

Statistiques descriptives

Solène Colin, Vivien Roussez & Pascal Irz

26 September 2019

Contents

| | | |
|----------|---|-----------|
| 1 | Introduction | 5 |
| 1.1 | Le parcours de formation | 5 |
| 1.2 | Objectif du module 3 | 6 |
| 1.3 | Notions et méthodes présentées | 7 |
| 1.4 | Fondamentaux R présentés | 8 |
| 2 | Bien commencer | 9 |
| 2.1 | Créer un projet sous Rstudio pour vous permettre de recenser vos travaux. | 9 |
| 2.2 | Intégrer vos données | 12 |
| 2.3 | Créer votre arborescence de projet | 12 |
| 2.4 | Activer les packages nécessaires | 12 |
| 2.5 | Bien structurer ses projets data | 13 |
| 3 | Quelques préliminaires | 15 |
| 3.1 | Vocabulaire | 15 |
| 3.2 | Données utilisées pour la formation | 16 |
| 3.3 | Variables quantitatives | 18 |
| 3.4 | Variables qualitatives | 18 |
| 4 | Une variable qualitative | 25 |
| 4.1 | Tableaux de synthèse | 26 |
| 4.2 | Graphiques | 29 |
| 5 | Deux variables quantitatives | 35 |
| 5.1 | Représentations graphiques | 35 |
| 5.2 | Coefficients de corrélation | 47 |
| 5.3 | Régression | 49 |
| 6 | Deux variables qualitatives | 51 |
| 6.1 | Définitions | 51 |
| 6.2 | Tableaux de synthèse | 51 |
| 6.3 | Graphiques | 55 |
| 6.4 | Avec pondération | 58 |

| | |
|--|------------|
| 7 Lien variable quantitative - variable qualitative | 61 |
| 7.1 Statistiques en fonction d'un facteur | 61 |
| 7.2 Eléments théoriques | 61 |
| 7.3 Représentation graphique | 62 |
| 7.4 Calcul du rapport de corrélation | 64 |
| 8 Tests | 65 |
| 8.1 Sur une variable quantitative | 65 |
| 8.2 Sur une variable qualitative | 69 |
| 8.3 Sur 2 variables qualitatives | 70 |
| 8.4 Sur un croisement quantitatif/qualitatif | 72 |
| 8.5 Résumé | 81 |
| 9 Exercice de synthèse | 83 |
| 9.1 Enoncé | 83 |
| 9.2 Corrigé | 84 |
| 10 Une variable quantitative | 105 |
| 10.1 Statistiques de distribution | 106 |
| 10.2 Calcul sur plusieurs variables | 111 |
| 10.3 Représentations graphiques | 114 |
| 10.4 Discrétisation d'une variable continue | 121 |

Chapter 1

Introduction



Crédit photographique Pascal Boulin

1.1 Le parcours de formation

Ce dispositif de formation vise à faire monter en compétence les agents du MTES (Ministère de la transition écologique et solidaire) et du MCT (Ministère de la cohésion des territoires) dans le domaine de la science de la donnée avec le

logiciel R. Il est conçu pour être déployé à l'échelle nationale par le réseau des CVRH (Centre de Valorisation des Ressources Humaines).

Le parcours proposé est structuré en modules de 2 jours chacun. Les deux premiers (ou un niveau équivalent) sont des pré-requis pour suivre les suivants qui sont proposés “à la carte” :

1. Socle : Premier programme en R
2. Socle : Préparation des données
3. Statistiques descriptives
4. Analyses multivariées
5. Datavisualisation : Produire des graphiques, des cartes et des tableaux
6. Documents reproductibles avec RMarkdown (2^{ème} semestre 2019)

... et en perspective : analyse spatiale, applis interactives avec Shiny, big data, etc.

La mise à disposition des supports de formation se fait désormais par la page d'accueil du parcours de formation. Ces supports sont en licence ouverte.

Si vous souhaitez accéder aux sources et aux données mobilisées pendant les formations, il faut directement les télécharger depuis le Github du ministère.

Pour vous tenir au courant de l'offre de formation proposée par le réseau des CVRH, consultez la plateforme OUPS. Vous pouvez vous y abonner pour recevoir les annonces qui vous intéressent.

Il existe une liste pour diffuser de l'information, échanger autour de R ou lever des points de blocage. Pour s'inscrire, envoyer un message vide avec le titre “subscribe labo.communaute-r” à l'adresse sympa@developpement-durable.gouv.fr.

1.2 Objectif du module 3

Ce qui est visé est une autonomie en matière de statistiques de base avec le logiciel R.

Le module comprend, pour chacune des parties ci-dessous, l'acquisition ou le rappel des notions statistiques abordées, ainsi que la maîtrise de la production et de l'interprétation, avec le logiciel R, des statistiques descriptives, des représentations graphiques et des tests usuels.

1.3 Notions et méthodes présentées

1.3.1 Analyse univariée d'une variable quantitative

- Histogramme
- Courbe de densité
- Diagramme quantile-quantile
- Statistiques de tendance centrale (moyenne, médiane)
- Statistiques de dispersion (variance, coefficient de variation, intervalle inter-quartiles)
- Méthodes de discréétisation

1.3.2 Analyse univariée d'une variable qualitative

- Diagrammes en barres et en secteurs
- Tableau de fréquences pondérées ou non pondérées

1.3.3 Relation entre 2 variables quantitatives

- Nuage de points
- Corrélation paramétrique ou non paramétrique

1.3.4 Relation entre 2 variables qualitatives

- Graphique en barres empilées ou juxtaposées
- Graphique en mosaïque
- Tableau de contingence
- Profils-lignes et profils-colonnes
- Test du χ^2 , V de Cramer

1.3.5 Relation entre une variable qualitative et une variable quantitative

- Agrégation d'une variable quantitative selon une variable qualitative
- Boxplot, violin plot
- ANOVA

1.4 Fondamentaux R présentés

Objets R, scripts, graphiques avec `ggplot2`, tests avec les packages de `base`, `dplyr` et `lsr`.

Chapter 2

Bien commencer

2.1 Créer un projet sous Rstudio pour vous permettre de recenser vos travaux.

Pourquoi travailler avec les projets Rstudio plutôt que les scripts R ?

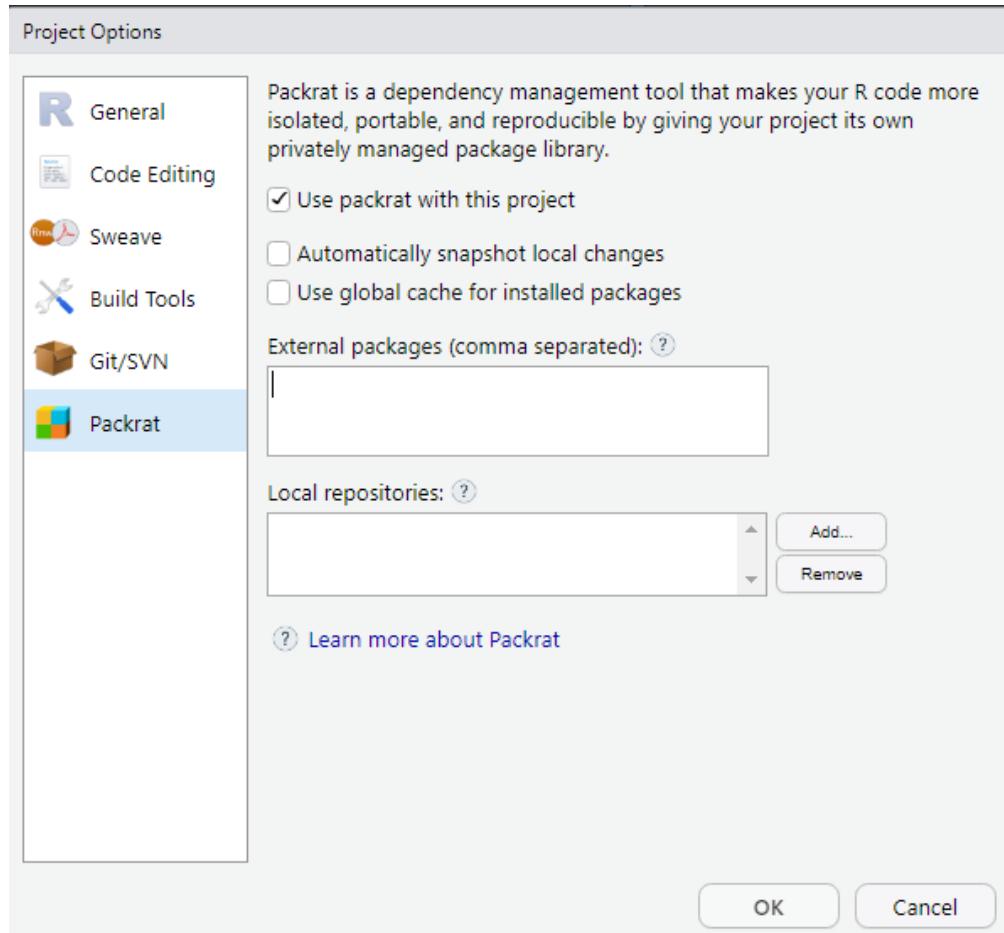
- Cela permet la portabilité : le répertoire de travail par défaut d'un projet est le répertoire où est ce projet. Si vous transmettez celui-ci à un collègue, le fait de lancer un programme ne dépend pas de l'arborescence de votre machine.

Finis les `setwd("chemin/qui/marche/uniquement/sur/mon/poste")` !

- Toujours sur la portabilité, un projet peut être utilisé avec un outil comme `packrat` qui va vous intégrer en interne au projet l'ensemble des packages nécessaires au projet. Cela permet donc à votre collègue à qui vous passez votre projet de ne pas avoir à les installer et, surtout, si vous mettez à jour votre environnement R, votre projet restera toujours avec les versions des packages avec lesquelles vous avez fait tourner votre projet à l'époque. Cela évite d'avoir à subir les effets d'une mise à jour importante d'un package qui casserait votre code.

Pour activer `packrat` sur un projet, aller dans Tools/Project Options->Packrat

En savoir plus sur Packrat

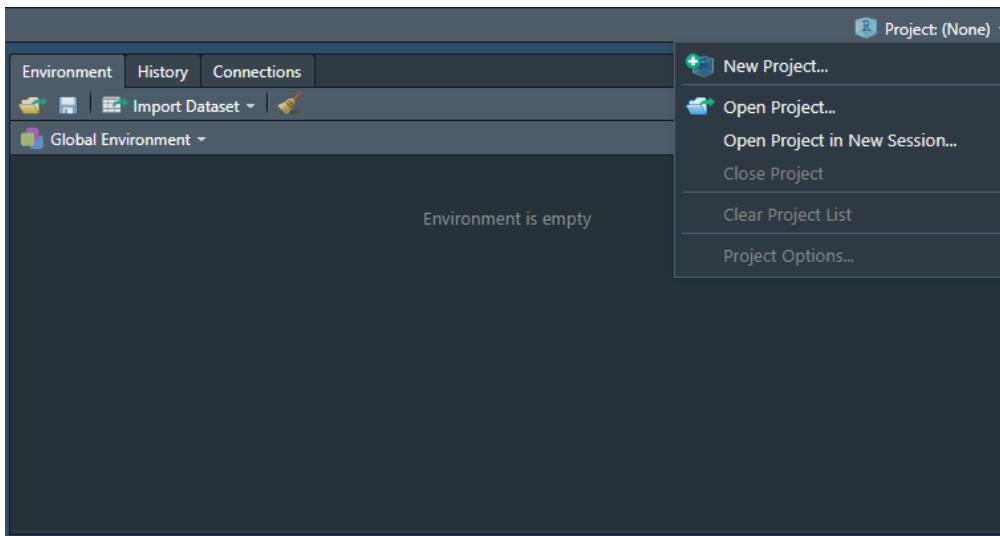


- Cela permet de se forcer à travailler en mode projet : on intègre à un seul endroit tout ce qui est lié à un projet : données brutes, données retravaillées, scripts, illustrations, documentations, publications... et donc y compris les packages avec **packrat**.
- On peut travailler sur plusieurs projets en même temps, Rstudio ouvre autant de sessions que de projets dans ce cas.
- Les projets Rstudio intègrent une interface avec les outils de gestion de version Git et SVN. Cela veut dire que vous pouvez versionner votre projet et l'héberger simplement comme répertoire sur des plateformes de gestion de code telle que Github ou Gitlab.

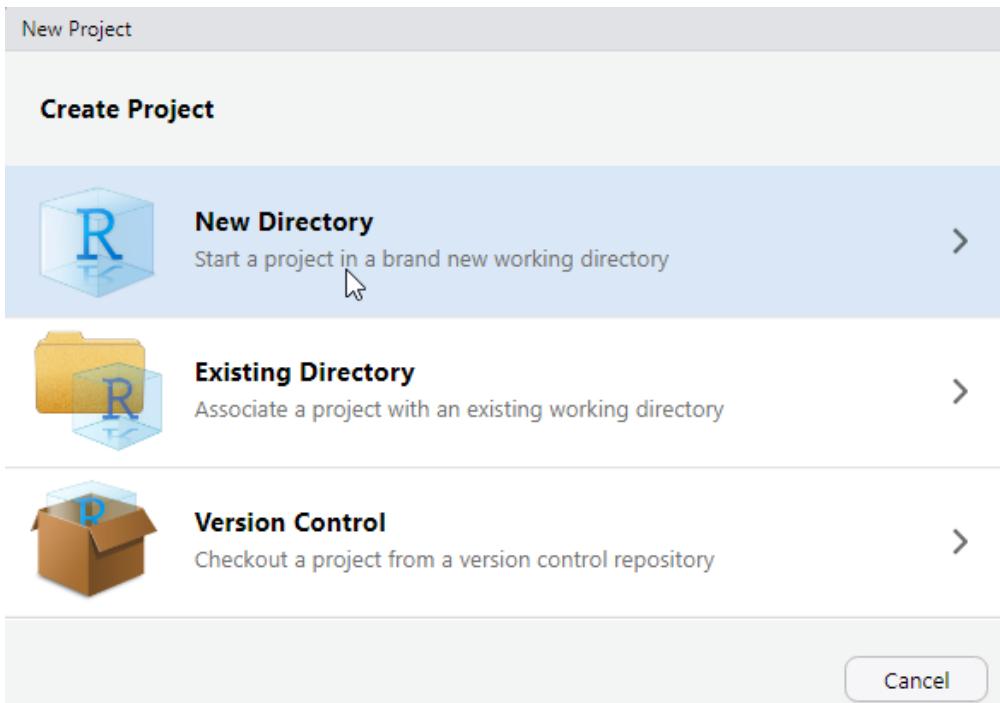
Pour créer un projet :

- Cliquez sur *Project* en haut à droite puis *New Project*.

2.1. CRÉER UN PROJET SOUS RSTUDIO POUR VOUS PERMETTRE DE RECENCER VOS TRAVAUX.11



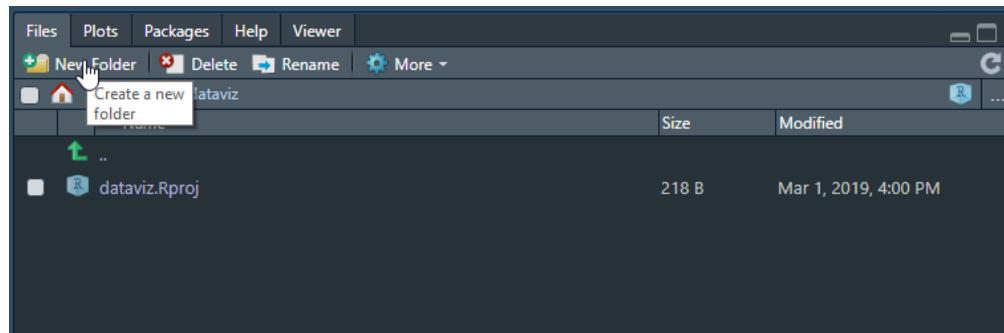
- Cliquez sur *New Directory*.



2.2 Intégrer vos données

Une bonne pratique est de créer un sous répertoire `/data` pour stocker les données sur lesquelles vous aurez à travailler.

Vous pouvez le faire depuis l'explorateur de fichier de votre système d'exploitation ou directement à partir de l'explorateur de fichier de RStudio.



Cela marche bien quand on a un seul type de données, mais en général on va avoir à travailler sur des données brutes que l'on va retravailler ensuite et vouloir stocker à part. Si par la suite vous souhaitez avoir des exemples de bonnes pratiques sur comment structurer vos données, vous pouvez vous référer au chapitre data du livre d'Hadley Wickham sur la construction de packages R (tout package R étant aussi un projet !).

2.3 Créez votre arborescence de projet

- Créez un répertoire `/src` où vous mettrez vos scripts R.
- Créez un répertoire `/figures` où vous mettrez vos illustrations issues de R.

2.4 Activer les packages nécessaires

Commencez par rajouter un script dans le répertoire `/src` à votre projet qui commencera par :

- activer l'ensemble des packages nécessaires
- charger les données dont vous aurez besoin.

```
library (plyr)
library (tidyverse)
library (forcats)
```

```
library (lsr)
library (sp)
library (cartography)
library (GGally)
library (ggmosaic)
library (DT)
library (plotly)
library (grid)

dat <- read.csv (file = "data/Base_synth_territoires.csv", sep = ';',
                 dec = ',', colClasses = c ("REG" = "factor")) %>%
  select (-starts_with ("ET"))
```

2.5 Bien structurer ses projets data

Plusieurs documents peuvent vous inspirer sur la structuration de vos projets data par la suite.

En voici quelques-uns :

- <https://github.com/pavopax/new-project-template>
- <https://nicercode.github.io/blog/2013-04-05-projects/>
- <https://www.inwt-statistics.com/read-blog/a-meaningful-file-structure-for-r-projects.html>
- <http://projecttemplate.net/architecture.html>

A partir du moment où quelques grands principes sont respectés (un répertoire pour les données brutes en lecture seule par exemple), le reste est surtout une question d'attraction plus forte pour l'une ou l'autre solution. L'important est de vous tenir ensuite à conserver toujours la même arborescence dans vos projets afin de vous y retrouver plus simplement.

Chapter 3

Quelques préliminaires

3.1 Vocabulaire

- Statistique : un *résumé* de l'information contenue dans l'ensemble des observations. Par exemple : la somme, la moyenne, l'écart-type, les quantiles...
- Description des données : un tableau peut être regardé comme un ensemble de lignes (des individus statistiques) ou comme un ensemble de colonnes (des variables).

| | V1 | V2 | V3 | V4 | V5 | V6 | ... | Vp |
|----|----|----|----|----|----|----|-----|----|
| 1 | | | | | | | | |
| 2 | | | | | | | | |
| 3 | | | | | | | | |
| 4 | | | | | | | | |
| 5 | | | | | | | | |
| 6 | | | | | | | | |
| 7 | | | | | | | | |
| 8 | | | | | | | | |
| 9 | | | | | | | | |
| 10 | | | | | | | | |
| 11 | | | | | | | | |
| 12 | | | | | | | | |
| : | | | | | | | | |
| n | | | | | | | | |

→ structure du tableau (`dataframe`) à modifier ? Voir le module 2 sur la préparation des données.

3.2 Données utilisées pour la formation

Nous utiliserons la base Insee de comparaison des territoires. Pour chaque commune, nous disposons d'un certain nombre d'informations, contenues dans les différentes variables (population, naissance, décès, nombre de logements, etc...).

```
str (dat) # premier aperçu de la base
```

```
## 'data.frame':   36689 obs. of  29 variables:
## $ CODGEO      : Factor w/ 36689 levels "01001","01002",...: 1 2 3 4 5 6 7 8 9 10 ...
## $ LIBGEO      : Factor w/ 34174 levels "\xc8ve","\xc8vres",...: 13683 13685 813 813 ...
## $ REG         : Factor w/ 17 levels "01","02","03",...: 15 15 15 15 15 15 15 15 15 15 ...
## $ DEP         : Factor w/ 100 levels "01","02","03",...: 1 1 1 1 1 1 1 1 1 1 ...
## $ ZAU         : Factor w/ 9 levels "111 - Grand p\xf4le (plus de 10 000 emplois)"...
## $ ZE          : Factor w/ 321 levels "0050 - Mont-de-Marsan",...: 260 248 248 260 ...
## $ P14_POP     : int  767 239 14022 1627 109 2570 743 338 1142 397 ...
## $ P09_POP     : int  787 207 13350 1592 120 2328 660 336 960 352 ...
## $ SUPERF      : num  15.95 9.15 24.6 15.92 5.88 ...
## $ NAIS0914    : int  40 16 1051 117 8 175 59 12 56 25 ...
## $ DECE0914    : int  25 7 551 41 3 78 20 11 32 10 ...
## $ P14_MEN     : num  306 99.3 6161.1 621.1 52.5 ...
## $ NAISD15     : int  13 5 222 15 2 21 11 2 18 4 ...
## $ DECESD15    : int  5 1 121 7 2 9 3 3 5 0 ...
## $ P14_LOG     : num  342.7 161.2 6838.4 661.8 71.5 ...
## $ P14_RP      : num  306 99.3 6161.1 621.1 52.5 ...
## $ P14_RSECOCC : num  14 47.3 121.6 10.9 10.9 ...
## $ P14_LOGVAC  : num  22.74 14.55 555.64 29.85 8.14 ...
## $ P14_RP_PROP : num  260 84.6 2769 473.3 37.7 ...
## $ NBMENFISC13: int  297 99 6034 617 47 1014 299 140 431 137 ...
## $ PIMP13      : num  NA NA 57.4 NA NA ...
## $ MED13       : num  22130 23213 19554 22388 21872 ...
## $ TP6013      : num  NA NA 15.1 NA NA ...
## $ P14_EMPLT   : num  85.16 12.81 7452.93 280.57 5.95 ...
## $ P14_EMPLT_SAL: num  52.19 4.95 6743.37 206.38 3.96 ...
## $ P09_EMPLT   : num  65.57 17.64 7551.68 286.61 5.29 ...
## $ P14_POP1564 : num  463 141.6 8962.8 1043.1 71.3 ...
## $ P14_CHOM1564: num  33 9.84 1059.73 66.33 7.93 ...
## $ P14_ACT1564 : num  376 121 6681.9 842.1 57.5 ...
```

Liste des variables :

- *CODGEO* : Code du département suivi du numéro de commune ou du numéro d'arrondissement municipal
- *LIBGEO* : Libellé de la commune ou de l'arrondissement municipal pour Paris, Lyon et Marseille
- *REG* : Région

- *DEP* : Département
- *ZAU* : Classe du zonage en aires urbaines
- *ZE* : Zone d'emploi
- *P14_POP* : Population en 2014
- *P09_POP* : Population en 2009
- *SUPERF* : Superficie (en km^2)
- *NAIS0914* : Nombre de naissances entre le 01/01/2009 et le 01/01/2014
- *DECE0914* : Nombre de décès entre le 01/01/2009 et le 01/01/2014
- *P14_MEN* : Nombre de ménages en 2014
- *NAISD15* : Nombre de naissances domiciliées en 2015
- *DECESD15* : Nombre de décès domiciliés en 2015
- *P14_LOG* : Nombre de logements en 2014
- *P14_RP* : Nombre de résidences principales en 2014
- *P14_RSECOCC* : Nombre de résidences secondaires et logements occasionnels en 2014
- *P14_LOGVAC* : Nombre de logements vacants en 2014
- *P14_RP_PROP* : Nombre de résidences principales occupées par propriétaires en 2014
- *NBMENFISC14* : Nombre de ménages fiscaux en 2014
- *PIMP14* : Part des ménages fiscaux imposés en 2014
- *MED14* : Médiane du niveau de vie en 2014
- *TP6014* : Taux de pauvreté en 2014
- *P14_EMPLT* : Nombre d'emplois au lieu de travail en 2014
- *P15_EMPLT_SAL* : Nombre d'emplois salariés au lieu de travail en 2015
- *P09_EMPLT* : Nombre d'emplois au lieu de travail en 2009
- *P15_POP1564* : Nombre de personnes de 15 à 64 ans en 2015
- *P15_CHOM1564* : Nombre de chômeurs de 15 à 64 ans en 2015
- *P15_ACT1564* : Nombre de personnes actives de 15 à 64 ans en 2015

3.3 Variables quantitatives

Une variable **quantitative** permet de mesurer une grandeur (quantité). Elle peut être :

- **discrète** (un nombre fini de valeurs possibles). *Exemple : un nombre de logements*
- **continue** (*a priori*, toutes les valeurs possibles). *Exemple : une taille, une surface, un revenu*

On peut calculer des statistiques (somme, moyenne, ...) sur les variables quantitatives.

Dans R, il s'agit des variables de type **numeric**. Par exemple :

```
v <- c(12.5, 25.38, 14.9)
class(v)

## [1] "numeric"
```

3.4 Variables qualitatives

3.4.1 Définition

Une variable **qualitative** indique des caractéristiques qui ne sont pas des quantités. Les différentes valeurs que peut prendre cette variable sont appelées les catégories ou modalités (**levels** dans R). Elle peut être :

- **ordonnée** (exprimer un ordre). *Exemple : “petit - moyen - grand”*
- **non ordonnée**. *Exemple : une couleur, un groupe sanguin...*

Une variable qualitative ne permet pas de faire des calculs (la moyenne d'un groupe sanguin n'a aucun sens).

Dans R, il s'agit des variables **factor**. Elles peuvent être générée par la fonction **factor()**. Par exemple :

```
v <- factor(x = c("un peu", "beaucoup", "passionnément", "beaucoup",
                  "un peu", "un peu", "un peu"))
class(v)

## [1] "factor"
```

Les modalités de la variable peuvent être obtenues grâce à la fonction **levels()**.

```
levels(v)

## [1] "beaucoup"      "passionnément" "un peu"
```

```
v %>% table ()
## .
##      beaucoup passionnément      un peu
##            2                  1                  4
```

NB : Ce sont des étiquettes mais la variable est stockée sous forme d'entiers.

```
as.integer (v) %>% table ()
```

```
## .
## 1 2 3
## 2 1 4
```

3.4.2 Manipulation des factor



La manipulation des **factor** fait intervenir le package **forcats**, du “métapackage” **tidyverse**, qui propose de nombreuses fonctions.

Les fonctions de ce package sont reconnaissables à leur préfixe **fct_**.

On peut trouver des exemples d'utilisation (en français) sur ce blog.

- Dans un dataframe contenant une variable de type **factor**, on filtre comme sur une chaîne de caractère :

```
d <- filter (dat, str_sub (string = ZAU, start = 1, end = 3) != "120")
# ou bien :
d <- filter (dat, ZAU != "111 - Grand pôle (plus de 10 000 emplois)")
```

- A la suite d'une opération de sélection des parmi les lignes, certains `levels` peuvent disparaître. Ils seront toutefois toujours présents dans la liste des modalités de la variables. La fonction `fct_drop()`, appliquée sur un vecteur, permet de se débarasser des modalités désormais inutilisées.

```
d %>% pull (ZAU) %>% levels ()
```

```
## [1] "111 - Grand p\xf4le (plus de 10 000 emplois)"
## [2] "112 - Couronne d'un grand p\xf4le"
## [3] "120 - Multipolaris\xe9e des grandes aires urbaines"
## [4] "211 - Moyen p\xf4le (5 000 \xe0 10 000 emplois)"
## [5] "212 - Couronne d'un moyen p\xf4le"
## [6] "221 - Petit p\xf4le (de 1 500 \xe0 5 000 emplois)"
## [7] "222 - Couronne d'un petit p\xf4le"
## [8] "300 - Autre commune multipolaris\xe9e"
## [9] "400 - Commune isol\xe9e hors influence des p\xf4les"
```

```
d %>% pull (ZAU) %>% fct_drop () %>% levels ()
```

```
## [1] "112 - Couronne d'un grand p\xf4le"
## [2] "120 - Multipolaris\xe9e des grandes aires urbaines"
## [3] "211 - Moyen p\xf4le (5 000 \xe0 10 000 emplois)"
## [4] "212 - Couronne d'un moyen p\xf4le"
## [5] "221 - Petit p\xf4le (de 1 500 \xe0 5 000 emplois)"
## [6] "222 - Couronne d'un petit p\xf4le"
## [7] "300 - Autre commune multipolaris\xe9e"
## [9] "400 - Commune isol\xe9e hors influence des p\xf4les"
```

La modalité 111 - Grand pôle (plus de 10 000 emplois) a bien été supprimée.

