Manipulation avancée des dataframes

Joël K. Kazadi

2023-02-24

Nous avons vu dans les leçons précédentes que les objets de classe "dataframe" sont hétérogènes, i.e. ils permettent de stocker des données de plusieurs modes. Les dataframes sont des tableaux de données où chaque ligne correspond à un individu, et chaque colonne à une variable. Les variables d'un dataframe peuvent être de nature variée. Rappelons le code de création d'un dataframe :

```
library(knitr)

# Creation du dataframe
data <- data.frame(Econometrie = c(18,16,17,18),
                   Statistique = c(14,13,15,19),
                   Niveau = c("Bon","Moyen",NA,"Excellent"),
                   row.names = c("Kadima","Kazadi","Nsamba","Malu"))

# La fonction kable du package knitr permet de generer
# des tableaux de maniere plus elegante
kable(data, label = NA, caption = "Tableau des données")
```

Table 1: Tableau des données

	Econometrie	Statistique	Niveau
Kadima	18	14	Bon
Kazadi	16	13	Moyen
Nsamba	17	15	NA
Malu	18	19	Excellent

Dans cette section, il sera question d'aborder quelques notions relatives à la manipulation des dataframes. Il s'agit notamment de : (i) l'ajout et la suppression des variables, (ii) la sélection des individus par tranche, (iii) la subdivision du dataframe, (iv) la concaténation des dataframes, etc.

1 Ajout et Suppression des variables

Considérons un tableau des données relatif à l'évolution des indicateurs clés au sein d'une industrie entre 1970 et 2019.

```
# Specification du repertoire de travail
setwd("E:/Data_Science/RStudio/")

# Importation du dataset
library(readxl)
data_industrie <- read_excel("Data_frame.xlsx",
                             sheet = "time_series",
                             col_names = TRUE)

# Impression des 10 premieres lignes du dataset
kable(head(data_industrie, n = 10L),
      label = NA, caption = "Evolution des indicateurs")
```

Table 2: Evolution des indicateurs

date	environnement	effectif	satisfaction	produits	charges
1970	Very favorable	2	3	7550	5185
1971	Very favorable	1	2	7975	5000
1972	High risk	3	2	4490	6104
1973	Low risk	1	4	4691	4568
1974	Favorable	3	1	7676	5925
1975	Favorable	4	3	7277	5833
1976	Favorable	2	1	7155	5802
1977	Favorable	3	5	7043	5463
1978	Favorable	1	3	7409	5555
1979	Very favorable	1	5	6777	5308

Supposons que l'on veuille créer une colonne nommé "gain" dans ce dataset. Il faudra ajouter une nouvelle variable, sachant que le bénéfice correspond à la différence entre les produits et les charges.

```
# Ajout de la variable GAIN
data_industrie$gain <- data_industrie$produits - data_industrie$charges

# Impression des 10 premieres lignes du dataset
kable(head(data_industrie, n = 10L),
      label = NA, caption = "Evolution des indicateurs")
```

Table 3: Evolution des indicateurs

date	environnement	effectif	satisfaction	produits	charges	gain
1970	Very favorable	2	3	7550	5185	2365
1971	Very favorable	1	2	7975	5000	2975
1972	High risk	3	2	4490	6104	-1614
1973	Low risk	1	4	4691	4568	123
1974	Favorable	3	1	7676	5925	1751
1975	Favorable	4	3	7277	5833	1444
1976	Favorable	2	1	7155	5802	1353
1977	Favorable	3	5	7043	5463	1580
1978	Favorable	1	3	7409	5555	1854
1979	Very favorable	1	5	6777	5308	1469

Supposons à présent qu'il nous soit demandé de supprimer la variable "environnement" du dataset. Tout comme pour le cas des matrices, les colonnes d'un dataframe sont indicées par des nombres entiers naturels à partir du chiffre "1". La variable "environnement" a donc pour indice dans le dataframe le chiffre "2". Supprimer cette variable dans le dataframe consistera tout simplement à imprimer ce dataframe en renseignant le nombre opposé à l'indice de la variable à retirer, i.e. il faut indiquer "-2".

```
# Suppression de la variable ENVIRONNEMENT
kable(head(data_industrie[,-2], n = 10L),
      label = NA, caption = "Evolution des indicateurs")
```

Table 4: Evolution des indicateurs

date	effectif	satisfaction	produits	charges	gain
1970	2	3	7550	5185	2365
1971	1	2	7975	5000	2975
1972	3	2	4490	6104	-1614
1973	1	4	4691	4568	123
1974	3	1	7676	5925	1751
1975	4	3	7277	5833	1444
1976	2	1	7155	5802	1353
1977	3	5	7043	5463	1580
1978	1	3	7409	5555	1854
1979	1	5	6777	5308	1469

2 Sélection des lignes par tranche

Les fonctions `head` et `tail` permettent respectivement de sélectionner les premières et les dernières lignes d'un dataframe. Ainsi, pour sélectionner des lignes suivant d'autres critères, on fait recours une fois de plus au principe d'indexation des dataframes (*cf.* la leçon sur les matrices).

