TP_Sirene

Benoît Blanc

10/10/2019

R Markdown

This is an R Markdown document. Markdown is a simple formatting syntax for authoring HTML, PDF, and MS Word documents. For more details on using R Markdown see http://rmarkdown.rstudio.com.

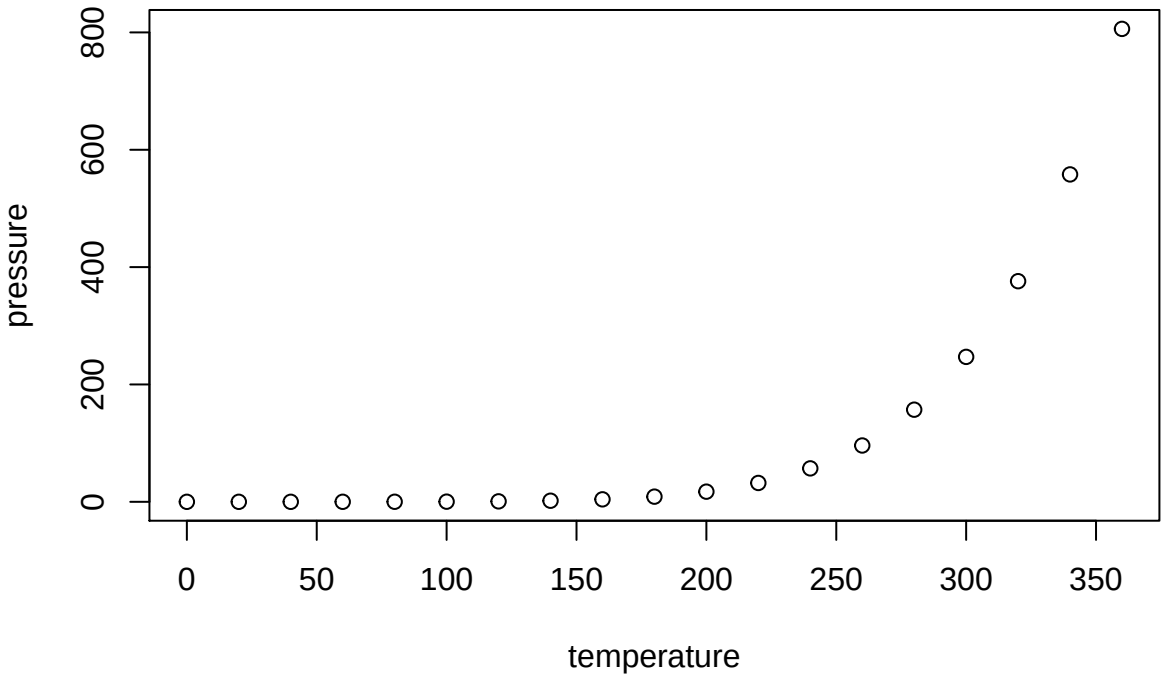
When you click the **Knit** button a document will be generated that includes both content as well as the output of any embedded R code chunks within the document. You can embed an R code chunk like this:

```
summary(cars)
```

```
##      speed           dist       
##  Min.   : 4.0   Min.   :  2.00  
##  1st Qu.:12.0   1st Qu.: 26.00  
##  Median :15.0   Median : 36.00  
##  Mean   :15.4   Mean   : 42.98  
##  3rd Qu.:19.0   3rd Qu.: 56.00  
##  Max.   :25.0   Max.   :120.00
```