Pour effectuer l'opération sur l'ensemble des variables `factor` d'un `dataframe`, il faut utiliser la fonction `droplevels()` du package `base`.

- Les fonctions `levels()` et `fct_recode()` permettent de modifier les `levels`. La fonction `levels()` renomme toutes les modalités et s'applique sur un vecteur de type `factor`. `fct_recode()` permet de renommer seulement les `levels` voulus et peut être imbriqué dans un `mutate()`.

```
vec_ZAU <- pull (dat, ZAU)
levels(vec_ZAU) <- c("111 - Grand pôle",
                      "112 - Couronne GP",
                      "120 - Multipol grandes AU",
                      "211 - Moyen pôle",
                      "212 - Couronne MP" ,
                      "221 - Petit pôle",
                      "222 - Couronne PP",
                      "300 - Autre multipol.",
                      "400 - Commune isolée")
```

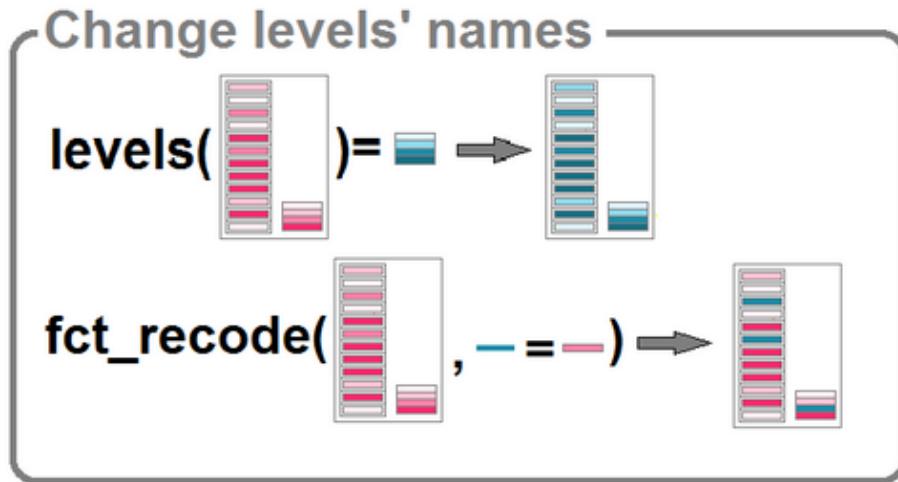


Figure 3.1: [http://perso.ens-lyon.fr/lise.vaudor/
manipulation-de-facteurs-avec-forcats/](http://perso.ens-lyon.fr/lise.vaudor/manipulation-de-facteurs-avec-forcats/)

```
levels (vec_ZAU)

## [1] "111 - Grand pôle"           "112 - Couronne GP"
## [3] "120 - Multipol grandes AU" "211 - Moyen pôle"
## [5] "212 - Couronne MP"         "221 - Petit pôle"
## [7] "222 - Couronne PP"         "300 - Autre multipol."
## [9] "400 - Commune isolée"

dat$ZAU2 <- vec_ZAU # création de la variable ZAU2 dans le data frame dat

dat <- dat %>%
  mutate (DEP2 = fct_recode (DEP, "Ain" = "01", "Aisne" = "02"))

• Pour agréger des facteurs et compter le nombre de modalités, on peut se servir de fct_count() :

pull (dat, ZAU2) %>%
  fct_recode (urbain = "111 - Grand pôle", urbain = "211 - Moyen pôle",
             urbain = "221 - Petit pôle",
             periurbain = "112 - Couronne GP", periurbain = "212 - Couronne MP",
             periurbain = "120 - Multipol grandes AU", periurbain = "300 - Autre multipol.",
             periurbain = "222 - Couronne PP",
             rural = "400 - Commune isolée") %>%
  fct_count ()

## # A tibble: 3 x 2
```

```
##   f          n
##   <fct>     <int>
## 1 urbain    4629
## 2 periurbain 24677
## 3 rural     7383
```

- Pour modifier l'ordre des facteurs (cela peut être utile en particulier pour les représentations graphiques), il existe plusieurs fonctions :

```
# renverser l'ordre
dat <- dat %>%
  mutate (ZAU3 = fct_rev (ZAU2))

# ordonner "à la main"
dat <- dat %>%
  mutate (ZAU3 = fct_relevel (ZAU2, "221 - Petit pôle", "111 - Grand pôle"))

# ordonner selon ordre d'apparition
dat <- dat %>%
  mutate (ZAU3 = fct_inorder (ZAU2))

# ordonner selon la fréquence
dat <- dat %>%
  mutate (ZAU3 = fct_infreq (ZAU2))
```

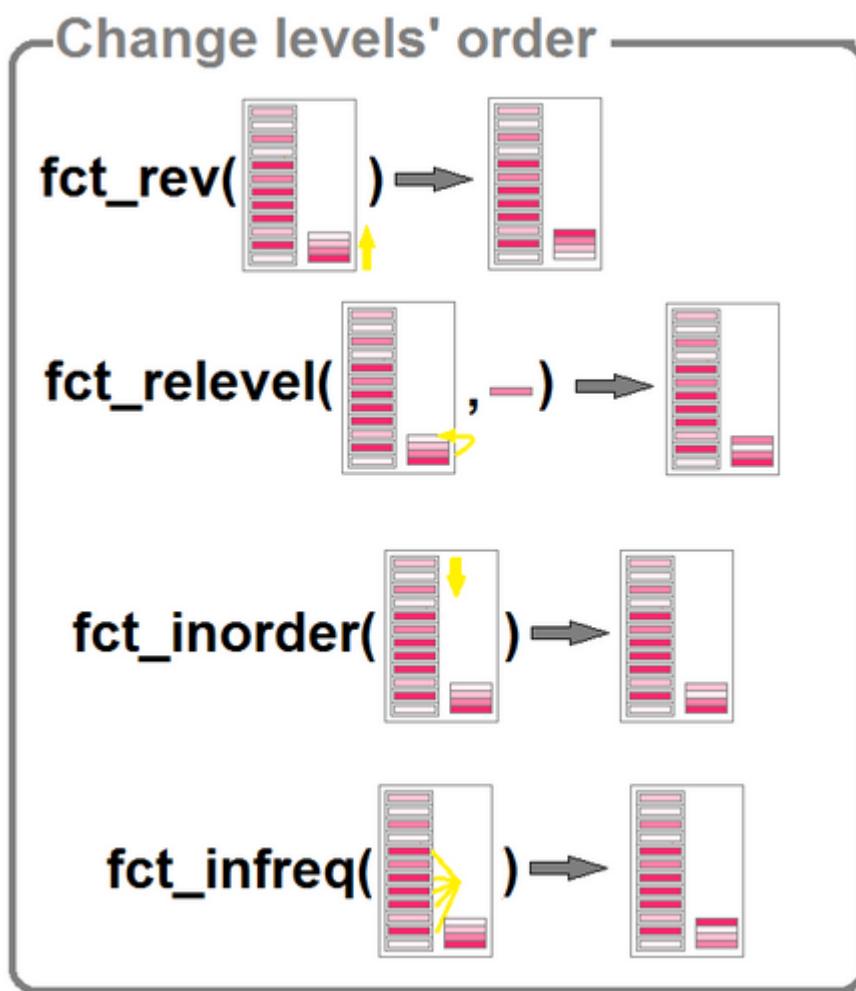


Figure 3.2: [http://perso.ens-lyon.fr/lise.vaudor/
manipulation-de-facteurs-avec-forcats/](http://perso.ens-lyon.fr/lise.vaudor/manipulation-de-facteurs-avec-forcats/)

Chapter 4

Une variable qualitative

Les données qualitatives peuvent faire l'objet de dénombrement, en effectif ou en proportion.

Petit rappel : cours en ligne et liens avec Excel.

Pour décrire une variable qualitative, on calcule :

- Le nombre d'occurrence de chacune des modalités dans la base (ex : le nombre de communes de chaque type ZAU) : N_i où $i \in \{1 \dots k\}$ représente l'ensemble des modalités
- La proportion (ou fréquence) de chacune des modalités : $f_i = \frac{N_i}{N}$

On peut calculer des effectifs pondérés (exemple : la population des communes de chaque type ZAU). Dans ce cas $N_i = \sum_c w_i \cdot \mathbb{I}_{c=i}$ et $N = \sum_c w_i$, où c est l'ensemble des communes.

Pour un premier coup d'œil à une variable qualitative, on peut utiliser la fonction générique `summary()`.

```
select (dat, ZAU) %>%
  summary ()  
  
##      ZAU
## 112 - Couronne d'un grand p\xf4le          :12297
## 400 - Commune isol\xe9e hors influence des p\xf4les: 7383
## 300 - Autre commune multipolaris\xe9e        : 7021
## 120 - Multipolaris\xe9e des grandes aires urbaines : 3962
## 111 - Grand p\xf4le (plus de 10 000 emplois)    : 3285
## 221 - Petit p\xf4le (de 1 500 \xe0 5 000 emplois)   :  888
## (Other)                                         : 1853
```

4.1 Tableaux de synthèse

4.1.1 Comptage

On cherche le nombre d'occurrence de chacune des modalités. Cela s'effectue grâce à la fonction `table()`.

```
pull (dat, ZAU) %>%
  table ()
```

```
## .
##      111 - Grand p\xf4le (plus de 10 000 emplois)
##                                         3285
##      112 - Couronne d'un grand p\xf4le
##                                         12297
## 120 - Multipolaris\xe9e des grandes aires urbaines
##                                         3962
##      211 - Moyen p\xf4le (5 000 \xe0 10 000 emplois)
##                                         456
##      212 - Couronne d'un moyen p\xf4le
##                                         815
##      221 - Petit p\xf4le (de 1 500 \xe0 5 000 emplois)
##                                         888
##      222 - Couronne d'un petit p\xf4le
##                                         582
##      300 - Autre commune multipolaris\xe9e
##                                         7021
## 400 - Commune isol\xe9e hors influence des p\xf4les
##                                         7383
```

La fonction `DT::datatable()` permet une mise en forme moins austère des dataframes.

```
pull (dat, ZAU) %>%
  table () %>%
  as.data.frame () %>%
  datatable (rownames = FALSE, colnames = c ("ZAU", "Nombre"))
```

Show 10 entries Search:

| ZAU | Nombre |
|--|--------|
| 111 - Grand pôle (plus de 10 000 emplois) | 3285 |
| 112 - Couronne d'un grand pôle | 12297 |
| 120 - Multipolarise des grandes aires urbaines | 3962 |
| 211 - Moyen pôle (5 000 à 10 000 emplois) | 456 |
| 212 - Couronne d'un moyen pôle | 815 |
| 221 - Petit pôle (de 1 500 à 5 000 emplois) | 888 |
| 222 - Couronne d'un petit pôle | 582 |
| 300 - Autre commune multipolarise | 7021 |
| 400 - Commune isolée hors influence des pôles | 7383 |

Showing 1 to 9 of 9 entries Previous 1 Next

4.1.2 Comptage pondéré

On ne cherche plus à afficher le nombre d'occurrence de chaque modalités, mais à connaître le *poids* d'une variable sur ces modalités. Il s'obtient avec la fonction `xtabs()`. Par exemple, si on veut connaître la population pour chaque ZAU :

```
xtabs (formula = P14_POP ~ ZAU, data = dat) %>%
  as.data.frame () %>%
  datatable (rownames = FALSE, colnames = c ("ZAU", "Population"))
```

Show 10 entries Search:

| ZAU | Population |
|--|------------|
| 111 - Grand pôle (plus de 10 000 emplois) | 38738103 |
| 112 - Couronne d'un grand pôle | 12437519 |
| 120 - Multipolarise des grandes aires urbaines | 3414201 |
| 211 - Moyen pôle (5 000 à 10 000 emplois) | 1931948 |
| 212 - Couronne d'un moyen pôle | 359717 |
| 221 - Petit pôle (de 1 500 à 5 000 emplois) | 2422164 |
| 222 - Couronne d'un petit pôle | 167718 |
| 300 - Autre commune multipolarise | 3415222 |
| 400 - Commune isolée hors influence des pôles | 3020568 |

Showing 1 to 9 of 9 entries Previous 1 Next

4.1.3 Fréquences

On souhaite maintenant connaître le pourcentage de communes que représente chaque ZAU. Cela se fait grâce à la fonction `table()` suivi cette fois de `prop.table()`. Par exemple :

```
pull (dat, ZAU) %>%
  table () %>%
  prop.table () %>%
  round (3) %>%
  as.data.frame () %>%
  datatable (rownames = FALSE, colnames = c ("ZAU", "Fréquence")) %>%
  formatPercentage ('Freq', 1)
```

| ZAU | Fréquence |
|--|-----------|
| 111 - Grand pôle (plus de 10 000 emplois) | 9.0% |
| 112 - Couronne d'un grand pôle | 33.5% |
| 120 - Multipolarise des grandes aires urbaines | 10.8% |
| 211 - Moyen pôle (5 000 à 10 000 emplois) | 1.2% |
| 212 - Couronne d'un moyen pôle | 2.2% |
| 221 - Petit pôle (de 1 500 à 5 000 emplois) | 2.4% |
| 222 - Couronne d'un petit pôle | 1.6% |
| 300 - Autre commune multipolarise | 19.1% |
| 400 - Commune isolée hors influence des pôles | 20.1% |

Showing 1 to 9 of 9 entries

Previous 1 Next

4.1.4 Fréquences pondérées

On souhaite finalement connaître le pourcentage de population que représente chaque ZAU. En respectant la même logique que précédemment : cela se fait grâce à la fonction `xtabs()` suivi de `prop.table()`. Par exemple :

```
xtabs (formula = P14_POP ~ ZAU, data = dat) %>%
  prop.table () %>%
  round (3) %>%
  as.data.frame () %>%
  datatable (rownames = FALSE, colnames = c("ZAU", "% de la Population")) %>%
  formatPercentage ('Freq', 1)
```

| ZAU | % de la Population |
|--|--------------------|
| 111 - Grand pôle (plus de 10 000 emplois) | 58.8% |
| 112 - Couronne d'un grand pôle | 18.9% |
| 120 - Multipolarise des grandes aires urbaines | 5.2% |
| 211 - Moyen pôle (5 000 à 10 000 emplois) | 2.9% |
| 212 - Couronne d'un moyen pôle | 0.5% |
| 221 - Petit pôle (de 1 500 à 5 000 emplois) | 3.7% |
| 222 - Couronne d'un petit pôle | 0.3% |
| 300 - Autre commune multipolarise | 5.2% |
| 400 - Commune isolée hors influence des pôles | 4.6% |

Show 10 entries Search:

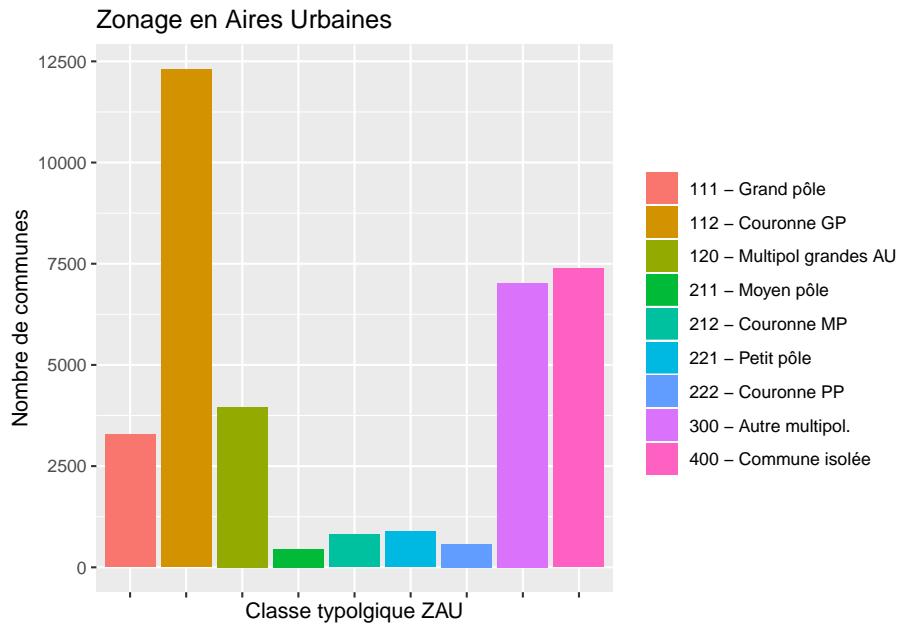
Showing 1 to 9 of 9 entries Previous 1 Next

4.2 Graphiques

4.2.1 Diagramme en barres (ou en bâtons)

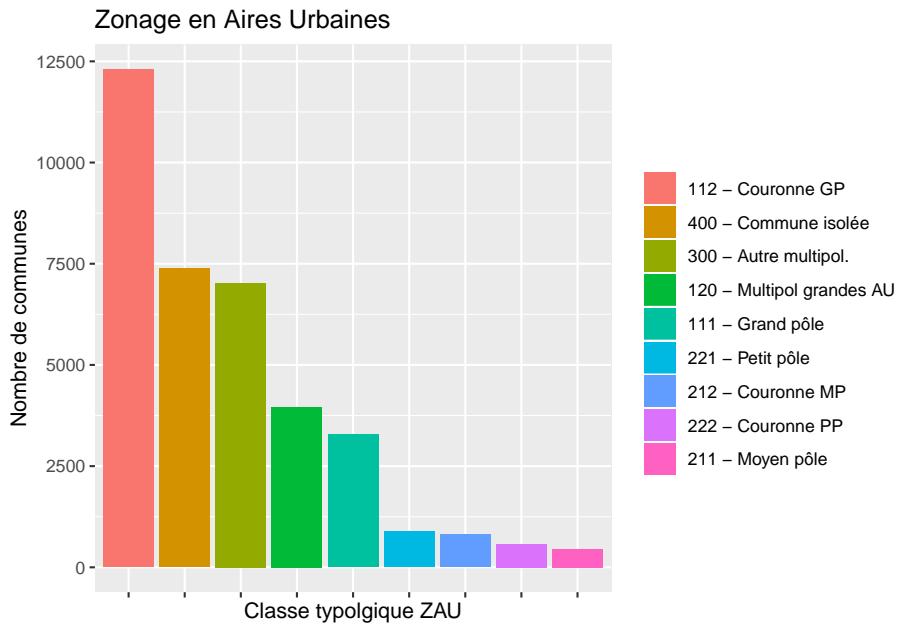
On souhaite connaître le nombre de communes dans chaque ZAU. Grâce à notre base communale, on cherche donc le nombre de lignes pour chaque ZAU.

```
ggplot (dat, aes (x = ZAU2, fill = ZAU2)) +
  geom_bar () +
  ggtitle ("Zonage en Aires Urbaines") +
  xlab (label = "Classe typologique ZAU") +
  ylab (label = "Nombre de communes") +
  theme (axis.text.x = element_blank ()) +
  theme(legend.title = element_blank())
```



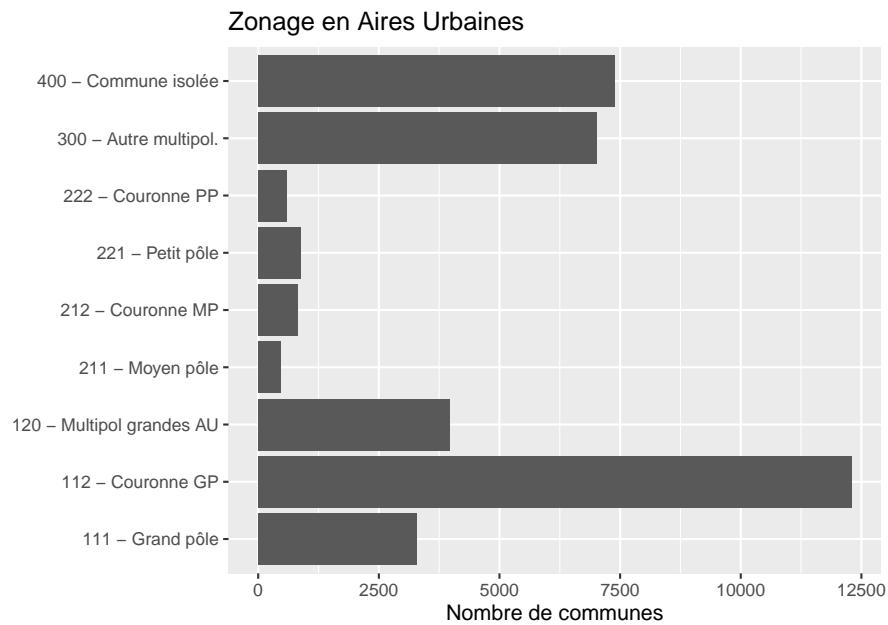
Ceci n'est pas un histogramme ! C'est un diagramme en barres. La lisibilité est compliquée en raison de l'ordre des facteurs. Grâce aux fonctions vues précédemment, on peut modifier cet ordre :

```
ggplot (dat, aes (x = fct_infreq (ZAU2), fill = fct_infreq (ZAU2))) +
  geom_bar () +
  ggtitle ("Zonage en Aires Urbaines") +
  xlab (label = "Classe typologique ZAU") +
  ylab (label = "Nombre de communes") +
  theme (axis.text.x = element_blank ()) +
  theme(legend.title = element_blank())
```



D'autres présentations sont possibles. Par exemple, avec les intitulés des modalités sur l'axe (obtenu grâce à `coord_flip()`, il n'y donc plus besoin de différencier par les couleurs, ni de la légende. La disposition en barres horizontales permet d'afficher ces intitulés longs.

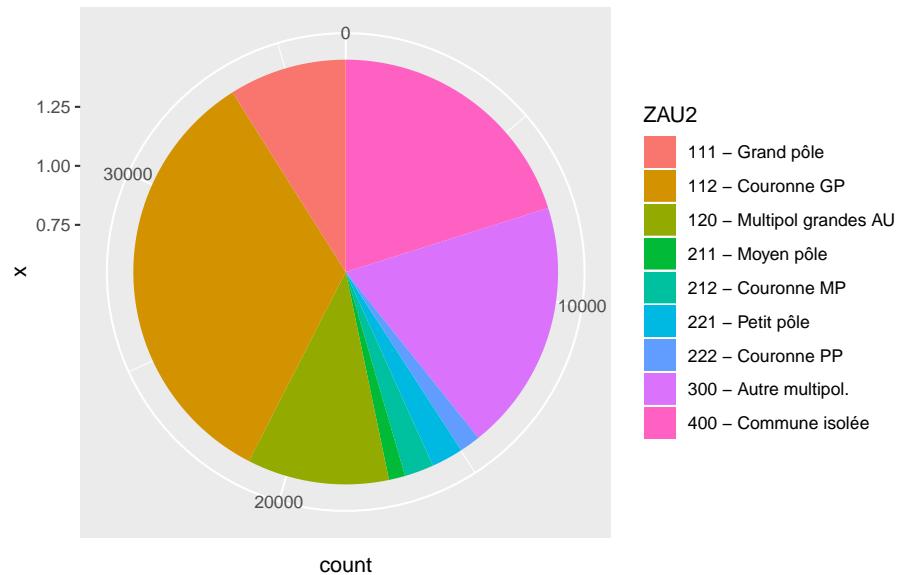
```
ggplot (dat, aes (x = ZAU2)) +
  geom_bar () +
  ggtitle ("Zonage en Aires Urbaines") +
  ylab (label = "Nombre de communes") +
  xlab ("") +
  theme (legend.position = "none") +
  coord_flip ()
```



4.2.2 Diagramme en secteurs

Cette forme est à éviter autant que possible car sa lecture est moins facile que celle des diagrammes en barres. Elle s'obtient grâce à `coord_polar()`.

```
ggplot (dat, aes (x = 1, fill = ZAU2)) +
  geom_bar () +
  coord_polar (theta = "y")
```



Chapter 5

Deux variables quantitatives

On cherche à quantifier le lien entre deux variables quantitatives X et Y sur un échantillon de n individus. Par exemple : le lien entre les revenus médians et le nombre d'emploi au lieu de travail des communes.

Plusieurs coefficient permettent de quantifier ce lien :

- Coefficient de corrélation linéaire (Pearson) : $\rho_{XY} = \frac{\sigma_{XY}}{\sigma_X \cdot \sigma_Y}$ avec $\sigma_{XY} = \frac{1}{n} \sum_{i=1}^n (x_i - \bar{x})(y_i - \bar{y})$ et σ_x l'écart-type de X.
- Coefficient de corrélation des rangs (Spearman) : c'est le coefficient précédent appliqué aux rangs \Rightarrow détecte des relations non linéaires

Dans certains cas (distributions asymétriques), il faut penser à appliquer la transformation logarithmique

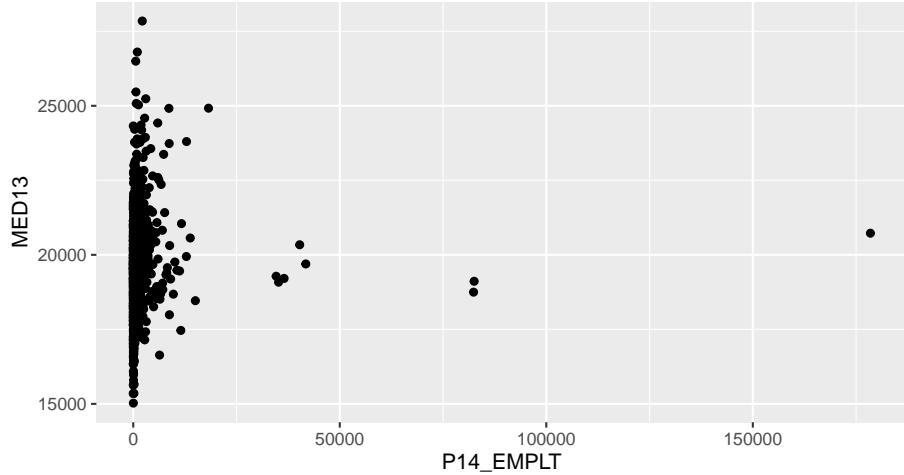
5.1 Représentations graphiques

Des représentations graphiques permettent de visualiser d'éventuels effets entre variables, qu'il faudra ensuite étudier plus précisément en les quantifier.