Supposons qu'il nous soit demandé de sélectionner les lignes d'indice 11 à 20. Le code de sélection par tranche est le suivant :

```
# Format du dataframe
dim(data_industrie)
```

```
## [1] 50  7
```

```
# Selection par tranche
kable(data_industrie[11:20,], label = NA,
      caption = "Evolution des indicateurs")
```

Table 5: Evolution des indicateurs

date	environnement	effectif	satisfaction	produits	charges	gain
1980	Favorable	4	3	5648	5062	586
1981	Low risk	3	4	5447	4938	509
1982	Low risk	2	2	4954	4769	185
1983	Low risk	4	5	4718	4260	458
1984	High risk	1	1	4649	4538	111
1985	Very favorable	4	3	5093	3952	1141
1986	Favorable	4	4	5185	4000	1185
1987	Favorable	2	5	6185	4908	1277
1988	Low risk	3	2	5040	4630	410
1989	High risk	1	3	5185	4960	225

Supposons à présent qu'il nous soit demandé de sélectionner les lignes d'indice 11 à 20 et d'indice 31 à 40.

```
# Selection par tranche
kable(data_industrie[c(11:20,31:40),], label = NA,
      caption = "Evolution des indicateurs")
```

Table 6: Evolution des indicateurs

date	environnement	effectif	satisfaction	produits	charges	gain
1980	Favorable	4	3	5648	5062	586
1981	Low risk	3	4	5447	4938	509
1982	Low risk	2	2	4954	4769	185
1983	Low risk	4	5	4718	4260	458
1984	High risk	1	1	4649	4538	111
1985	Very favorable	4	3	5093	3952	1141
1986	Favorable	4	4	5185	4000	1185
1987	Favorable	2	5	6185	4908	1277
1988	Low risk	3	2	5040	4630	410
1989	High risk	1	3	5185	4960	225
2000	Low risk	4	1	5254	4445	809
2001	Low risk	2	1	5320	5198	122
2002	Very favorable	4	6	5463	4075	1388
2003	Very favorable	4	6	5807	4260	1547
2004	High risk	4	1	6295	6630	-335
2005	High risk	3	1	6166	6260	-94
2006	Favorable	2	6	5589	3983	1606
2007	High risk	1	1	5608	5693	-85
2008	Very favorable	1	6	6073	3335	2738
2009	Favorable	1	2	6203	3566	2637

3 Subdivision d'un dataframe

Au-délà de la sélection par tranche à l'aide de l'indexation, il est possible d'avoir accès aux données d'un dataframe suivant d'autres critères beaucoup plus poussés. Il s'agit notamment des critères se rapportant aux opérateurs logiques. Pour cela, on va recourir à la fonction `subset`.

Supposons qu'il nous soit demandé d'extraire les données de notre dataframe relatif aux indicateurs économiques d'une industrie uniquement à partir de l'année 2000. Le code est le suivant :

```
kable(subset(data_industrie, date >= 2000), label = NA,
      caption="Evolution des indicateurs entre 2000 et 2019")
```

Table 7: Evolution des indicateurs entre 2000 et 2019

date	environnement	effectif	satisfaction	produits	charges	gain
2000	Low risk	4	1	5254	4445	809
2001	Low risk	2	1	5320	5198	122
2002	Very favorable	4	6	5463	4075	1388
2003	Very favorable	4	6	5807	4260	1547
2004	High risk	4	1	6295	6630	-335
2005	High risk	3	1	6166	6260	-94
2006	Favorable	2	6	5589	3983	1606
2007	High risk	1	1	5608	5693	-85
2008	Very favorable	1	6	6073	3335	2738
2009	Favorable	1	2	6203	3566	2637
2010	High risk	4	2	6480	6630	-150
2011	Low risk	3	5	6559	5723	836
2012	High risk	4	1	6480	6538	-58
2013	High risk	4	4	6564	7137	-573
2014	Very favorable	3	6	6648	4568	2080
2015	Very favorable	2	5	6838	4630	2208
2016	Low risk	1	3	6916	6630	286
2017	Favorable	2	1	6918	4075	2843
2018	Very favorable	3	4	6529	3952	2577
2019	Low risk	4	1	6480	5890	590