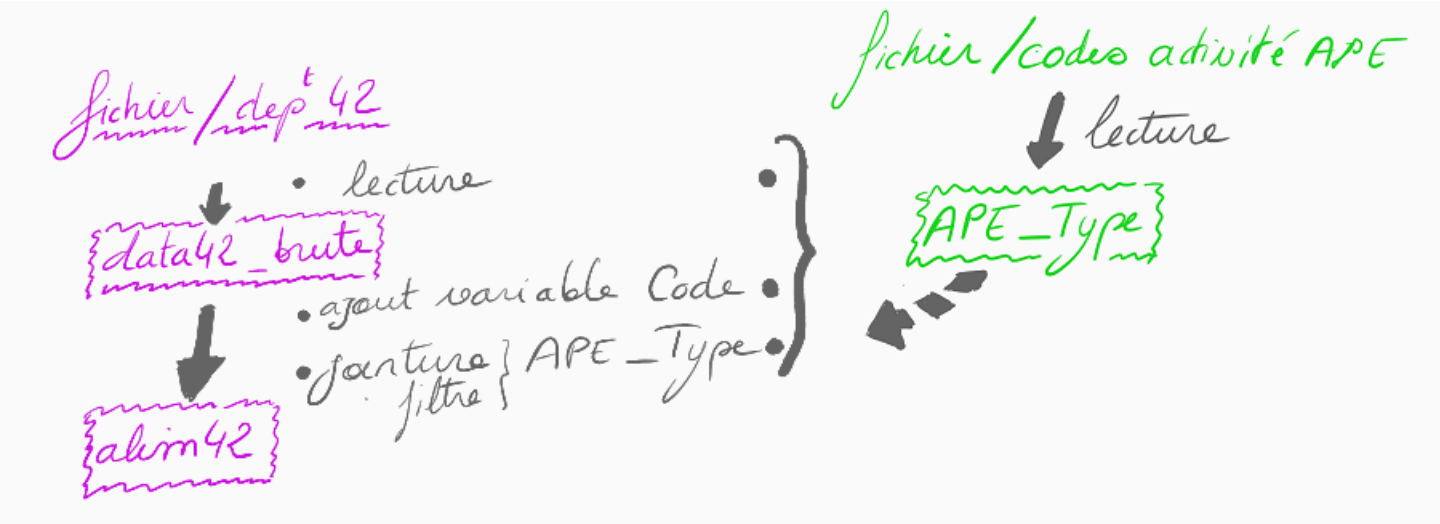
Including Plots

You can also embed plots, for example:



Note that the `echo = FALSE` parameter was added to the code chunk to prevent printing of the R code that generated the plot.

Mise en place



Installations, téléchargements, premiers tests sur le département 42

Téléchargez les données du département 42, geo_siret_42.csv dans ce répertoire et dézippez le dossier sur votre machine.

Téléchargez la table qui renseigne les codes correspondant à l'Activité Principale de l'Etablissement (APE)APE_Type.csv

```
library(dplyr)
```

```
##
## Attaching package: 'dplyr'
```

```
## The following objects are masked from 'package:stats':
##
##     filter, lag
```

```
## The following objects are masked from 'package:base':
##
##     intersect, setdiff, setequal, union
```

```
library(readr)
```

Lecture de tableaux de données

Depuis RStudio, créez un projet qui comprendra l'ensemble des données et documents nécessaires à réaliser l'ensemble des traitements qui vous seront demandés pour ce TP.

Créez le data.frame data42 en lisant la table geo-siret_42.csv.

```
data42 <- read_csv("data/geo_siret_42.csv")
```

```
## Parsed with column specification:
## cols(
##   .default = col_character(),
##   dateCreationEtablissement = col_date(format = ""),
##   anneeEffectifsEtablissement = col_double(),
##   activitePrincipaleRegistreMetiersEtablissement = col_logical(),
```

```
##   dateDernierTraitementEtablissement = col_datetime(format = ""),
##   etablissementSiege = col_logical(),
##   nombrePeriodesEtablissement = col_double(),
##   numeroVoieEtablissement = col_double(),
##   codePostalEtablissement = col_double(),
##   libelleCommuneEtrangerEtablissement = col_logical(),
##   distributionSpecialeEtablissement = col_logical(),
##   codeCommuneEtablissement = col_double(),
##   codeCedexEtablissement = col_logical(),
##   libelleCedexEtablissement = col_logical(),
##   codePaysEtrangerEtablissement = col_logical(),
##   libellePaysEtrangerEtablissement = col_logical(),
##   complementAdresse2Etablissement = col_logical(),
##   numeroVoie2Etablissement = col_double(),
##   indiceRepetition2Etablissement = col_logical(),
##   codePostal2Etablissement = col_double(),
##   libelleCommuneEtranger2Etablissement = col_logical()
##   # ... with 14 more columns
## )
```

```
## See spec(...) for full column specifications.
```

```
## Warning: 45378 parsing failures.
##  row                                          col           expected
## 1492 denominationUsuelleEtablissement             1/0/T/F/TRUE/FALSE PARTEDIS
## 1508 denominationUsuelleEtablissement             1/0/T/F/TRUE/FALSE POINT P RHONE ALPES - POINT P
## 1548 denominationUsuelleEtablissement             1/0/T/F/TRUE/FALSE LIGNALITHE
## 1604 activitePrincipaleRegistreMetiersEtablissement 1/0/T/F/TRUE/FALSE 4311ZZ
## 1605 distributionSpecialeEtablissement            1/0/T/F/TRUE/FALSE BP 729
## .... ................................................ .................. ..........................
## See problems(...) for more details.
```

Créez l'objet APE_Type en lisant le fichier relatif aux codes d'APE.

```
APE_Type <- read_csv("data/APE_Type.csv")
```

```
## Parsed with column specification:
## cols(
##   Code = col_double(),
##   Type = col_character(),
##   TypeAbreg = col_character()
## )
```

Code et types d'activités => commerces alimentaires

```
library(stringr)
```

Combien d'entreprises ont un nom (enseigne1Etablissement) qui comprend le terme "BOULANGERIE"?

Il y a 84 entreprises qui comprennent le terme "BOULANGERIE" dans leur nom.

Ajoutez une variable Code à votre table en ne conservant que les quatre premiers caractères de la variable activitePrincipaleEtablissement (cela correspond à un pattern "^....", à savoir le début de chaîne de caractère suivi de quatre caractères quelconques -cf ce billet de blog sur les expressions régulières-).

```
data42 <- mutate(data42, Code=as.numeric(str_extract(activitePrincipaleEtablissement, "^....")))
```

Filtrez les lignes de data42 pour ne retenir que celles pour lesquelles l'APE correspond aux commerces "alimentaires" -alimentation, boisson, restaurant, bar- (voir la liste contenue dans le fichier APE_Type).