5.1.1 Nuage de points

Un croisement de deux variables quantitatives peut se faire facilement avec un nuage simple (`geom_point()`):

```
dat_pays_loire <- dat %>% filter(REG == "52")
ggplot (data = dat_pays_loire, aes (x = P14_EMPLT, y = MED13)) +
  geom_point ()
## Warning: Removed 159 rows containing missing values (geom_point).
```



On peut ajouter un peu de mise en forme :

```
g <- ggplot (data = dat_pays_loire, aes (x = P14_EMPLT, y = MED13)) +
  geom_point (colour = "blue", cex = 0.2) +
  ggtitle ("Revenu médian en fonction du nombre d'emplois") +
  ylab ("Revenu médian") +
  xlab ("Nombre d'emplois") +
  scale_x_continuous (trans = 'log10',
                      breaks = c(0,1,10,100,1000, 10000, 100000, 1000000),
                      labels = function(x) format(x, big.mark = " ", scientific = FALSE))
  scale_y_continuous (labels = function(x) paste (format (x, big.mark = " ",
                                                       scientific = FALSE), " €"))
g
## Warning: Removed 159 rows containing missing values (geom_point).

## Warning in grid.Call(C_textBounds, as.graphicsAnnot(x$label), x$x, x$y, :
## conversion failure on '15 000 €' in 'mbcsToSbcs': dot substituted for <e2>

## Warning in grid.Call(C_textBounds, as.graphicsAnnot(x$label), x$x, x$y, :
## conversion failure on '15 000 €' in 'mbcsToSbcs': dot substituted for <82>

## Warning in grid.Call(C_textBounds, as.graphicsAnnot(x$label), x$x, x$y, :
## conversion failure on '15 000 €' in 'mbcsToSbcs': dot substituted for <ac>

## Warning in grid.Call(C_textBounds, as.graphicsAnnot(x$label), x$x, x$y, :
```



```

## Warning in grid.Call(C_textBounds, as.graphicsAnnot(x$label), x$x, x$y, :
## conversion failure on '20 000 €' in 'mbcsToSbcs': dot substituted for <82>

## Warning in grid.Call(C_textBounds, as.graphicsAnnot(x$label), x$x, x$y, :
## conversion failure on '20 000 €' in 'mbcsToSbcs': dot substituted for <ac>

## Warning in grid.Call(C_textBounds, as.graphicsAnnot(x$label), x$x, x$y, :
## conversion failure on '25 000 €' in 'mbcsToSbcs': dot substituted for <e2>

## Warning in grid.Call(C_textBounds, as.graphicsAnnot(x$label), x$x, x$y, :
## conversion failure on '25 000 €' in 'mbcsToSbcs': dot substituted for <82>

## Warning in grid.Call(C_textBounds, as.graphicsAnnot(x$label), x$x, x$y, :
## conversion failure on '25 000 €' in 'mbcsToSbcs': dot substituted for <ac>

## Warning in grid.Call.graphics(C_text, as.graphicsAnnot(x$label), x$x, x
## $y, : conversion failure on '15 000 €' in 'mbcsToSbcs': dot substituted for
## <e2>

## Warning in grid.Call.graphics(C_text, as.graphicsAnnot(x$label), x$x, x
## $y, : conversion failure on '15 000 €' in 'mbcsToSbcs': dot substituted for
## <82>

## Warning in grid.Call.graphics(C_text, as.graphicsAnnot(x$label), x$x, x
## $y, : conversion failure on '15 000 €' in 'mbcsToSbcs': dot substituted for
## <ac>

## Warning in grid.Call.graphics(C_text, as.graphicsAnnot(x$label), x$x, x
## $y, : conversion failure on '20 000 €' in 'mbcsToSbcs': dot substituted for
## <e2>

## Warning in grid.Call.graphics(C_text, as.graphicsAnnot(x$label), x$x, x
## $y, : conversion failure on '20 000 €' in 'mbcsToSbcs': dot substituted for
## <82>

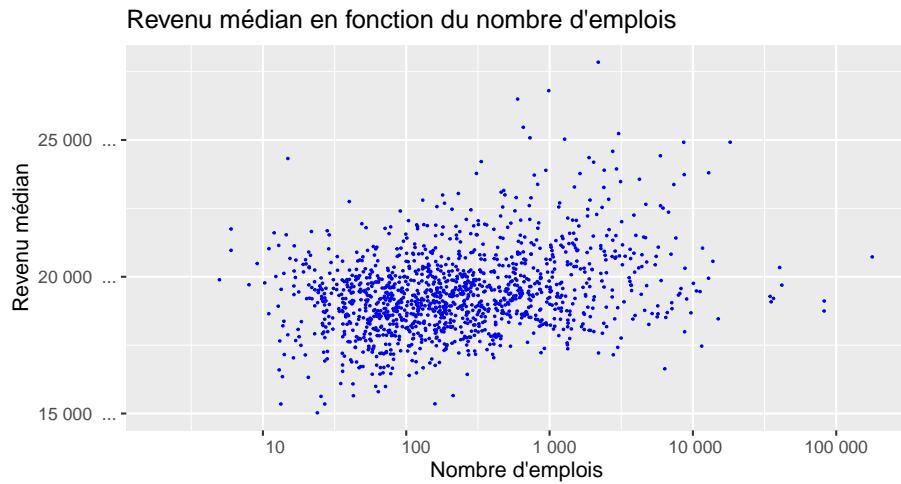
## Warning in grid.Call.graphics(C_text, as.graphicsAnnot(x$label), x$x, x
## $y, : conversion failure on '20 000 €' in 'mbcsToSbcs': dot substituted for
## <ac>

## Warning in grid.Call.graphics(C_text, as.graphicsAnnot(x$label), x$x, x
## $y, : conversion failure on '25 000 €' in 'mbcsToSbcs': dot substituted for
## <e2>

## Warning in grid.Call.graphics(C_text, as.graphicsAnnot(x$label), x$x, x
## $y, : conversion failure on '25 000 €' in 'mbcsToSbcs': dot substituted for
## <82>

## Warning in grid.Call.graphics(C_text, as.graphicsAnnot(x$label), x$x, x
## $y, : conversion failure on '25 000 €' in 'mbcsToSbcs': dot substituted for
## <ac>

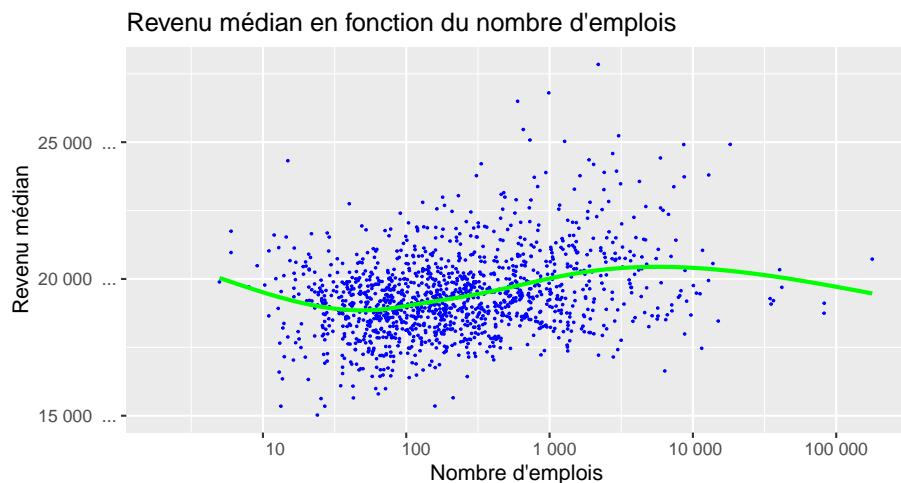
```



Les fonctions `scale_._continuous()` permettent de modifier les échelles des axes. Ici, nous avons effectué une transformation logarithmique de l'axe des X.

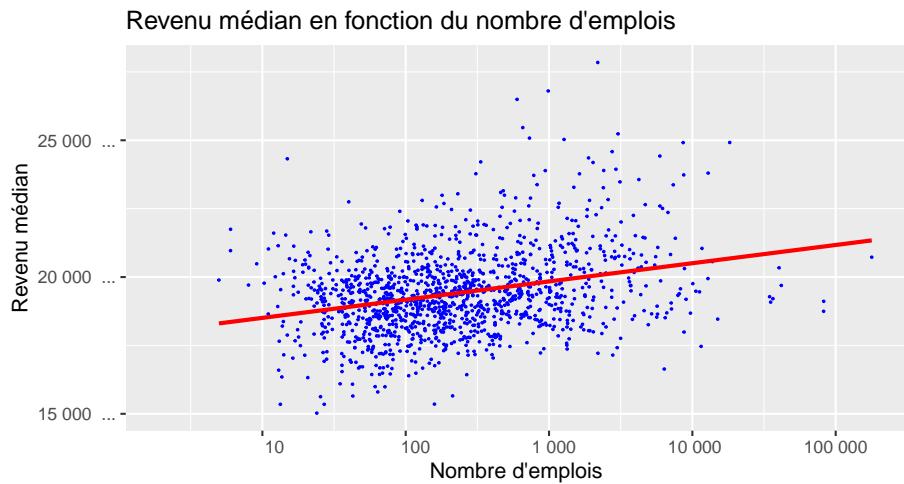
Pour visualiser, sans *a priori* sur sa forme, la relation entre X et Y (elle n'est pas forcément linéaire) on peut appliquer une fonction de lissage avec `geom_smooth`.

```
g + geom_smooth (se = FALSE, color = 'green')
```



Si on veut la droite de régression, c'est aussi la fonction `geom_smooth` en ajoutant l'argument `method = "lm"`.

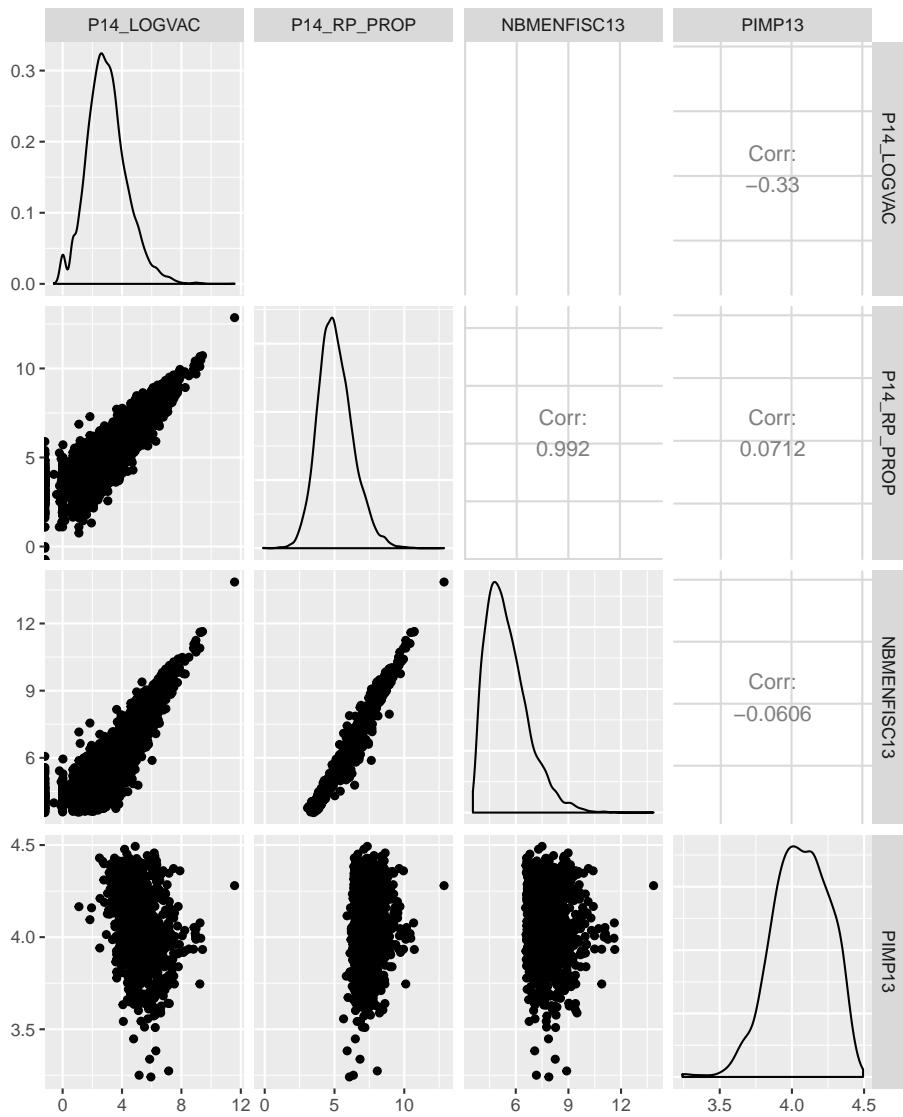
```
g + geom_smooth (method = "lm", se = FALSE, color = 'red')
```



5.1.2 Matrice de nuages de points

Une matrice permet d'afficher plusieurs nuages de points d'un coup. Cela est pratique quand on explore un jeu de données avec différentes variables susceptibles d'être corrélées entre elles. Elle s'obtient avec la fonction `ggpairs()` (du package `GGally`).

```
num <- select (dat, P14_LOGVAC:PIMP13) %>%
  sample_n (10000) %>%
  log ()
ggpairs (num) # package "GGally" nécessaire
```



5.2 Coefficients de corrélation

Le **coefficient de corrélation** entre deux variables quantifie l'intensité de la liaison entre ces variables. Il varie entre +1 et -1. S'il est proche de zéro, il n'y a pas de liaison, les variables sont indépendantes l'une de l'autre. S'il est de signe positif, les variables ont tendance à varier dans le même sens, c'est-à-dire que quand l'une augmente, l'autre tend aussi à augmenter. S'il est négatif, quand

l'une augmente l'autre tend à diminuer.

Le coefficient de corrélation *linéaire* classique, par défaut dans la fonction `cor()` est celui de Pearson. Attention, il est très sensible aux valeurs extrêmes et il ne rend pas bien compte des relations non linéaires.

Il entre dans la catégorie des outils paramétriques, donc attention à son interprétation quand les variables ne sont pas distribuées selon une loi normale.

$$r = \frac{Cov(X, Y)}{\sigma_X \times \sigma_Y}$$

```
cor (na.omit (num))
```

```
##          P14_LOGVAC P14_RP_PROP NBMENFISC13      PIMP13
## P14_LOGVAC    1.0000000  0.77856461  0.85282185 -0.32965564
## P14_RP_PROP   0.7785646  1.00000000  0.96351460  0.07121211
## NBMENFISC13   0.8528219  0.96351460  1.00000000 -0.06146151
## PIMP13        -0.3296556  0.07121211 -0.06146151  1.00000000
cor (na.omit (log (num)))
```

```
##          P14_LOGVAC P14_RP_PROP NBMENFISC13      PIMP13
## P14_LOGVAC    1.0000000  0.71718454  0.79910611 -0.34343271
## P14_RP_PROP   0.7171845  1.00000000  0.95855944  0.07587788
## NBMENFISC13   0.7991061  0.95855944  1.00000000 -0.06772624
## PIMP13        -0.3434327  0.07587788 -0.06772624  1.00000000
```

Pour tester si vous arrivez bien à estimer à l'oeil l'intensité du lien linéaire entre deux variables, un peu d'entraînement sur cette appli shiny ou celle-ci.

Quand la relation n'est pas linéaire, on peut utiliser en alternative le coefficient de corrélation de Spearman. Celui-ci est basé sur les rangs. Il permet donc de mettre en évidence des relations non linéaires (pour peu qu'elles soient monotones) entre les deux variables.

```
cor (na.omit (num), method = "spearman")
```

```
##          P14_LOGVAC P14_RP_PROP NBMENFISC13      PIMP13
## P14_LOGVAC    1.0000000  0.68444428  0.79551319 -0.37853187
## P14_RP_PROP   0.6844443  1.00000000  0.94334127  0.06173166
## NBMENFISC13   0.7955132  0.94334127  1.00000000 -0.07941188
## PIMP13        -0.3785319  0.06173166 -0.07941188  1.00000000
```

Quelques exemples de comparaison entre les corrélations de Pearson et de Spearman ici

5.3 Régression

Le principe de la régression est de décrire la manière dont une variable est dépendante d'une autre variable. À la différence de la corrélation, où on cherche à quantifier la manière dont 2 variables s'orientent l'une par rapport à l'autre, la régression a pour objectif d'estimer les valeurs d'une variable en connaissant les valeurs d'une autre variable. Le modèle usuel de régression linéaire est : $y = \beta_0 + \beta_1 x + \varepsilon$, avec :

- y variable à expliquer
- x variable explicative
- ε un terme d'erreur aléatoire de loi normale, d'espérance nulle et d'écart-type .

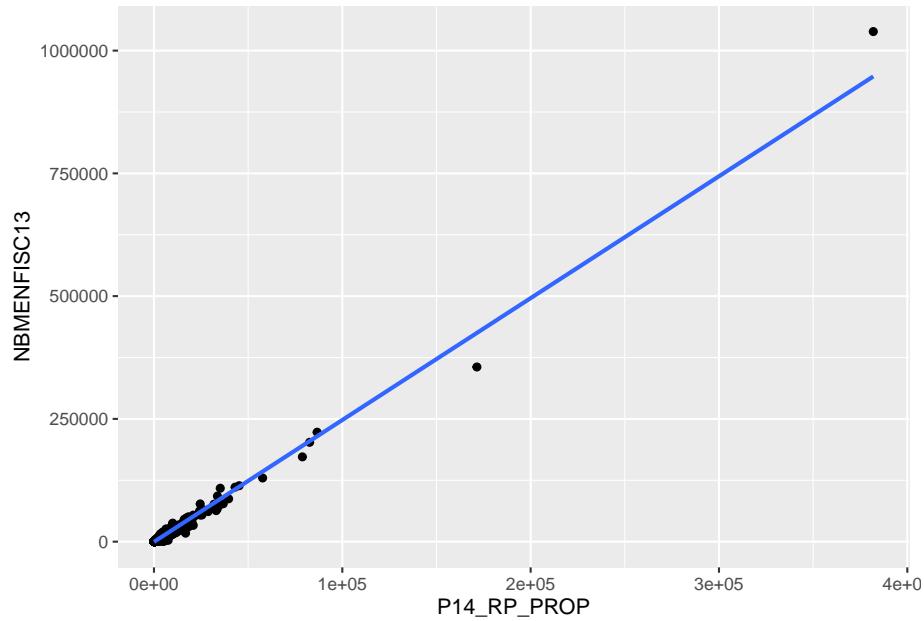
On veut une estimation des coefficients β_0 et β_1 . Sur R, on se sert de la fonction `lm()` pour créer un modèle ("linear model"). Cet objet permettra d'accéder à plusieurs informations de la régression.

```
model <- lm(formula = NBMENFISC13 ~ P14_RP_PROP, data = dat)
summary(model)

##
## Call:
## lm(formula = NBMENFISC13 ~ P14_RP_PROP, data = dat)
##
## Residuals:
##    Min     1Q Median     3Q    Max
## -69374   -22    210    312  91368
##
## Coefficients:
##             Estimate Std. Error t value Pr(>|t|)
## (Intercept) -4.199e+02  5.955e+00 -70.52   <2e-16 ***
## P14_RP_PROP  2.482e+00  2.071e-03 1198.05   <2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 1052 on 32187 degrees of freedom
##   (4500 observations deleted due to missingness)
## Multiple R-squared:  0.9781, Adjusted R-squared:  0.9781
## F-statistic: 1.435e+06 on 1 and 32187 DF,  p-value: < 2.2e-16
```

On peut encore une fois visualiser la droite de régression grâce à la fonction `geom_smooth()`

```
ggplot(data = dat, aes(x = P14_RP_PROP, y = NBMENFISC13)) +
  geom_point() +
  geom_smooth(method = "lm", se = FALSE)
```



5.3.1 Exercice

Effectuer une nouvelle régression avec 2 autres variables, puis tracer le nuage de points de ces variables, et enfin y ajouter la droite de regression.

Chapter 6

Deux variables qualitatives

6.1 Définitions

- On calcule les effectifs de chaque couple possible de modalités i et j. $i \in \{1 \dots k\}, j \in \{1 \dots r\}$. Exemple : le nombre de communes du type ZAU i et de la région j.
- $N_{ij} = \sum_c \mathbb{I}_{c=i} \cdot \mathbb{I}_{c=j}$ (nombre de communes du type ZAU i et de la région j)
- $N_{i \cdot} = \sum_c \mathbb{I}_{c=i}$ et $N_{\cdot j} = \sum_c \mathbb{I}_{c=j}$ sont appelées effectifs marginaux (nombre de communes du type ZAU i, puis nombre de communes de la région j)
- $f_{ij} = \frac{N_{ij}}{N}$: la fréquence
- $f_{ij}^* = \frac{N_{ij}}{N_i}$ les profils ligne : la part de communes de type i et de la région j parmi les communes du type i
- $f_{ij}^{**} = \frac{N_{ij}}{N_{\cdot j}}$ les profils colonne : la part de communes de type i et de la région j parmi les communes de la région j

6.2 Tableaux de synthèse

6.2.1 Tableau de contingence (effectifs)

Le tableau à double entrée, avec les lignes et colonnes correspondant aux modalités des variables qui sont croisées est appelé tableau de contingence. Les cellules contiennent le nombre d'occurrences correspondant à chaque case.

```
tab <- select (dat, ZAU2, REG) %>%
  filter (REG %in% c("11", "24", "27", "52")) %>%
  droplevels () %>%
  table ()
```

```
as.data.frame (tab) %>%
  spread (key = REG, value = Freq) %>%
  datatable (caption = "VENTILATION DES COMMUNES PAR REGION ET PAR CLASSE DE ZAU")
```

Show 10 entries Search:

VENTILATION DES COMMUNES PAR REGION ET PAR CLASSE DE ZAU

| | ZAU2 | 11 | 24 | 27 | 52 |
|---|---------------------------|-----|-----|------|-----|
| 1 | 111 - Grand pôle | 413 | 103 | 140 | 108 |
| 2 | 112 - Couronne GP | 853 | 734 | 1299 | 535 |
| 3 | 120 - Multipol grandes AU | 3 | 188 | 336 | 186 |
| 4 | 211 - Moyen pôle | 3 | 30 | 31 | 23 |
| 5 | 212 - Couronne MP | 2 | 72 | 122 | 44 |
| 6 | 221 - Petit pôle | 0 | 46 | 60 | 69 |
| 7 | 222 - Couronne PP | 0 | 19 | 137 | 8 |
| 8 | 300 - Autre multipol. | 7 | 375 | 737 | 386 |
| 9 | 400 - Commune isolée | 0 | 275 | 969 | 143 |

Showing 1 to 9 of 9 entries Previous Next

La fonction `table()` fonctionne aussi au-delà de 2 variables, à l'image des tableaux croisés dynamiques dans Excel®, ou de la PROC FREQ en SAS®.

6.2.2 Tableau des fréquences

Le tableau des fréquences est obtenu, comme auparavant, avec la fonction `prop.table()`.

```
freq <- tab %>% prop.table() %>% round(digits = 3) * 100
addmargins(freq) # permet d'ajouter les fréquences totales pour chaque lignes et colonnes
```

| | REG | 11 | 24 | 27 | 52 | Sum |
|------------------------------|-----|------|------|------|------|-------|
| ## ZAU2 | | 11 | 24 | 27 | 52 | Sum |
| ## 111 - Grand pôle | | 4.9 | 1.2 | 1.7 | 1.3 | 9.1 |
| ## 112 - Couronne GP | | 10.1 | 8.7 | 15.4 | 6.3 | 40.5 |
| ## 120 - Multipol grandes AU | | 0.0 | 2.2 | 4.0 | 2.2 | 8.4 |
| ## 211 - Moyen pôle | | 0.0 | 0.4 | 0.4 | 0.3 | 1.1 |
| ## 212 - Couronne MP | | 0.0 | 0.9 | 1.4 | 0.5 | 2.8 |
| ## 221 - Petit pôle | | 0.0 | 0.5 | 0.7 | 0.8 | 2.0 |
| ## 222 - Couronne PP | | 0.0 | 0.2 | 1.6 | 0.1 | 1.9 |
| ## 300 - Autre multipol. | | 0.1 | 4.4 | 8.7 | 4.6 | 17.8 |
| ## 400 - Commune isolée | | 0.0 | 3.3 | 11.5 | 1.7 | 16.5 |
| ## Sum | | 15.1 | 21.8 | 45.4 | 17.8 | 100.1 |

```
addmargins (freq, 1) # sommes en colonnes
```

| | REG | 11 | 24 | 27 | 52 |
|------------------------------|-----|------|------|------|------|
| ## ZAU2 | | 11 | 24 | 27 | 52 |
| ## 111 - Grand pôle | | 4.9 | 1.2 | 1.7 | 1.3 |
| ## 112 - Couronne GP | | 10.1 | 8.7 | 15.4 | 6.3 |
| ## 120 - Multipol grandes AU | | 0.0 | 2.2 | 4.0 | 2.2 |
| ## 211 - Moyen pôle | | 0.0 | 0.4 | 0.4 | 0.3 |
| ## 212 - Couronne MP | | 0.0 | 0.9 | 1.4 | 0.5 |
| ## 221 - Petit pôle | | 0.0 | 0.5 | 0.7 | 0.8 |
| ## 222 - Couronne PP | | 0.0 | 0.2 | 1.6 | 0.1 |
| ## 300 - Autre multipol. | | 0.1 | 4.4 | 8.7 | 4.6 |
| ## 400 - Commune isolée | | 0.0 | 3.3 | 11.5 | 1.7 |
| ## Sum | | 15.1 | 21.8 | 45.4 | 17.8 |

```
addmargins (freq, 2) # sommes en lignes
```

| | REG | 11 | 24 | 27 | 52 | Sum |
|------------------------------|-----|------|-----|------|-----|------|
| ## ZAU2 | | 11 | 24 | 27 | 52 | Sum |
| ## 111 - Grand pôle | | 4.9 | 1.2 | 1.7 | 1.3 | 9.1 |
| ## 112 - Couronne GP | | 10.1 | 8.7 | 15.4 | 6.3 | 40.5 |
| ## 120 - Multipol grandes AU | | 0.0 | 2.2 | 4.0 | 2.2 | 8.4 |
| ## 211 - Moyen pôle | | 0.0 | 0.4 | 0.4 | 0.3 | 1.1 |
| ## 212 - Couronne MP | | 0.0 | 0.9 | 1.4 | 0.5 | 2.8 |
| ## 221 - Petit pôle | | 0.0 | 0.5 | 0.7 | 0.8 | 2.0 |
| ## 222 - Couronne PP | | 0.0 | 0.2 | 1.6 | 0.1 | 1.9 |
| ## 300 - Autre multipol. | | 0.1 | 4.4 | 8.7 | 4.6 | 17.8 |
| ## 400 - Commune isolée | | 0.0 | 3.3 | 11.5 | 1.7 | 16.5 |

Ici, on peut voir que, dans les communes que nous avons filtré :

- environ 10% des communes sont dans la région 11 ET dans une couronne de grand pôle
- il y a 16,5% des communes qui sont isolées
- la région 27 totalise plus de 45% des communes

6.2.3 Profils-ligne

Grâce à l'argument *margin* de la fonction `prop.table()`, on peut afficher les profils-ligne (avec *margin = 1*). On regarde ainsi la proportion des régions pour chaque ZAU.

```
freq <- tab %>% prop.table(margin = 1) %>% round(digits = 3) * 100
addmargins (freq)
```

| | REG | 11 | 24 | 27 | 52 | Sum |
|---------|-----|----|----|----|----|-----|
| ## ZAU2 | | 11 | 24 | 27 | 52 | Sum |

```

##   111 - Grand pôle      54.1 13.5 18.3 14.1 100.0
##   112 - Couronne GP     24.9 21.5 38.0 15.6 100.0
##   120 - Multipol grandes AU 0.4 26.4 47.1 26.1 100.0
##   211 - Moyen pôle      3.4 34.5 35.6 26.4 99.9
##   212 - Couronne MP      0.8 30.0 50.8 18.3 99.9
##   221 - Petit pôle      0.0 26.3 34.3 39.4 100.0
##   222 - Couronne PP      0.0 11.6 83.5 4.9 100.0
##   300 - Autre multipol.  0.5 24.9 49.0 25.6 100.0
##   400 - Commune isolée   0.0 19.8 69.9 10.3 100.0
##   Sum                      84.1 208.5 426.5 180.7 899.8

```

Ici, on voit que, parmi les communes que nous avons filtré :

- 70% des communes classées comme “Commune isolée” sont dans la région 27
- 54% des communes classées “Grand pôle” sont dans la région 11

6.2.4 Profils-colonne

Pour obtenir les profils-colonnes, on se sert de *margin = 2*. On regarde alors la proportion de ZAU pour chaque région.

```
freq <- tab %>% prop.table(margin = 2) %>% round(digits = 3) * 100
addmargins (freq)
```

| | REG | | | | |
|------------------------------|------|------|-------|------|-------|
| ## ZAU2 | 11 | 24 | 27 | 52 | Sum |
| ## 111 - Grand pôle | 32.2 | 5.6 | 3.7 | 7.2 | 48.7 |
| ## 112 - Couronne GP | 66.6 | 39.8 | 33.9 | 35.6 | 175.9 |
| ## 120 - Multipol grandes AU | 0.2 | 10.2 | 8.8 | 12.4 | 31.6 |
| ## 211 - Moyen pôle | 0.2 | 1.6 | 0.8 | 1.5 | 4.1 |
| ## 212 - Couronne MP | 0.2 | 3.9 | 3.2 | 2.9 | 10.2 |
| ## 221 - Petit pôle | 0.0 | 2.5 | 1.6 | 4.6 | 8.7 |
| ## 222 - Couronne PP | 0.0 | 1.0 | 3.6 | 0.5 | 5.1 |
| ## 300 - Autre multipol. | 0.5 | 20.4 | 19.2 | 25.7 | 65.8 |
| ## 400 - Commune isolée | 0.0 | 14.9 | 25.3 | 9.5 | 49.7 |
| ## Sum | 99.9 | 99.9 | 100.1 | 99.9 | 399.8 |

