# distrr: Un paquet R pour la création de cubes de données

## Journées suisses de la statistique publique 2017

Sandro Petrillo

USTAT, Ufficio di statistica del Canton Ticino

Ittingen, 21 novembre 2017

Qu'est-ce que c'est distrr

Qu'est-ce qu'on entend par cube de données

Créer un cube de données avec `distrr`

Conclusions / Utilisations / Développements

Annexe: exemples avec des sources de statistique publique

# Qu'est-ce que c'est distrr

# Qu'est-ce que c'est distrr

- `distrr` est un paquet `R` qui fournit quelques instruments pour estimer et gérer des distributions empiriques.
- En particulier, une des fonctionnalités principales est la création de cubes de données avec des estimations de statistiques, qui comprennent toutes les combinaisons des variables d'intérêt.
- L'idée de créér `distrr` est née lorsque à l'Ustat on répondait à plusieurs requêtes et on s'est aperçus qu'il fallait "beacoup de croisements".
- Le paquet fait une forte utilisation des outils fournis par `dplyr`, qui est une "grammar of data manipulation".
- `distrr` est disponible sur `CRAN` et peut être installé en digitant dans `R`:

```
install.packages("distrr")
```

# Qu'est-ce qu'on entend par cube de données

# Caractéristiques d'un cube de données dans la statistique publique

- Un conteneur de **résultats de statistique publique**
- Qui comprend toutes les combinaisons des modalités des variables ("croisements"), y compris la modalité "Total" de chaque variable
- Peut contenir une ou plusieurs statistiques pour chaque croisement (estimation de la numérosité, nombre d'entreprises, salaires médians, etc. . . )
- Contient uniquement les résultats de statistique publique qui peuvent être publiés (cette partie est facilement "automatisable")
- Un cube de données est quelque chose entre des données individuelles (pour la richesse de son contenu) et des tableaux "standard" (parce qu'il peut être donné à quiconque sans problèmes de protection des données)

# Créer un cube de données avec `distrr`

# Estimation de statistiques par groupes avec dplyr

- Quand on veut estimer des statistiques par groupes, avec `dplyr` on fait quelque chose du style (en pseudo-code):

```
data %>%
  group_by(une ou plusieurs variables) %>%
  summarise(une ou plusieurs estimations de statistiques)
```

# Exemple avec dplyr

- On utilise les données `invented_wages` du paquet `distrr`, qui contiennent 1.000 observations de salariés inventés.

```
library(distrr)
str(invented_wages)
```

```
## Classes 'tbl_df' and 'data.frame':   1000 obs. of  5 variables:
##  $ gender        : Factor w/ 2 levels "men","women": 1 2 1 2 1 1 1 2 2 1
##  $ sector        : Factor w/ 2 levels "secondary","tertiary": 2 1 2 2 1
##  $ education     : Factor w/ 3 levels "I","II","III": 3 2 2 2 2 1 3 1 2
##  $ wage          : num  8400 4200 5100 7400 4300 4900 5400 2900 4500 300
##  $ sample_weights: num  105 32 36 12 21 46 79 113 34 32 ...
```

# Estimation de la numerosité et des salaires moyens par sexe

- Supposons qu'on veut estimer la numerosité et les salaires moyens par sexe:

```
library(dplyr)
invented_wages %>%
  group_by(gender) %>%
  summarise(n = n(), av_wage = mean(wage))
```

```
## # A tibble: 2 x 3
##   gender     n  av_wage
##   <fctr> <int>    <dbl>
## 1    men   547 5435.466
## 2  women   453 4440.618
```

# Et la même chose mais avec le secteur économique

- On veut estimer la même chose mais selon le secteur économique:

```
library(dplyr)
invented_wages %>%
  group_by(sector) %>%
  summarise(n = n(), av_wage = mean(wage))
```

```
## # A tibble: 2 x 3
##       sector     n  av_wage
##       <fctr> <int>    <dbl>
## 1 secondary   455 4566.154
## 2  tertiary   545 5334.312
```

