Janitor: Reproduction des tableaux sur RmarkDown

NFEGUE, CREPIN, JEAN-PIERRE, TIENDREBEOGO

DESCRIPTION

Les principales fonctions du Package Janitor peuvent :

- formater parfaitement les noms de colonnes des data.frame ;
- fournir des comptages rapides des combinaisons de variables (c.-à-d. la fréquence) de colonnes data.frame;
- explorer les enregistrements en double. D'autres fonctions de concierge formatent joliment les résultats de la tabulation.

Ces fonctions de tabulation et de rapport se rapprochent des caractéristiques populaires de SPSS et de Microsoft Excel.

- Ce package suit les principes de "l'inverse" et fonctionne bien avec la fonction pipe %>%. janitor a été conçu pour les utilisateurs débutants et intermédiaires de R et est optimisé pour être facile à utiliser, optimisé pour la convivialité.

OPERATIONNALITE

- Le package janitor comprend (mais pas seulement) un certain nombre de fonctions pour améliorer vos tableaux de contingence. En Anglais, "janitor" veut dire (plus ou moins) "homme à tout faire".
- Et en effet, outre les tâches liées à la mise en forme de ces tableaux, le package janitor vise à prendre en charge pour vous un certain nombre de tâches aussi variées que pénibles .

IMPORTATION DES PACKAGES NECESSAIRES

Nous allons d'abord installer et importer le package Janitor.

```
#install.package(janitor)
library(janitor)
```

Toutefois, d'autres packages nous ont été utiles dans notre exposé.

```
library(tidyverse)
library(dplyr)
library(gt)
library(knitr)
library(tinytex)
library(readxl)
library(flextable)
```

VUE D'ENSEMBLE DE QUELQUES FONCTIONS DU PACKAGE JANITOR

add_totals_col()

add_totals_row()

adorn_ns()

adorn_pct_formatting ()

adorn_percentages()

adorn_rounding()

adorn_title()

adorn_totals()

clean_names()

compare_df_cols()

compare_df_cols_same()

convert_to_date()

excel_numeric_to_date()

get_dupes()

get_one_to_one()

make_clean_names()

remove_constant()

remove_empty()

remove_empty_cols()

remove_empty_rows()

round_half_up()

round_to_fraction()

row_to_names()

tabyl()

top_levels()

untabyl()

use_first_valid_of()

I. LES FONCTIONS DE NETTOYAGE

1. La fonction clean_names()

Appelez cette fonction à chaque fois que vous lisez des données. Elle fonctionne dans un pipeline %>% et gère les noms de variables problématiques, en particulier ceux qui sont si bien préservés par readxl::read_excel() et readr::read_csv().

- Analyse la casse des lettres et les séparateurs dans un format cohérent.

La valeur par défaut est snake_case, mais d'autres cas comme camelCase sont disponibles.

- Gère les caractères spéciaux et les espaces, y compris la translittération de caractères tels que œ en oe.

- Ajoute des chiffres aux noms dupliqués
- Convertit “%” en “pourcentage” et “#” en “nombre” pour conserver la signification. L’espacement (ou l’absence d’espacement) autour des nombres est préservé.

```
# Create a data.frame with dirty names

test_df <- as.data.frame(matrix(ncol = 6))
names(test_df) <- c("firstName", "ábc@!*", "% successful (2009)",
                    "REPEAT VALUE", "REPEAT VALUE", "")
test_df
```

```
##   firstName ábc@!* % successful (2009) REPEAT VALUE REPEAT VALUE
## 1        NA     NA                   NA           NA           NA NA
```

Cette fonction nettoie les noms des variables et renvoie un data.frame :

```
test_df %>%
  clean_names()
```

```
##   first_name abc percent_successful_2009 repeat_value repeat_value_2  x
## 1         NA  NA                      NA           NA             NA NA
```

Comparez à ce que la base R produit:

```
make.names(names(test_df))
```

```
## [1] "firstName"            "ábc..."               "X..successful..2009."
## [4] "REPEAT.VALUE"         "REPEAT.VALUE"         "X"
```

- D’autres fonctionnalités de la fonction clean_names()

```
# --- Simple Usage ---
x <- data.frame(caseID = 1, DOB = 2, Other = 3)
clean_names(x)
```

```
##   case_id dob other
## 1       1   2     3
```

```r
# or pipe in the input data.frame:
x %>%
clean_names()
```

```
##   case_id dob other
## 1       1   2     3
```

```r
# if you prefer camelCase variable names:
x %>%
clean_names(., "lower_camel")
```

```
##   caseId dob other
## 1      1   2     3
```

```r
# --- Taking advantage of the underlying snakecase::to_any_case arguments ---
# Restore column names to Title Case, e.g., for plotting

mtcars %>%
clean_names(case = "title") %>%
  head()
```

```
##                    Mpg Cyl Disp  Hp Drat    Wt  Qsec Vs Am Gear Carb
## Mazda RX4         21.0   6  160 110 3.90 2.620 16.46  0  1    4    4
## Mazda RX4 Wag     21.0   6  160 110 3.90 2.875 17.02  0  1    4    4
## Datsun 710        22.8   4  108  93 3.85 2.320 18.61  1  1    4    1
## Hornet 4 Drive    21.4   6  258 110 3.08 3.215 19.44  1  0    3    1
## Hornet Sportabout 18.7   8  360 175 3.15 3.440 17.02  0  0    3    2
## Valiant           18.1   6  225 105 2.76 3.460 20.22  1  0    3    1
```

```r
# Tell clean_names to leave certain abbreviations untouched:
x %>%
clean_names(case = "upper_camel", abbreviations = c("ID", "DOB"))
```

```
##   CaseID DOB Other
## 1      1   2     3
```