De la même manière que précédemment, parmi nos communes :

- 66,6% des communes de la région 11 sont classées comme “Courrone GP”
- 2,5% des communes de la région 24 sont classées comme “Petit pôle”

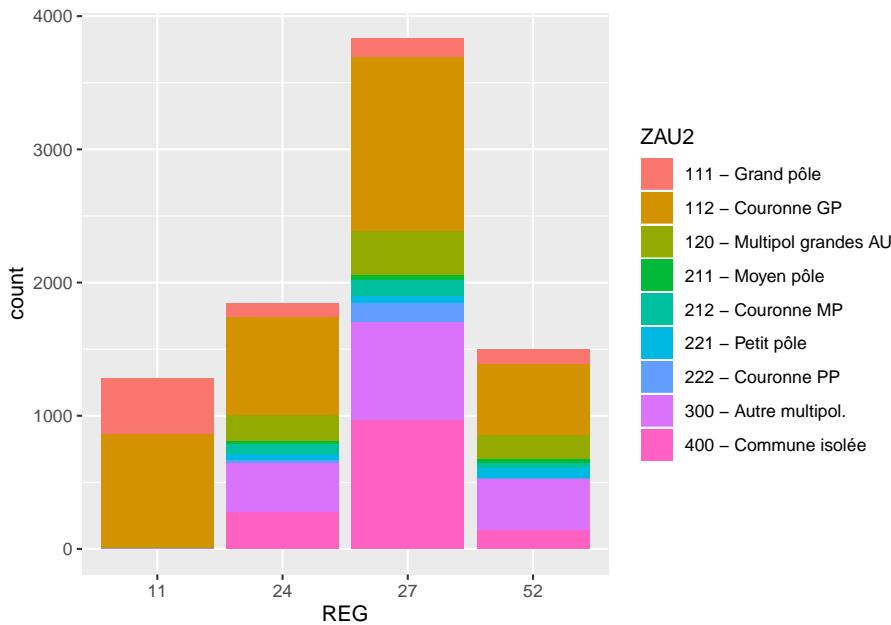
6.3 Graphiques

6.3.1 Diagramme en bâtons

Pour visualiser un croisement de deux variables qualitatives, il est possible de faire un *diagramme superposé*. La fonction est similaire aux diagrammes en bâtons déjà tracés. Cette fois, on précise dans *aes* que les régions sont sur l'axe, et que les différents ZAU sont colorés (argument *fill*).

```
dat4 <- dat %>%
  filter (REG %in% c("11", "24", "27", "52"))

bar <- ggplot (dat4, aes(x = REG, fill = ZAU2))
bar + geom_bar (position = "stack") # précise que les ZAU sont "empilées"
```

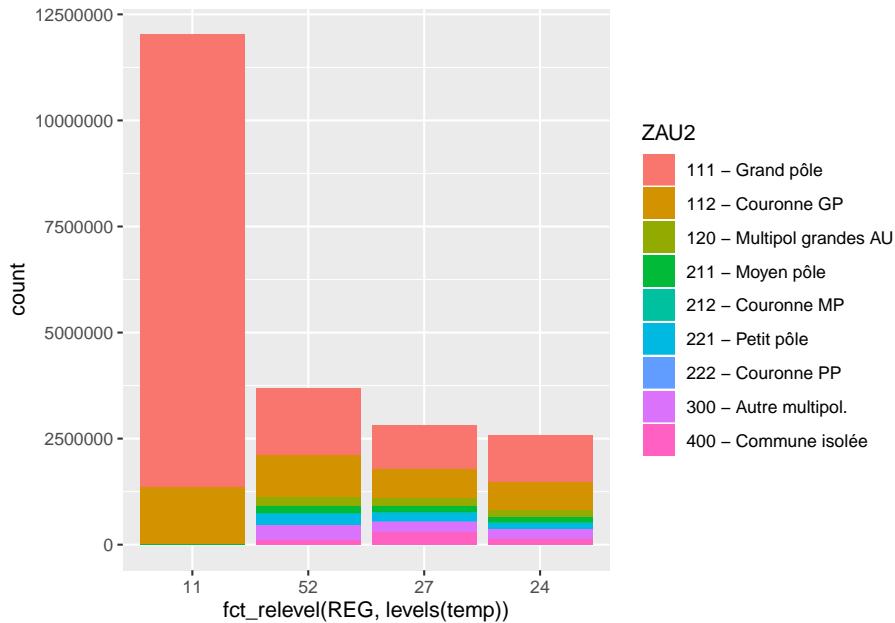


Si l'on veut ordonner les modalités par ordre décroissant de la hauteur totale des barres, c'est un peu plus compliqué :

```
# On calcule la population totale par région
prov <- dat4 %>%
  group_by (REG) %>%
  summarise (pop_reg = sum (P14_POP, na.rm = T))

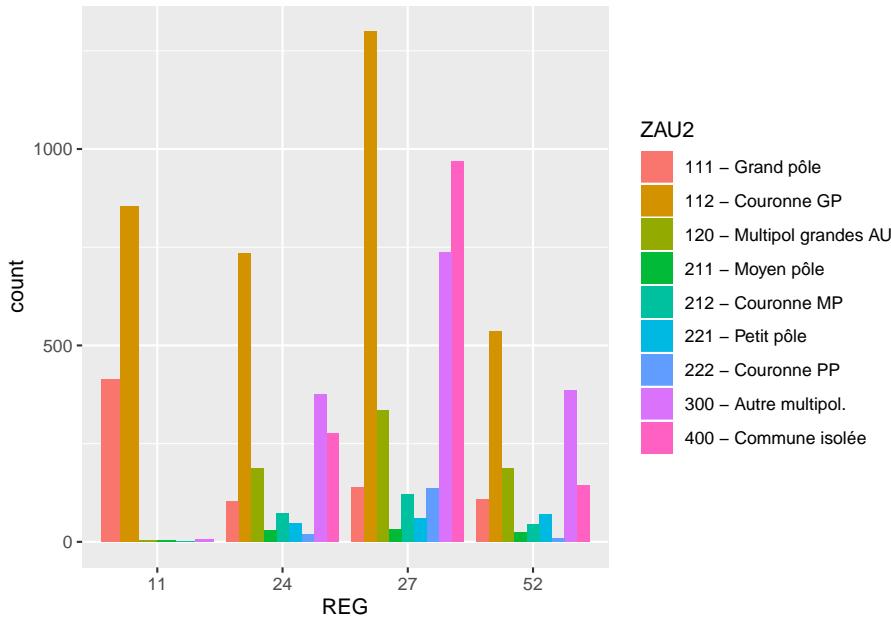
# On crée un vecteur contenant les modalités de REG par ordre décroissantes de population
temp <- fct_reorder (prov$REG, -prov$pop_reg)
```

```
# On produit le graphique en ordonnant les régions sur la base de leur ordre dans l'objet
ggplot (data = dat4, aes (x = fct_relevel (REG, levels (temp)),
                         fill = ZAU2,
                         weight = P14_POP)) +
  geom_bar (position = "stack")
```



Il est aussi possible de faire un *diagramme juxtaposé* en changeant la valeur *position* de la fonction *geom_bar* :

```
bar + geom_bar(position = "dodge") # précise que les ZAU sont "côte à côte"
```

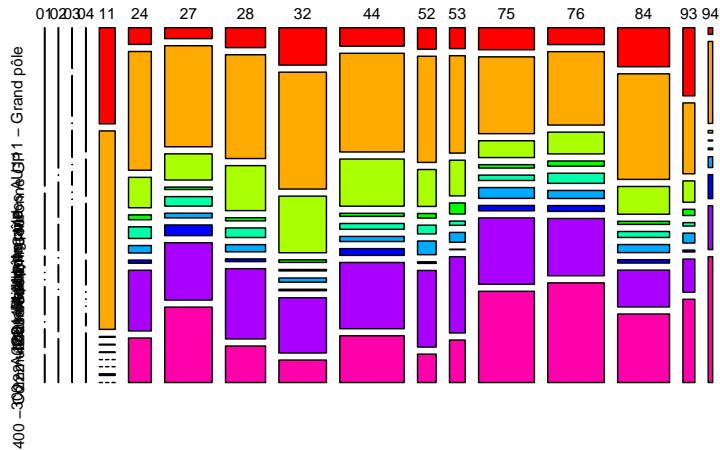


6.3.2 Graphique mosaïque

Pour afficher graphiquement les proportions de deux modalités, il est possible de faire un graphique mosaïque :

```
# On inverse colonnes et lignes pour avoir une représentation plus lisible
tab <- table(dat$REG, dat$ZAU2)
cols <- rainbow(nlevels(dat$ZAU2))
plot(tab, col = cols, main = "Répartition des communes par type ZAU et région")
```

Répartition des communes par type ZAU et région



```
#ggplot(dat) + geom_mosaic(aes(x = product(ZAU2, REG), fill = ZAU2)) # package ggmosaic
```

6.4 Avec pondération

Si on désire connaître le “poids” d’une variable suivant le croisement de 2 variables qualitatives, on peut créer un tableau pondéré. Si par exemple, on souhaite connaître la population pour chaque croisement REG / ZAU : on se sert de la fonction `xtabsen` précisant la *formula* qu’on souhaite (P14_POP par REG et ZAU).

```
dat <- mutate (dat, ZAU_COURT = as.factor (substr (ZAU, start = 1, stop = 3)))
tab <- xtabs (formula = P14_POP ~ REG + ZAU_COURT, data = dat)
tab
```

| | ZAU_COURT | | | | | | | |
|--------|-----------|---------|--------|--------|-------|--------|-------|--|
| ## REG | 111 | 112 | 120 | 211 | 212 | 221 | 222 | |
| ## 01 | 307294 | 59632 | 4149 | 0 | 0 | 15995 | 0 | |
| ## 02 | 299891 | 0 | 19388 | 33187 | 0 | 11311 | 0 | |
| ## 03 | 109538 | 17223 | 0 | 70037 | 0 | 10984 | 0 | |
| ## 04 | 623948 | 41639 | 71880 | 75100 | 0 | 0 | 0 | |
| ## 11 | 10659489 | 1349018 | 985 | 12927 | 757 | 0 | 0 | |
| ## 24 | 1088230 | 676997 | 142859 | 130552 | 41751 | 121185 | 7883 | |
| ## 27 | 1025799 | 689277 | 194488 | 119122 | 41438 | 172248 | 30457 | |
| ## 28 | 1443395 | 800329 | 286367 | 117695 | 40818 | 168795 | 10085 | |

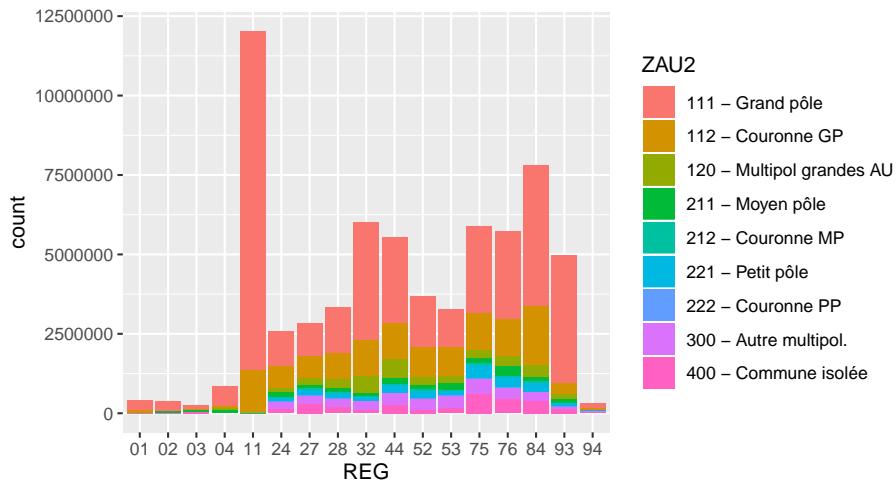
```

##   32  3709369  1124937  521043  108449    5885  129443  4876
##   44  2699363  1134602  621246  177829   26099  242055 26455
##   52  1575655  973415  225904  149431   37815  250004 3036
##   53  1169052  914047  244926  214774   17052  145728  786
##   75  2708020  1190761  225545  171744   40468  410303 33199
##   76  2748178  1179129  315400  293385   54187  302203 24407
##   84  4432062  1857298  374734  144472   43441  288745 15498
##   93  4003370  365184  164985  101619    7703  119567  2163
##   94  135450   64031     302   11625    2303  33598  8873
##      ZAU_COURT
## REG      300      400
## 01      8886     4230
## 02     11478     8656
## 03        0     44556
## 04      5950     24250
## 11      4389       0
## 24     223588    144390
## 27     249496    298298
## 28     279772    188389
## 32     286099    116055
## 44     368481    258515
## 52     372614    102959
## 53     387382    182796
## 75     494269    604835
## 76     348798    465066
## 84     282066    382650
## 93     81279     137568
## 94    10675     57355

```

Une fois l'objet `table` généré, tout est identique pour les tables. Il faut ajouter le paramètre `weight` pour les graphiques.

```
ggplot(data = dat, aes (REG, fill = ZAU2, weight = P14_POP)) +
  geom_bar(position = "stack")
```



Ici, la hauteur de chaque rectangle coloré représente la population totale des communes concernées par le croisement.

Chapter 7

Lien variable quantitative - variable qualitative

7.1 Statistiques en fonction d'un facteur

Par exemple, calculer la population totale (moyenne, médiane...) des communes pour chaque type ZAU.

```
## # A tibble: 9 x 4
##   ZAU          pop_moy dens_med nb_com
##   <fct>        <dbl>    <dbl>   <int>
## 1 "111 - Grand p\xf4le (plus de 10 000 emplois)" 11956.   4022.   3285
## 2 "112 - Couronne d'un grand p\xf4le"                1034.    556    12297
## 3 "120 - Multipolaris\xe9e des grandes aires urbaines"
## 4 "211 - Moyen p\xf4le (5 000 \xe0 10 000 emplois)"   4322.   1860     456
## 5 "212 - Couronne d'un moyen p\xf4le"                  455.    338     815
## 6 "221 - Petit p\xf4le (de 1 500 \xe0 5 000 emploi~"
## 7 "222 - Couronne d'un petit p\xf4le"                   293.    222     582
## 8 "300 - Autre commune multipolaris\xe9e"            500.    296    7021
## 9 "400 - Commune isol\xe9e hors influence des p\xf4les"
```

7.2 Eléments théoriques

- Soit X une variable continue, et $Y \in \{1, \dots, k\}$ une variable qualitatives à k modalités
- Dans chaque classe j : $\bar{X}_j = \mathbb{E}(X|Y = j)$ et $\sigma_j^2 = \mathbb{V}(X|Y = j)$

- Variance intraclasse : $V_{intra} = \frac{1}{n} \sum_{j=1}^k n_j \sigma_j^2$, moyenne (pondérée) des variances de chaque classe
- Variance interclasse : $V_{inter} = \frac{1}{n} \sum_{j=1}^k n_j (\bar{x}_j - \bar{x})^2$, variance (pondérée) des moyennes de chaque classe
- Rapport de corrélation : $\eta^2 = \frac{V_{inter}}{V_{Totale}} = \frac{V_{inter}}{V_{inter} + V_{intra}}$
- C'est le R^2 de l'anova de X par Y

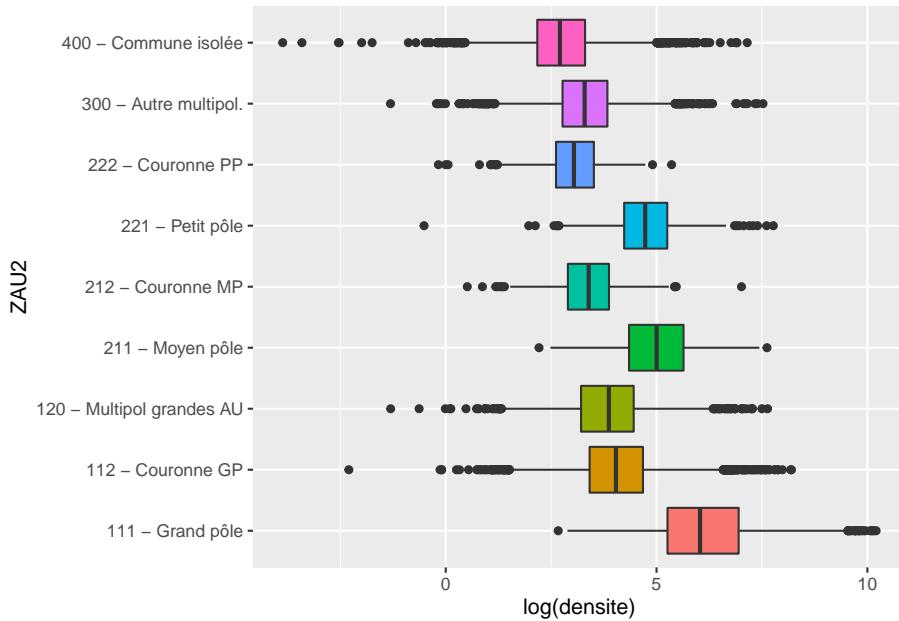
7.3 Représentation graphique

Pour représenter graphiquement le croisement d'une variable qualitative avec une variable quantitative, il existe plusieurs moyens.

- La fonction `geom_boxplot()` produit la boîte à moustaches pour visualiser, pour chaque modalités de la variable qualitative, la distribution de la variable quantitative. La barre la plus basse de la boîte indique Q1 (pourcentile 25%), le trait au milieu indique la médiane, et la barre supérieur de la boîte indique Q3.

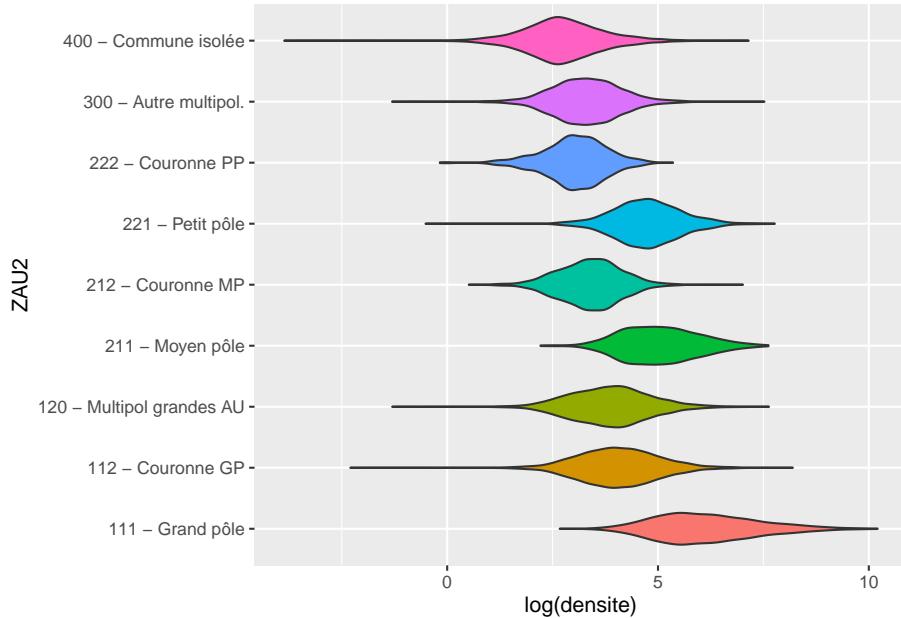
```
dat <- dat %>%
  mutate (densite = P14_POP / SUPERF,
         log_dens = log10 (densite+0.00000001))

ggplot (data = dat, aes (y = log (densite), x = ZAU2, fill = ZAU2)) +
  geom_boxplot () +
  coord_flip () + # pour plus de lisibilité
  theme (legend.position = "none") # supprime la légende
```



- Le *violinplot* (fonction `geom_violin()`) fonctionne sur le même principe. Les boîtes à moustaches sont remplacées par des graphiques en *violon*, qui représentent la densité de distribution de la variables quantitatives.

```
ggplot (data = dat, aes (y = log (densite), x = ZAU2, fill = ZAU2)) +
  geom_violin () +
  coord_flip () +
  theme (legend.position = "none")
```



7.4 Calcul du rapport de corrélation

Le **rapport de corrélation** est une mesure de la force de la liaison existant entre une variable quantitative et une variable qualitative. Il est similaire au coefficient de corrélation. Il se définit comme suit : $\hat{\eta}^2 = \frac{VarInter}{VarTotale}$. Pour le calculer, on peut appliquer la fonction `etaSquared()` sur un objet de type `anova`. Si on veut quantifier le lien éventuel entre la densité de population et les ZAU, on fait donc :

```
anova <- aov (densite ~ ZAU2, data = dat)
etaSquared (anova) # package lsR comme le v de cramer
```

```
##           eta.sq eta.sq.part
## ZAU2  0.1689881   0.1689881
```

Chapter 8

Tests

Les tests sortent du champ de la statistique descriptive “pure” pour entrer dans celui de la statistique inférentielle. L’inférence statistique consiste à induire les caractéristiques d’une population à partir de celles d’un échantillon de cette population.

8.1 Sur une variable quantitative

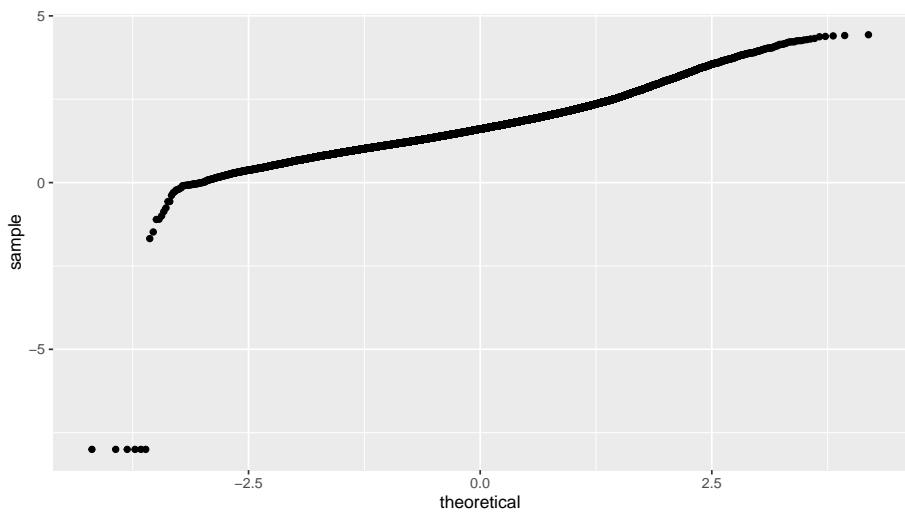
Plusieurs questions peuvent se poser à propos de la distribution d’une variable quantitative. La forme de la distribution (normale ou pas), ainsi que la tendance centrale peuvent être étudiées.

8.1.1 La distribution est-elle normale ?

Plusieurs approches sont envisageables pour se faire une idée de la réponse à cette question. La première est de visualiser graphiquement la distribution, et son “écart” à une loi normale.

Le diagramme quantile-quantile sert à comparer les quantiles de la variable à ceux d’une distribution attendue, par exemple la loi normale. La fonction `stat_qq()` permet d’afficher ce graphique.

```
ggplot (data = dat, aes (sample = log_dens)) +  
  stat_qq ()
```

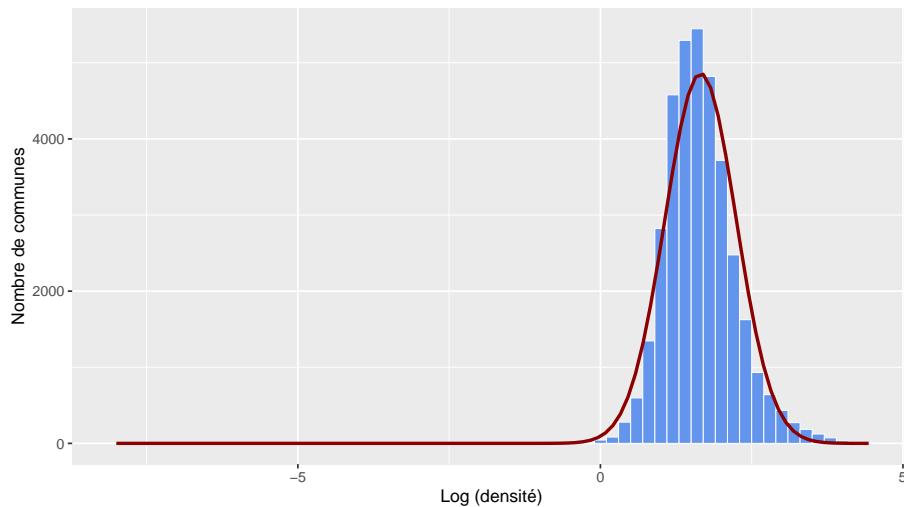


Plus la courbe est rectiligne, plus cela signifie que les quantiles de la variable sont similaires à ceux de la distribution attendue.

On peut aussi apprécier la normalité en repartant de l'histogramme :

```
vec_logd <- pull (dat, log_dens)
n <- sum (!is.na (vec_logd)) # nombre d'éléments non-NA dans vec_logd
moy <- mean (vec_logd, na.rm = T)
ecart_t <- sd (vec_logd, na.rm = T)
n_binwidth <- 0.2

ggplot (dat, aes (x = log_dens)) +
  geom_histogram (binwidth = n_binwidth,
                  colour = "white", fill = "cornflowerblue", size = 0.1) +
  stat_function (fun = function(x) dnorm (x, mean = moy, sd = ecart_t) * n * n_binwidth,
                 color = "darkred", size = 1) +
  xlab (label = "Log (densité)") +
  ylab (label = "Nombre de communes")
```



`stat_function()` trace la fonction indiquée. Ici, on trace une loi normale (`dnorm()`) de même moyenne et écart-type que la variable dont on veut tester la normalité.

Après une visualisation, il est possible de “tester” l'affirmation. On se place dans l'hypothèse la plus simple (appelée **hypothèse nulle**, ici : c'est que la distribution est bien normale), et on regarde si cela semble cohérent avec nos données.

Tests de normalité : Shapiro-Wilk, préconisé par Razali et al. 2011 Journal of Statistical Modeling and Analytics 2 (1): 21–33

```
dat$log_dens[1:4000] %>% # maxi 5000 données pour ce test
  shapiro.test ()
```

```
## 
##  Shapiro-Wilk normality test
## 
## data: .
## W = 0.98639, p-value < 2.2e-16
```

p < 5% donc l'hypothèse de normalité est rejetée.

8.1.2 La moyenne diffère-t-elle de la valeur attendue ?

De la même manière, si on s'intéresse à la moyenne d'une distribution, on “teste” le fait qu'elle diffère ou non d'une valeur fixe. Le test de Student (fonction `t.test()`) pour échantillon unique permet de comparer la moyenne observée à une valeur de référence. Il est utilisable si et seulement si la distribution des

valeurs dans l'échantillon est normale. Si ce n'est pas le cas, il faut utiliser le test de Wilcoxon (sur les rangs) avec la fonction `wilcox.test()`.

