"))]
pretendants_freq_chap
```

	chapitre	nom	freq
	<fctr>	<char>	<int>
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

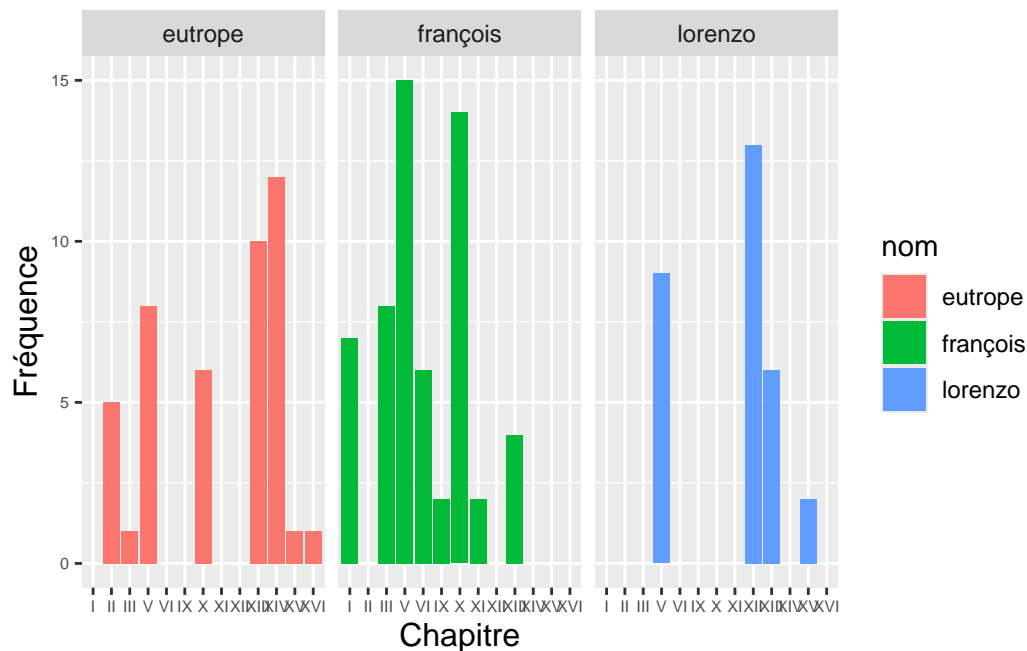
On ne peut séparer la valeur numérique (**freq**) des deux autres variables dont elle est la mesure, cela n'aurait pas de sens. La valeur de la première ligne, par exemple, fournit la fréquence du mot “françois” dans le segment du roman composé du chapitre I. Les trois variables sont solidaires. Pour 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 catégorielles. 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 catégorielle possède de valeurs uniques:

```
# La première couche constituée des données et des éléments esthétiques sera la même pour les deux facettes
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 selon
p + geom_bar(stat = "identity", position = "stack")+
  xlab("Chapitre")+
  ylab("Fréquence")
```



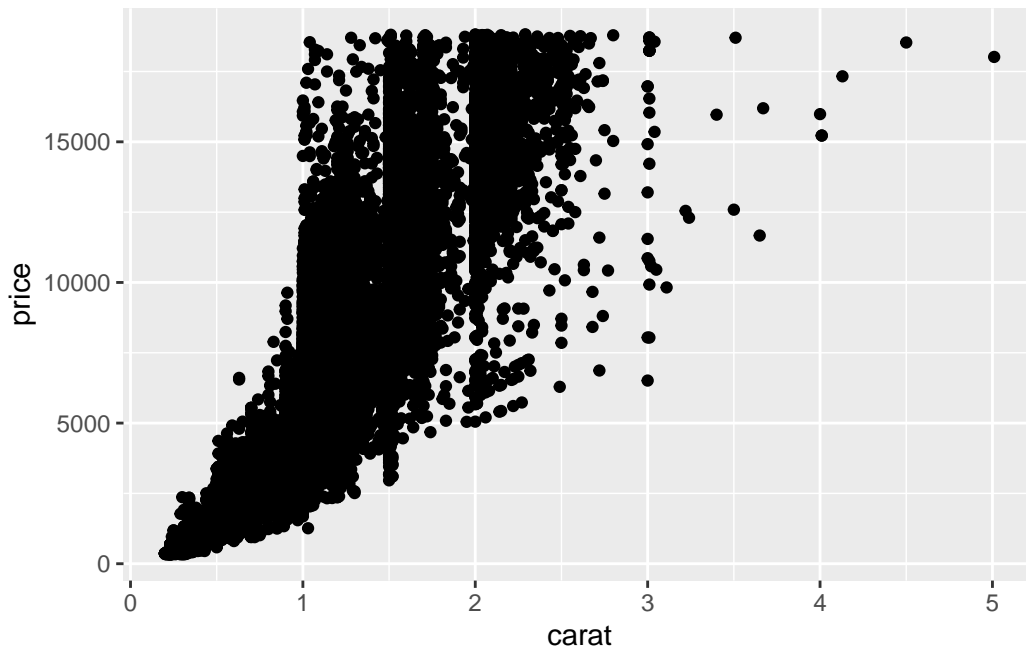
```
# Option 2: le diagramme distribue les données dans trois facettes
p + geom_bar(stat = "identity") +
  facet_wrap(~ nom) +
  theme(axis.text=element_text(size=6),      #On diminue la taille des titres d'axes pour éviter
        axis.title=element_text(size=12))+  #les chevauchements
  xlab("Chapitre")+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