Cette fonction est alimentée par la fonction exportée sous-jacente make_clean_names(), qui accepte et renvoie un vecteur de caractères de noms (voir ci-dessous). Cela permet de nettoyer les noms de n'importe

quel objet, et pas seulement d'un data.frame. clean_names() est conservée pour sa commodité dans les flux de travail en pipeline, et peut être appelée sur un objet sf simple features ou un objet tbl_graph tidygraph en plus d'un data.frame.

**Ces data.frames contiennent-ils réellement les mêmes colonnes ?

2. La fonction compare_df_cols()

Pour les cas où vous disposez d'un ensemble de fichiers de données qui devraient être identiques et que vous souhaitez lire et combiner pour l'analyse. Mais dplyr::bind_rows() ou rbind() échoue, parce que les colonnes sont différentes ou parce que les classes de colonnes ne correspondent pas d'un data.frame à l'autre. La fonction compare_df_cols() prend des noms de data.frames / tibbles non quotés, ou une liste de data.frames, et renvoie un résumé de la façon dont ils se comparent. Voyez quels sont les types de colonnes, lesquels sont absents ou présents dans les différentes entrées, et comment les types de colonnes diffèrent.

```
df1 <- data.frame(a = 1:2, b = c("big", "small"))
df2 <- data.frame(a = 10:12, b = c("medium", "small", "big"), c = 0, stringsAsFactors = TRUE) # here, c
df3 <- df1 %>%
  dplyr::mutate(b = as.character(b))

compare_df_cols(df1, df2, df3)
```

```
##   column_name       df1     df2       df3
## 1           a   integer integer   integer
## 2           b character  factor character
## 3           c      <NA> numeric      <NA>
```

```
compare_df_cols(df1, df2, df3, return = "mismatch")
```

```
##   column_name       df1    df2       df3
## 1           b character factor character
```

```
compare_df_cols(df1, df2, df3, return = "mismatch", bind_method = "rbind") # default is dplyr::bind_row
```

```
##   column_name       df1     df2       df3
## 1           b character  factor character
## 2           c      <NA> numeric      <NA>
```

compare_df_cols_same() renvoie VRAI ou FAUX indiquant si les data.frames peuvent être reliés avec succès à l'aide de la méthode de liaison donnée :

```
compare_df_cols_same(df1, df3)
```

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

```
compare_df_cols_same(df2, df3)
```

```
##   column_name    ..1       ..2
## 1           b factor character
```

```
## [1] FALSE
```

3. La fonction make_clean_names()

Comme la fonction make.names() de R de base, mais avec le style et le choix des cas de la fonction clean_names() de l'administrateur de longue date. Alors que clean_names() est toujours proposée pour une utilisation dans le pipe line data.frame avec %>%, make_clean_names() permet une utilisation plus générale, par exemple sur un vecteur.

Il peut également être utilisé comme argument de .name_repair dans la dernière version de tibble::as_tibble :

```
tibble::as_tibble(iris, .name_repair = janitor::make_clean_names)
```

```
## # A tibble: 150 x 5
##    sepal_length sepal_width petal_length petal_width species
##           <dbl>       <dbl>        <dbl>       <dbl> <fct>
##  1          5.1         3.5          1.4         0.2 setosa
##  2          4.9         3            1.4         0.2 setosa
##  3          4.7         3.2          1.3         0.2 setosa
##  4          4.6         3.1          1.5         0.2 setosa
##  5          5           3.6          1.4         0.2 setosa
##  6          5.4         3.9          1.7         0.4 setosa
##  7          4.6         3.4          1.4         0.3 setosa
```

```
##  8          5           3.4          1.5         0.2 setosa
##  9          4.4         2.9          1.4         0.2 setosa
## 10          4.9         3.1          1.5         0.1 setosa
## # i 140 more rows
```

4. La fonction remove_empty()

Fait ce qu'elle dit. Pour des cas comme le nettoyage de fichiers Excel qui contiennent des lignes et des colonnes vides après avoir été lus dans R.

```
q <- data.frame(v1 = c(1, NA, 3),
                    v2 = c(NA, NA, NA),
                    v3 = c("a", NA, "b"))
q
```

```
##   v1 v2   v3
## 1  1 NA    a
## 2 NA NA <NA>
## 3  3 NA    b
```

```
q %>%remove_empty(c("rows", "cols")) %>%
    flextable::flextable() %>%    # convertir en une belle image
  flextable::autofit()
```

```
## Warning: fonts used in ‘flextable‘ are ignored because the ‘pdflatex‘ engine is
## used and not ‘xelatex‘ or ‘lualatex‘. You can avoid this warning by using the
## ‘set_flextable_defaults(fonts_ignore=TRUE)‘ command or use a compatible engine
## by defining ‘latex_engine: xelatex‘ in the YAML header of the R Markdown
## document.
```

v1	v3
1	a
3	b

5. La fonction remove_constant()

Supprime les colonnes d'un data.frame qui ne contiennent qu'une seule valeur constante (avec une option na.rm pour contrôler si les NA doivent être considérés comme des valeurs différentes de la constante).