Il est également possible de combiner plusieurs critères logiques. Par exemple, pour extraire les données de 2000 à 2019, mais uniquement pour les années où l'état de l'environnement dans lequel l'industrie évolue est "favorable", on procède de la manière suivante :

```
kable(subset(data_industrie, date >= 2000 & environnement == "Favorable"),
      label = NA,
      caption="Evolution des indicateurs entre 2000 et 2019 (Environnement favorable)")
```

Table 8: Evolution des indicateurs entre 2000 et 2019 (Environnement favorable)

date	environnement	effectif	satisfaction	produits	charges	gain
2006	Favorable	2	6	5589	3983	1606
2009	Favorable	1	2	6203	3566	2637
2017	Favorable	2	1	6918	4075	2843

Partant de la subdivision précédente, supposons que l'on ne veuille afficher que les 3 dernières colonnes du dataset. Pour y arriver, l'on va utiliser l'argument `select` de la fonction `subset`.

```
kable(subset(data_industrie, date >= 2000 & environnement == "Favorable",
             select = c(date, produits, charges)),
      label = NA,
      caption="Evolution des indicateurs entre 2000 et 2019 (Environnement favorable)")
```

Table 9: Evolution des indicateurs entre 2000 et 2019 (Environnement favorable)

date	produits	charges
2006	5589	3983
2009	6203	3566
2017	6918	4075

4 Quelques fonctions raccourcies

Ces fonctions permettent d'effectuer des opérations directement sur les objets R. Dans le cadre de cette leçon, nous en verrons de quatre types : (i) la fonction `apply`, (ii) la fonction `tapply`, (iii) la fonction `by`, et (iv) la fonction `aggregate`.

4.1 La fonction `apply`

Cette fonction permet d'effectuer des calculs directement sur les rangées (lignes ou colonnes) d'un dataframe. Supposons que l'on veuille calculer la moyenne de toutes les variables quantitatives dans notre dataset. On procède comme suit :

```
kable(apply(X = subset(data_industrie,
                       select = c(produits, charges, gain)),
            MARGIN = 2,
            FUN = "mean"),
      col.names = "Moyenne",
      caption = "Moyenne des variables quantitatives")
```

Table 10: Moyenne des variables quantitatives

	Moyenne
produits	5925.70
charges	5046.94
gain	878.76

L'argument `MARGIN` indique la rangée par laquelle les calculs doivent s'effectuer. Si cet argument est spécifié à "2", les opérations s'effectueront par colonne. Si cet argument est spécifié à "1", les opérations s'effectueront par ligne.

4.2 La fonction `tapply`

La fonction `apply` ne permet pas d'effectuer des opérations sur différentes sous-populations dans le dataset. La fonction `tapply` prend en compte cette particularité en donnant la possibilité de subdiviser les individus dans le dataset suivant les modalités d'une variable qualitative quelconque. Supposons qu'il nous soit demandé de calculer le bénéfice médian pour chaque type d'environnement où l'industrie a évolué.

```
kable(tapply(X = data_industrie$gain,
             INDEX = data_industrie$environnement,
             FUN = "median"),
      row.names = TRUE,
      col.names = "Mediane",
      caption = "Bénéfice médian selon le type d'environnement")
```

Table 11: Bénéfice médian selon le type d'environnement

	Mediane
Favorable	1398.5
High risk	-122.0
Low risk	410.0
Very favorable	2080.0

4.3 La fonction `by`

Tout comme la fonction `tapply`, la fonction `by` permet également d'effectuer des calculs sur plusieurs sous-populations. A la différence de la fonction `tapply` où les opérations portent essentiellement sur une seule variable (un vecteur), la fonction `by` autorise les calculs simultanément sur plusieurs variables (une matrice).