Stockez le résultat de ces opérations dans un objet alim42.

```
alim42 <- filter(data42, Code=="47.1" | Code=="47.2" | Code=="56.1" | Code=="56.2" | Code=="56.3")
```

Réalisez une jointure entre data42_alim (variable codeAPE) et APE_Type (variable Code), de manière à compléter alim42 avec les types de commerces (variables Type et TypeAbreg).

```
alim42 <- left_join(alim42, APE_Type, by="Code")
```

Résumé, classement

On crée une fonction pour charger le SHP d'un département pour visualiser les prochains résultats sur une carte

```
library(ggplot2)
library(sf)
```

```
## Linking to GEOS 3.7.2, GDAL 2.4.2, PROJ 5.2.0
```

```
load_SHP_dept <- function(dept) {
  filename <- "data/GEOFLA_XX.shp"
  filename <- str_replace(filename, "X{2}", dept)
  dept_shp <- st_read(filename,
                      stringsAsFactors = FALSE)
  st_crs(dept_shp)<-2154  # on définit la projection Lambert93
  return(dept_shp)
}
```

On charge le SHP de la Loire

```
dept42_SHP <- load_SHP_dept("42")
```

```
## Reading layer `GEOFLA_42' from data source `/Users/benoitblanc/Documents/M2 Geonum/2A1 - Statistique
## Simple feature collection with 327 features and 26 fields
## geometry type:  MULTIPOLYGON
## dimension:      XY
## bbox:           xmin: 753769 ymin: 6460313 xmax: 837633.1 ymax: 6575508
## epsg (SRID):    2154
## proj4string:    +proj=lcc +lat_1=49 +lat_2=44 +lat_0=46.5 +lon_0=3 +x_0=700000 +y_0=6600000 +ellps=GI
```

```
plot(dept42_SHP$geometry)
```

Quelles sont les 3 communes de votre base de données qui comptent le plus de magasins alimentaires?

Les trois communes comptant le plus de commerces alimentaires sont :SAINT ETIENNE, ROANNE, SAINT CHAMOND

Pour les communes qui ne comptent qu'un seul commerce "alimentaire", de quel type est-il, le plus fréquemment?

Les communes ne comptant qu'un seul commerce "alimentaire" sont au nombre de 15. Avec en détail : APINAC, BOYER, CEZAY, LA CHAPELLE EN LAFAYE, GREZIEUX LE FROMENTAL, JARNOSSE, PINAY, SAINTE AGATHE EN DONZY, SAINT LAURENT ROCHEFORT, SAINT PIERRE LA NOAILLE, SAINT PRIEST LA VETRE, SAINT SIXTE, TARTARAS, URBISE, VIRICELLES

Lorsqu'il y a un seul commerce alimentaire dans une commune, celui-ci est de type Restaurants et services de restauration mobile.

Quelles communes de plus de 100 commerces comptent au moins 10 commerces de type "viande"?

```
data42 <- inner_join(data42, APE_Type, by="Code")
```

```
morethan100commerces <- group_by(data42, codeCommuneEtablissement, libelleCommuneEtablissement) %>%
  summarise(nb_commerces=n())
morethan100commerces <- filter(morethan100commerces, nb_commerces>=100)
```

Les communes de plus de 100 commerces sont ANDREZIEUX BOUTHEON, LE CHAMBON FEUGEROLLES, CHARLIEU, CHAZELLES SUR LYON, LE COTEAU, FEURS, FIRMINY, MONTBRISON, LA RICAMARIE, RIVE DE GIER, ROANNE, ROCHE LA MOLIERE, SAINT CHAMOND, SAINT ETIENNE, SAINT PRIEST EN JAREZ, SAINT JUST SAINT RAMBERT, LA TALAUDIERE

```
morethan10viandes <- filter(data42, TypeAbreg=='viande') %>%
  group_by(codeCommuneEtablissement, libelleCommuneEtablissement) %>%
  summarise(nb_commerces_viande=n())
morethan10viandes <- filter(morethan10viandes, nb_commerces_viande>=10)
```

Les communes qui ont plus de 10 commerces de type 'viande' sont FEURS, MONTBRISON, ROANNE, SAINT ETIENNE, LA TALAUDIERE

On retrouve les 5 communes ayant plus de 10 commerces de type 'viande' dans les villes ayant plus de 100 commerces. On a donc la réponse. Mais pour s'assurer, on peut réaliser une jointure entre les 2 tableaux :

```
more100commerces10viandes <- left_join(morethan10viandes, morethan100commerces,
                                       by=c("codeCommuneEtablissement", "libelleCommuneEtablissement"))
```

On retrouve le même résultat : FEURS, MONTBRISON, ROANNE, SAINT ETIENNE, LA TALAUDIERE