Exemple : La densité (log-transformée) en Corse diffère-t-elle de la moyenne nationale ?

```

dens_corse <- dat %>% filter (REG == "94") %>% pull(log_dens)
dens_nat <- dat %>% pull(log_dens)
moy_dens_nat <- dens_nat %>% mean(na.rm = T) # la valeur de référence

shapiro.test (dens_corse) # non normal -> Wilcoxon

## 
## Shapiro-Wilk normality test
##
## data: dens_corse
## W = 0.96621, p-value = 2.124e-07
wilcox.test (dens_corse, mu = moy_dens_nat)

## 
## Wilcoxon signed rank test with continuity correction
##
## data: dens_corse
## V = 6180, p-value < 2.2e-16
## alternative hypothesis: true location is not equal to 1.655789
t.test (dens_corse, mu = moy_dens_nat) # à titre indicatif

## 
## One Sample t-test
##
## data: dens_corse
## t = -18.247, df = 359, p-value < 2.2e-16
## alternative hypothesis: true mean is not equal to 1.655789
## 95 percent confidence interval:
##  1.086866 1.197566
## sample estimates:
## mean of x
##  1.142216

```

La moyenne observée en Corse est de 1.14, contre 1.66 au niveau national. Cette différence est significative (p-value << 5%).

8.2 Sur une variable qualitative

On travaille cette fois sur une variable qualitative. On peut s'interroger sur la répartition des observations.

8.2.1 La distribution des observations diffère-t-elle de celle attendue ?

Par exemple sur dans un échantillon de personnes, on veut savoir si le sex-ratio est significativement différent de celui de la population française dans son ensemble, soit 51.4%, ou bien si au contraire le pourcentage observé est dans l'intervalle d'incertitude compte tenu de la taille de l'échantillon.

```
femmes <- 601
hommes <- 514

observed_n <- c(femmes, hommes)      # Observation
expected_p <- c(0.514, 1-0.514)       # Proportions attendues
```

Ici l'échantillon est constitué de 601 femmes et 514 hommes donc un total de 1115 personnes.

```
expected_n <- round (expected_p * (femmes + hommes))
tab <- cbind (Sexe = c('Femmes', 'Hommes'), `Observé` = observed_n, Attendu = expected_n) %>%
  as.data.frame () %>%
  datatable ()
tab
```

| | | Show 10 entries | | | Search: |
|---|--------|-----------------|---------|--|---------|
| | Sexe | | Observé | | Attendu |
| 1 | Femmes | | 601 | | 573 |
| 2 | Hommes | | 514 | | 542 |

Showing 1 to 2 of 2 entries

Previous 1 Next

En appliquant le pourcentage théorique, on s'attendrait à ce que ces 1115 personnes soient réparties en 573 femmes et 542 hommes.

Question : Est-ce que les proportions observées peuvent être dues au hasard, ou bien notre échantillon est-il différent de la population ? On va tester au moyen du test du χ^2 pour échantillon unique. Ce test s'appelle grâce à la fonction `chisq.test()`. L'hypothèse nulle est que l'échantillon provient de la population générale.

```
test <- chisq.test(x = observed_n, p = expected_p)
test

##
## Chi-squared test for given probabilities
##
## data: observed_n
## X-squared = 2.7927, df = 1, p-value = 0.0947
```

Les proportions observées sont dans l'intervalle d'incertitude pour un échantillon de 1115 personnes ($p = 0.095 > 5\%$). L'hypothèse nulle ne peut donc pas être rejetée au risque de 5%.

8.3 Sur 2 variables qualitatives

On étudie désormais 2 variables qualitatives.

8.3.1 Les deux variables qualitatives sont-elles indépendantes l'une de l'autre ?

Pour mesurer le lien entre 2 variables qualitatives, on peut se servir du χ^2 : - Le χ^2 de Pearson constitue la base de toute mesure de liaison entre des variables qualitatives $\chi^2 = \sum_{i,j} \frac{(N_{ij} - N_{ij}^*)^2}{N_{ij}^*} \hookrightarrow \chi^2_{(k-1)(r-1)}$ où $N_{ij}^* = \frac{N_{i\cdot} \cdot N_{\cdot j}}{N}$ - Il mesure l'écart par rapport à la situation d'indépendance : plus le χ^2 est élevé, plus les variables sont liées (elles ne sont pas indépendantes) - La comparaison de la valeur avec le fractile d'ordre $1 - \alpha$ d'une loi $\chi^2_{(k-1)(r-1)}$ permet de connaître la significativité du lien - On le réalise sur le tableau de contingence des effectifs - *Il faut un effectif > 5 dans chaque case*

Cette vidéo explique bien, étape par étape, ce que représente le test du χ^2

```
tab <- table(dat$REG, dat$ZAU2)
chisq.test (tab)
```

```
## Warning in chisq.test(tab): Chi-squared approximation may be incorrect
```

```
##
## Pearson's Chi-squared test
##
## data: tab
## X-squared = 6100.4, df = 128, p-value < 2.2e-16
```

La p-value est inférieure à 5%, on a donc une probabilité très faible de nous tromper en rejetant l'hypothèse d'indépendance. Donc on rejette l'indépendance, c'est à dire qu'on peut considérer que les variables sont liées.

Cependant, on a des effectifs nuls : il faut donc regrouper des modalités pour satisfaire aux conditions d'utilisation du test. Ici on écarte aussi les outre-mer (codes 01 à 04) qui ont des effectifs faibles.

```
dat_ss_om <- dat %>%
  filter (!REG %in% c('01', '02', '03', '04')) %>%
  droplevels () %>%
  mutate (ZAU3 = fct_recode (ZAU2, urbain = "111 - Grand pôle",
                            urbain = "211 - Moyen pôle",
                            urbain = "221 - Petit pôle",
                            "periurbain ou rural" = "112 - Couronne GP",
                            "periurbain ou rural" = "212 - Couronne MP",
                            "periurbain ou rural" = "120 - Multipol grandes AU",
                            "periurbain ou rural" = "300 - Autre multipol.",
                            "periurbain ou rural" = "222 - Couronne PP",
                            "periurbain ou rural" = "400 - Commune isolée"))

levels (dat_ss_om$ZAU3)

## [1] "urbain"                 "periurbain ou rural"
tab <- table(dat_ss_om$REG, dat_ss_om$ZAU3)
chisq.test (tab)

##
## Pearson's Chi-squared test
##
## data: tab
## X-squared = 1053.7, df = 12, p-value < 2.2e-16
```

8.3.2 Un couple de variables qualitatives est-il “lié plus fortement” qu'un autre couple ?

- La statistique du χ^2 dépend du nombre d'observations n et du nombre de modalités des variables. On ne peut donc pas comparer la force du lien entre plusieurs couples de variables

- Le **V de Cramer** quantifie l'intensité du lien entre deux variables qualitatives
- $V = \sqrt{\frac{\chi^2}{\min(k-1, r-1)}} \in [0; 1]$
- Analogie avec le coefficient de corrélation

```
cramersV (tab)      # package lsr
## [1] 0.169728
```

On peut comparer cette grandeur pour plusieurs couples de variables qualitatives.

8.4 Sur un croisement quantitatif/qualitatif

On cherche à comparer les moyennes d'une variable de deux groupes indépendants. La variable à comparer est quantitative, les deux (ou plus) groupes peuvent être vus comme issus d'une variable quantitative. Par exemple, on peut s'intéresser à la comparaison de la **densité (variable quantitative)** selon le **département (variable qualitative)**.

8.4.1 2 échantillons indépendants

On ne regarde que 2 départements pour le moment.

Exemple : On veut comparer la densité (log-transformée) des communes du Haut Rhin (68) et de celles du Val de Marne (94). Ce sont deux groupes indépendants (les communes du Haut Rhin ne sont pas liées à celles du Val de Marne).

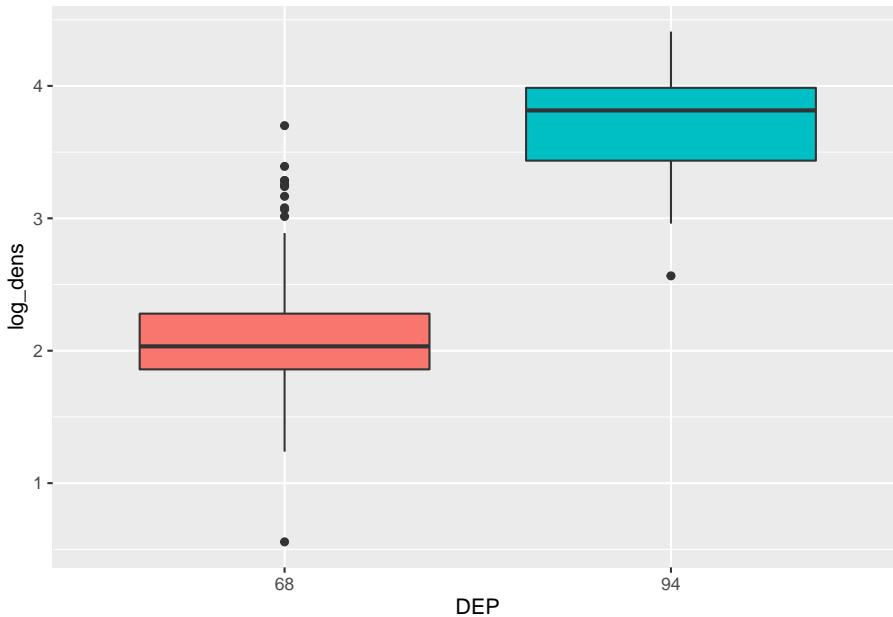
8.4.1.1 Distribution normale

Si les deux groupes à observer sont normalement distribués, on se servira du *test de Student* (`t.test()` sur R).

```
# travail préalable : on peut visualiser les données
data_test <- dat %>% filter(DEP %in% c(94, 68)) %>%
  select(CODEGEO, log_dens, densite, DEP)

ggplot(data = data_test, aes(x = DEP, y = log_dens, fill = DEP)) +
  geom_boxplot() + theme (legend.position = "none")

## Warning: Removed 11 rows containing non-finite values (stat_boxplot).
```



```
res <- t.test(log_dens ~ DEP, data = data_test, var.equal = TRUE)
res

##
##  Two Sample t-test
##
## data: log_dens by DEP
## t = -27.271, df = 411, p-value < 2.2e-16
## alternative hypothesis: true difference in means is not equal to 0
## 95 percent confidence interval:
## -1.717797 -1.486801
## sample estimates:
## mean in group 68 mean in group 94
##           2.087342          3.689640
```

La p-value est inférieure à 0.05. La moyenne de la log-densité dans le Val de Marne est significativement différente celle dans le Haut Rhin.

8.4.1.2 Distribution non normale

Dans le cas où les distributions des groupes ne sont pas normales, on se servira du *test de Wilcoxon* (`wilcox.test()` sur R). Le test de Wilcoxon se base sur les rangs des valeurs des observations des deux groupes, ce qui permet de s'affranchir de l'hypothèse de normalité. Si on regarde directement la densité :

```
res <- wilcox.test(densite ~ DEP, data = data_test)
res

##
## Wilcoxon rank sum test with continuity correction
##
## data: densite by DEP
## W = 111, p-value < 2.2e-16
## alternative hypothesis: true location shift is not equal to 0
```

La p-value est inférieure à 0.05. Les moyennes sont donc significativement différentes entre les deux départements.

8.4.2 2 échantillons dépendants

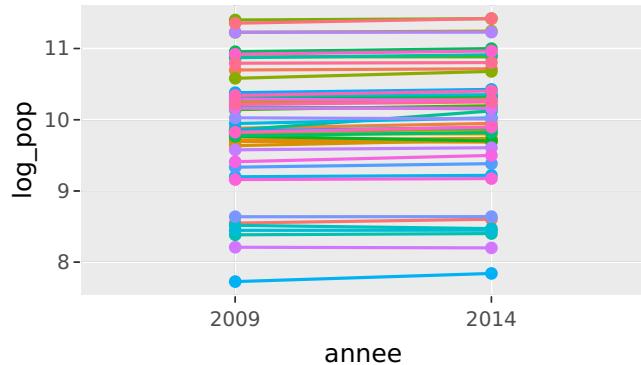
Cette fois, on cherche à comparer les moyennes entre deux groupes dépendants. On a alors 2 variables par observations. Par exemple, on veut comparer la population moyenne en 2014 par rapport à celle en 2009. Pour chaque commune, on dispose de la population en 2009 et 2014. On se servira des mêmes tests que ceux vus précédemment, et on précisera l'argument `paired = TRUE`.

8.4.2.1 Distribution normale

Si les distributions sont normales, on se servira du *test de Student*

```
# travail préalable : on peut visualiser les données
data_test_d <- dat %>% filter(DEP == 94) %>%
  select(CODGEO, P14_POP, P09_POP) %>%
  gather(key = annee, value = population, -CODGEO) %>%
  mutate(log_pop = log(population),
        annee = ifelse(annee == "P14_POP", "2014", "2009")) %>%
  arrange(desc(annee), CODGEO)

g <- ggplot(data = data_test_d, aes(x = annee, y = log_pop, color = CODGEO)) +
  geom_line(aes(group = CODGEO)) + geom_point() +
  theme (legend.position = "none")
ggplotly(g) # ajout de dynamisme
```



```

res <- t.test(log_pop ~ annee, data = data_test_d, paired = TRUE)
res

##
## Paired t-test
##
## data: log_pop by annee
## t = -4.6323, df = 46, p-value = 2.985e-05
## alternative hypothesis: true difference in means is not equal to 0
## 95 percent confidence interval:
## -0.04960983 -0.01955511
## sample estimates:
## mean of the differences
## -0.03458247

```

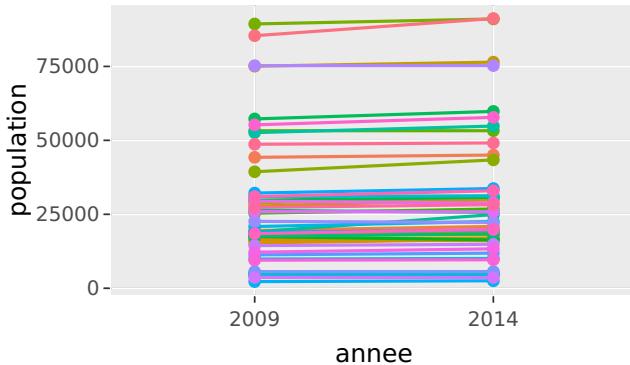
8.4.2.2 Distribution non normale

Si les distributions ne sont pas normales, on se servira du *test de Wilcoxon*

```

# travail préalable : on peut visualiser les données
g <- ggplot(data = data_test_d, aes(x = annee, y = population, color = CODGEO)) +
  geom_line(aes(group = CODGEO)) + geom_point() +
  theme (legend.position = "none")
ggplotly(g)

```



```
res <- wilcox.test(population ~ annee, data = data_test_d, paired = TRUE)

## Warning in wilcox.test.default(x = c(5161L, 44278L, 19548L, 16248L,
## 16594L, : cannot compute exact p-value with ties
res

##
## Wilcoxon signed rank test with continuity correction
##
## data: population by annee
## V = 126, p-value = 3.661e-06
## alternative hypothesis: true location shift is not equal to 0
```

La p-value est inférieure à 0.05. La moyenne des populations est significativement différente d'une année sur l'autre.

8.4.3 Plusieurs échantillons indépendants

Si on cherche à comparer plus de 2 groupes, on doit se servir d'autres tests que ceux vus précédemment. De la même manière, on dispose d'un test qui fonctionne avec hypothèse de normalité, et un qui permettra de s'affranchir de cette contrainte.

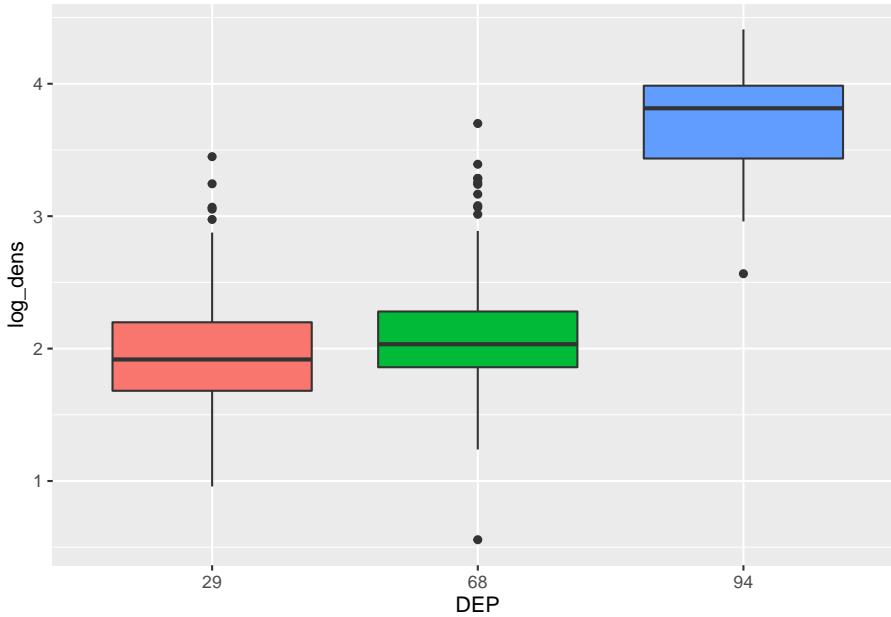
8.4.3.1 Echantillons normalement distribués (ANOVA)

Dans le cas où les données suivent une distribution normale, on se servira alors de l'ANOVA (Analysis of Variance). La fonction permettant d'effectuer une ANOVA sur R est `aov()`. Si on veut comparer la densité dans 3 départements différents, on fera :

```
# travail préalable : on peut visualiser les données
data_test_m <- dat %>% filter(DEP %in% c(94, 68, 29)) %>%
  select(CODGEO, densite, log_dens, DEP)

ggplot(data = data_test_m, aes(x = DEP, y = log_dens, fill = DEP)) +
  geom_boxplot() + theme (legend.position = "none")

## Warning: Removed 13 rows containing non-finite values (stat_boxplot).
```



```
res <- aov(log_dens ~ DEP, data = data_test_m)
summary(res)

##          Df Sum Sq Mean Sq F value Pr(>F)
## DEP          2   124.0    61.98   399.1 <2e-16 ***
## Residuals  691   107.3     0.16
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
## 13 observations deleted due to missingness
```

La p-value est inférieure à 0.05 : il y a bien une différence significative entre les

groupes. Pour compléter l'analyse, il faudrait ensuite repartir sur les tests entre deux échantillons. Pour connaître les groupes deux à deux distincts, on peut aussi se servir de la fonction `TukeyHSD()`, qui calcule les comparaisons entre les groupes.

```
TukeyHSD(res)
```

```
## Tukey multiple comparisons of means
## 95% family-wise confidence level
##
## Fit: aov(formula = log_dens ~ DEP, data = data_test_m)
##
## $DEP
##      diff      lwr      upr   p adj
## 68-29 0.136662 0.06324928 0.2100747 4.21e-05
## 94-29 1.738961 1.59309691 1.8848246 0.00e+00
## 94-68 1.602299 1.45888251 1.7457150 0.00e+00
```

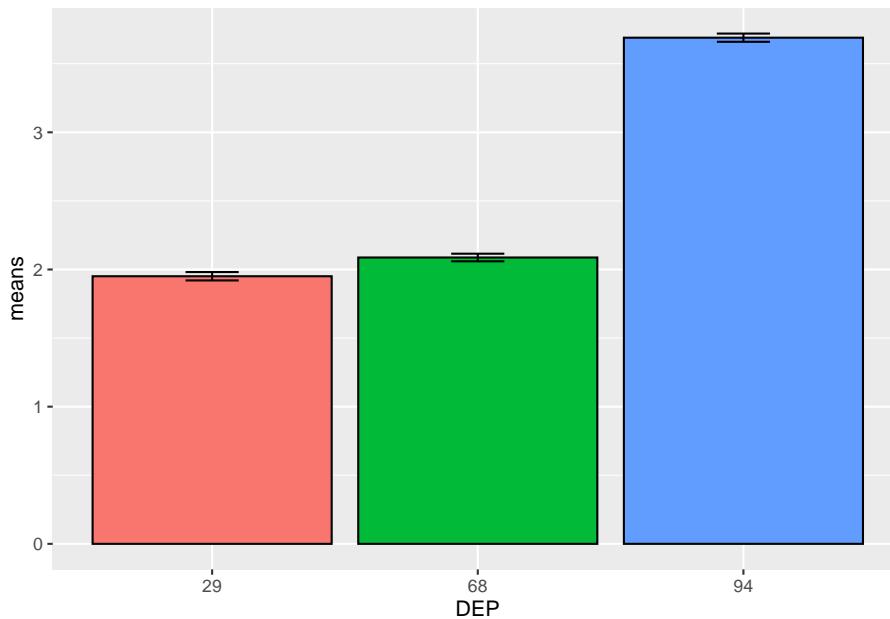
Les p-value étant toutes inférieures à 0.05, la différence est donc significative entre chaque groupes pris deux à deux.

Une autre représentation graphique utilisée pour une ANOVA est le graphique `errorbar`.

```
n <- nrow(data_test_m)

# creation des données pour chaque DEP
graph <- data_test_m %>% group_by(DEP) %>%
  summarise(sd = sd(log_dens, na.rm = T),
            means = mean(log_dens, na.rm = T)) %>%
  mutate(se = 1.96 * sd / sqrt(n))

ggplot(graph, aes(x = DEP, y = means, fill = DEP)) +
  geom_bar(stat="identity", color="black",
           position = position_dodge()) +
  geom_errorbar(aes(ymax = means + se, ymin = means - se),
                width = 0.2) +
  theme (legend.position = "none")
```

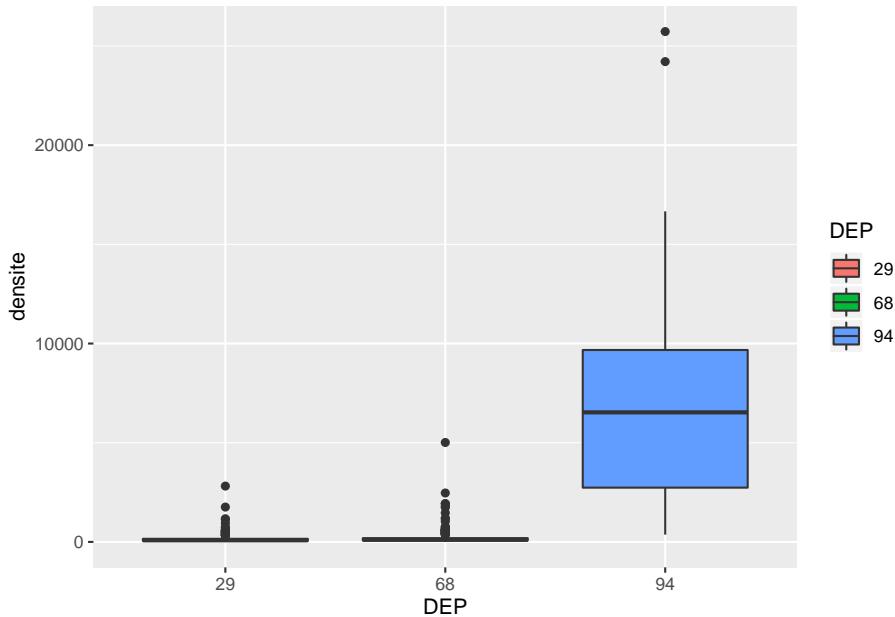


8.4.3.2 Echantillons non normalement distribués (Kruskal-Wallis)

Si les distributions ne sont pas normales, on se servira du *test de Kruskal-Wallis*. Il généralise le test de Wilcoxon sur plus de deux groupes. Il fonctionne de manière similaire, en comparant les rangs des valeurs des différents groupes.

```
# travail préalable : on peut visualiser les données
ggplot(data = data_test_m, aes(x = DEP, y = densite, fill = DEP)) + geom_boxplot()

## Warning: Removed 13 rows containing non-finite values (stat_boxplot).
```



```
res <- kruskal.test(densite ~ DEP, data = data_test_m)
res

## 
##  Kruskal-Wallis rank sum test
##
##  data:  densite by DEP
##  Kruskal-Wallis chi-squared = 145.34, df = 2, p-value < 2.2e-16
```

La p-value est inférieure à 0.05 : il y a bien une différence significative entre les groupes. Pour compléter l'analyse, il faudrait ensuite repartir sur les tests entre deux échantillons. Pour connaître les groupes deux à deux distincts, on peut aussi se servir de la fonction pairwise.wilcox.test, qui calcule les comparaisons entre les groupes.

```
pairwise.wilcox.test(data_test_m$densite, data_test_m$DEP,
                      p.adjust.method = "BH")
```

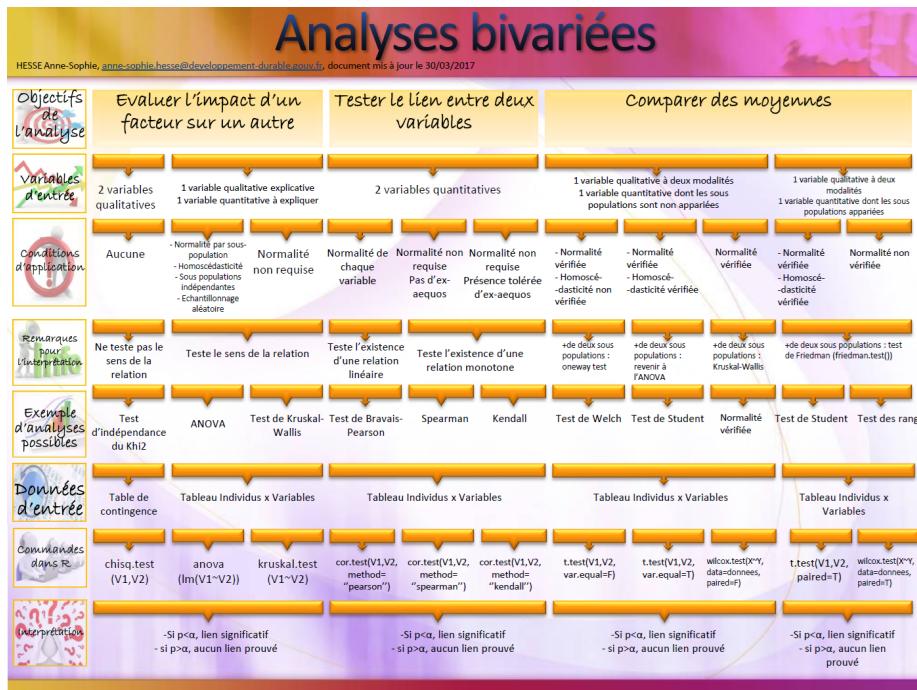
```
## 
##  Pairwise comparisons using Wilcoxon rank sum test
##
##  data:  data_test_m$densite and data_test_m$DEP
##
##    29      68
## 68 1.2e-05 -
## 94 < 2e-16 < 2e-16
##
```

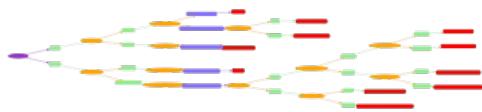
```
## P value adjustment method: BH
```

Ici, tous les départements sont significativement différents deux à deux, car les *p-values* sont toutes inférieures à 0,05.

8.5 Résumé

Afin de savoir quel test s'applique dans chaque situation :





Chapter 9

Exercice de synthèse

9.1 Enoncé

Créer un sous-jeu de données constitué des 15 premières colonnes de la base Insee restreint à la région des Pays-de-la-Loire (code REG 52). Supprimer les modalités inutiles des variables qualitatives.

9.1.1 Partie 1

L'objectif est de bien décrire la variable de population issue du recensement 2014.

Produire un graphique simple pour décrire la distribution.

Calculer les statistiques simples décrivant cette variable.

On a des indicateurs de tendance centrale qui sont très différents. Pourquoi ?

Quel est le nombre de communes de moins de 1000 habitants ?

Produire un graphique plus riche que le précédent.

Produire un graphique comparant les distributions entre les départements des Pays-de-la-Loire.

9.1.2 Partie 2

En 2014, la population communale moyenne en Pays-de-la-Loire est-elle supérieure ou inférieure à la population communale moyenne nationale ?

Cette différence est-elle significative au seuil de 5% ?

Aide : Réfléchir aux questions :

- un ou deux échantillons ?
- test paramétrique ou non paramétrique ?

La population moyenne diffère-t-elle entre PDL et Centre-val-de-Loire ?