Le code ci-après calcule les coefficients de corrélation entre les variables quantitatives prises deux-à-deux, en fonction du type d'environnement.

```
by(data = subset(data_industrie,
                 select = c(produits, charges, gain)),
   INDICES = data_industrie$environnement,
   FUN = cor)
```

```
## data_industrie$environnement: Favorable
##           produits    charges       gain
## produits 1.0000000  0.7232593  0.5429649
## charges  0.7232593  1.0000000 -0.1872110
## gain     0.5429649 -0.1872110  1.0000000
## ------------------------------------------------------------
## data_industrie$environnement: High risk
##           produits    charges       gain
```

```
## produits 1.0000000  0.7226370  0.2137048
## charges  0.7226370  1.0000000 -0.5208282
## gain     0.2137048 -0.5208282  1.0000000
## ------------------------------------------------------------
## data_industrie$environnement: Low risk
##           produits    charges       gain
## produits 1.0000000  0.9186233  0.2754853
## charges  0.9186233  1.0000000 -0.1267775
## gain     0.2754853 -0.1267775  1.0000000
## ------------------------------------------------------------
## data_industrie$environnement: Very favorable
##           produits    charges       gain
## produits 1.0000000 0.69284700 0.76774753
## charges  0.6928470 1.00000000 0.06989481
## gain     0.7677475 0.06989481 1.00000000
```

4.4 La fonction aggregate

Cette fonction permet d'effectuer des calculs sur plusieurs variables dans un dataset subdivisé en sous-populations. Contrairement à la fonction by qui peut aussi s'appliquer sur les matrices, la fonction aggregate s'applique exclusivement sur les objets de classe "dataframe".

Le code ci-après permet de générer le résumé statistique des variables quantitatives selon le type d'environnement de l'industrie.

```
tab <- aggregate(x=subset(data_industrie,
                          select = c(produits, charges, gain)),
                 by = list(Environnement = data_industrie$environnement),
                 FUN = summary)

kable(cbind(tab$Environnement, tab$produits),
      caption = "Resumé statistique : Produits")
```

Table 12: Resumé statistique : Produits

	Min.	1st Qu.	Median	Mean	3rd Qu.	Max.
Favorable	5053	5546.25	6194	6289.85714285714	7127	7676
High risk	4490	5013.5	5471.5	5597	6341.25	6564
Low risk	4691	5040	5254	5471.15384615385	5447	6916
Very favorable	5093	5635	6529	6358	6807.5	7975

```
kable(cbind(tab$Environnement, tab$charges),
      caption = "Resumé statistique : Charges")
```

Table 13: Resumé statistique : Charges

	Min.	1st Qu.	Median	Mean	3rd Qu.	Max.
Favorable	3566	4136.75	4723	4813.21428571429	5532	5925
High risk	4445	5449.75	6182	5922.33333333333	6561	7137
Low risk	4260	4568	4938	5035.46153846154	5198	6630
Very favorable	3335	4013.5	4260	4403	4815	5308

```
kable(cbind(tab$Environnement, tab$gain),
      caption = "Resumé statistique : Bénéfices")
```

Table 14: Resumé statistique : Bénéfices

	Min.	1st Qu.	Median	Mean	3rd Qu.	Max.
Favorable	586	1041.75	1398.5	1476.64285714286	1714.75	2843
High risk	-1614	-622.75	-122	-325.333333333333	-15.75	626
Low risk	102	185	410	435.692307692308	590	925
Very favorable	1017	1428.5	2080	1955	2471	2975

5 Le package `dplyr`

Développé par Hadley Wickham, le package `dplyr` fournit des outils simples pour les tâches de manipulation avancée de données. Ce package fonctionne comme une "grammaire" de manipulation de données, fournissant un ensemble cohérent de "verbes" qui aide à résoudre les problèmes de manipulation de données les plus courants. Les principaux verbes sont les suivants : (i) `select`, (ii) `filter`, (iii) `arrange`, et (iv) `mutate`.

5.1 Le verbe `select`

Nous avons vu précédemment comment sélectionner les variables dans un dataframe suivant le principe d'indexation. Cependant, dans un dataset contenant plusieurs variables, il devient difficile de recourir aux indices de chacune de ces variables. Le verbe `select` facilite cet exercice en donnant la possibilité d'appeler les variables directement par leurs noms. Voici quelques exemples d'utilisation de ce verbe.

```r
# Specification de la variable qualitative comme variable categorielle
library(dplyr)
data_industrie$environnement <- as.factor(data_industrie$environnement)
str(data_industrie)
```

```
## tibble [50 x 7] (S3: tbl_df/tbl/data.frame)
##  $ date         : num [1:50] 1970 1971 1972 1973 1974 ...
##  $ environnement: Factor w/ 4 levels "Favorable","High risk",..: 4 4 2 3 1 1 1 1 1 4
##  $ effectif     : num [1:50] 2 1 3 1 3 4 2 3 1 1 ...
##  $ satisfaction : num [1:50] 3 2 2 4 1 3 1 5 3 5 ...
##  $ produits     : num [1:50] 7550 7975 4490 4691 7676 ...
##  $ charges      : num [1:50] 5185 5000 6104 4568 5925 ...
##  $ gain         : num [1:50] 2365 2975 -1614 123 1751 ...
```

```r
# Selection de quelques variables
kable(head(select(data_industrie, date, environnement, produits, charges)),
      caption = "Sélection des variables")
```