## Le diagramme de dispersion

Le diagramme de dispersion, ou nuage de points, transpose chaque valeur 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` contenu dans l'extension `ggplot2`. Celui-ci propose une multitude d'informations sur un lot de 53 940 diamants. Si on voulait vérifier avec un diagramme de dispersion la corrélation entre les variables `carat` et `price`, deux variables continues, on donnerait à R les instructions suivantes:

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

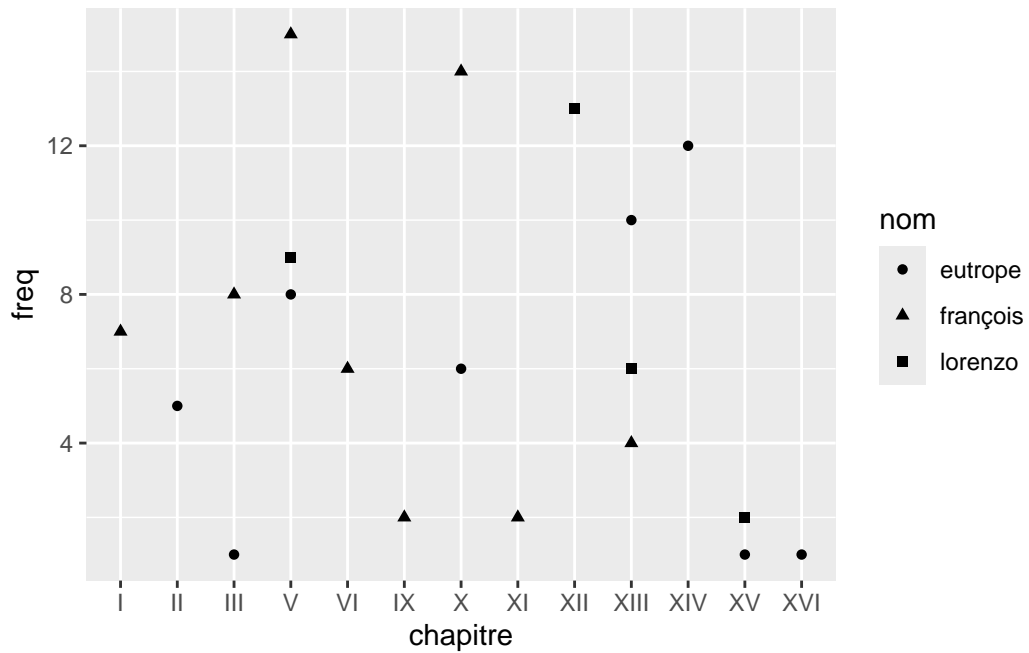


Chaque point de ce graphique représente un diamant défini par sa qualité, exprimée en carats, et son prix, exprimé en dollars.

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 noms des prétendants et la fréquence de leur mention. Comme nous n'avons que deux axes (x et y), nous devons utiliser un troisième élément esthétique pour représenter l'une des trois variables. En x, on mettra la variable indépendante, `chapitre`, on placera `freq` en y 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.