remove_constant et remove_empty fonctionnent aussi bien sur les matrices que sur les data.frames.

```
a <- data.frame(good = 1:3, boring = "the same")
a
```

```
##   good   boring
## 1    1 the same
## 2    2 the same
## 3    3 the same
```

```
a %>% remove_constant()
```

```
##   good
## 1    1
## 2    2
## 3    3
```

6. La fonction get_dupes()

Cette fonction permet de rechercher et d'examiner les enregistrements en double lors du nettoyage des données - généralement lorsqu'il ne devrait pas y en avoir. Par exemple, dans un data.frame bien rangé, on peut s'attendre à ce qu'un identifiant unique soit répété pour chaque année, mais qu'il n'y ait pas de paires d'identifiants uniques et d'années dupliquées. Supposons que vous souhaitiez vérifier et étudier ces enregistrements dupliqués. get_dupes() renvoie les enregistrements (et insère un nombre de doublons) afin que vous puissiez examiner les cas problématiques :

```r
get_dupes(mtcars, wt, cyl) # or mtcars %>% get_dupes(wt, cyl) if you prefer to pipe
```

```
##     wt cyl dupe_count  mpg  disp  hp drat  qsec vs am gear carb
## 1 3.44   6          2 19.2 167.6 123 3.92 18.30  1  0    4    4
## 2 3.44   6          2 17.8 167.6 123 3.92 18.90  1  0    4    4
## 3 3.57   8          2 14.3 360.0 245 3.21 15.84  0  0    3    4
## 4 3.57   8          2 15.0 301.0 335 3.54 14.60  0  1    5    8
```

7. La fonction get_one_to_one()

Cette fonction montre quelles colonnes d'un data.frame ont, le cas échéant, une relation biunivoque entre elles. Voici un petit exemple qui examine les quatre premières lignes du data.frame starwars du package dplyr. Les variables sont regroupées en trois ensembles de groupes univoques

```r
library(dplyr)
starwars[1:4,]
```

```
## # A tibble: 4 x 14
##   name      height  mass hair_color skin_color eye_color birth_year sex   gender
##   <chr>      <int> <dbl> <chr>      <chr>      <chr>          <dbl> <chr> <chr>
## 1 Luke Sky~    172    77 blond      fair       blue            19   male  mascu~
## 2 C-3PO        167    75 <NA>       gold       yellow         112   none  mascu~
## 3 R2-D2         96    32 <NA>       white, bl~ red             33   none  mascu~
## 4 Darth Va~    202   136 none       white      yellow          41.9 male  mascu~
## # i 5 more variables: homeworld <chr>, species <chr>, films <list>,
## #   vehicles <list>, starships <list>
```

```r
starwars[1:4,] %>%
  get_one_to_one()
```

```
## [[1]]
## [1] "name"       "height"     "mass"       "skin_color" "birth_year"
## [6] "films"
##
```

```
## [[2]]
## [1] "hair_color" "starships"
##
## [[3]]
## [1] "sex"     "species"
```

8. La fonction round_half_up()

R utilise « l'arrondissement du banquier », c'est-à-dire que les moitiés sont arrondies au nombre pair le plus proche. Cette fonction, une implémentation exacte de https://stackoverflow.com/questions/12688717/round-up-from-5/12688836#12688836, arrondira toutes les moitiés. Comparer:

```
nums <- c(2.5, 3.5)
round(nums)
```

```
## [1] 2 4
```

```
round_half_up(nums)
```

```
## [1] 3 4
```

9. La fonction round_to_fraction()

Supposons que vos données ne doivent avoir que des valeurs de trimestres 0, 0,25, 0,5, 0,75, 1, etc. Mais il existe soit des valeurs incorrectes entrées par l'utilisateur, comme ou des problèmes de précision en virgule flottante comme . volonté Appliquer la distribution fractionnaire souhaitée en arrondissant les valeurs à la valeur la plus proche étant donné le dénominateur spécifié.0.20.25000000001round_to_fraction()

Il y a aussi un argument pour la suite facultative arrondi.digits

10. La fonction excel_numeric_to_date()

Avez-vous déjà chargé des données à partir d'Excel et vu une valeur comme l'endroit où une date devrait être? Cette fonction convertit ces numéros de série en classe , avec des options pour différents encodage de date Excel systèmes, préservant des fractions d'une date comme temps (auquel cas le la valeur renvoyée est de classe), et en spécifiant une heure zone.42223DatePOSIXl

```
excel_numeric_to_date(41103)
```

```
## [1] "2012-07-13"
```

```
excel_numeric_to_date(41103.01) # ignores decimal places, returns Date object
```

```
## [1] "2012-07-13"
```

```
excel_numeric_to_date(41103.01, include_time = TRUE) # returns POSIXlt object
```

```
## [1] "2012-07-13 00:14:24 GMT"
```

```
excel_numeric_to_date(41103.01, date_system = "mac pre-2011")
```

```
## [1] "2016-07-14"
```