9.1.3 Partie 3

Combien de communes par département dans les Pays-de-Loire ?

En 2014, quelle est la population communale moyenne par département dans la région ?

Est-ce que la ventilation des communes dans les catégories de ZAU diffère selon les départements de la région PDL ?

9.1.4 Partie 4

Y a-t-il un lien significatif entre la densité de population et le taux d'emploi en région PDL ? Si oui, est-il positif ou négatif ?

9.2 Corrigé

Les questions étant ouvertes, il y a de nombreuses façons d'y répondre. Ci-dessous certaines d'entre elles.

Créer un sous-jeu de données constitué des 15 premières colonnes de la base Insee restreint à la région des Pays-de-la-Loire (code REG 52)

```
pdl <- dat %>%
  select (1:15) %>%
  filter (REG == "52")

summary (pdl)

##      CODGEO                      LIBGEO          REG          DEP
## 44001   :  1  Allonnes           :  2  52    :1502   72   :375
## 44002   :  1  Avrill\xe9         :  2  01    :  0   49   :363
## 44003   :  1  Beaumont-Pied-de-Boeuf:  2  02    :  0   85   :282
## 44004   :  1  Ch\xe9ranc\xe9        :  2  03    :  0   53   :261
## 44005   :  1  Chang\xe9          :  2  04    :  0   44   :221
## 44006   :  1  Cherr\xe9          :  2  11    :  0   01   :  0
## (Other):1496 (Other)            :1490 (Other):  0 (Other):  0
##      ZAU
```

```

## 112 - Couronne d'un grand p\xf4le :535
## 300 - Autre commune multipolaris\xe9e :386
## 120 - Multipolaris\xe9e des grandes aires urbaines :186
## 400 - Commune isol\xe9e hors influence des p\xf4les:143
## 111 - Grand p\xf4le (plus de 10 000 emplois) :108
## 221 - Petit p\xf4le (de 1 500 \xe0 5 000 emplois) : 69
## (Other) : 75
## ZE P14_POP P09_POP
## 5213 - Le Mans :230 Min. : 12 Min. : 17
## 5205 - Angers :182 1st Qu.: 495 1st Qu.: 481
## 5209 - Laval :157 Median : 1029 Median : 978
## 5203 - Nantes :153 Mean : 2720 Mean : 2608
## 5218 - La Roche-sur-Yon:106 3rd Qu.: 2353 3rd Qu.: 2243
## 5207 - Saumur : 86 Max. :298029 Max. :282047
## (Other) :588 NA's :145 NA's :145
## SUPERF NAIS0914 DECE0914 P14_MEN
## Min. : 0.20 Min. : 1.0 Min. : 0.0 Min. : 5.45
## 1st Qu.: 11.76 1st Qu.: 33.0 1st Qu.: 15.0 1st Qu.: 196.52
## Median : 17.96 Median : 69.0 Median : 31.0 Median : 402.51
## Mean : 23.64 Mean : 166.3 Mean : 113.1 Mean : 1183.32
## 3rd Qu.: 28.53 3rd Qu.: 147.0 3rd Qu.: 107.0 3rd Qu.: 952.00
## Max. :323.98 Max. :19814.0 Max. :9968.0 Max. :155272.72
## NA's :145 NA's :145 NA's :145 NA's :145
## NAISD15 DECESD15 P14_LOG
## Min. : 0.00 Min. : 0.00 Min. : 11.45
## 1st Qu.: 5.00 1st Qu.: 3.00 1st Qu.: 246.94
## Median : 12.00 Median : 7.00 Median : 483.02
## Mean : 30.99 Mean : 24.96 Mean : 1427.59
## 3rd Qu.: 26.00 3rd Qu.: 23.00 3rd Qu.: 1108.68
## Max. :4011.00 Max. :2146.00 Max. :171979.65
## NA's :145 NA's :145 NA's :145

```

On a toujours les anciens codes régions et départements qui sont désormais inutiles.

```

pdl <- pdl %>%
  droplevels () %>%
  select (-REG)

summary (pdl)

## CODGEO LIBGEO DEP
## 44001 : 1 Allonnes : 2 44:221
## 44002 : 1 Avrill\xe9e : 2 49:363
## 44003 : 1 Beaumont-Pied-de-Boeuf: 2 53:261
## 44004 : 1 Ch\xe9ranc\xe9e : 2 72:375
## 44005 : 1 Chang\xe9e : 2 85:282

```

```

## 44006 : 1 Cherr\xe9e : 2
## (Other):1496 (Other) :1490
## ZAU
## 112 - Couronne d'un grand p\xf4le :535
## 300 - Autre commune multipolaris\xe9e :386
## 120 - Multipolaris\xe9e des grandes aires urbaines :186
## 400 - Commune isol\xe9e hors influence des p\xf4les:143
## 111 - Grand p\xf4le (plus de 10 000 emplois) :108
## 221 - Petit p\xf4le (de 1 500 \xe0 5 000 emplois) : 69
## (Other) : 75
## ZE P14_POP P09_POP
## 5213 - Le Mans :230 Min. : 12 Min. : 17
## 5205 - Angers :182 1st Qu.: 495 1st Qu.: 481
## 5209 - Laval :157 Median : 1029 Median : 978
## 5203 - Nantes :153 Mean : 2720 Mean : 2608
## 5218 - La Roche-sur-Yon:106 3rd Qu.: 2353 3rd Qu.: 2243
## 5207 - Saumur : 86 Max. :298029 Max. :282047
## (Other) :588 NA's :145 NA's :145
## SUPERF NAIS0914 DECE0914 P14_MEN
## Min. : 0.20 Min. : 1.0 Min. : 0.0 Min. : 5.45
## 1st Qu.: 11.76 1st Qu.: 33.0 1st Qu.: 15.0 1st Qu.: 196.52
## Median : 17.96 Median : 69.0 Median : 31.0 Median : 402.51
## Mean : 23.64 Mean : 166.3 Mean : 113.1 Mean : 1183.32
## 3rd Qu.: 28.53 3rd Qu.: 147.0 3rd Qu.: 107.0 3rd Qu.: 952.00
## Max. :323.98 Max. :19814.0 Max. :9968.0 Max. :155272.72
## NA's :145 NA's :145 NA's :145 NA's :145
## NAISD15 DECESD15 P14_LOG
## Min. : 0.00 Min. : 0.00 Min. : 11.45
## 1st Qu.: 5.00 1st Qu.: 3.00 1st Qu.: 246.94
## Median : 12.00 Median : 7.00 Median : 483.02
## Mean : 30.99 Mean : 24.96 Mean : 1427.59
## 3rd Qu.: 26.00 3rd Qu.: 23.00 3rd Qu.: 1108.68
## Max. :4011.00 Max. :2146.00 Max. :171979.65
## NA's :145 NA's :145 NA's :145

```

9.2.1 Partie 1

Objectif : bien décrire la variable population communale (P14_POP)

Quelle est la population moyenne des communes de la région ?

```

pop_moy_pdl <- pdl %>%
  pull (P14_POP) %>%
  mean (na.rm = T) %>%
  round ()
pop_moy_pdl

```

```
## [1] 2720
```

Quelle est la la médiane ?

```
pop_med_pdl <- pdl %>%
  pull (P14_POP) %>%
  median (na.rm = T) %>%
  round ()
pop_med_pdl
```

```
## [1] 1029
```

On a deux indicateurs de tendance centrale qui sont très différents. Pourquoi ?

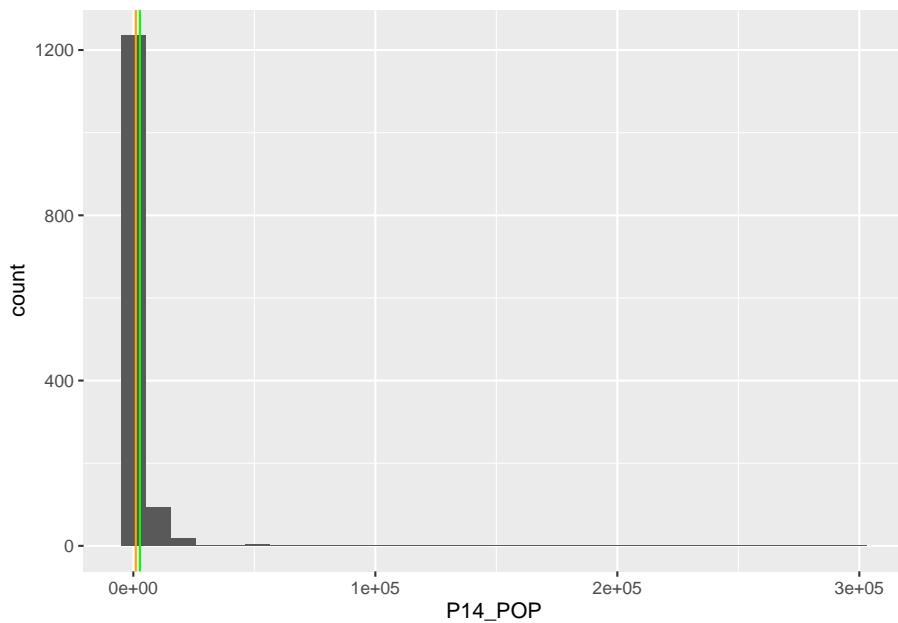
Quel est le nombre de communes de moins de 1000 habitants ?

```
pop_inf_1000 <- pdl %>%
  filter (P14_POP < 1000) %>%
  nrow ()
pop_inf_1000
```

```
## [1] 664
```

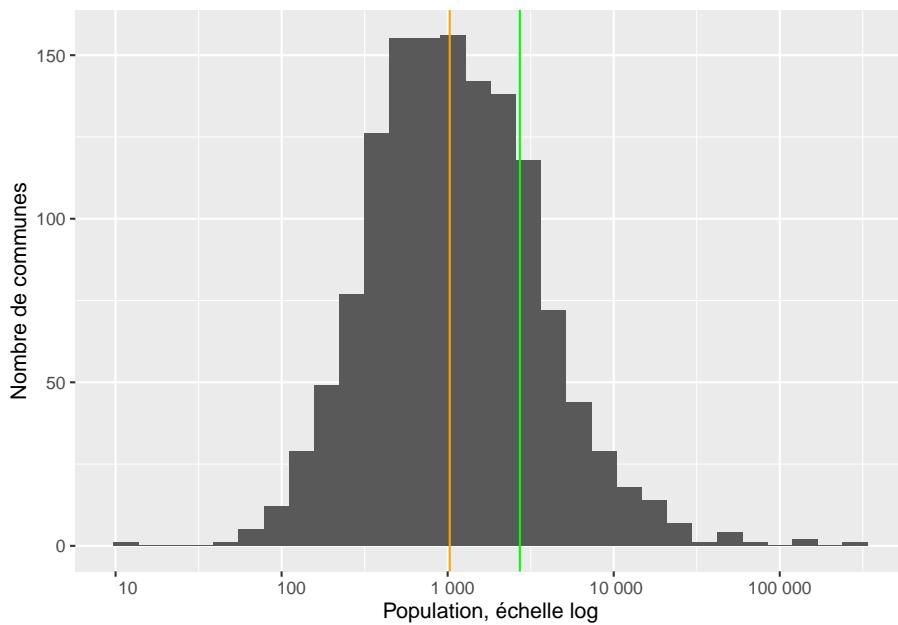
L'histogramme de base ...

```
graphique <- ggplot (data = pdl, aes (x = P14_POP)) +
  geom_histogram () +
  geom_vline (xintercept = pop_med_pdl, color = 'orange') +
  geom_vline (xintercept = pop_moy_pdl, color = 'green')
graphique
```



```
graphique <- graphique +
  scale_x_continuous (trans = 'log10',
                      labels = function(x) format(x, big.mark = " ", scientific = F),
                      breaks = c(10, 100, 1000, 10000, 100000)) +
  xlab ('Population, échelle log') +
  ylab ('Nombre de communes')

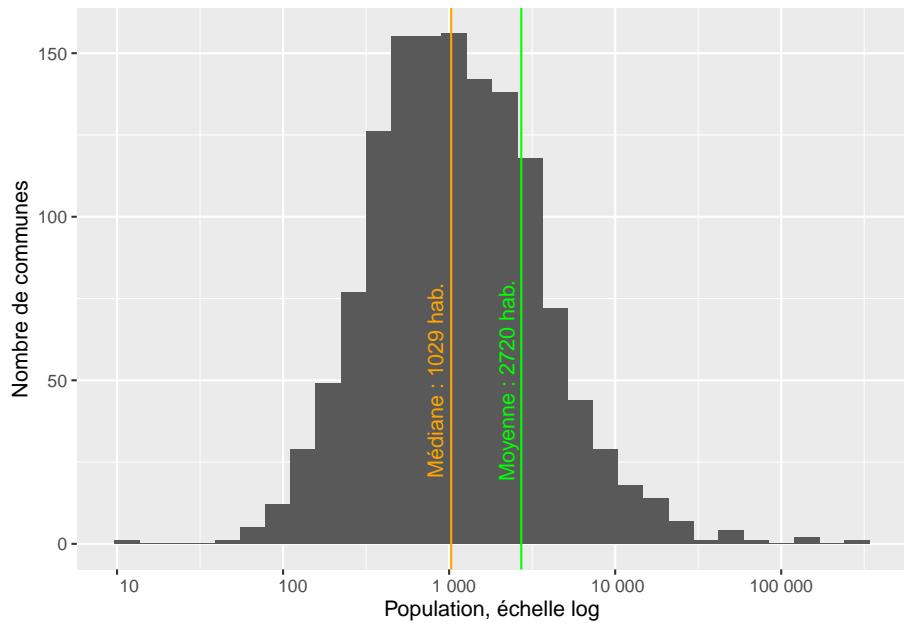
graphique
```



On peut encore rendre le graphique plus “auto-porteur”.

```
graphique <- graphique +
  annotate (geom = 'text', x = pop_med_pdl-200, y = 50,
            label = paste ('Médiane', ':', pop_med_pdl, 'hab.'),
            angle = 90, color = 'orange') +
  annotate (geom = 'text', x = pop_moy_pdl-500, y = 50,
            label = paste ('Moyenne', ':', pop_moy_pdl, 'hab.'),
            angle = 90, color = 'green')
```

graphique

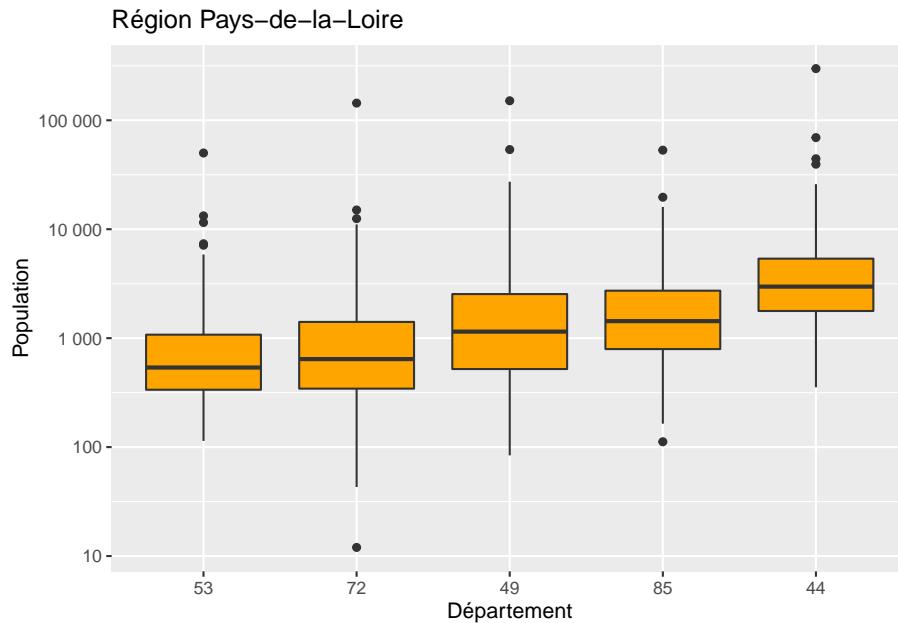


Produire un graphique comparant les distributions entre les départements des Pays-de-la-Loire

```
medianes <- pdl %>%
  group_by (DEP) %>%
  summarise (pop_med = median (P14_POP, na.rm = T))
pdl <- inner_join (x = pdl, y = medianes)
```

```
## Joining, by = "DEP"
ggplot (data = pdl, aes (x = fct_reorder (DEP, pop_med), y = P14_POP)) +
  geom_boxplot (fill = 'orange') +
  scale_y_continuous (trans = 'log10',
                      breaks = c(10, 100, 1000, 10000, 100000),
                      labels = function(x) format(x, big.mark = " ", scientific = FALSE))
  xlab ('Département') +
  ylab ('Population') +
  ggtitle ('Région Pays-de-la-Loire')
```

```
## Warning: Removed 145 rows containing non-finite values (stat_boxplot).
```



9.2.2 Partie 2

La population moyenne en Pays-de-la-Loire diffère-t-elle de la moyenne nationale ?

```
pop_moy_nat <- dat %>%
  pull (P14_POP) %>%
  mean (na.rm = T) %>%
  round ()
pop_moy_nat
```

```
## [1] 1837
```

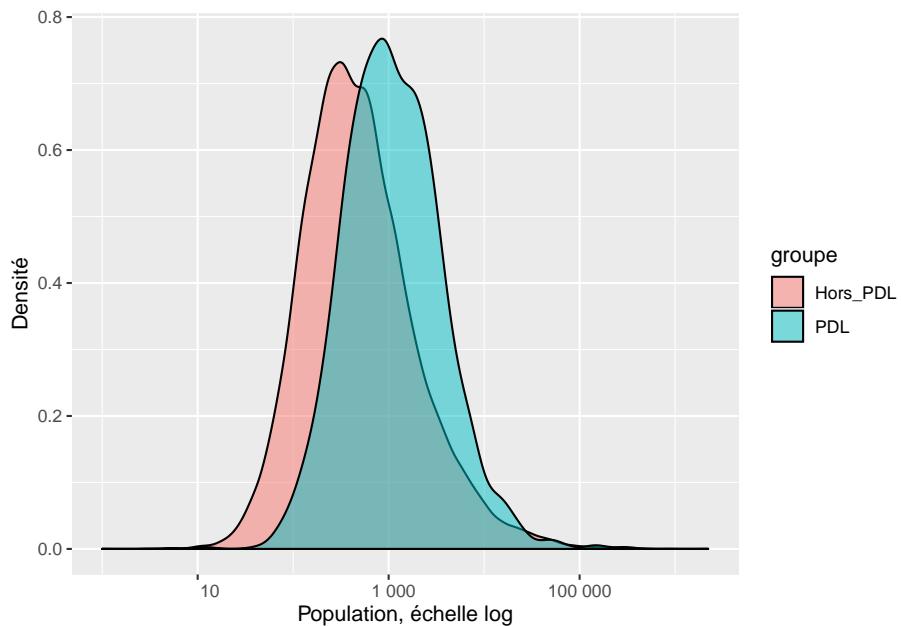
Cette différence est-elle significative au seuil de 5% ? Réfléchir aux questions :
- un ou deux échantillons ? - test paramétrique ou non paramétrique ?

On peut déjà comparer graphique les distributions PDL et hors PDL

```
dat <- dat %>%
  mutate (groupe = ifelse (REG == '52', 'PDL', 'Hors_PDL'))

ggplot (data = dat, aes (x = P14_POP, group = groupe, fill = groupe)) +
  geom_density (alpha = 0.5) +
  scale_x_continuous (trans = 'log10', labels = function(x) format (x, big.mark = " ", scientific = TRUE))
  xlab ('Population, échelle log') +
  ylab ('Densité')
```

```
## Warning: Transformation introduced infinite values in continuous x-axis
## Warning: Removed 827 rows containing non-finite values (stat_density).
```



Alors, un ou deux échantillons ?

Si l'on considère 2 échantillons :

```
t.test (P14_POP ~ groupe, var.equal = TRUE, data = dat) # Student

##
##  Two Sample t-test
##
## data: P14_POP by groupe
## t = -2.2175, df = 35866, p-value = 0.0266
## alternative hypothesis: true difference in means is not equal to 0
## 95 percent confidence interval:
## -1727.6282 -106.4727
## sample estimates:
## mean in group Hors_PDL      mean in group PDL
##           1802.797              2719.847

t.test (P14_POP ~ groupe, var.equal = FALSE, data = dat) # Welch

##
##  Welch Two Sample t-test
##
## data: P14_POP by groupe
```

```

## t = -3.0193, df = 1572.4, p-value = 0.002575
## alternative hypothesis: true difference in means is not equal to 0
## 95 percent confidence interval:
## -1512.814 -321.287
## sample estimates:
## mean in group Hors_PDL      mean in group PDL
##           1802.797             2719.847
wilcox.test (P14_POP ~ groupe, data = dat) # Wilcoxon

##
## Wilcoxon rank sum test with continuity correction
##
## data: P14_POP by groupe
## W = 14420160, p-value < 2.2e-16
## alternative hypothesis: true location shift is not equal to 0

```

Si l'on considère 1 échantillon :

```

t.test (x = pdl$P14_POP, mu = pop_moy_nat) # Student

##
## One Sample t-test
##
## data: pdl$P14_POP
## t = 3.0164, df = 1356, p-value = 0.002605
## alternative hypothesis: true mean is not equal to 1837
## 95 percent confidence interval:
## 2145.694 3294.001
## sample estimates:
## mean of x
## 2719.847
wilcox.test (x = pdl$P14_POP, mu = pop_moy_nat) # Wilcoxon

##
## Wilcoxon signed rank test with continuity correction
##
## data: pdl$P14_POP
## V = 343305, p-value = 4.264e-16
## alternative hypothesis: true location is not equal to 1837

```

On peut faire plein de tests ; à chaque fois R donne un résultat, mais le(s)quel(s) choisir ?

Ici la moyenne nationale n'est pas calculée sur un échantillon : elle l'est sur l'exhaustivité des communes. C'est donc la moyenne sur la population. Notre problème est donc la comparaison de la moyenne d'un échantillon unique avec la moyenne de la population.

Comme les distributions sont très asymétriques donc non gaussiennes, on ne peut pas lire les tests de Student. C'est donc le test de Wilcoxon qui nous indique une différence significative au seuil de 5%. Conclusion : Les communes de la région comptent en moyenne significativement plus d'habitants que celles de la France entière. Il est peu probable qu'il s'agisse d'un effet du hasard, donc les populations des communes des PDL ne sont pas distribuées comme celles des autres communes de France.

La population moyenne diffère-t-elle entre PDL et Centre-val-de-Loire ?

Traduire : Si je regroupe les communes des PDL + de CVDL (1502 + 1842) puis que je répartis aléatoirement ces communes dans deux groupes, est-il possible que par un effet du hasard, les distributions des deux groupes diffèrent autant que diffèrent celles observées dans les deux régions ?

Centre-val-de-Loire : code 24

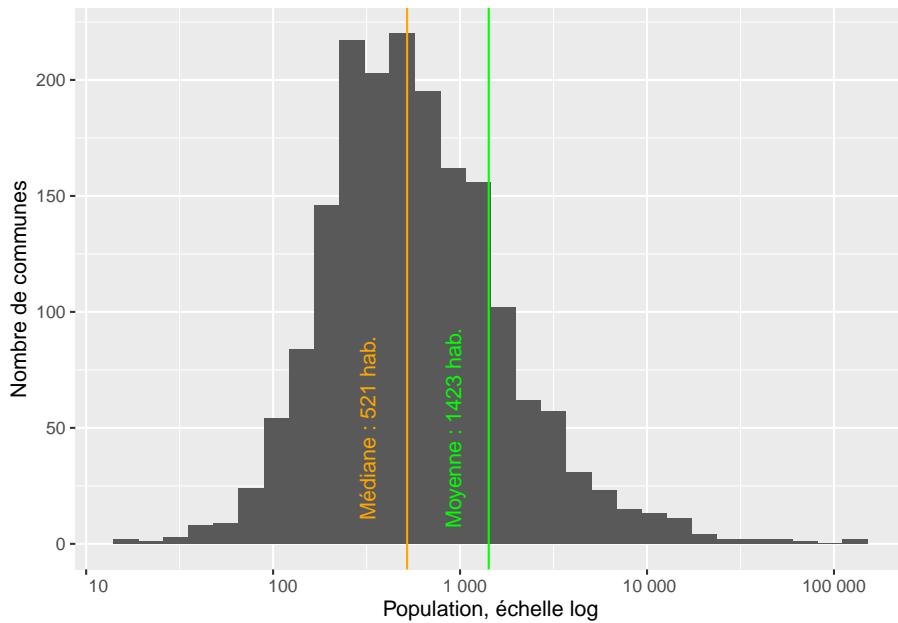
```
cvdl <- dat %>%
  select (1:15) %>%
  filter (REG == "24")

pop_moy_cvdl <- cvdl %>%
  pull (P14_POP) %>%
  mean (na.rm = T) %>%
  round ()

pop_med_cvdl <- cvdl %>%
  pull (P14_POP) %>%
  median (na.rm = T) %>%
  round ()
```

A quoi ressemble la distribution en Centre-val-de-Loire ?

```
ggplot (data = cvdl, aes (x = P14_POP)) +
  geom_histogram () +
  geom_vline (xintercept = pop_med_cvdl, color = 'orange') +
  geom_vline (xintercept = pop_moy_cvdl, color = 'green') +
  scale_x_continuous (trans = 'log10',
    labels = function (x) format(x, big.mark = " ", scientific = F),
    breaks = c (10, 100, 1000, 10000, 100000)) +
  xlab ('Population, échelle log') +
  ylab ('Nombre de communes') +
  annotate (geom = 'text', x = pop_med_cvdl-200, y = 50,
    label = paste ('Médiane', ':', pop_med_cvdl, 'hab.'),
    angle = 90, color = 'orange') +
  annotate (geom = 'text', x = pop_moy_cvdl-500, y = 50,
    label = paste ('Moyenne', ':', pop_moy_cvdl, 'hab.'),
    angle = 90, color = 'green')
```



Tester la significativité

```
cvdl_vs_pdl <- dat %>%
  filter (REG %in% c(52, 24))

wilcox.test (P14_POP ~ REG, data = cvdl_vs_pdl)

## 
## Wilcoxon rank sum test with continuity correction
##
## data: P14_POP by REG
## W = 824661, p-value < 2.2e-16
## alternative hypothesis: true location shift is not equal to 0
```

Pas de surprise, la différence est très significative. A noter, que “p-value < 2.2e-16” est un message qui apparaît souvent : c'est une valeur “plancher” et il ne sert à rien d'aller voir plus loin au microscope des valeurs encore plus faibles.