Rapport, statistiques descriptives

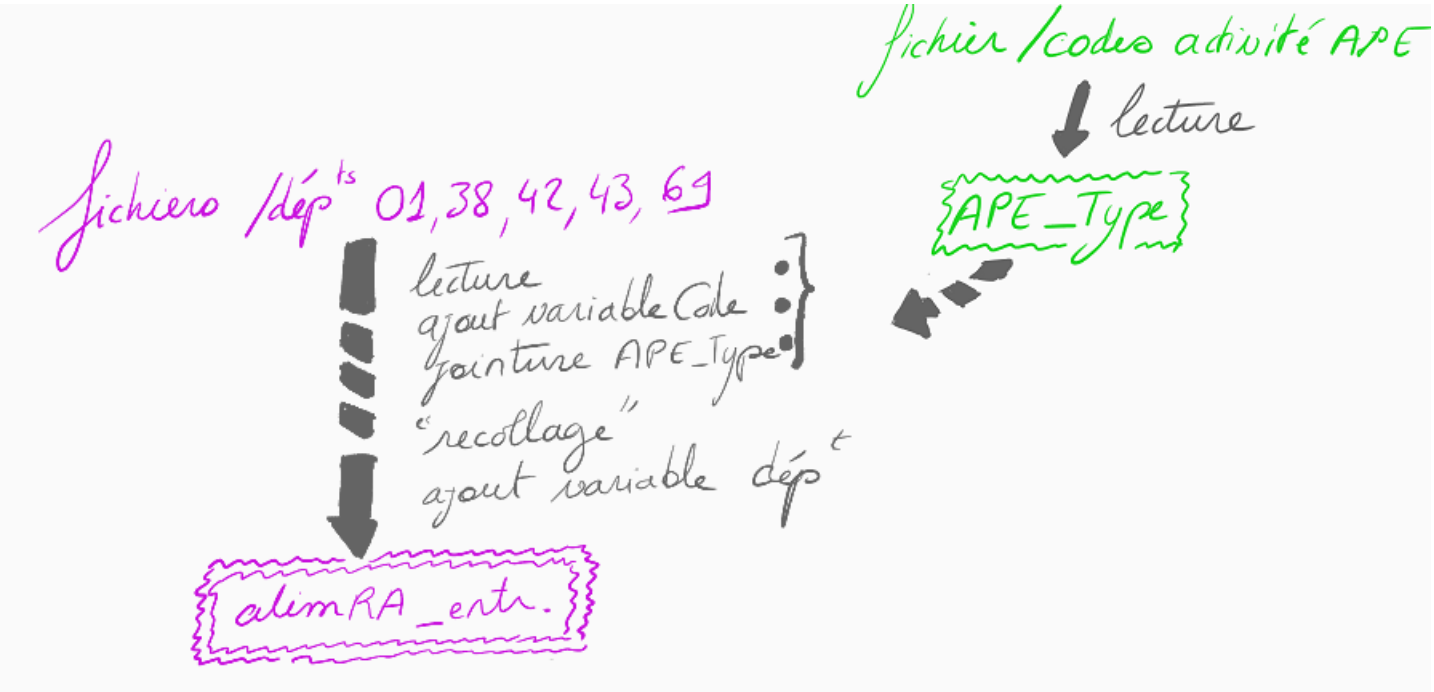
A ce stade, votre script commence à être un peu long et (peut-être) un peu désordonné… Ne serait-ce pas plus agréable de continuer votre projet sous la forme d'un rapport Rmarkdown? (Ne répondez pas à cette question, elle est rhétorique…).

Créez un document _____.Rmd, structurez-le avec quelques titres, et répartissez les différents morceaux de code de votre script de manière pertinente dans différents chunks. Vous pouvez maintenant rédiger des paragraphes en y intégrant des éléments de réponses aux questions posées précédemment. Rédigez un petit paragraphe pour nommer les 3 communes qui comptent le plus d'entreprises (exercice précédent) en utilisant l'insertion d'"inline chunks".

Les 3 communes qui ocmptent le plus d'entreprises sont SAINT ETIENNE, ROANNE, SAINT CHAMOND.

A partir de maintenant, votre document de travail sera un document'_____.Rmd' et non le script que vous avez créé initialement…

Programmation: automatisation pour plusieurs départements



Fonction

Pour obtenir la table alim42, vous avez réalisé un certain nombre d'opérations. On voudrait réaliser l'ensemble de ces opérations pour les 5 départements suivants:

- l'Ain (01)
- l'Isère (38)
- la Loire (42)
- la Haute-Loire (43)
- le Rhône (69)

Réutilisez les commandes que vous avez mises au point sur data42 pour écrire une fonction get_clean_data() qui réalisera l'ensemble de ces opérations sur le département de votre choix. L'input correspondra à un

numéro de département (c'est-à-dire que vous pourrez utiliser la fonction en faisant, par exemple get_clean_data("01")).