11. La fonction convert_to_date()

Convertir un Mélange de formats date et datetime à la date

S'appuyant sur , les nouvelles fonctions et sont plus robustes à un mélange d'entrées. Pratique lors de la lecture de nombreuses feuilles de calcul Cela devrait avoir les mêmes formats de colonne, mais ce n'est pas le cas.excel_numeric_to_date()convert_to_date()convert_to_datetime()

Par exemple, ici un vecteur avec une date et une date Excel heure voit les deux valeurs ont été converties avec succès en classe Date :

```
convert_to_date(c("2020-02-29", "40000.1"))
```

```
## [1] "2020-02-29" "2009-07-06"
```

12. La fonction row_to_names()

Si un fichier data.frame a les noms de variables prévus stockés dans l'un de ses lignes, élèvera la ligne spécifiée à deviennent les noms du fichier data.frame et éventuellement (par défaut) suppriment la ligne dans laquelle les noms ont été stockés et/ou les lignes au-dessus.row_to_names()

```
dirt <- data.frame(X_1 = c(NA, "ID", 1:3),
                   X_2 = c(NA, "Value", 4:6))

dirt %>%
  find_header()
```

```
## [1] 2
```

```
row_to_names(dirt, 2)
```

```
##   ID Value
## 3  1     4
## 4  2     5
## 5  3     6
```

13. La fonction find_header()

La fonction est une fonction compagnon de . Par défaut, il recherchera un data.frame pour la première ligne sans valeurs manquantes et renvoyer ce numéro de ligne.find_header()row_to_names()

Il peut également être utilisé pour renvoyer le numéro de ligne où une chaîne donnée est présent dans la première colonne ou dans une colonne spécifique. Ensuite, ce résultat peut être fourni à .row_to_names()

14. La fonction top_levels()

Conçu à l'origine pour être utilisé avec les données d'enquête de Likert stockées en tant que Facteurs. Renvoie une table de fréquences avec lignes nommées de manière appropriée, regroupées en groupes tête/milieu/queue.tbl_df

Prend une taille spécifiée par l'utilisateur pour les groupes tête/queue Calcule automatiquement une colonne de pourcentage Prise en charge du tri Peut afficher ou masquer des valeurs.NA

```
f <- factor(c("strongly agree", "agree", "neutral", "neutral", "disagree", "strongly agree"),
            levels = c( "agree", "neutral", "disagree", "strongly disagree"))
top_levels(f)
```

```
##                             f n percent
##               agree, neutral 3    0.75
##  disagree, strongly disagree 1    0.25
```

```
top_levels(f, n = 1)
```

```
##                   f n percent
##              agree 1    0.25
##  neutral, disagree 3    0.75
##  strongly disagree 0    0.00
```

II. LES FONCTIONS DE TABLEAUX

1. La fonction tabyl():un meilleure version de table()

tabyl() est un remplacement de table() orienté tidyverse. Elle compte les combinaisons d'une, deux ou trois variables, et peut ensuite être formatée à l'aide d'une série de fonctions adorn_* pour ressembler à ce que vous voulez. Par exemple :

```
mtcars %>%
  tabyl(gear, cyl) %>%
  adorn_totals("col") %>%
  adorn_percentages("row") %>%
  adorn_pct_formatting(digits = 2) %>%
  adorn_ns() %>%
  adorn_title(placement = "combined") %>%
```

```
    flextable::flextable() %>%
  flextable::autofit()
```

```
## Warning: fonts used in ‘flextable‘ are ignored because the ‘pdflatex‘ engine is
## used and not ‘xelatex‘ or ‘lualatex‘. You can avoid this warning by using the
## ‘set_flextable_defaults(fonts_ignore=TRUE)‘ command or use a compatible engine
## by defining ‘latex_engine: xelatex‘ in the YAML header of the R Markdown
## document.
```

gear/cyl4	6	8	Total
36.67% (1)	13.33% (2)	80.00% (12)	100.00% (15)
466.67% (8)	33.33% (4)	0.00% (0)	100.00% (12)
540.00% (2)	20.00% (1)	40.00% (2)	100.00% (5)

Comme vous avez pu le constater, l'utilisation de tabyl() s'accompagne des fonctions suivantes: - adorn_totals() - adorn_percentages() - adorn_pct_formatting() - adorn_ns() - adorn_title()

2. La fonction adorn_totals()

Cette donction ajoute une colonne de totaux à un data.frame.

- Utilisation adorn_totals(dat, na.rm = TRUE) ou adorn_totals("col") pour spécifier en colonne
- Arguments dat un data.frame en entrée avec au moins une colonne numérique. na.rm Les valeurs manquantes (y compris NaN) doivent-elles être omises des calculs ?
- Valeur Renvoie un data.frame avec une colonne de totaux contenant des sommes sur une ligne.

3. La fonction adorn_percentages()

Cette fonction convertit un data.frame d'effectifs en pourcentages.

- Description de la fonction Cette fonction exclut par défaut la première colonne du data.frame d'entrée, en supposant qu'elle contienne une variable descriptive, mais il est possible d'y déroger en spécifiant les colonnes à orner dans l'argument.