# Et puis par sexe et secteur économique

```
library(dplyr)
invented_wages %>%
  group_by(gender, sector) %>%
  summarise(n = n(), av_wage = mean(wage))
```

```
## # A tibble: 4 x 4
## # Groups:   gender [?]
##   gender    sector     n  av_wage
##   <fctr>    <fctr> <int>    <dbl>
## 1    men secondary   289 5070.588
## 2    men  tertiary   258 5844.186
## 3  women secondary   166 3687.952
## 4  women  tertiary   287 4875.958
```

# Toutes les combinaisons avec distrr

- On peut faire toutes les trois étapes précedentes avec la fonction dcc5 du paquet distrr:

```
invented_wages %>%
  dcc5(.variables = c("gender", "sector"),
       salaire_moyen = ~mean(wage), p50 = ~quantile(wage, probs = 0.5))
```

```
## # A tibble: 9 x 5
##    gender    sector     n salaire_moyen   p50
## * <fctr>    <fctr> <int>         <dbl> <dbl>
## 1 Totale    Totale  1000      4984.800  4700
## 2 Totale secondary   455      4566.154  4400
## 3 Totale  tertiary   545      5334.312  5000
## 4    men    Totale   547      5435.466  5200
## 5    men secondary   289      5070.588  5000
## 6    men  tertiary   258      5844.186  5450
## 7  women    Totale   453      4440.618  4000
```

# Approche générale avec `distrr`

- Le point de départ est toujours:
  - Une source de données dont on dispose des données individuelles (ESS, STATENT, ESPA, RS, ...) (argument `.data`)
  - Des variables catégorielles qui définissent les croisements qui nous intéressent (argument `.variables`)
- Deux possibilités pour créer un cube:
  - avec la fonction `dcc5`
  - avec la fonction `dcc6`

# Avec la fonction dcc5

- Données individuelles (`.data`)
- Variables pour les "croisements" (`.variables`)
- Une séquence d'appels à des fonctions (vectorielles) qui retournent une valeur (par exemple: `sum`, `mean`, `weighted.mean`, ou une fonction écrite par nous)

```
library(distrr)
dcc5(.data = invented_wages,
     .variables = c("gender", "sector"),
     salaire_moyen = ~weighted.mean(wage, sample_weights),  # Appel 1
     p50 = ~wq(wage, sample_weights)                        # Appel 2
     )
```

- (résultat à la page suivante...)

# Résultat avec dcc5

```
## # A tibble: 9 x 5
##   gender    sector     n salaire_moyen   p50
## * <fctr>    <fctr> <int>         <dbl> <dbl>
## 1 Totale    Totale  1000      4645.323  4300
## 2 Totale secondary   455      4444.305  4100
## 3 Totale  tertiary   545      4871.707  4400
## 4    men    Totale   547      5323.053  5100
## 5    men secondary   289      5334.285  5300
## 6    men  tertiary   258      5312.575  4800
## 7  women    Totale   453      3613.789  3400
## 8  women secondary   166      3356.970  3100
## 9  women  tertiary   287      4001.054  3800
```

# Avec la fonction dcc6

- Données individuelles (`.data`)
- Variables pour les "croisements" (`.variables`)
- Une liste d'appel à une ou plusieurs fonctions (vectorielles) (`sum`, `mean`, ...)

```
library(distrr)
# On crée une liste d'appel à deux fonctions
liste_fonctions <- list(
  salaire_moyen = ~weighted.mean(wage, sample_weights),
  p50 = ~wq(wage, sample_weights)
)

# Et on l'utilise comme argument .funs_list dans la fonction dcc6
dcc6(.data = invented_wages,
     .variables = c("gender", "sector"),
     .funs_list = liste_fonctions
     )
```