Table 15: Sélection des variables

date	environnement	produits	charges
1970	Very favorable	7550	5185
1971	Very favorable	7975	5000
1972	High risk	4490	6104
1973	Low risk	4691	4568
1974	Favorable	7676	5925
1975	Favorable	7277	5833

```r
# Selection de quelques variables par tranche
kable(head(select(data_industrie, date, produits : gain)),
      caption = "Sélection des variables par tranche")
```

Table 16: Sélection des variables par tranche

date	produits	charges	gain
1970	7550	5185	2365
1971	7975	5000	2975
1972	4490	6104	-1614
1973	4691	4568	123
1974	7676	5925	1751
1975	7277	5833	1444

```
# Selection des variables dont les noms commencent par la lettre E
kable(head(select(data_industrie, starts_with("e")), n = 5L),
      caption="Sélection des variables dont les noms commencent par la lettre \"e\" ")
```

Table 17: Sélection des variables dont les noms commencent par la lettre “e”

environnement	effectif
Very favorable	2
Very favorable	1
High risk	3
Low risk	1
Favorable	3

```
# Selection des variables dont les noms s'achevent par la lettre S
kable(head(select(data_industrie, ends_with("s")), n = 5L),
      caption="Sélection des variables dont les noms s'achèvent par la lettre \"s\" ")
```

Table 18: Sélection des variables dont les noms s’achèvent par la lettre “s”

produits	charges
7550	5185
7975	5000
4490	6104
4691	4568
7676	5925

```
# Deselection d'une variable
kable(head(select(data_industrie, -effectif), n = 5L),
      caption = "Désélection d'une variables")
```

Table 19: Désélection d’une variables

date	environnement	satisfaction	produits	charges	gain
1970	Very favorable	3	7550	5185	2365
1971	Very favorable	2	7975	5000	2975
1972	High risk	2	4490	6104	-1614
1973	Low risk	4	4691	4568	123
1974	Favorable	1	7676	5925	1751

```
# Deselection des variables dont les noms contiennent la suite de lettres AT
kable(head(select(data_industrie, -contains("at")), n = 5L),
      caption="Désélection des variables dont les noms contiennent
      la suite de lettres \"at\" ")
```

Table 20: Désélection des variables dont les noms contiennent la suite de lettres “at”

environnement	effectif	produits	charges	gain
Very favorable	2	7550	5185	2365
Very favorable	1	7975	5000	2975
High risk	3	4490	6104	-1614
Low risk	1	4691	4568	123
Favorable	3	7676	5925	1751

5.2 Le verbe `filter`

Comme l’indique son nom, ce verbe permet de filtrer les lignes d’un dataframe en fonction des conditions logiques appliquées sur les colonnes de ce dataframe. Voici quelques exemples d’utilisation de ce verbe.

```
# Impression des annees de perte ou l'environnement n'etait pas tres favorable,
# et pendant lesquelles le chiffre d'affaires variait entre 5000 et 6000
kable(filter(data_industrie, gain < 0,
             between(produits, 5000, 6000),
             environnement != "Very favorable"),
      caption = "Filtrage avancee")
```

Table 21: Filtrage avancee

date	environnement	effectif	satisfaction	produits	charges	gain
1996	High risk	2	6	5335	6520	-1185
2007	High risk	1	1	5608	5693	-85

```
# Impression des annees ou l'environnement etait soit favorable, soit peu risque
kable(head(filter(data_industrie,
                  environnement %in% c("Favorable", "Low risk")),
           n = 10L),
      caption = "Filtrage avancee")
```

Table 22: Filtrage avancee

date	environnement	effectif	satisfaction	produits	charges	gain
1973	Low risk	1	4	4691	4568	123
1974	Favorable	3	1	7676	5925	1751
1975	Favorable	4	3	7277	5833	1444
1976	Favorable	2	1	7155	5802	1353
1977	Favorable	3	5	7043	5463	1580
1978	Favorable	1	3	7409	5555	1854
1980	Favorable	4	3	5648	5062	586
1981	Low risk	3	4	5447	4938	509
1982	Low risk	2	2	4954	4769	185
1983	Low risk	4	5	4718	4260	458

```
# Impression des lignes pour lesquelles au moins une valeur des
# colonnes EFFECTIF et SATISFACTION est superieure a 2
kable(head(filter_all(select(data_industrie, effectif, satisfaction),
                      any_vars(. > 2)),
           n = 5L),
      caption = "Filtrage avancee")
```