- Utilisation adorn_percentages(dat, denominator = “row”, na.rm = TRUE, . . .)
- Arguments denominator la direction à utiliser pour calculer les pourcentages. L’un des choix suivants : “row”, “col” ou “all”.

4. La fonction adorn_pct_formatting()

Elle formate un data.frame de décimales en pourcentages.

5. La fonction adorn_ns()

Cette fonction ajoute les N sous-jacents à un tableau affichant les pourcentages.

6. La fonction adorn_title()

Elle permet d’ajouter le nom d’une colonne en haut d’un tableau à double sens.

```
mtcars %>%
tabyl(am, cyl) %>%
adorn_title(placement = "top")
```

```
##     cyl
##  am   4 6  8
##   0   3 4 12
##   1   8 3  2
```

III. PHASE PRATIQUE : APPLICATION DE QUELQUE FONCTIONS DE JANITOR

Exemple sur une base

```
My_data <- read_excel("My_data.xlsx")
My_data %>%
  gt() %>%
  tab_header(title =md("**My_base**"),subtitle = md("Une base de donnée **fictive** sur d'une entrepris
  tab_source_note("My_data")
```

My_base

Une base de donnée **fictive** sur d'une entreprise quelconque

ID	Prenom	Âge	Sexe	Salaire	Nationalité	Marié	Langues	Diplôme	Expérience
34EDJN	Alice	28	F	50000	France	Marié	Français	Master	5
34TYUI	Bob	32	M	60000	États-Unis	Celibataire	Anglais	Bachelor	3
456TFRD	Charlie	45	M	75000	Royaume-Uni	Marié	Anglais, Espagnol	Doctorat	12
65R45RT	David	19	M	40000	Canada	Celibataire	Anglais, Français	Bachelor	3
FRE345	Eve	54	F	90000	Australie	Celibataire	Anglais	Master	15
6YGTFR	Frank	41	M	65000	Allemagne	Marié	Allemand	Bachelor	10
567UH	George	36	M	70000	Italie	Marié	Italien	Master	7
Y6765	Hannah	29	F	55000	Brésil	Celibataire	Portugais	Doctorat	3
FRE345	Isaac	50	M	80000	Espagne	Marié	Espagnol	Bachelor	9
5RT65	Julia	33	F	60000	Chine	Celibataire	Chinois	Master	6
34RT6	Kate	27	F	55000	France	Marié	Français, Anglais	Bachelor	3
677ED	Liam	31	M	65000	États-Unis	Celibataire	Anglais	Master	11
234BH	Mia	40	F	70000	Allemagne	Marié	Allemand, Français	Doctorat	14
56YGHN	Toh	NA	F	140000	NA	NA	Chinois	Master	NA
34EDJN	Abou	55	M	200000	France	NA	Anglais	Bachelor	20

My_data

```
My_data %>%
  names()
```

```
##  [1] "ID"           "Prenom"       "Âge"          "Sexe"         "Salaire"
##  [6] "Nationalité"  "Marié"        "Langues"      "Diplôme"      "Expérience"
## [11] "Bonus %"      "Nbre d'annee"
```

La fonction clean_names()

```
clean = My_data %>%
  janitor:: clean_names()
data.frame(colnames(My_data),colnames(clean)) %>%
  gt()
```

colnames.My_data.	colnames.clean.
ID	id
Prenom	prenom
Âge	age
Sexe	sexe
Salaire	salaire
Nationalité	nationalite
Marié	marie
Langues	langues
Diplôme	diplome
Expérience	experience
Bonus %	bonus_percent
Nbre d'annee	nbre_dannee

La fonction remove_empty()

```
clean[,ncol(clean)]
```

```
## # A tibble: 15 x 1
##    nbre_dannee
##    <lgl>
##  1 NA
##  2 NA
##  3 NA
##  4 NA
##  5 NA
```

```
##  6 NA
##  7 NA
##  8 NA
##  9 NA
## 10 NA
## 11 NA
## 12 NA
## 13 NA
## 14 NA
## 15 NA
```

```
clean_x = clean %>%
  janitor::remove_empty(which = "cols")
clean_x %>%
  names
```

```
##  [1] "id"            "prenom"        "age"           "sexe"
##  [5] "salaire"       "nationalite"   "marie"         "langues"
##  [9] "diplome"       "experience"    "bonus_percent"
```

La fonction compare_df_cols()

```
clean %>%
compare_df_cols(clean,clean_x) %>%
  gt()
```

column_name	.	clean	clean_x
age	numeric	numeric	numeric
bonus_percent	numeric	numeric	numeric
diplome	character	character	character
experience	numeric	numeric	numeric
id	character	character	character
langues	character	character	character

marie	character	character	character
nationalite	character	character	character
nbre_dannee	logical	logical	NA
prenom	character	character	character
salaire	numeric	numeric	numeric
sexe	character	character	character