# Résultat avec dcc6

```
## # A tibble: 9 x 4
##   gender    sector salaire_moyen   p50
## * <fctr>    <fctr>         <dbl> <dbl>
## 1 Totale    Totale      4645.323  4300
## 2 Totale secondary      4444.305  4100
## 3 Totale  tertiary      4871.707  4400
## 4    men    Totale      5323.053  5100
## 5    men secondary      5334.285  5300
## 6    men  tertiary      5312.575  4800
## 7  women    Totale      3613.789  3400
## 8  women secondary      3356.970  3100
## 9  women  tertiary      4001.054  3800
```

# Conclusions / Utilisations / Développements

# Conclusions / Utilisations / Développements

- Nous utilisons `distrr` à l'USTAT pour répondre aux requêtes de nos utilisateurs (environ depuis deux ans)
- Les cubes créés avec `distrr` peuvent être transformés en format "PX-win" (fichiers .px), qui est le format utilisé par STAT-TAB de l'OFS (ou dans d'autres formats, un exemple simple est un feuil de calcul avec des filtres)
- On pourrait construire une interface graphique pour la création de cubes de données (par exemple avec `shiny`)
- Les cubes de données peuvent être facilement exploités pour des graphiques
- Les cubes de données sont aussi un outil de diffusion interne à l'office
- ...

Grazie per l'attenzione
Merci de votre attention
Herzlichen Dank für die Aufmerksamkeit
https://CRAN.R-project.org/package=distrr
www.ti.ch/ustat

# Annexe: exemples avec des sources de statistique publique

# Enquête suisse sur la structure des salaires (ESS, 2014) (1/7)

```r
library(distrr)
load("rss14.rda")
# Variables pour les "croisements" (argument .variables)
vars <- c("GR", "GESCHLE")

# Liste d'appel à des fonctions (argument .funs_list)
liste_fonctions <- list(
  n = ~n(),                                  # nombre d'observations
  n_entr = ~length(unique(ENTID_N))          # nombre d'entreprises
  p25 = ~wq(MBLS, GEWIBGRS, probs = 0.25),   # 25ème percentile
  p50 = ~wq(MBLS, GEWIBGRS),                 # médiane des salaires
  p75 = ~wq(MBLS, GEWIBGRS, probs = 0.75)    # 75ème percentile
)

# Et on l'utilise comme argument .funs_list dans la fonction dcc6
# (page suivante)
```

# ESS 2014 (2/7)

```
c01_rss <- dcc6(.data = rss14,
                .variables = vars,
                .funs_list = liste_fonctions
                )
```

```
# On peut vérifier les statistiques publiables et 'masquer' celles
# qui ne sont pas publiables. Les résultats de l'ESS sont publiables
# s'ils reposent aux moins de 60 personnes salariées et 5 entreprises
k <- with(c01_rss, n < 60 | n_entr < 5)
table(k, useNA = "always")
```

```
## k
## FALSE  <NA>
##    24     0
```

# ESS 2014 (3/7)

- Dans ce cas il n'y a pas de résultats non publiables. Il suffit d'enlever les colonnes du nombre d'observations (`n`) et du nombre d'entreprises (`n_entr`):

```
c01_rss <- c01_rss %>% select(-n, -n_entr)
str(c01_rss)
```

```
## Classes 'tbl_df', 'tbl' and 'data.frame':    24 obs. of  5 variables:
##  $ GR     : Factor w/ 8 levels "Total","1","2",..: 1 1 1 2 2 2 3 3 3 4
##  $ GESCHLE: Factor w/ 3 levels "Total","1","2": 1 2 3 1 2 3 1 2 3 1 ...
##  $ p25    : int  5101 5444 4643 5052 5291 4716 5208 5539 4733 5202 ...
##  $ p50    : int  6427 6751 5907 6498 6682 6189 6358 6677 5879 6578 ...
##  $ p75    : int  8383 9000 7568 8599 9065 8061 8089 8656 7295 8588 ...
##  - attr(*, ".variables")= chr  "GR" "GESCHLE"
```