Pour lire le fichier, il faudra indiquer son chemin… Pensez à réutiliser ce que vous venez d'apprendre sur les chaînes de caractères pour reformer le chemin du fichier que vous intéresse à partir du numéro de département…

Certaines chaînes de caractère sont interprétées comme des chaînes de caractère pour certains jeux de données (par exemple pour les codes postaux de l'Ain, à cause du "0" en début de chaîne) tandis qu'elle est interprétée comme un numérique pour les autres jeux de données. Faites en sorte que votre fonction transforme bien cette variable pour qu'elles soient toujours de classe "character" en sortie (conversion par as.character()…).

```
library(purrr)
departements <- list("01", "38", "42", "43", "69")
```

```
get_clean_data <- function(dept) {
  dept <- as.character(dept)
  print(dept)
  # Read CSV file for dept
  filename <- "data/geo_siret_XX.csv"
  filename <- str_replace(filename, "X{2}", dept)
  print(filename)
  data <- read_csv(filename)
  data <- mutate(data, Code=as.numeric(str_extract(activitePrincipaleEtablissement, "^....")))
  alim <- filter(data, Code=="47.1" | Code=="47.2" | Code=="56.1" | Code=="56.2" | Code=="56.3")
  alim <- left_join(alim, APE_Type, by="Code")

  return(alim)
}
```

```
convert_to_str <- function(param) {
  str_param = as.character(param)
  return(str_param)
}
```

Itération

Appelez cette fonction de manière itérative pour chacun des départements cités ci-dessus. Vous pouvez pour ce faire soit écrire une boucle for, soit utiliser la fonction map() du package purrr.

```
if (!file.exists("data/alimRA_entr.csv")) {
  commercesRA <- departements %>% map(get_clean_data)
  #commercesRA <- mutate(commercesRA, codePostalEtablissement=convert_to_str(codePostalEtablissement))
} else {
  alimRA_entr <- read_csv("data/alimRA_entr.csv")
}
```

```
## Parsed with column specification:
## cols(
##   .default = col_character(),
##   dateCreationEtablissement = col_date(format = ""),
##   anneeEffectifsEtablissement = col_double(),
##   activitePrincipaleRegistreMetiersEtablissement = col_logical(),
##   dateDernierTraitementEtablissement = col_datetime(format = ""),
##   etablissementSiege = col_logical(),
##   nombrePeriodesEtablissement = col_double(),
##   numeroVoieEtablissement = col_double(),
```

```
##   libelleCommuneEtrangerEtablissement = col_logical(),
##   codePaysEtrangerEtablissement = col_logical(),
##   libellePaysEtrangerEtablissement = col_logical(),
##   complementAdresse2Etablissement = col_logical(),
##   numeroVoie2Etablissement = col_logical(),
##   indiceRepetition2Etablissement = col_logical(),
##   typeVoie2Etablissement = col_logical(),
##   libelleVoie2Etablissement = col_logical(),
##   codePostal2Etablissement = col_logical(),
##   libelleCommune2Etablissement = col_logical(),
##   libelleCommuneEtranger2Etablissement = col_logical(),
##   distributionSpeciale2Etablissement = col_logical(),
##   codeCommune2Etablissement = col_logical()
##   # ... with 11 more columns
## )
```

```
## See spec(...) for full column specifications.
```

A partir des 5 jeux de données obtenus vous créerez un seul et même jeu de données alimRA_entr (données pour l'ancienne région Rhône-Alpes, où 1 ligne=1 entreprise).

Vous pourrez si vous le souhaitez vous servir de la commande do.call("rbind",...) ou bind_rows().

```
if (!file.exists("data/alimRA_entr.csv")) {
  #alimRA_entr <- bind_rows(commercesRA)
  alimRA_entr <- do.call("rbind", commercesRA)
}
```

Rajoutez une variable departement (correspondant au numéro de département) à votre jeu de données alimRA_entr. Peut-être par des manipulations sur le code postal?...

```
if (!file.exists("data/alimRA_entr.csv")) {
  alimRA_entr <- mutate(alimRA_entr, codePostalEtablissement= as.character(codePostalEtablissement))
  alimRA_entr <- mutate(alimRA_entr, codePostalEtablissement=gsub("^([0-9]{1,4})", "0\\1", as.character
  alimRA_entr <- mutate(alimRA_entr, departement=str_extract(codePostalEtablissement, "^.{2}"))
}
```

If et écriture de fichier

Vous avez dû remarquer que l'exécution de l'étape précédente prenait un peu de temps car les 5 fichiers geosirene lus sont très volumineux... En revanche la table alimRA_entr est de taille beaucoup plus raisonnable. Or, nous n'aurons besoin que de cette table pour la suite du projet. Pour éviter d'exécuter cette étape chronophage à chaque fois que vous travaillerez sur ce projet:

- exportez alimRA_entr dans un fichier alimRA_entr.csv.
- entourez la boucle for d'une structure conditionnelle if de sorte que la boucle ne soit exécutée que si le fichier alimRA_entr.csv n'existe pas (voir fonction file.exists()...)
- écrivez à la suite la commande qui servira à lire alimRA_entr.csv à chaque "tricotage" de votre rapport Rmarkdown.

```
write_csv(alimRA_entr, "data/alimRA_entr.csv")
alimRA_entr <- read_csv("data/alimRA_entr.csv")
```