La fonction get_dupes()

```
clean %>%
  janitor::get_dupes(id) %>%
  gt()
```

id	dupe_count	prenom	age	sexe	salaire	nationalite	marie	langues	diplome	experience	b
34EDJN	2	Alice	28	F	5e+04	France	Marié	Français	Master	5	
34EDJN	2	Abou	55	M	2e+05	France	NA	Anglais	Bachelor	20	
FRE345	2	Eve	54	F	9e+04	Australie	Celibataire	Anglais	Master	15	
FRE345	2	Isaac	50	M	8e+04	Espagne	Marié	Espagnol	Bachelor	9	

```
clean %>%
  janitor::get_dupes(diplome,experience) %>%
  gt()
```

diplome	experience	dupe_count	id	prenom	age	sexe	salaire	nationalite	marie	langues
Bachelor	3	3	34TYUI	Bob	32	M	60000	États-Unis	Celibataire	Anglais
Bachelor	3	3	65R45RT	David	19	M	40000	Canada	Celibataire	Anglais, Fra
Bachelor	3	3	34RT6	Kate	27	F	55000	France	Marié	Français, A

Base de donnéé

```
Data_USA <- read_excel("Data_MCA.xlsx")
Data_USA %>%
  gt() %>%
  tab_header(title =md("**Data_USA**"),subtitle = md("Une base de donnée reprensatant les caracteristiqu
  tab_source_note("Source : ")
```

caseid	weight	Q1	Q44	Q58	age	gend
1	0.4011694	I definitely voted in the November General Election	Somewhat confident	Yes	[40;45[	Fema
2	0.5278851	I definitely voted in the November General Election	Somewhat confident	No	[50;55[	Fema
3	0.9246396	I definitely voted in the November General Election	Very confident	No	[30;35[	Male
4	0.5855338	I definitely voted in the November General Election	Very confident	No	[55;60[	Fema
5	0.9622483	I did not vote in the election this November	NA	Yes	[50;55[	Fema
6	0.8993339	I definitely voted in the November General Election	Very confident	No	[20;25[	Fema
7	1.0951118	I definitely voted in the November General Election	Very confident	No	[65;70[	Male
8	1.0190446	I definitely voted in the November General Election	Very confident	No	[25;30[	Fema
9	1.1308900	I definitely voted in the November General Election	Very confident	No	[65;70[	Male
10	1.4160690	I definitely voted in the November General Election	Very confident	No	[70;75[	Male
11	0.2495224	I definitely voted in the November General Election	Very confident	No	[25;30[	Fema
12	3.3318090	I definitely voted in the November General Election	Very confident	No	[35;40[	Male
13	0.4502857	I definitely voted in the November General Election	Very confident	No	[30;35[	Fema
14	0.8975509	I definitely voted in the November General Election	Very confident	No	80 et plus	Male
15	0.5999998	I definitely voted in the November General Election	Very confident	No	[45;50[	Fema
16	1.5624177	I usually vote, but didn't this time	NA	No	[35;40[	Male
17	1.4087807	I definitely voted in the November General Election	Somewhat confident	No	[20;25[	Male
18	1.5459984	I definitely voted in the November General Election	Somewhat confident	Yes	[40;45[	Male
19	0.4665837	I definitely voted in the November General Election	Not at all confident	No	[40;45[	Fema
20	1.2460998	I definitely voted in the November General Election	Very confident	No	[40;45[	Male
21	0.6629623	I definitely voted in the November General Election	Very confident	No	[55;60[	Male
22	0.3898009	I definitely voted in the November General Election	Somewhat confident	Yes	[25;30[	Fema
23	0.5674344	I definitely voted in the November General Election	Very confident	Yes	[55;60[	Fema
24	0.4557532	I definitely voted in the November General Election	Somewhat confident	No	[45;50[	Fema
25	0.5325010	I definitely voted in the November General Election	Very confident	Yes	[50;55[	Fema
26	0.6146746	I definitely voted in the November General Election	Somewhat confident	Yes	[60;65[	Male

27	1.0621582	I definitely voted in the November General Election	Somewhat confident	Yes	[60;65[	Male
28	0.6818706	I did not vote in the election this November	NA	Yes	[60;65[	Fema
29	1.7093862	I definitely voted in the November General Election	Somewhat confident	Yes	[40;45[	Fema
30	0.5244768	I did not vote in the election this November	NA	No	[40;45[	Fema
31	1.4535836	I definitely voted in the November General Election	Very confident	No	[25;30[	Fema
32	0.7745307	I definitely voted in the November General Election	Somewhat confident	No	[30;35[	Male
33	0.9873696	I definitely voted in the November General Election	Very confident	No	[40;45[	Male
34	0.9058769	I definitely voted in the November General Election	Very confident	No	80 et plus	Male
35	1.0133174	I definitely voted in the November General Election	Very confident	Yes	80 et plus	Fema
36	0.6912497	I definitely voted in the November General Election	Very confident	No	[55;60[	Male
37	0.6932394	I definitely voted in the November General Election	Somewhat confident	No	[55;60[	Fema
38	0.7388797	I definitely voted in the November General Election	Somewhat confident	No	[65;70[	Male
39	0.5717101	I definitely voted in the November General Election	Very confident	No	[30;35[	Fema
40	0.7314152	I definitely voted in the November General Election	Very confident	No	80 et plus	Male
41	1.7275241	I definitely voted in the November General Election	Very confident	No	[30;35[	Fema
42	0.9246396	I definitely voted in the November General Election	Not at all confident	No	[30;35[	Male
43	0.6251186	I definitely voted in the November General Election	Somewhat confident	No	[75;80[	Male
44	3.0314992	I definitely voted in the November General Election	Very confident	No	[20;25[	Male
45	1.8504624	I definitely voted in the November General Election	Very confident	No	[20;25[	Male
46	0.8603879	I definitely voted in the November General Election	Not too confident	No	[55;60[	Male
47	1.5084816	I definitely voted in the November General Election	Somewhat confident	No	[65;70[	Male
48	1.7100829	I definitely voted in the November General Election	Very confident	Yes	[50;55[	Male
49	0.4697354	I definitely voted in the November General Election	Very confident	Yes	[50;55[	Male
50	1.1929907	I definitely voted in the November General Election	Very confident	No	[40;45[	Male
51	0.7177975	I definitely voted in the November General Election	Very confident	No	[50;55[	Male