9.2.3 Partie 3

Combien de communes par département dans les Pays-de-Loire ?

```
table (pdl$DEP)

## 
## 44 49 53 72 85
```

```
## 221 363 261 375 282
```

Quelle est la population communale moyenne par département dans la région ?

```
pop_com_moy_dept <- pdl %>% group_by (DEP) %>%
  summarise (pop_moy_dept = round (mean (P14_POP, na.rm = T)))
```

```
pop_com_moy_dept
```

```
## # A tibble: 5 x 2
##   DEP    pop_moy_dept
##   <fct>     <dbl>
## 1 44        6352
## 2 49        3224
## 3 53        1192
## 4 72        1546
## 5 85        2461
```

Est-ce que la ventilation des communes dans les catégories de ZAU diffère selon les départements de la région PDL ?

```
croisement <- table (pdl$DEP, pdl$ZAU)
```

```
DEP_vs_ZAU <- croisement %>%
  as.data.frame () %>%
  spread (key = Var1, value = Freq) %>%
  rename (ZAU = Var2)
DEP_vs_ZAU
```

| | | ZAU | 44 | 49 | 53 | 72 | 85 |
|------|---|-----|-----|----|-----|-----|----|
| ## 1 | 111 - Grand p\xf4le (plus de 10 000 emplois) | 40 | 22 | 7 | 27 | 12 | |
| ## 2 | 112 - Couronne d'un grand p\xf4le | 102 | 166 | 65 | 157 | 45 | |
| ## 3 | 120 - Multipolaris\xe9e des grandes aires urbaines | 26 | 31 | 20 | 59 | 50 | |
| ## 4 | 211 - Moyen p\xf4le (5 000 \xe0 10 000 emplois) | 8 | 2 | 4 | 1 | 8 | |
| ## 5 | 212 - Couronne d'un moyen p\xf4le | 12 | 5 | 21 | 2 | 4 | |
| ## 6 | 221 - Petit p\xf4le (de 1 500 \xe0 5 000 emplois) | 6 | 27 | 5 | 14 | 17 | |
| ## 7 | 222 - Couronne d'un petit p\xf4le | 0 | 0 | 4 | 4 | 0 | |
| ## 8 | 300 - Autre commune multipolaris\xe9e | 27 | 83 | 80 | 90 | 106 | |
| ## 9 | 400 - Commune isol\xe9e hors influence des p\xf4les | 0 | 27 | 55 | 21 | 40 | |

```
chisq.test (croisement)
```

```
## Warning in chisq.test(croisement): Chi-squared approximation may be
## incorrect
```

```
##
## Pearson's Chi-squared test
##
## data: croisement
```

```
## X-squared = 316.14, df = 32, p-value < 2.2e-16
```

Problème : il y a dans certaines cases du tableau de contingence des effectifs inférieurs à 5. On va donc créer des regroupements :

- 111 et 112 (grands pôles)
- 120 et 300 (multipolarisées)
- 211 212 221 22 et 400 (rurale ou petits pôles)

```
levels (pdl$ZAU)
```

```
## [1] "111 - Grand pôle (plus de 10 000 emplois)"
## [2] "112 - Couronne d'un grand pôle"
## [3] "120 - Multipolarisée des grandes aires urbaines"
## [4] "211 - Moyen pôle (5 000 à 10 000 emplois)"
## [5] "212 - Couronne d'un moyen pôle"
## [6] "221 - Petit pôle (de 1 500 à 5 000 emplois)"
## [7] "222 - Couronne d'un petit pôle"
## [8] "300 - Autre commune multipolarisée"
## [9] "400 - Commune isolée hors influence des pôles"

pdl <- pdl %>%
  mutate (ZAU_regroupées = fct_collapse (ZAU, 'Grand pôle' = levels (pdl$ZAU)[1:2],
                                         'Multipolarisées' = levels (pdl$ZAU)[c(3,8)],
                                         'Rurales et petits pôles' = levels (pdl$ZAU)[c(4:7)]))

croisement <- table (pdl$DEP, pdl$ZAU_regroupées)

DEP_vs_ZAU_regroupées <- croisement %>%
  as.data.frame () %>%
  spread (key = Var1, value = Freq) %>%
  rename (ZAU = Var2)

DEP_vs_ZAU_regroupées

##          ZAU 44 49 53 72 85
## 1 Grand pôle 142 188 72 184 57
## 2 Multipolarisées 53 114 100 149 156
## 3 Rurales et petits pôles 26 61 89 42 69

test <- chisq.test (croisement)
test

##
## Pearson's Chi-squared test
##
## data: croisement
## X-squared = 174.29, df = 8, p-value < 2.2e-16
```

Il y a bien un lien entre les variables *ZAU_regroupées* et *DEP*. Si l'on veut en savoir plus on peut comparer les effectifs observés dans le tableau de contingence aux effectifs attendus si les variables étaient indépendantes l'une de l'autre.

```
predict <- t(round(test$expected)) %>% as.data.frame()
names(predict) = paste0(names(predict), 'p')

prov <- bind_cols(x = DEP_vs_ZAU_regroupées, y = predict)
prov
```

| | ZAU | 44 | 49 | 53 | 72 | 85 | 44p | 49p | 53p | 72p | 85p |
|------|-------------------------|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|
| ## 1 | Grand pôle | 142 | 188 | 72 | 184 | 57 | 95 | 155 | 112 | 161 | 121 |
| ## 2 | Multipolarisées | 53 | 114 | 100 | 149 | 156 | 84 | 138 | 99 | 143 | 107 |
| ## 3 | Rurales et petits pôles | 26 | 61 | 89 | 42 | 69 | 42 | 69 | 50 | 72 | 54 |

9.2.4 Partie 4

Y a-t-il un lien entre la densité de population et le taux d'emploi en région PDL ?

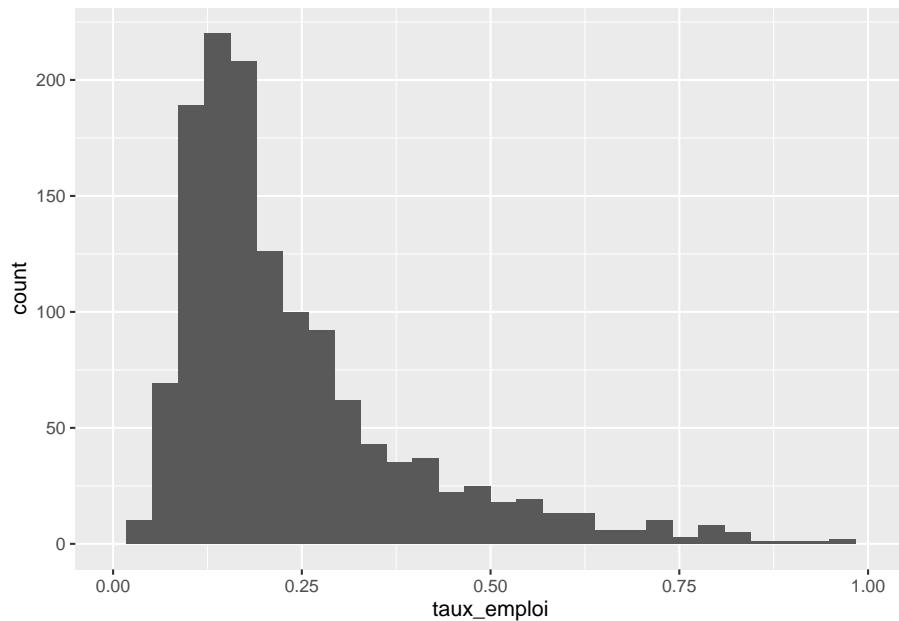
On crée un sous-jeu de données ad hoc.

```
data_p4 <- dat %>%
  select(1:15, P09_EMPLT) %>%
  filter(REG == '52') %>%
  mutate(taux_emploi = P09_EMPLT / P09_POP,
         densite_pop = P14_POP / SUPERF)
summary(data_p4$taux_emploi)
```

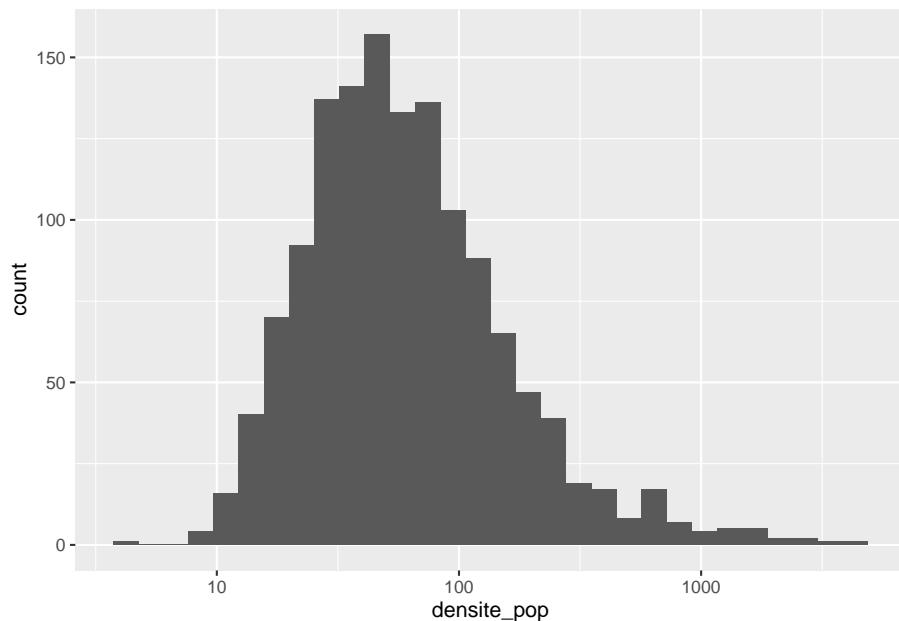
| | Min. | 1st Qu. | Median | Mean | 3rd Qu. | Max. | NA's |
|----|---------|---------|---------|---------|---------|---------|------|
| ## | 0.00979 | 0.13147 | 0.18493 | 0.24570 | 0.29610 | 1.59801 | 145 |

Avant de regarder le lien entre les deux variables, on les examine chacune à leur tour.

```
ggplot(data = data_p4, aes(x = taux_emploi)) +
  geom_histogram() +
  scale_x_continuous(limits = c(0, 1))
```



```
ggplot (data = data_p4, aes (x = densite_pop)) +  
  geom_histogram () +  
  scale_x_continuous (trans = 'log', breaks = c(1,10,100,1000))
```



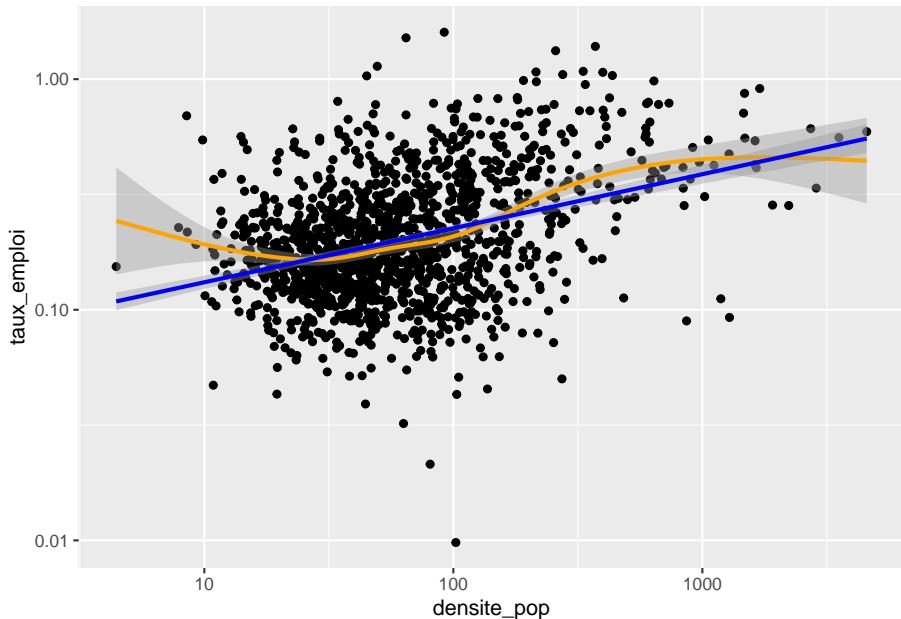
Ces distributions ne sont pas trop normales à première vue. Mais en testant ?

```
shapiro.test (data_p4$taux_emploi)
```

```
##  
## Shapiro-Wilk normality test  
##  
## data: data_p4$taux_emploi  
## W = 0.77523, p-value < 2.2e-16  
shapiro.test (data_p4$densite_pop)  
  
##  
## Shapiro-Wilk normality test  
##  
## data: data_p4$densite_pop  
## W = 0.31672, p-value < 2.2e-16
```

C'est confirmé -> en toute rigueur, test non paramétrique. On regarde aussi le nuage de points.

```
ggplot (data=data_p4, aes (x = densite_pop, y = taux_emploi)) +  
  geom_point () +  
  scale_x_continuous (trans = 'log10') +  
  scale_y_continuous (trans = 'log10') +  
  geom_smooth (color = 'orange') +  
  geom_smooth (method = 'lm', color = 'blue')
```



```
cor (x = data_p4$taux_emploi,
     y = data_p4$densite,
     use = "pairwise.complete.obs",
     method = "spearman")
```

```
## [1] 0.3121057
```

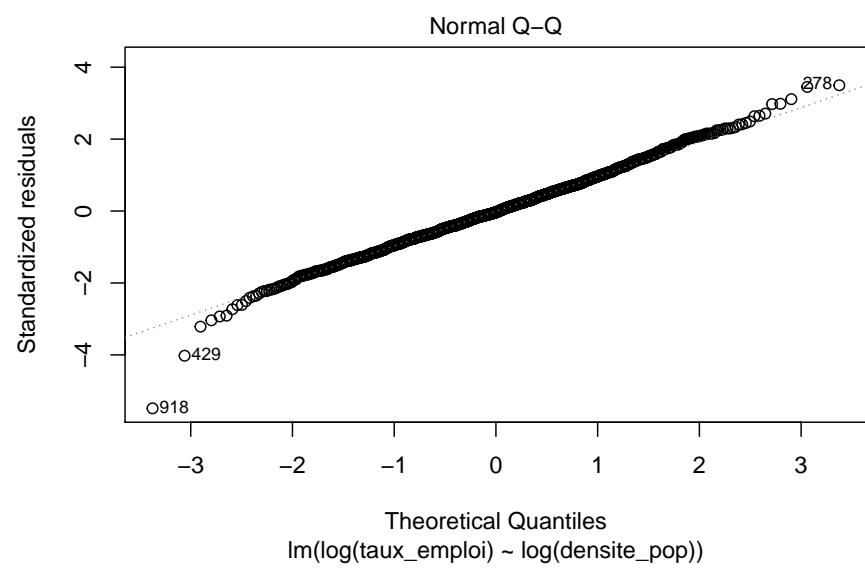
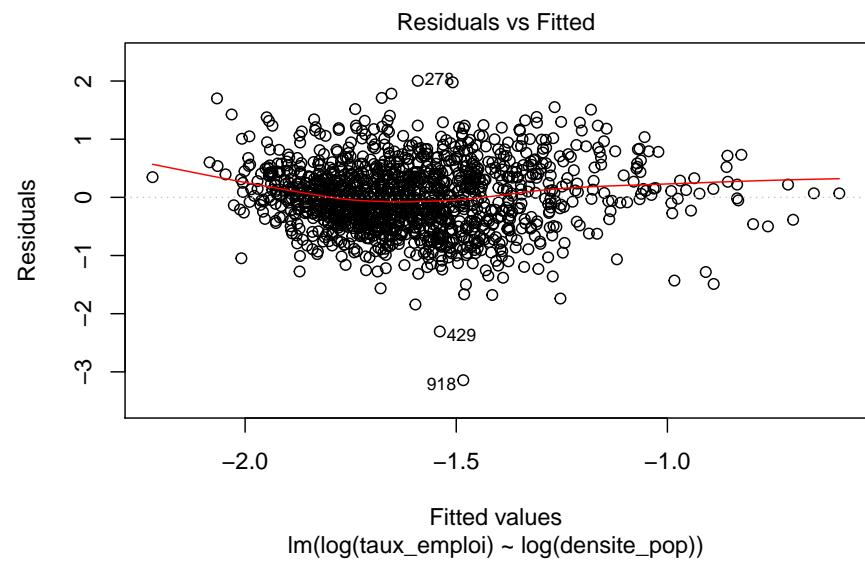
Conclusion : Le coefficient de corrélation est positif, faible mais non négligeable. Les variables tendent à varier dans le même sens.

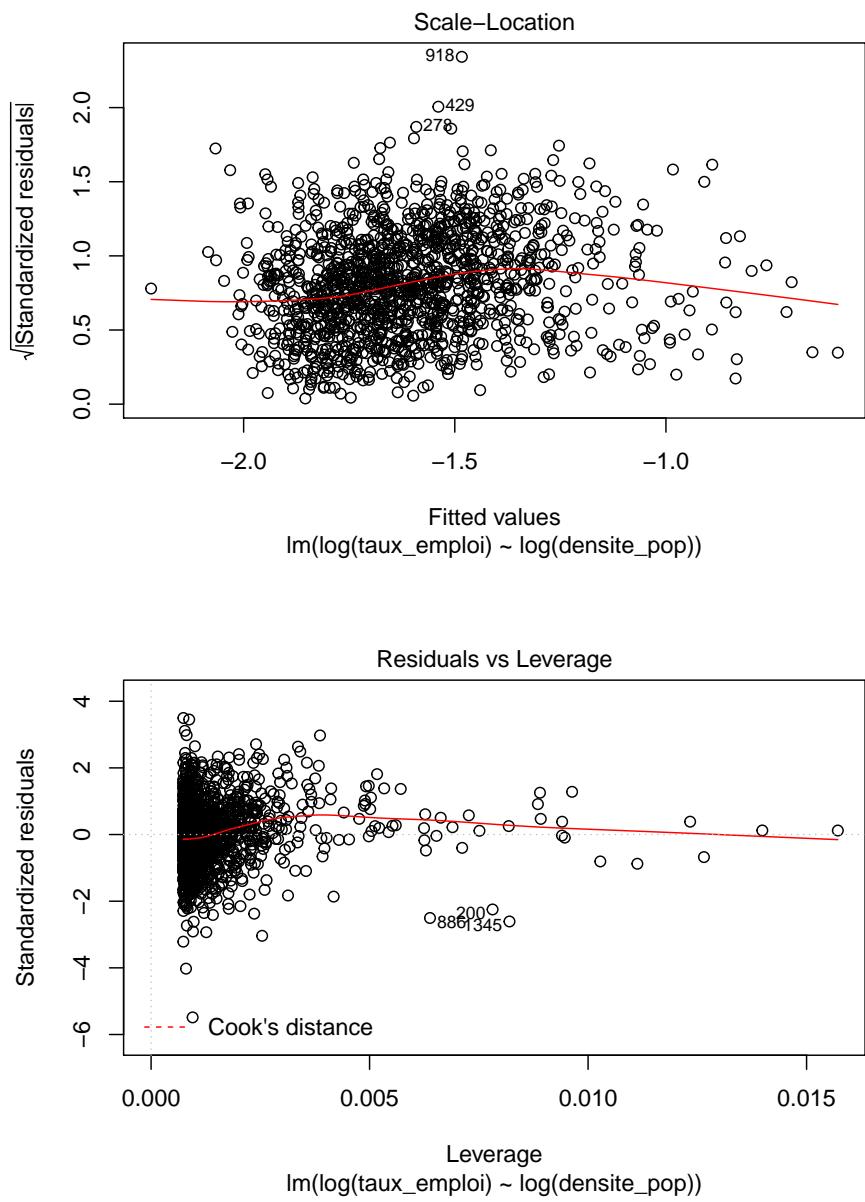
Est-ce que la variable *densite* contribue significativement à expliquer la variabilité de la variable *taux_emploi* ?

Pour répondre à cette question, on est bien embêtés dans un cadre non paramétrique. Pour l'approcher, on peut donc faire comme si on n'avait pas vu que les distributions des variables n'étaient pas normales.

```
modele <- lm (log (taux_emploi) ~ log (densite_pop), data = data_p4)
summary (modele)
```

```
##
## Call:
## lm(formula = log(taux_emploi) ~ log(densite_pop), data = data_p4)
##
## Residuals:
##      Min       1Q   Median       3Q      Max
## -3.14328 -0.37613 -0.02266  0.36720  2.00382
##
## Coefficients:
##             Estimate Std. Error t value Pr(>|t|)
## (Intercept) -2.56785   0.06862 -37.42  <2e-16 ***
## log(densite_pop) 0.23434   0.01625  14.42  <2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 0.5732 on 1355 degrees of freedom
##   (145 observations deleted due to missingness)
## Multiple R-squared:  0.133, Adjusted R-squared:  0.1323
## F-statistic: 207.8 on 1 and 1355 DF,  p-value: < 2.2e-16
plot (modele)
```





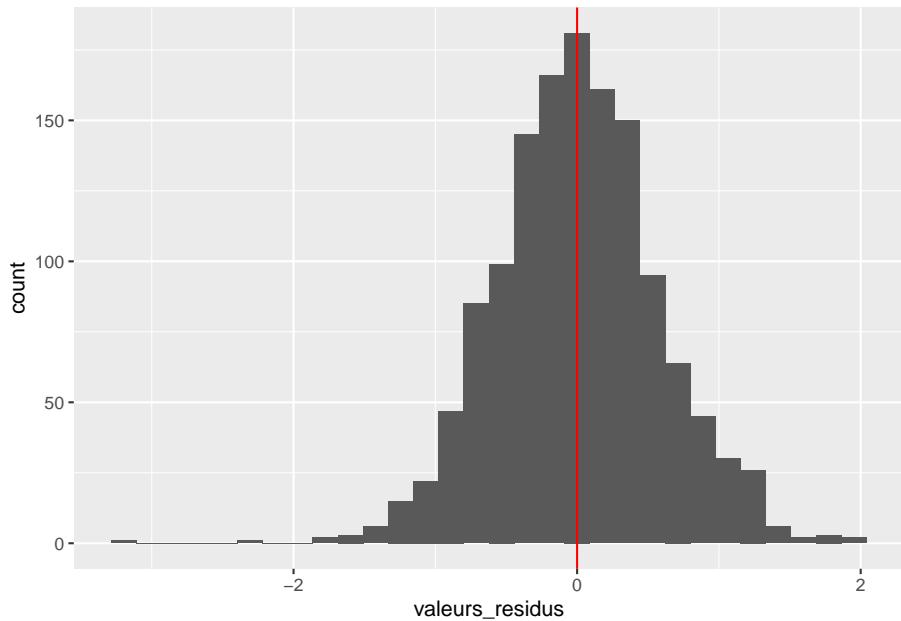
```

residus <- modele$residuals %>%
  as.data.frame () %>%
  rename (valeurs_residus = ' ')
  
```

```

ggplot (data = residus, aes (x = valeurs_residus)) +
  
```

```
geom_histogram () +
  geom_vline (xintercept = 0, color = 'red')
```



Les graphiques montrent que :

- on a à peu près indépendance entre les valeurs prédites et les résidus
- les résidus sont à peu près distribués normalement avec une distribution centrée en zéro
- on a quelques points avec des bras de leviers (distance de Cook) importants, mais le modèle les prédit bien.

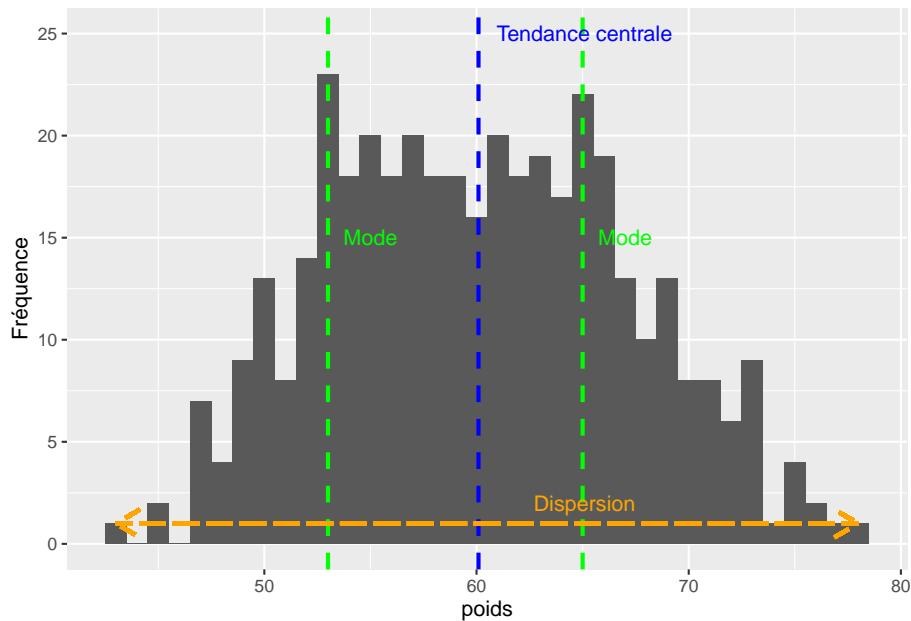
Donc ce modèle n'est pas catastrophique. On peut s'aventurer à lire les valeurs des coefficients et le R^2 ajusté qui vaut 0,1323. Le taux d'emploi tend à augmenter avec la densité de population, selon une relation log-log.

Chapter 10

Une variable quantitative

Il existe plusieurs caractéristiques à étudier sur la distribution d'une variable quantitative :

- La **tendance centrale** : le point autour duquel se regroupent les différentes valeurs d'une variable (mesurée par la médiane, le mode, la moyenne, etc...)
- La **dispersion** : l'étendue des différentes valeurs (donc l'hétérogénéité de la population) que peut prendre une variable (mesurée par l'écart-type, la variance, l'écart interquartiles, etc...)
- L'éventuelle présence de plusieurs **modes**



10.1 Statistiques de distribution

10.1.1 Statistiques robustes aux valeurs extrêmes

Ce sont des statistiques qui sont peu influencées par les valeurs extrêmes de la distribution. Elles sont dites “non paramétriques” car elles ne supposent aucune distribution particulière de la variable.

- La **médiane** est une statistique de tendance centrale : après classement par ordre croissant des n valeurs, c'est la valeur de l'observation “du milieu”. Il y a autant d'observations inférieures à la médiane que d'observations supérieures. Avec R, la fonction `median()` permet d'obtenir la médiane d'une variable quantitative. Si le nombre de valeurs n est impair ($n = 2k + 1$), c'est la valeur exacte observée au rang $k + 1$. Si le nombre de valeurs est pair ($n = 2k$), c'est la moyenne entre les valeurs observées aux rangs k et $k + 1$. En cas de valeurs manquantes sur la variable, l'argument `na.rm` permet de ne prendre en compte que les valeurs effectives.

```
vec_superf <- pull (dat, SUPERF)
median (vec_superf)
```

```
## [1] NA
```

```
median (vec_superf, na.rm = T)
```

```
## [1] 10.81
```

- Les **quartiles** : après classement par ordre croissant, on partage en 4, il y a donc 3 bornes. Le premier quartile est la valeur qui sépare les observations telle que 25% soient en dessous et 75% au dessus. Le deuxième quartile est la médiane. Le troisième quartile divise enfin les 75% observations les plus basses des 25% plus élevées.
- On peut généraliser aux déciles, aux centiles ou tout autre pourcentile. La fonction **quantile()** permet d'obtenir les percentile de son choix (par défaut, la fonction retourne les quartiles).

```
quantile (vec_superf, na.rm = T)
```

```
##      0%      25%      50%      75%     100%
##    0.04    6.44   10.81   18.58 18360.00
```

```
quantile (vec_superf, probs = c (0.05, 0.1, 0.25, 0.5, 0.75, 0.90, 0.95), na.rm = T)
```

```
##      5%     10%     25%     50%     75%     90%     95%
##  3.170  4.140  6.440 10.810 18.580 30.743 41.240
```

- L'**intervalle interquartile** (Interquartile range en anglais) est une statistique de dispersion. C'est l'intervalle entre le 1^{er} et le 3^{ème} quartile : il encadre 50% des observations. Il se calcule facilement par Q3 - Q1, ou au moyen de la fonction **IQR()**.

```
q <- quantile (vec_superf, na.rm = T)
print (q)
```

```
##      0%      25%      50%      75%     100%
##    0.04    6.44   10.81   18.58 18360.00
```

```
q["75%"] - q["25%"]
```

```
##    75%
## 12.14
```

```
IQR (vec_superf, na.rm = T)
```

```
## [1] 12.14
```

10.1.2 Statistiques sensibles aux valeurs extrêmes

Ce sont des statistiques qui sont influencées par les valeurs extrêmes de la distribution.

- Quand le terme de moyenne est employé sans plus de précision, il désigne la **moyenne arithmétique**. C'est une statistique de tendance centrale :

$$\bar{x} = \frac{1}{n} \sum_{i=1}^n x_i$$

C'est la valeur qu'aurait chacune des observations si la répartition était égale entre les individus statistiques. La fonction `mean()` permet d'obtenir la moyenne.

```
mean (vec_superf, na.rm = T)
```

```
## [1] 17.64062
```

Quand la variable est discrète, la moyenne arithmétique peut être pondérée en utilisant la fonction `weighted.mean()`.