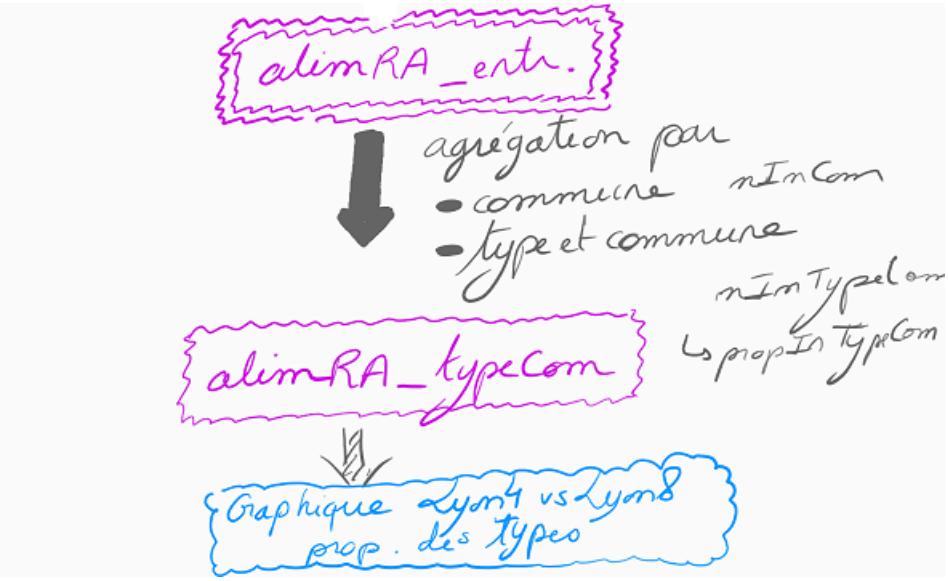
```
## Parsed with column specification:
## cols(
##   .default = col_character(),
##   dateCreationEtablissement = col_date(format = ""),
##   anneeEffectifsEtablissement = col_double(),
```

```
##   activitePrincipaleRegistreMetiersEtablissement = col_logical(),
##   dateDernierTraitementEtablissement = col_datetime(format = ""),
##   etablissementSiege = col_logical(),
##   nombrePeriodesEtablissement = col_double(),
##   numeroVoieEtablissement = col_double(),
##   libelleCommuneEtrangerEtablissement = col_logical(),
##   codePaysEtrangerEtablissement = col_logical(),
##   libellePaysEtrangerEtablissement = col_logical(),
##   complementAdresse2Etablissement = col_logical(),
##   numeroVoie2Etablissement = col_logical(),
##   indiceRepetition2Etablissement = col_logical(),
##   typeVoie2Etablissement = col_logical(),
##   libelleVoie2Etablissement = col_logical(),
##   codePostal2Etablissement = col_logical(),
##   libelleCommune2Etablissement = col_logical(),
##   libelleCommuneEtranger2Etablissement = col_logical(),
##   distributionSpeciale2Etablissement = col_logical(),
##   codeCommune2Etablissement = col_logical()
##   # ... with 11 more columns
## )

## See spec(...) for full column specifications.
```

```
#alimRA_entr <- map(., mutate(., codePostalEtablissement=convert_code_postal(codePostalEtablissement)))
```

Résumé par commune et type de commerce



Agrégation des données par commune et type de commerce

Agrégez la table alimRA_entr par commune et type de commerce, pour créer une table alimRA_typeCom (où une ligne correspondra à un type de commerce pour une commune):

```
alimRA_Com <- group_by(alimRA_entr, codeCommuneEtablissement, libelleCommuneEtablissement)%>%
  summarise(nInCom=n())
alimRA_type <- group_by(alimRA_entr, codeCommuneEtablissement, libelleCommuneEtablissement, TypeAbreg)%
```

```
  summarise(nInTypeCom=n()) %>%
  mutate(propInTypeCom = round(nInTypeCom / sum(nInTypeCom)*100))
```

- une variable nInCom correspondant au nombre de commerces par commune
- une variable nInTypeCom correspondant au nombre de commerces par type et commune
- une variable propInTypeCom correspondant à la proportion d'un type de commerce dans une commune

```
alimRA_typeCom <- left_join(alimRA_Com, alimRA_type,
                                        by=c("codeCommuneEtablissement", "libelleCommuneEtablissement"))
```

Quelles communes comptant plus de 100 commerces comptent au moins 15% de commerces de type "bar"?