Source :

```
Data_USA %>%
  names()
```

```
##  [1] "caseid"          "weight"          "Q1"              "Q44"
##  [5] "Q58"             "age"             "gender"          "race"
##  [9] "educ"            "marstat"         "employ"          "pid3"
## [13] "presvote16post" "inputstate"      "ideo5"           "newsint"
## [17] "religpew"       "pew_religimp"
```

La fonction tabyl() de janitor

Un aperçu de table()

```
table(Data_USA$pid3)
```

```
##
##    Democrat Independent    Not sure       Other  Republican
##          10          11          10          10          10
```

Tableau de contingence

```
table(Data_USA$gender,Data_USA$pid3)
```

```
##
##          Democrat Independent Not sure Other Republican
##   Female        6           4        8     4          0
##   Male          4           7        2     6         10
```

janitor : : tabyl()

```
Data_USA %>%
  janitor::tabyl(pid3) %>%
  gt(rowname_col ="pid3" ) %>%
   tab_header(title =md("**Tableau des effectifs**"))
```

Tableau des effectifs

	n	percent
Democrat	10	0.1960784
Independent	11	0.2156863
Not sure	10	0.1960784

Other	10	0.1960784
Republican	10	0.1960784

Tableau de contingence

```
Data_USA %>%
  janitor:: tabyl(gender,pid3) %>%
  gt("gender") %>%
  tab_header(title =md("**Tableau de contingence : Genre & Part_politique**"))
```

Tableau de contingence : Genre & Part_politique

	Democrat	Independent	Not sure	Other	Republican
Female	6	4	8	4	0
Male	4	7	2	6	10

```
Data_USA %>%
  janitor:: tabyl(gender,pid3) %>%
  knitr:: kable()
```

gender	Democrat	Independent	Not sure	Other	Republican
Female	6	4	8	4	0
Male	4	7	2	6	10

Selection de colonne

table()

```
table(Data_USA$gender,Data_USA$pid3)$Independent
# Cette syntaxe nous retourne une erreur
```

tabyl()

```
tabyl(Data_USA,gender,pid3)$Independent
```

```
## [1] 4 7
```

```r
Data_USA %>%
  janitor:: tabyl(gender,pid3,marstat )
```

```
## $Divorced
##  gender Democrat Independent Not sure Other Republican
##  Female        0           0        0     0          0
##    Male        0           0        0     1          1
##
## $`Domestic / civil partnership`
##  gender Democrat Independent Not sure Other Republican
##  Female        0           0        3     0          0
##    Male        0           1        0     0          1
##
## $Married
##  gender Democrat Independent Not sure Other Republican
##  Female        2           2        0     2          0
##    Male        2           2        2     4          5
##
## $`Never married`
##  gender Democrat Independent Not sure Other Republican
##  Female        3           2        3     1          0
##    Male        1           4        0     1          3
##
## $Separated
##  gender Democrat Independent Not sure Other Republican
##  Female        1           0        1     0          0
##    Male        0           0        0     0          0
##
## $Widowed
##  gender Democrat Independent Not sure Other Republican
##  Female        0           0        1     1          0
##    Male        1           0        0     0          0
```

La fonction adorn_pct_formatting()

```
Data_USA %>%
  janitor::tabyl(pid3) %>%
  janitor::adorn_pct_formatting(digits = 0, affix_sign = TRUE) %>%
  gt(rowname_col ="pid3")
```

	n	percent
Democrat	10	20%
Independent	11	22%
Not sure	10	20%
Other	10	20%
Republican	10	20%

La fonction adorn_totals()

```
# total des lignes, par defaut
Data_USA %>%
  janitor::tabyl(gender,pid3 ) %>%
  janitor::adorn_totals() %>%
  gt(rowname_col ="gender")
```

	Democrat	Independent	Not sure	Other	Republican
Female	6	4	8	4	0
Male	4	7	2	6	10
Total	10	11	10	10	10

```
Data_USA %>%
  janitor::tabyl(gender,pid3 ) %>%
  janitor::adorn_totals(where = c("row","col"))%>%
  gt(rowname_col ="gender") %>%
  tab_header(title =md("**Tableau de contingence**"))
```

Tableau de contingence

	Democrat	Independent	Not sure	Other	Republican	Total
Female	6	4	8	4	0	22
Male	4	7	2	6	10	29
Total	10	11	10	10	10	51