Table 23: Filtrage avancee

effectif	satisfaction
2	3
3	2
1	4
3	1
4	3

```
# Impression des lignes pour lesquelles toutes les valeurs des
# colonnes EFFECTIF et SATISFACTION sont superieures a 2
kable(head(filter_all(select(data_industrie, effectif, satisfaction),
                      all_vars(. > 2)),
           n = 5L),
      caption = "Filtrage avancee")
```

Table 24: Filtrage avancee

effectif	satisfaction
4	3
3	5
4	3
3	4
4	5

5.3 Le verbe arrange

Ce verbe permet de classer les lignes d'un dataframe par ordre croissant ou décroissant des valeurs d'une colonne quelconque.

```
# Classement par ordre croissant des produits
kable(tail(arrange(data_industrie, produits), n = 5L),
      caption = "Ordre croissant")
```

Table 25: Ordre croissant

date	environnement	effectif	satisfaction	produits	charges	gain
1975	Favorable	4	3	7277	5833	1444
1978	Favorable	1	3	7409	5555	1854
1970	Very favorable	2	3	7550	5185	2365
1974	Favorable	3	1	7676	5925	1751
1971	Very favorable	1	2	7975	5000	2975

```
# Classement par odre decroissant des charges
kable(tail(arrange(data_industrie, desc(charges)), n = 5L),
      caption = "Ordre decroissant")
```

Table 26: Ordre decroissant

date	environnement	effectif	satisfaction	produits	charges	gain
2006	Favorable	2	6	5589	3983	1606
1985	Very favorable	4	3	5093	3952	1141
2018	Very favorable	3	4	6529	3952	2577
2009	Favorable	1	2	6203	3566	2637
2008	Very favorable	1	6	6073	3335	2738

5.4 Le verbe `mutate`

Ce verbe permet de créer de nouvelles colonnes dans le dataframe sur base des colonnes existantes, mais aussi de supprimer une ou plusieurs autres colonnes.

```
# Creation d'une colonne qui calcule le taux de marge beneficiaire
kable(head(mutate(data_industrie,
                  profit_margin = gain*100/produits)),
      caption = "Creation des variables")
```

Table 27: Creation des variables

date	environnement	effectif	satisfaction	produits	charges	gain	profit_margin
1970	Very favorable	2	3	7550	5185	2365	31.324503
1971	Very favorable	1	2	7975	5000	2975	37.304075
1972	High risk	3	2	4490	6104	-1614	-35.946548
1973	Low risk	1	4	4691	4568	123	2.622042
1974	Favorable	3	1	7676	5925	1751	22.811360
1975	Favorable	4	3	7277	5833	1444	19.843342

```
# Suppression de la colonne des effectifs
kable(head(mutate(data_industrie, effectif = NULL), n = 5L),
      caption = "Suppression des variables")
```

Table 28: Suppression des variables

date	environnement	satisfaction	produits	charges	gain
1970	Very favorable	3	7550	5185	2365
1971	Very favorable	2	7975	5000	2975
1972	High risk	2	4490	6104	-1614
1973	Low risk	4	4691	4568	123
1974	Favorable	1	7676	5925	1751

5.5 L'opérateur *pipe*

Il est possible d'utiliser de manière enchaînée plusieurs verbes `dplyr`. Pour y arriver, l'on utilise l'operateur *pipe* (`%>%`) qui permet de créer une liaison entre les verbes qui se succèdent[1]. Cette syntaxe est connue sous le nom de "tuyautage" (*piping*).

```
# Illustration 1
exemple1 <- data_industrie %>%
  select(-environnement) %>%
  arrange(charges)
kable(head(exemple1, n = 5L), caption = "Illustration 1")
```

Table 29: Illustration 1

date	effectif	satisfaction	produits	charges	gain
2008	1	6	6073	3335	2738
2009	1	2	6203	3566	2637
1985	4	3	5093	3952	1141
2018	3	4	6529	3952	2577
2006	2	6	5589	3983	1606

```
# Illustration 2
exemple2 <- data_industrie %>%
  group_by(environnement) %>%
  mutate(profit_margin = gain*100/produits,
         cost_ratio = charges*100/produits) %>%
  summarise_each(funs(mean, sd), profit_margin, cost_ratio)
kable(exemple2, caption = "Illustration 2")
```