Exemple : quel est le nombre moyen de pièces par appartement dans une localité ?

```
immo <- data.frame (nb_pieces = 1:7,
                     nb_apparts = c(38, 61, 92, 67, 43, 18, 5))

datatable (immo, width = 400,
           colnames = c("Nombre de pièces", "Nombre d'appartements"),
           rownames = FALSE)
```

Show entries
 Search:

| Nombr e de pièces | Nombr e d'appartements |
|---|--|
| 1 | 38 |
| 2 | 61 |
| 3 | 92 |
| 4 | 67 |
| 5 | 43 |
| 6 | 18 |
| 7 | 5 |

Showing 1 to 7 of 7 entries

Previous Next

```
weighted.mean (x = immo$nb_pieces, w = immo$nb_apparts)
```

```
## [1] 3.277778
```

- La **moyenne géométrique** est aussi une statistique de tendance centrale : $\bar{x} = \sqrt[n]{\prod_{i=1}^n x_i}$

Typiquement, la moyenne géométrique est utilisée pour calculer un taux de variation moyen sur une période. Par exemple si l'on a des données sur quatre années, et des taux de variation annuels de +12%, +4%, +7% et -2% avec une

valeur de départ de 100, la valeur finale est :

$$Vf = 100 \times 1,12 \times 1,04 \times 1,07 \times 0,98 = 122,1$$

Le taux de variation moyen annuel est de :

$$Tm = \sqrt[4]{1,12 \times 1,04 \times 1,07 \times 0,98} - 1 = 0,051$$

soit + 5,1% par an.

Cas où l'on a juste la valeur de départ, la valeur finale et le nombre de pas de temps :

```
Vd <- 100
Vf <- 100 * 1.12 * 1.04 * 1.07 * 0.98
Annees <- 4
taux_moyen_annuel <- (Vf / Vd) ^ (1 / Annees)

100 * taux_moyen_annuel ^ Annees # vérification
```

[1] 122.1409

On peut calculer la moyenne géométrique ainsi :

```
exp (mean (log (vec_superf), na.rm = T)) # alternative : la fonction heR::gm()
```

[1] 11.06955

- La **variance** est une statistique de dispersion : $S_x^2 = \frac{1}{n-1} \sum_{i=1}^n (x_i - \bar{x})^2$

C'est la moyenne des carrés des écarts à la moyenne. Son unité est le carré de celle de la variable sur laquelle on la calcule. La fonction **var()** permet de la calculer.

```
var (vec_superf, na.rm = T)
```

[1] 21166.58

- L'**écart type** est une statistique de dispersion. Il est défini par : $\sigma_x = \sqrt{S_x^2}$. C'est l'écart moyen à la moyenne. L'écart-type (*standard deviation* en anglais) est dans la même unité que la variable à laquelle on l'applique. L'écart-type s'obtient avec la fonction **sd()**

```
sd (vec_superf, na.rm = T)
```

[1] 145.4874

- L'**étendue** est l'écart entre les valeurs maximale et minimale.

```
etendue <- pull (dat, P14_RP_PROP) %>% range (na.rm = T) %>% round (0)
etendue
```

[1] 0 381934

- Le **coefficent de variation** est une statistique de dispersion relative. C'est une mesure de l'écart relatif des valeurs d'une distribution à une valeur centrale. Il permet de comparer l'hétérogénéité de distributions de variables qui ne seraient pas du même ordre de grandeur (une variable prenant ses valeurs entre 1000 et 2000 aura naturellement une "dispersion brute" plus importante qu'une variable prenant ses valeurs entre 10 et 20). Le coefficient de variation est obtenu comme ratio de l'écart-type sur la moyenne donc, comme ces deux valeurs, il est sensible aux valeurs extrêmes. Le coefficient de variation n'a pas d'unité donc il est souvent exprimé en pourcentage.

```
v <- pull (dat, P14_POP)
ecart_t <- sd (v, na.rm = T)
moy <- mean (v, na.rm = T)
ecart_t/moy

## [1] 8.132898
```

La fonction **summary()** fournit plusieurs statistiques (appliquée à une variable quantitative, on obtient : minimum, maximum, quartile, moyenne et nombre de valeurs manquantes) :

```
select (dat, P14_POP, SUPERF) %>% summary ()

##      P14_POP           SUPERF
##  Min.   :    0   Min.   :  0.04
##  1st Qu.: 197   1st Qu.:  6.44
##  Median : 444   Median : 10.81
##  Mean   : 1838   Mean   : 17.64
##  3rd Qu.: 1110   3rd Qu.: 18.58
##  Max.   :2220445  Max.   :18360.00
##  NA's    :821     NA's   :821
```

10.2 Calcul sur plusieurs variables

Il est souvent pratique d'appliquer une même fonction à plusieurs variables d'un **dataframe**.

Sur les variables quantitatives :

```
dat %>%
  summarise_if (is.numeric, funs(mean(., na.rm = TRUE))) %>%
  round (digits = 2) %>%
  t() %>%
  datatable (rownames = TRUE, colnames = c('Variable', 'Moyenne'), width = 300)
```

Show entries
 Search:

| Variable | Moyenne |
|----------|---------|
| P14_POP | 1837.49 |
| P09_POP | 1792.81 |
| SUPERF | 17.64 |
| NAIS0914 | 114.44 |
| DECE0914 | 77.35 |
| P14_MEN | 802 |
| NAISD15 | 21.97 |
| DECESD15 | 16.47 |
| P14_LOG | 970.23 |
| P14_RP | 802 |

Showing 1 to 10 of 25 entries

| | | | |
|----------|---|---|---|
| Previous | 1 | 2 | 3 |
|----------|---|---|---|

Next

Sur des variables appelées par leur nom :

```
dat %>%
  summarise_at (vars(P14_POP, SUPERF), funs(mean(., na.rm = TRUE)))

##      P14_POP    SUPERF
## 1 1837.492 17.64062
```

On peut également décliner selon les modalités d'une variable qualitative :

```
dat %>%
  group_by (REG) %>%
  summarise_at (vars(P14_POP, SUPERF), funs(mean(., na.rm = TRUE))) %>%
  mutate_if (is.numeric, round, digits = 2) %>%
  datatable (width = 500)
```

Show entries Search:

| | REG | P14_POP | SUPERF |
|----|-----|----------|--------|
| 1 | 01 | 12505.81 | 50.89 |
| 2 | 02 | 11291.5 | 33.18 |
| 3 | 03 | 11469.91 | 3797 |
| 4 | 04 | 35115.29 | 104.32 |
| 5 | 11 | 9411.24 | 9.4 |
| 6 | 24 | 1423.21 | 21.62 |
| 7 | 27 | 746.79 | 12.65 |
| 8 | 28 | 1156.2 | 10.37 |
| 9 | 32 | 1573.12 | 8.33 |
| 10 | 44 | 1078.15 | 11.15 |

Showing 1 to 10 of 17 entries

| | | | |
|----------|-------------------|---|------|
| Previous | 1 | 2 | Next |
|----------|-------------------|---|------|

10.2.1 Exercice

Comparer le coefficient de variation de plusieurs variables. Laquelle est la plus “relativement étendue” ?

10.3 Représentations graphiques

Plusieurs représentations permettent de visualiser la distribution des valeurs d'une variable quantitative. Les principales sont :

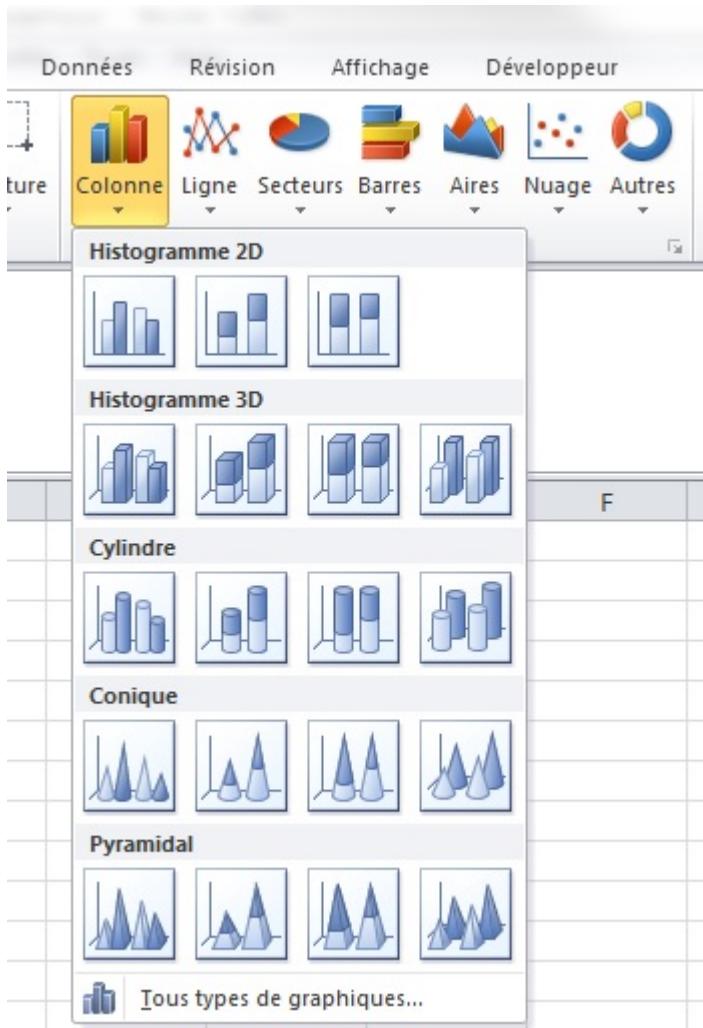
- l'histogramme
- le diagramme de densité

10.3.1 Histogramme

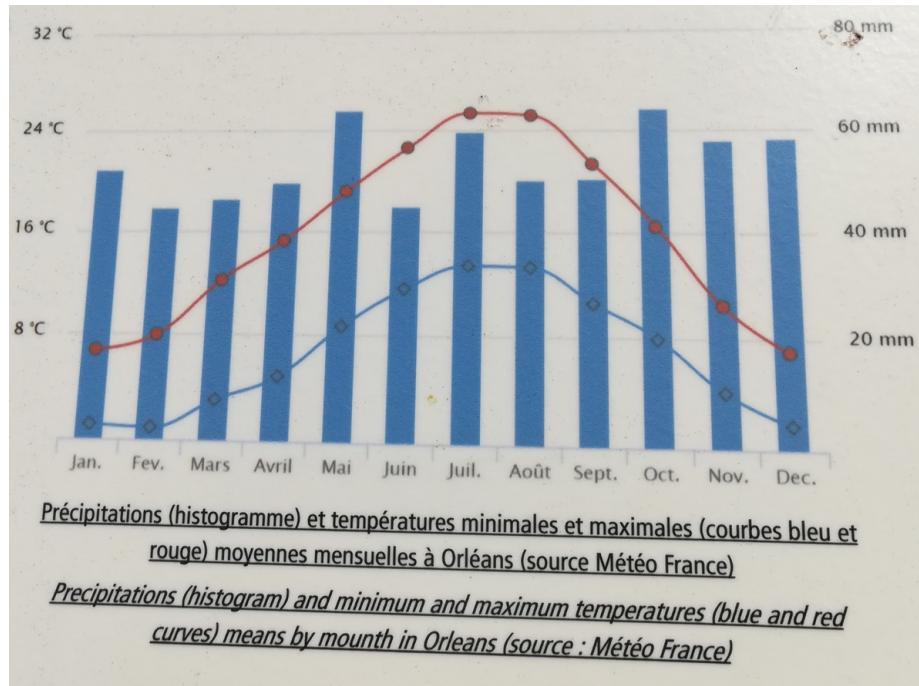
C'est la représentation la plus classique, qui permet d'observer la fréquence d'observations dans différents intervalles de valeurs.

A ne pas confondre avec le diagramme en barres / bâtons !

Beaucoup de confusions entre ces deux types de graphiques, même dans des journaux “de référence” comme dans cet article de la rubrique “Les décodeurs” du Monde et dans Excel®.



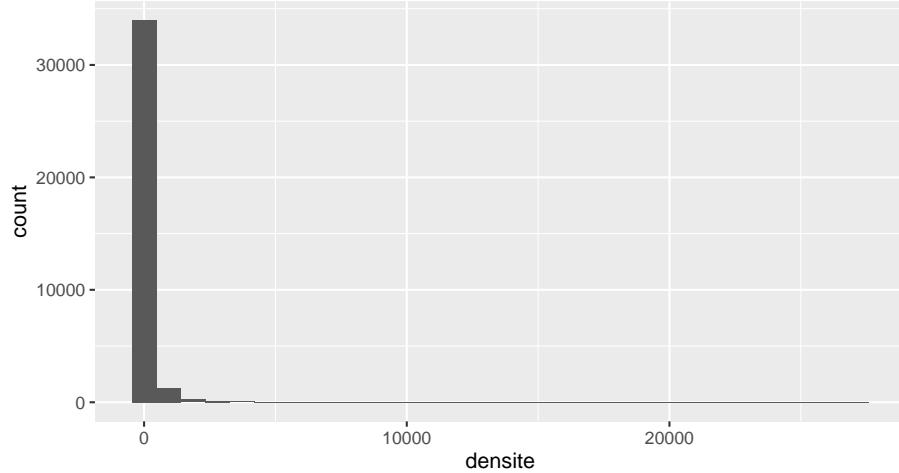
... ou dans l'*arboretum* des Barres, là où ont été formés la plupart des ingénieurs forestiers des ministères de l'agriculture et de l'environnement.



Ça c'est un diagramme en bâtons, contrairement à ce qu'indique la légende.

Un histogramme, c'est ça !

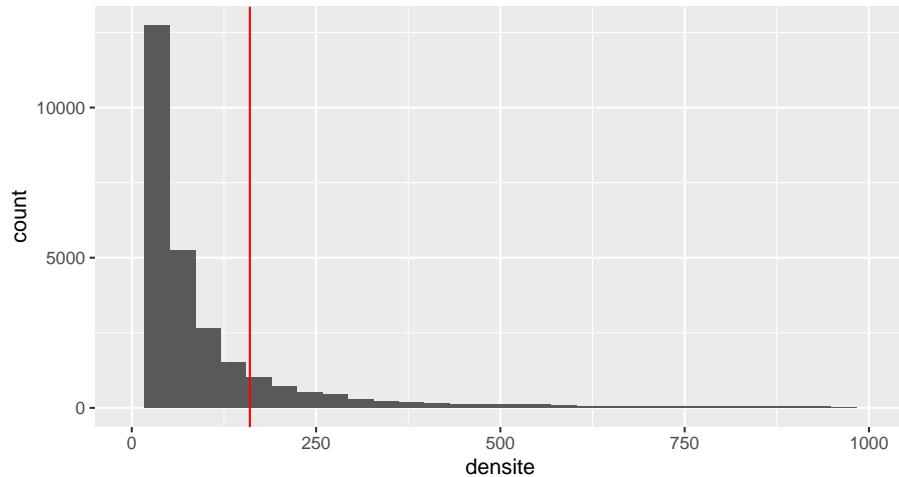
```
dat <- mutate (dat, densite = P14_POP/SUPERF,
               densite = replace (densite, densite == 0, 0.01))
# pour éviter les densités nulles (si on souhaite faire un passage au log, voir plus bas)
g <- ggplot (data = dat, aes (densite)) + geom_histogram ()
g
```



Il n'y a qu'une variable considérée. L'axe des abscisses est numérique et couvre la gamme des valeurs prises par la variable considérée. L'axe des ordonnées indique le nombre des valeurs (ou leur fréquence en pourcentage) dans chaque intervalle. L'histogramme est obtenu grâce à la fonction `geom_histogram()`, appliquée à la suite de la fonction `ggplot()` (qui, elle, permet de définir les variables qui seront prises en compte pour le graphique ; cf. le module 5 qui devrait être proposé à partir de 2019).

Les quelques valeurs très élevées rendent le graphique illisible. On limite la gamme des valeurs de l'axe des abscisses.

```
g + scale_x_continuous (limits = c (0, 1000)) +
  geom_vline (xintercept = mean (dat$densite, na.rm = T), color = 'red')
```



Ici, peu de villes ont une densité importante, et la plupart sont concentrées sur

la première barre : c'est un phénomène d'asymétrie (*skewness* en anglais).

La fonction `geom_vline()` permet d'ajouter une ligne verticale, afin de mettre en évidence des valeurs particulières (ex : moyenne, seuil réglementaire). Ici, la moyenne (ligne verticale rouge) est un mauvais indicateur de tendance centrale.

Si l'on est chagriné par cette distribution, on peut appliquer à la variable une *transformation logarithmique*. Le logarithme “tasse” les valeurs élevées et “étire” les valeurs faibles, cela permet de zoomer là où il y a de la donnée.

```
dat <- mutate (dat, log_dens = log10 (densite))
summarise (dat, mini = min (densite, na.rm = T), maxi = max (densite, na.rm = T))

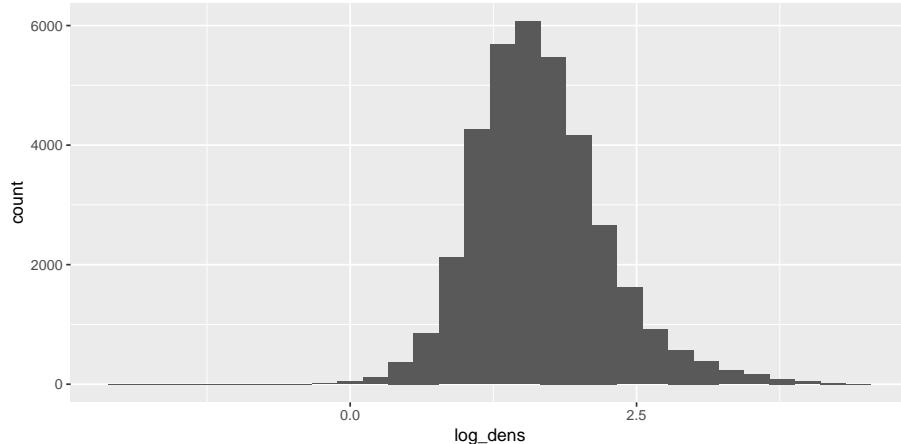
##    mini      maxi
## 1 0.01 27126.14

summarise (dat, mini = min (log_dens, na.rm = T), maxi = max (log_dens, na.rm = T))

##    mini      maxi
## 1   -2 4.433388
```

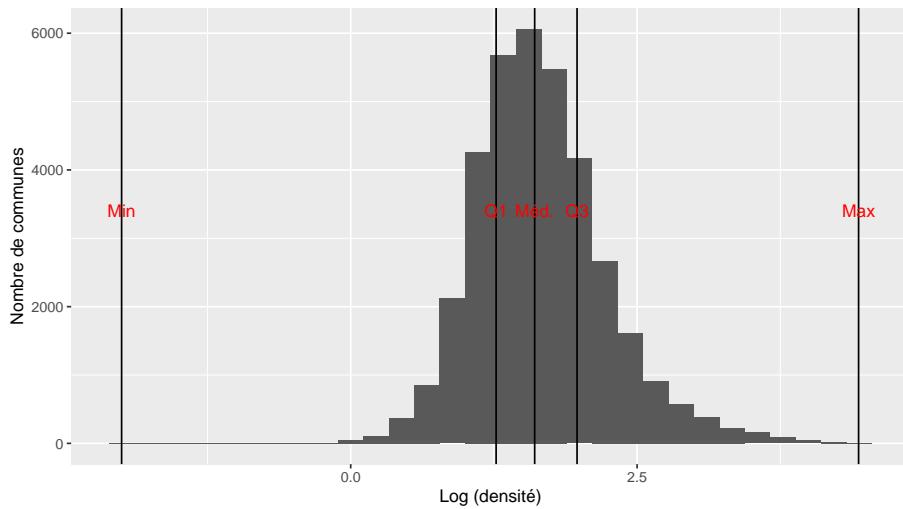
Voyons à quoi ressemble l'histogramme de la variable transformée.

```
ggplot (data = dat, aes (log_dens)) + geom_histogram ()
```



C'est déjà bien plus lisible ... mais quelques finitions peuvent le rendre plus informatif et “auto-porteur”.

```
quan <- pull (dat, log_dens) %>% quantile (na.rm = T)
ggplot (data = dat, aes (x = log_dens)) + geom_histogram () +
  geom_vline (xintercept = quan) +
  annotate (geom = "text", x = quan,
            y = 3400, label = c ("Min", "Q1", "Méd.", "Q3", "Max"), colour = "red") +
  xlab (label = "Log (densité)") +
  ylab (label = "Nombre de communes")
```

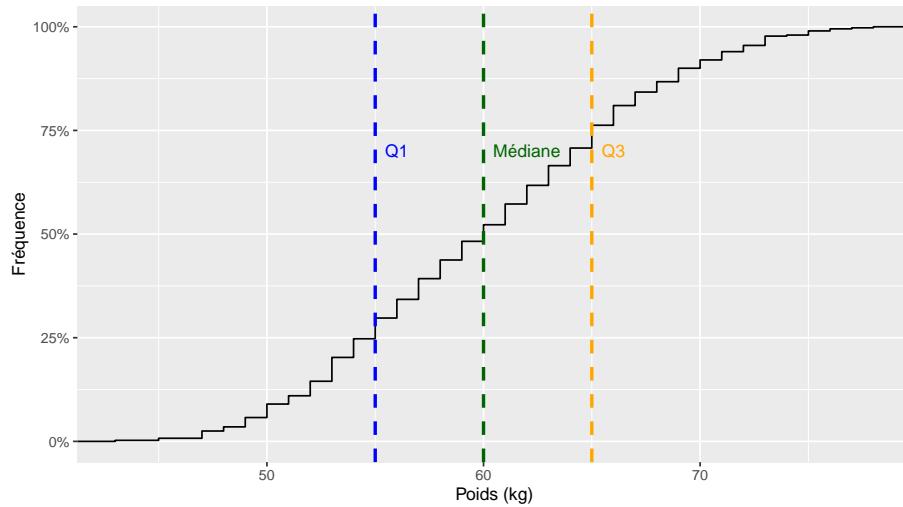


L'histogramme est plus lisible et informatif ! La fonction `annotate()` permet d'ajouter du texte, et enfin `xlab()` et `ylab()` servent à modifier le nom des axes.

10.3.2 La fonction de répartition (cumulative)

```
quan <- pull(poids_sex, poids) %>% quantile()

ggplot (data = poids_sex, aes (x = poids)) + stat_ecdf () +
  xlab (label = "Poids (kg)") +
  ylab (label = "Fréquence") +
  scale_y_continuous (labels = scales::percent) +
  geom_vline(aes(xintercept = quan[2]), color = "blue", linetype = "dashed", size = 1) +
  geom_vline(aes(xintercept = quan[3]), color = "darkgreen", linetype = "dashed", size = 1) +
  geom_vline(aes(xintercept = quan[4]), color = "orange", linetype = "dashed", size = 1) +
  annotate (geom = "text", x = c (quan[2]+1, quan[3]+2, quan[4]+1),
            y = 0.7, label = c ("Q1", "Médiane", "Q3"), colour = c("blue", "darkgreen", "orange"))
```



10.3.3 Graphique de densité

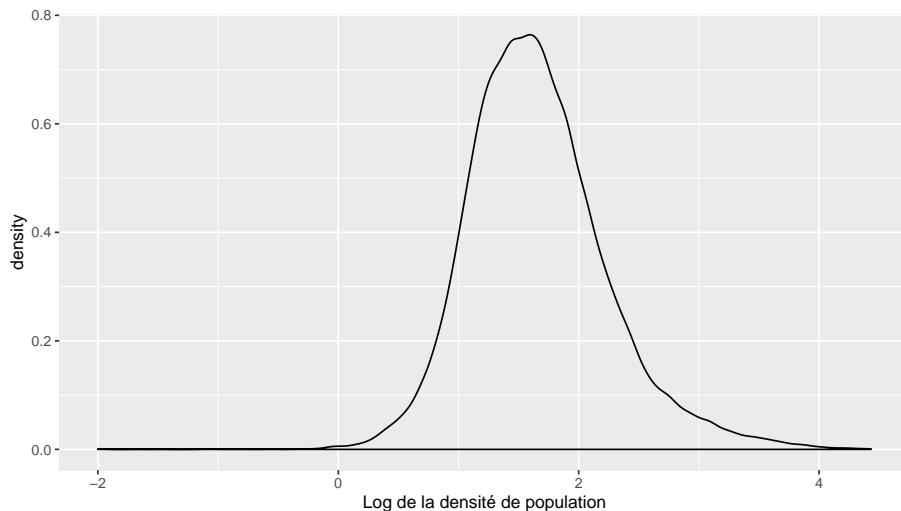
La **densité** d'une variable correspond à la probabilité de réalisation de chacune des valeurs possibles de cette variable : il s'agit d'une fonction continue (nombre infini de points). Exemple pour $X \hookrightarrow \mathcal{N}(m, \sigma)$: $f_X(t) = \frac{1}{\sigma\sqrt{2\pi}} \exp\left(-\frac{(t-m)^2}{2\sigma^2}\right)$

Comme on n'observe qu'un nombre fini de points, on recourt à l'estimation par “noyau” pour estimer la densité en chaque point “possible” de la variable.

La valeur de la densité en t se calcule comme une moyenne pondérée de l'ensemble des points observés. Plus la valeur est éloignée de t , moins leur poids est important. Le poids est donné par une fonction appelée *noyau*. Par exemple, le noyau gaussien : $K(u) = \frac{1}{\sqrt{2\pi}} \exp\left(-\frac{u^2}{2}\right)$

Grâce au package `ggplot2`, il est possible de créer un graphique de densité avec la fonction `geom_density()`

```
ggplot (data = dat, aes (log_dens)) +
  geom_density () +
  xlab ('Log de la densité de population')
```



10.4 Discréétisation d'une variable continue

La **discréétisation** est l'opération qui permet de découper une série de données quantitatives en classes : elle est très utilisée pour les représentations cartographiques. La création des classes dépend fortement de la forme de la distribution. Plusieurs méthodes existent, par exemple :

- Seuils naturels (Jenks, coûteuse en calcul)
- Moyenne et écart type
- Progression géométrique (adaptée aux distributions asymétriques)
- Méthode des quantiles (souvent par défaut)

10.4.1 Méthode des quantiles

Cette méthode vise une répartition égale des effectifs, c'est à dire le même nombre de données observations par classe. La fonction `cut()` permet de créer une variable selon le découpage voulu. Elle possède les options *breaks* et *labels*. *breaks* permet d'indiquer le découpage voulu grâce à un vecteur (les éléments dont la valeur est comprise entre le 1er et le 2ème élément de ce vecteur rentre dans la première classe, entre le 2ème et 3ème élément, dans la 2ème classe, etc...). Pour faire un découpage en *x* classes, le vecteur de *breaks* prend donc *x* + 1 éléments. *labels* permet d'indiquer les noms de classes grâce à un vecteur qui comporte autant d'élément que de classes à créer.

```
v_dens <- pull (dat, densite)
v_dens %>% quantile (na.rm = T)
```

```

##          0%      25%      50%      75%      100%
##    0.01000   18.59047   40.35457   94.57430  27126.14108
dat <- mutate (dat, tr_dens = cut (v_dens,
                                    breaks = c (0, 18.6, 40.4, 94.6, 27127),
                                    labels = c ("inf_18.6", "18.6_40.4", "40.4_94.6", "Sup_94.6"))

```

10.4.2 Méthode des seuils naturels (méthode de Jenks)

Cette méthode revient à séparer les classes à partir des fortes discontinuités observées dans la distribution. La fonction `getBreaks()` permet d'obtenir les bornes des classes, selon la méthode voulue.

```

library (cartography)
seuils <- getBreaks (v, nclass = 4, method = "fisher-jenks")

## Warning in classInt::classIntervals(v, nclass, style = method): N is large,
## and some styles will run very slowly; sampling imposed
dat <- mutate (dat, tr_dens2 = cut (densite, breaks = seuils))
select (dat, tr_dens2) %>% table ()

## .
##      (0,1.44e+04] (1.44e+04,5.98e+04] (5.98e+04,4.87e+05]
##            35849                 19                  0
## (4.87e+05,8.58e+05]
##                      0

```