# ESS 2014 (4/7)

- Et changer les codes des variables avec les “étiquettes” correspondantes:

```
levels(c01_rss$GR) <- c("Total Suisse", "Région lémanique",
                        "Espace Mittelland", "Suisse du Nord-Ouest",
                        "Zurich", "Suisse orientale", "Suisse centrale",
                        "Tessin")
levels(c01_rss$GESCHLE) <- c("Total", "Hommes", "Femmes")
```

- Pour obtenir finalement (voir page suivante)

# ESS 2014 (5/7)

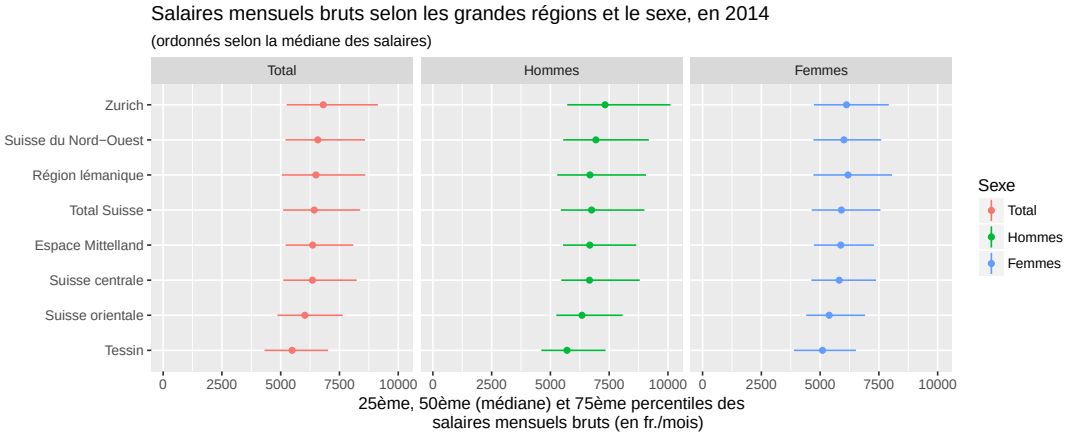
```
c01_rss
```

```
## # A tibble: 24 x 5
##                       GR GESCHLE   p25   p50   p75
##  *                 <fctr>  <fctr> <int> <int> <int>
##  1         Total Suisse   Total  5101  6427  8383
##  2         Total Suisse  Hommes  5444  6751  9000
##  3         Total Suisse  Femmes  4643  5907  7568
##  4     Région lémanique   Total  5052  6498  8599
##  5     Région lémanique  Hommes  5291  6682  9065
##  6     Région lémanique  Femmes  4716  6189  8061
##  7    Espace Mittelland   Total  5208  6358  8089
##  8    Espace Mittelland  Hommes  5539  6677  8656
##  9    Espace Mittelland  Femmes  4733  5879  7295
## 10 Suisse du Nord-Ouest   Total  5202  6578  8588
## # ... with 14 more rows
```

# Exemple de graphique avec le cube de l'ESS 2014 (6/7)

```
library(ggplot2)
p01_ess <- ggplot(c01_rss,
                  aes(x = reorder(GR, p50), y = p50, colour = GESCHLE))
p01_ess + geom_point() +
  geom_linerange(aes(ymin = p25, ymax = p75)) +
  expand_limits(y = 0) +
  coord_flip() +
  facet_wrap(~GESCHLE) +
  ggtitle("Salaires mensuels bruts selon les grandes régions et le sexe,
          en 2014", subtitle = "(ordonnés selon la médiane des
          salaires)") +
  labs(colour = "Sexe", y = "25ème, 50ème (médiane) et 75ème percentiles
       des salaires mensuels bruts (en fr./mois)", x = "")
```