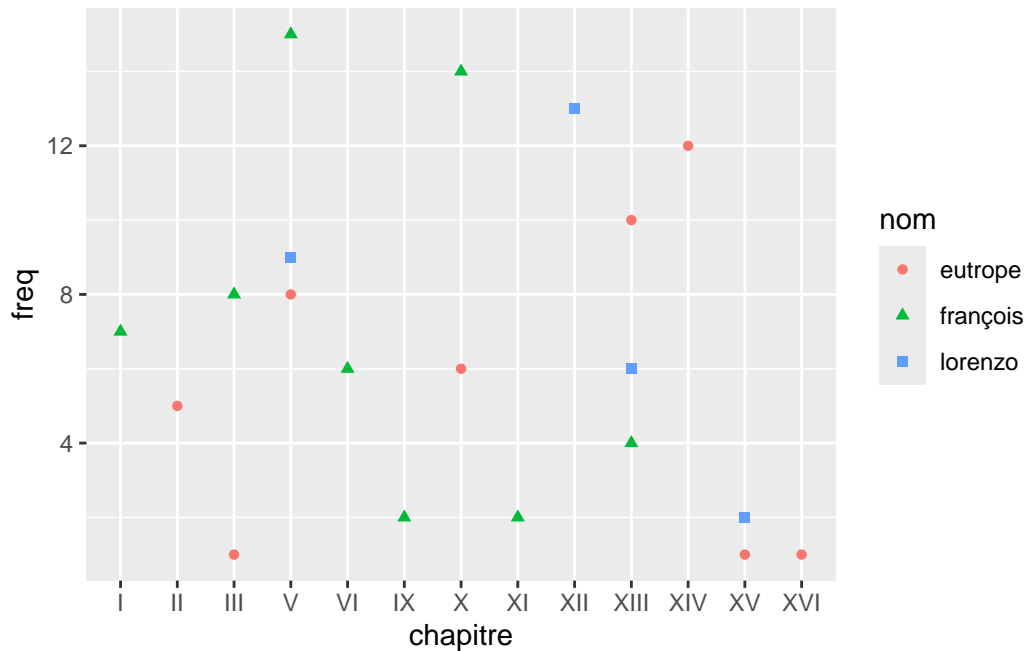
```
p <- ggplot(pretendants_freq_chap, aes(x=chapitre, y=freq, shape = nom))
p + geom_point()
```





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

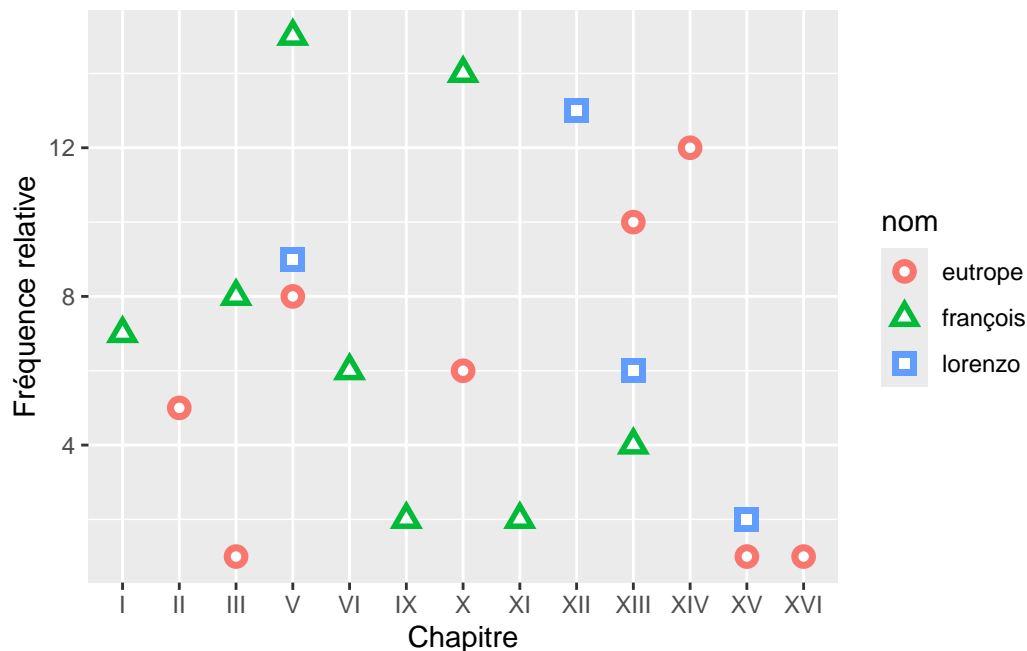
```
p <- ggplot(pretendants_freq_chap, aes(x=chapitre, y=freq, shape = nom, colour=nom))
p + geom_point()
```



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, respectivement, la couleur et la taille des points:

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

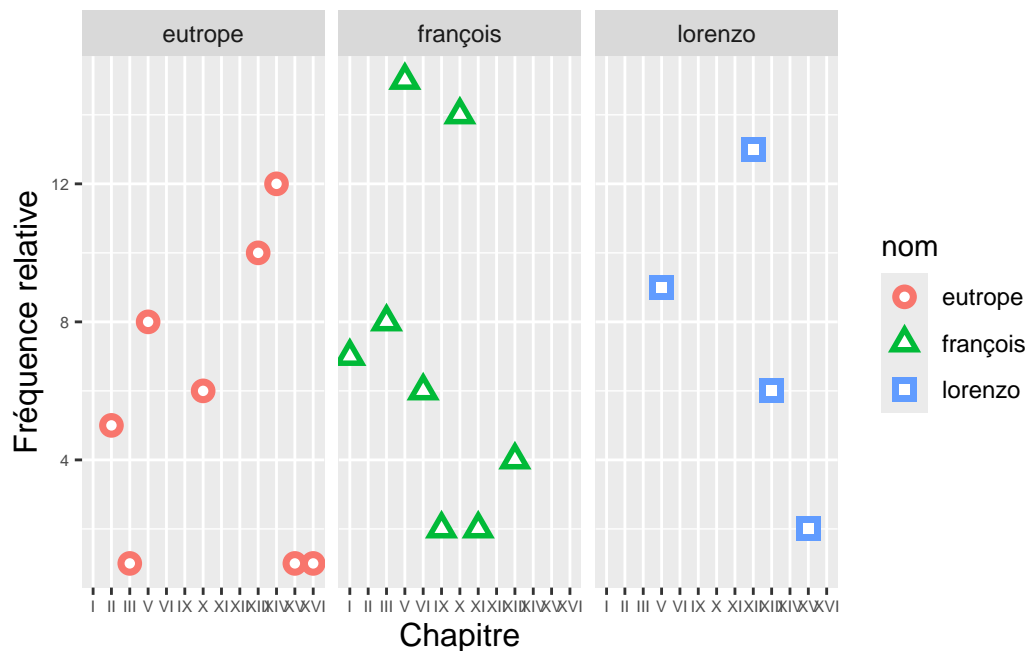
p + geom_point(aes(colour = nom), size = 4) +
  geom_point(colour = "white", size = 1.5) +
  xlab("Chapitre") +
  ylab("Fréquence relative")
```



Le diagramme gagne en lisibilité, mais il est encore difficile d'en tirer une information pertinente, mis à part que "françois" atteint au chapitre V, 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 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))

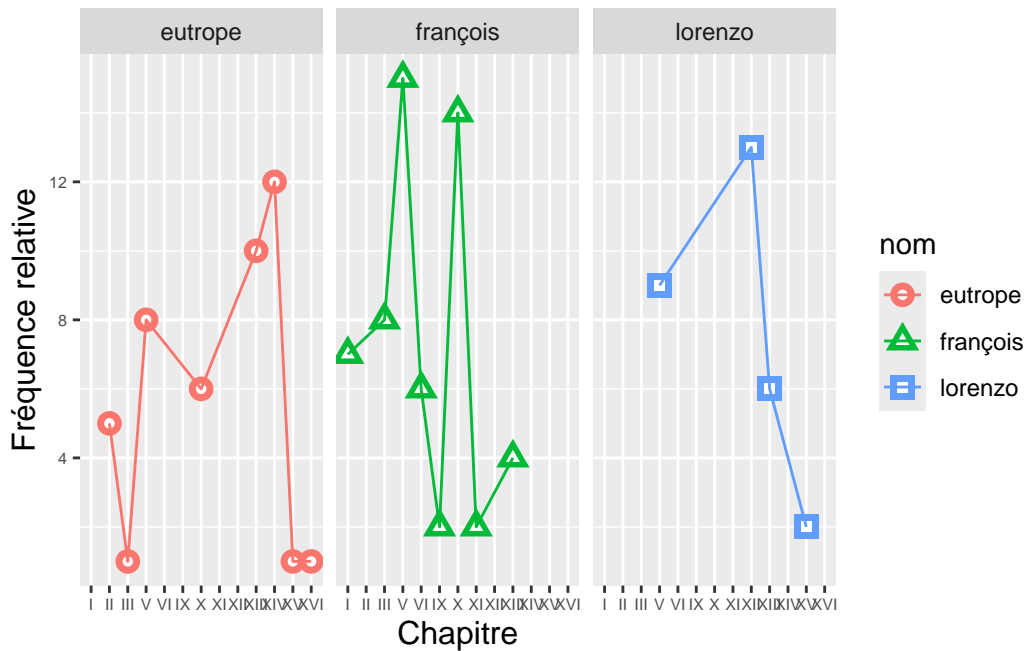
p + geom_point(aes(colour = nom), size = 4)+
  geom_point(colour = "white", size = 1.5)+
  facet_wrap(~nom)+
  theme(axis.text=element_text(size=6), # La fonction theme() permet de réduire la taille
        axis.title=element_text(size=12))+ # des caractères d'axes
  xlab("Chapitre")+
  ylab("Fréquence 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 symbolisent ou 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 avec `geom_bar()` qui montrera 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 catégorielle;
3. Faites en sorte que ce dernier diagramme prenne également en compte la variable du genre des auteurs `·e·s`.

## 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.