La fonction adorn_title()

```
Data_USA %>%
  janitor::tabyl(gender,pid3 ) %>%
  janitor::adorn_totals(where = c("row","col"))%>%
  janitor::adorn_title(
    row_name = "Gender",
    col_name = "P_poli",
    placement = "combined") %>%
  gt(rowname_col ="gender") %>%
  tab_header(title =md("**Tableau de contingence**"))
```

Tableau de contingence

Gender/P_poli	Democrat	Independent	Not sure	Other	Republican	Total
Female	6	4	8	4	0	22
Male	4	7	2	6	10	29
Total	10	11	10	10	10	51

La fonction adorn_percentages()

```
Data_USA %>%
  janitor::tabyl(religpew,pid3 ) %>%
  janitor::adorn_percentages("row")%>%
  janitor::adorn_totals(where = "col")%>%
  janitor::adorn_title(
    row_name = "Relig",
    col_name = "P_poli",
    placement = "combined") %>%
  gt(rowname_col ="gender") %>%
  tab_header(title =md("**Pourcentages par lignes**"))
```

Pourcentages par lignes

Relig/P_poli	Democrat	Independent	Not sure	Other	Republican	Total
Agnostic	0.16666667	0.3333333	0.3333333	0.1666667	0.0000000	1
Atheist	0.50000000	0.0000000	0.0000000	0.5000000	0.0000000	1
Jewish	0.00000000	1.0000000	0.0000000	0.0000000	0.0000000	1
Nothing in particular	0.08333333	0.3333333	0.3333333	0.0000000	0.2500000	1
Protestant	0.17647059	0.1176471	0.2352941	0.2352941	0.2352941	1
Roman Catholic	0.22222222	0.2222222	0.0000000	0.2222222	0.3333333	1
Something else	0.50000000	0.0000000	0.0000000	0.5000000	0.0000000	1

```r
Data_USA %>%
  janitor::tabyl(religpew,pid3 ) %>%
  janitor::adorn_percentages("row")%>%
  janitor::adorn_totals(where = c("row","col"))%>%
  janitor::adorn_pct_formatting(digits = 0, affix_sign = TRUE) %>%
  janitor::adorn_title(
    row_name = "Relig",
    col_name = "P_poli",
    placement = "combined") %>%
  gt() %>%
  tab_header(title =md("**TABLEAU DE CONTINGENCE**"))
```

TABLEAU DE CONTINGENCE

Relig/P_poli	Democrat	Independent	Not sure	Other	Republican	Total
Agnostic	17%	33%	33%	17%	0%	100%
Atheist	50%	0%	0%	50%	0%	100%
Jewish	0%	100%	0%	0%	0%	100%
Nothing in particular	8%	33%	33%	0%	25%	100%
Protestant	18%	12%	24%	24%	24%	100%
Roman Catholic	22%	22%	0%	22%	33%	100%
Something else	50%	0%	0%	50%	0%	100%
Total	165%	201%	90%	162%	82%	700%

```
Data_USA %>%
  janitor::tabyl(inputstate,pid3 ) %>%
  janitor::adorn_percentages("row")%>%
  janitor::adorn_totals(where = c("row","col"))%>%
  janitor::adorn_pct_formatting(digits = 0, affix_sign = TRUE) %>%
  janitor::adorn_title(
    row_name = "Ville",
    col_name = "P_poli",
    placement = "combined") %>%
  gt(rowname_col ="gender") %>%
  tab_header(title =md("**TABLEAU DE CONTINGENCE**"))
```

TABLEAU DE CONTINGENCE

Ville/P_poli	Democrat	Independent	Not sure	Other	Republican	Total
Arizona	0%	0%	0%	0%	100%	100%
California	40%	20%	0%	0%	40%	100%
Delaware	0%	0%	0%	100%	0%	100%
Florida	0%	75%	0%	25%	0%	100%
Georgia	0%	50%	0%	50%	0%	100%
Illinois	0%	0%	100%	0%	0%	100%
Iowa	0%	0%	100%	0%	0%	100%
Kansas	0%	0%	100%	0%	0%	100%
Kentucky	0%	0%	100%	0%	0%	100%
Louisiana	0%	0%	50%	50%	0%	100%
Massachusetts	0%	0%	0%	100%	0%	100%
Michigan	0%	100%	0%	0%	0%	100%
Minnesota	0%	0%	50%	50%	0%	100%
Missouri	67%	33%	0%	0%	0%	100%
Montana	100%	0%	0%	0%	0%	100%
New Jersey	0%	0%	0%	100%	0%	100%
New Mexico	0%	0%	50%	0%	50%	100%
New York	50%	0%	0%	0%	50%	100%
North Carolina	50%	0%	0%	0%	50%	100%
North Dakota	0%	0%	0%	0%	100%	100%
Ohio	0%	33%	0%	33%	33%	100%

Pennsylvania	20%	60%	0%	0%	20%	100%
South Carolina	0%	0%	0%	100%	0%	100%
Texas	50%	0%	50%	0%	0%	100%
Virginia	0%	0%	100%	0%	0%	100%
Washington	0%	0%	0%	100%	0%	100%
West Virginia	0%	0%	100%	0%	0%	100%
Wisconsin	50%	0%	0%	0%	50%	100%
Total	427%	372%	800%	708%	493%	2800%