# Et le résultat (7/7)



# Statistique structurelle des entreprises (STATENT, 2015p) (1/2)

```r
library(distrr)
load("statent15_p.rda")
# Variables pour les "croisements" (argument .variables)
vars <- c("CANTON_CD", "NOGA08_SECTION")

liste_fonctions <- list(
  établissements = ~n(),
  emplois = ~sum(EMPTOT),
  ept = ~sum(EMPFTE)
)

c01_statent <- dcc6(.data = statent15_p,
                    .variables = vars,
                    .funs_list = liste_fonctions,
                    .total = "Total"
                    )
```

# STATENT 2015 (données provisoires) (2/2)

```
c01_statent
```

```
## # A tibble: 540 x 5
##    CANTON_CD NOGA08_SECTION établissements emplois         ept
##  *    <fctr>         <fctr>          <int>   <int>       <dbl>
##  1     Total          Total         675506 5078915 3999207.457
##  2     Total              A          55843  164752  107205.864
##  3     Total              B            370    4979    4555.572
##  4     Total              C          43107  680125  623599.972
##  5     Total              D           1092   28862   25812.344
##  6     Total              E           1989   18469   16229.656
##  7     Total              F          49736  353695  325576.053
##  8     Total              G          95763  629176  510858.301
##  9     Total              H          18867  239364  198525.716
## 10     Total              I          32970  252085  190372.952
## # ... with 530 more rows
```

# Enquête suisse sur la population active (ESPA, données trimestrielles 2010-2016) (1/4)

```r
library(distrr)

# On crée une variable "statut_activité"
# (Occupés résidents, Chômeurs au sens du BIT, Personnes inactives)
espa10_16$statut_activite <- cut(espa10_16$TBD1, breaks = c(1, 5, 6, Inf),
                                 include.lowest = TRUE, right = FALSE)
levels(espa10_16$statut_activite) <- c("Occupés résidents",
                                       "Chômeurs au sens du BIT",
                                       "Personnes inactives")

# Variables pour les "croisements" (argument .variables)
vars <- c("B022", "B024", "B023", "statut_activite")
```

# ESPA, données trimestrielles 2010-2016 (2/4)

```r
# Liste d'appel à des fonctions (argument .funs_list)
liste_fonctions <- list(
  n = ~n(),  # numérosité, pour vérifier la publiabilité
  Nhat = ~sum(IXPXH)
)

# Et création du cube de données
c01_espa <- dcc6(.data = espa10_16,
                 .variables = vars,
                 .funs_list = liste_fonctions,
                 .total = "Total"
                 )
```

# ESPA, données trimestrielles 2010-2016 (3/4)

```
# On enlève les estimations avec toutes les années ensemble (B022 == "Tota
# et tous les trimestres ensemble (B024 == "Total")
c01_espa <- c01_espa %>%
  filter(B022 != "Total") %>%
  filter(B024 != "Total")

# Et on enlève les niveaux des variables catégorielles pour lesquels
# il n'y a pas de résultats
c01_espa[ , 1:2] <- lapply(c01_espa[ , 1:2], droplevels)
```

# ESPA, données trimestrielles 2010-2016 (4/4)

```
c01_espa
```

```
## # A tibble: 896 x 6
##      B022   B024   B023         statut_activite     n       Nhat
##    <fctr> <fctr> <fctr>                  <fctr> <int>      <dbl>
##  1   2010      1  Total                   Total 32759 6579352.21
##  2   2010      1  Total       Occupés résidents 19463 4157933.51
##  3   2010      1  Total Chômeurs au sens du BIT   987  237117.55
##  4   2010      1  Total     Personnes inactives 12309 2184301.15
##  5   2010      1      1                   Total  6109 1206955.92
##  6   2010      1      1       Occupés résidents  3433  704097.25
##  7   2010      1      1 Chômeurs au sens du BIT   244   60170.99
##  8   2010      1      1     Personnes inactives  2432  442687.67
##  9   2010      1      2                   Total  6887 1475931.29
## 10   2010      1      2       Occupés résidents  4091  932750.63
## # ... with 886 more rows
```