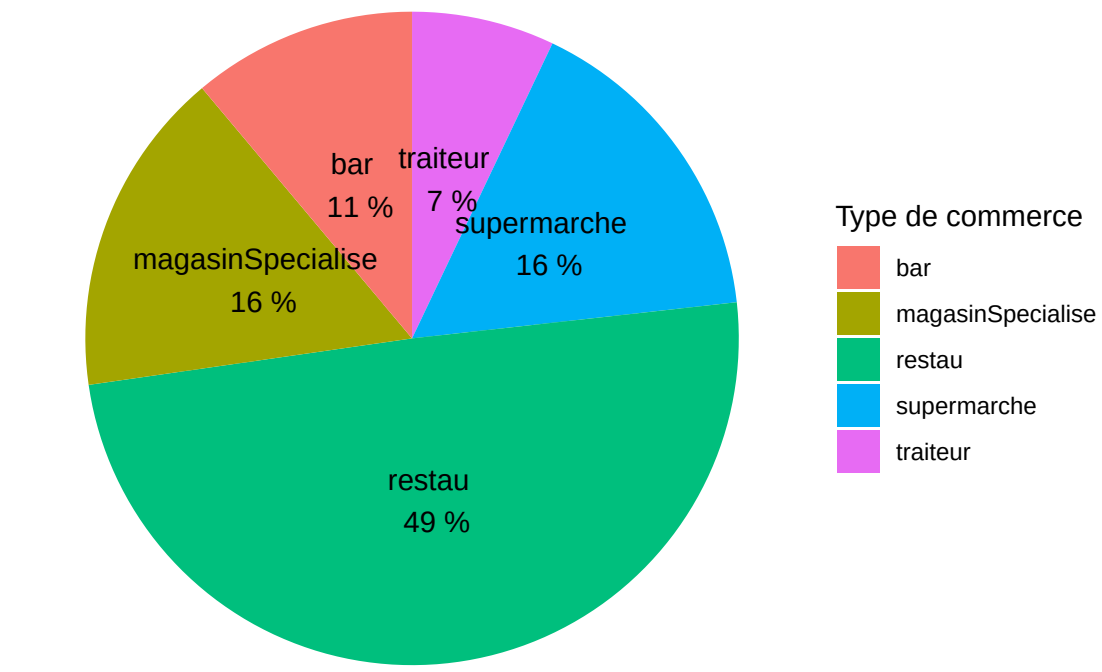
```
communes_15pct_bar <- filter(alimRA_typeCom, TypeAbreg=='bar' & nInCom >= 100 & propInTypeCom >= 15)
```

Les communes comptant plus de 100 commerces, dont les commerces de type 'bar' représentent au moins 15% sont LES DEUX ALPES, LA MURE, ROUSSILLON, SAINT MARCELLIN, CHAMROUSSE, LE CHAMBON FEUGEROLLES, CHARLIEU, FIRMINY, RIVE DE GIER, ROANNE, SAINT ETIENNE, BRIOUDE, LE PUY EN VELAY, YSSINGEAUX

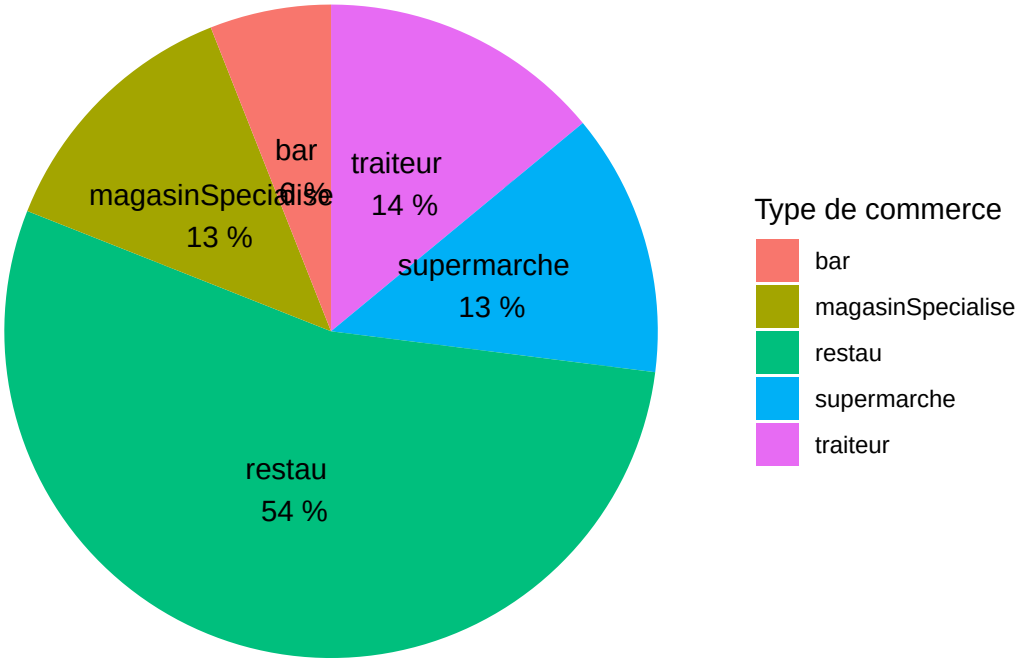
Graphique

Réalisez un graphique montrant les proportions des différents types de commerces pour LYON 4EME et LYON 8EME.