[1] *L'opérateur "pipe" s'obtient sur l'IDE RStudio à l'aide de la combinaison des touches* ***Ctrl + Shift + M*** *pour les laptops Windows. Pour les Macs, on utilise* ***Cmd + Shift + M****.*

Table 30: Illustration 2

environnement	profit_margin_mean	cost_ratio_mean	profit_margin_sd	cost_ratio_sd
Favorable	23.122985	76.87702	9.129057	9.129057
High risk	-6.277908	106.27791	13.000389	13.000389
Low risk	7.905382	92.09462	5.131687	5.131687
Very favorable	30.227506	69.77249	8.036847	8.036847

6 Concaténation des dataframes

La dernière opération relative à la manipulation des données, c'est la fusion des dataframes. Cette fusion peut se réaliser par lignes, par colonnes, ou suivant d'autres critères de jointure.

6.1 Concaténation par colonnes

Créons deux dataframes composés chacun de deux vecteurs générés aléatoirement, puis fusionnons-les par colonnes à l'aide de la fonction `cbind`.

```
# Graine de generation des nombres aleatoires
set.seed(999)

# Creation du premier dataframe
v1 <- sample(x=seq(from=0.1, to=0.9, by=0.2), size=4, replace=T)
v2 <- sample(x = 1:5, size = 4, replace = T)
df1 <- data.frame(v1, v2)
kable(df1, row.names = TRUE, caption = "DataFrame I")
```

Table 31: DataFrame I

	v1	v2
1	0.5	1
2	0.7	2
3	0.9	2
4	0.1	3

```
# Creation du deuxieme dataframe
v3 <- sample(x = c(TRUE, FALSE), size = 4, replace = T)
v4 <- sample(x = c("Blue", "Red"), size = 4, replace = T)
df2 <- data.frame(v3, v4)
kable(df2, row.names = TRUE, caption = "DataFrame II")
```

Table 32: DataFrame II

	v3	v4
1	TRUE	Blue
2	FALSE	Red
3	TRUE	Red
4	TRUE	Blue

```
# Concatenation par colonnes
df_col <- cbind(df1, df2)
kable(df_col, row.names = TRUE, caption = "Fusion par colonnes")
```

Table 33: Fusion par colonnes

	v1	v2	v3	v4
1	0.5	1	TRUE	Blue
2	0.7	2	FALSE	Red
3	0.9	2	TRUE	Red
4	0.1	3	TRUE	Blue

6.2 Concaténation par lignes

Créons un nouveau dataframe composé de quatre vecteurs générés aléatoirement, puis fusionnons-le avec le dataframe créé à la sous-section précédente par lignes à l'aide de la fonction `rbind`.

```
# Graine de generation des nombres aleatoires
set.seed(999)

# Creation du nouveau dataframe
v1 <- sample(x=seq(from=0.1, to=0.9, by=0.2), size=3, replace=T)
v2 <- sample(x = 5:9, size = 3, replace = T)
v3 <- sample(x = c(TRUE, FALSE), size = 3, replace = T)
v4 <- sample(x = c("Green", "Yellow"), size = 3, replace = T)
df3 <- data.frame(v1, v2, v3, v4)
kable(df3, row.names = TRUE, caption = "DataFrame III")
```

Table 34: DataFrame III

	v1	v2	v3	v4
1	0.5	5	FALSE	Green
2	0.7	5	FALSE	Yellow
3	0.9	6	TRUE	Green

```r
# Concatenation par lignes
df_row <- rbind(df_col, df3)
kable(df_row, row.names = TRUE, caption = "Fusion par lignes")
```

Table 35: Fusion par lignes

	v1	v2	v3	v4
1	0.5	1	TRUE	Blue
2	0.7	2	FALSE	Red
3	0.9	2	TRUE	Red
4	0.1	3	TRUE	Blue
5	0.5	5	FALSE	Green
6	0.7	5	FALSE	Yellow
7	0.9	6	TRUE	Green

6.3 Jointure des dataframes

Contrairement à la concaténation (*to bind*), la jointure (*to join*) concernent des dataframes différents, mais ayant une ou plusieurs colonnes communes. Nous verrons dans cette dernière sous-section comment fusionner deux dataframes ayant quelques colonnes communes, tout en gérant efficacement la présence des valeurs manquantes.

```r
# Creation d'un dataframe fictif
v1 <- select(df_row[-7,], v1)
v2 <- select(df_row[-7,], v2)
v5 <- seq(from = 0, to = 1, by = 0.2)
v6 <- c(rep(x = 9, times = 3), rep(x = 1, times = 3))
df4 <- data.frame(v1, v2, v5, v6)
kable(df4, row.names = TRUE, caption = "DataFrame IV")
```

Table 36: DataFrame IV

	v1	v2	v5	v6
1	0.5	1	0.0	9
2	0.7	2	0.2	9
3	0.9	2	0.4	9
4	0.1	3	0.6	1
5	0.5	5	0.8	1
6	0.7	5	1.0	1

Les dataframes présentés dans les tableaux 35 et 36 ont deux colonnes communes, ce sont les colonnes “v1” et “v2”. Ainsi, nous allons procéder à leur fusion partant de ces colonnes communes. Cependant, étant donné que ces dataframes ne comportent pas le même nombre

de lignes ($7 > 6$), il y aura nécessité de gérer les valeurs manquantes susceptibles d'être ajoutées après fusion. Pour cela, nous ferons recours aux fonctions de jointure disponibles dans le package `dplyr`.

```
library(dplyr)

# Jointure partielle
merging1 <- left_join(x = df_row, y = df4, by = c("v1","v2"))
kable(merging1, row.names = TRUE, caption = "Jointure I")
```

Table 37: Jointure I

	v1	v2	v3	v4	v5	v6
1	0.5	1	TRUE	Blue	0.0	9
2	0.7	2	FALSE	Red	0.2	9
3	0.9	2	TRUE	Red	0.4	9
4	0.1	3	TRUE	Blue	0.6	1
5	0.5	5	FALSE	Green	0.8	1
6	0.7	5	FALSE	Yellow	1.0	1
7	0.9	6	TRUE	Green	NA	NA

```
# Jointure interieure
merging2 <- inner_join(x = df_row, y = df4, by = c("v1","v2"))
kable(merging2, row.names = TRUE, caption = "Jointure II")
```

Table 38: Jointure II

	v1	v2	v3	v4	v5	v6
1	0.5	1	TRUE	Blue	0.0	9
2	0.7	2	FALSE	Red	0.2	9
3	0.9	2	TRUE	Red	0.4	9
4	0.1	3	TRUE	Blue	0.6	1
5	0.5	5	FALSE	Green	0.8	1
6	0.7	5	FALSE	Yellow	1.0	1

```r
# Jointure complete
merging3 <- full_join(x = df_row[-1,], y = df4, by = c("v1","v2"))
kable(merging3, row.names = TRUE, caption = "Jointure III")
```

Table 39: Jointure III

	v1	v2	v3	v4	v5	v6
1	0.7	2	FALSE	Red	0.2	9
2	0.9	2	TRUE	Red	0.4	9
3	0.1	3	TRUE	Blue	0.6	1
4	0.5	5	FALSE	Green	0.8	1
5	0.7	5	FALSE	Yellow	1.0	1
6	0.9	6	TRUE	Green	NA	NA
7	0.5	1	NA	NA	0.0	9

- Jointure partielle : Elle permet de joindre au dataframe de gauche (x) les individus dans le dataframe de droite (y), tout en conservant parmi les individus du dataframe de droite ceux qui n'ont pas d'observations pour quelques colonnes dans le dataframe de gauche. Ces valeurs manquantes sont complétées par des NAs.
- Jointure intérieure : Elle permet de joindre au dataframe de gauche (x) les individus dans le dataframe de droite (y), tout en supprimant parmi les individus du dataframe de droite ceux qui n'ont pas d'observations pour quelques colonnes dans le dataframe de gauche. Après fusion, il n'y aura aucune valeur manquante générée.
- Jointure complète : Elle permet de joindre au dataframe de gauche (x) les individus dans le dataframe de droite (y), tout en conservant parmi les individus du dataframe de droite ceux qui n'ont pas d'observations pour quelques colonnes dans le dataframe de gauche, mais en consérvant également les individus du dataframe de gauche qui n'ont pas d'observations pour quelques colonnes du dataframe de droite. Dans ce cas, il y aura des valeurs manquantes générées dans les deux sens.

Il existe d'autres fonctions de jointure disponibles dans le package `dplyr`. L'on peut citer notamment la demi-jointure (`semi_join`) et la jointure contraire (`anti_join`). Le lecteur intéressé par ces fonctions additionnelles peut se référer à l'excellent ouvrage de Christian Paroissin, "*Pratique de la Data Science avec R*" publié aux éditions *Ellipses* en 2021, pour de plus amples détails d'utilisation et spécificités par rapport aux trois fonctions de jointure abordées dans le cadre de cette leçon.