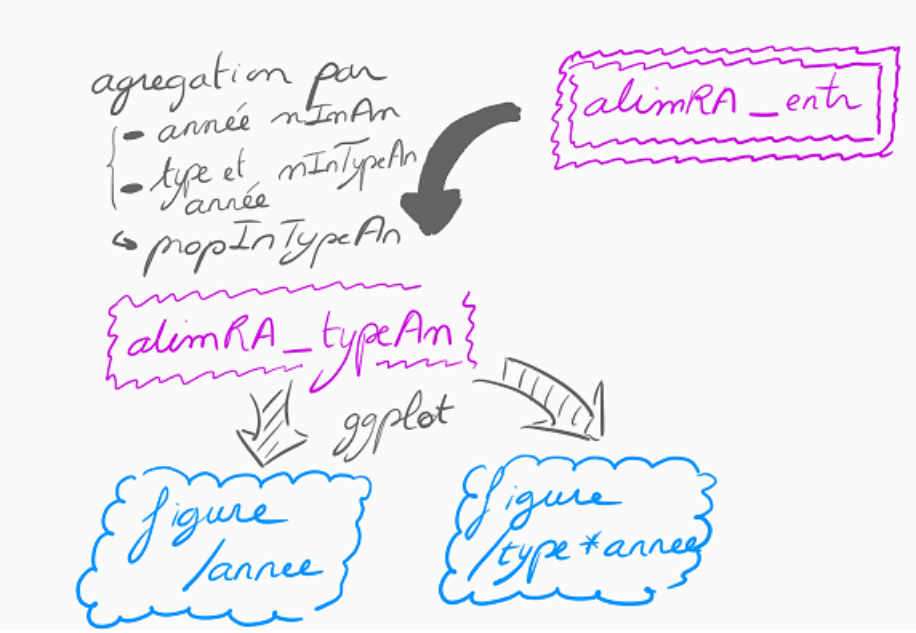
```
lyon4 <- filter(alimRA_typeCom, codeCommuneEtablissement == 69384)
propLyon4 <- ggplot(lyon4, aes(x="", y=propInTypeCom, fill=TypeAbreg))+
  geom_bar(width = 1, stat = "identity") + geom_text(aes(label=paste(TypeAbreg,"\n",propInTypeCom, "%"))
                                                     position = position_stack(vjust = 0.5)) + coord_pol
  ggtitle("Répartition des types de commerces à Lyon 4ème") + labs(fill = "Type de commerce") + theme_v
propLyon4
```



```
lyon8 <- filter(alimRA_typeCom, codeCommuneEtablissement == 69388)
propLyon8 <- ggplot(lyon8, aes(x="", y=propInTypeCom, fill=TypeAbreg)) +
  geom_bar(width = 1, stat = "identity") + geom_text(aes(label=paste(TypeAbreg,"\n",propInTypeCom, "%"))
                                                     position = position_stack(vjust = 0.5)) + coord_pol
  ggtitle("Répartition des types de commerces à Lyon 8ème") + labs(fill = "Type de commerce") + theme_v
propLyon8
```



Evolution dans le temps des créations d'entreprise



Manipuler des dates avec lubridate

Nous allons nous intéresser aux dates de création des entreprises de notre base alimRA_entr (variable dateCreationEtablissement).

Pour le moment, dateCreationEtablissement est considéré comme une variable de type "chaîne de caractères". Pour faire comprendre à R qu'il s'agit en réalité d'une date (et lui faire comprendre comment elle est mise en forme) nous allons faire appel au package lubridate.

- Installez et chargez le package lubridate.

```
library(lubridate)
```

```
##
## Attaching package: 'lubridate'
```

```
## The following object is masked from 'package:base':
##
##     date
```

- Transformez le tableau alimRA_entr en modifiant la classe de dateCreationEtablissement à l'aide d'une fonction de lubridate.

```
alimRA_entr <- mutate(alimRA_entr, dateCreationEtablissement = ymd(dateCreationEtablissement))
```

- Ajoutez une variable annee au tableau alimRA_entr à l'aide, à nouveau, d'une des fonctions de lubridate.

```
alimRA_entr <- mutate(alimRA_entr, annee = year(dateCreationEtablissement))
```

Résumé, filtre

- Créez une table alimRA_typeAn qui recense le nombre d'entreprises par année (nInAn), et par type*année (nInTypeAn).

```
alimRA_annee <- group_by(alimRA_entr, annee)%>%
  summarise(nInAn=n())
alimRA_type_annee <- group_by(alimRA_entr, annee, TypeAbreg)%>%
  summarise(nInTypeAn=n())
alimRA_typeAn <- left_join(alimRA_type_annee, alimRA_annee,
                                        by=c("annee"))
```

- Filtrez les données de alimRA_typeAn pour ne garder que les entreprises dont la création correspond aux années >=1970.

```
alimRA_ap1970 <- filter(alimRA_typeAn, annee >= 1970)
```

Graphiques: évolution du nombre d'entreprises au cours du temps

- Installez et chargez le package ggplot2
- Réalisez un graphique représentant l'évolution des proportions d'entreprises (par type) par année.
- Réalisez ce même graphique, mais en représentant le nombre de créations d'entreprises par année et par type, pour les 5 types comptant le plus de créations d'entreprises (au total).