RPU Quotidiens

JcB

10/01/2014

Contents

1 Fichier RPU quotdien

Depuis février 2014, Alsace e-sante transmet quotidiennement un fichier contenant les RPU des 7 derniers jours (j-7 à j-1). Les données correspondant à J-7 sont considérées comme consolidées. Elles peuvent être extraites et stockées. Les données sont transmises de manière habituelle, c'est à dire un fichier .sql qu'il faut transcoder en R pour le nettoyer avant stockage.

Au mois de mai 2014 la clinique des 3 frontières (C3F) a changé de N°FINESS. (voir le paragraphe C3F)

1.0.0.1 Méthode rapide: voir En Pratique

1.1 Descriptif

1. Le fichier des données est récupéré sur le serveur de test des HUS. Il st déposé dans le dossier de stockage (**/home/jcb/Documents/Resural/Stat Resural/Archives_Sagec/dataQ**) et dézippé.
2. le nom du fichier est construit de la manière suivante:

- date.jour <- "2014-02-21"
- file <- paste0("rpu_", date.jour, "_dump.sql")
- *date.jour est du type AAAA-MM-JJ*

2. le fichier est ensuite transféré dans la base de données **archives** dans la table **RPU___** via R

- il est important que le répertoire de travail temporaire soit positionné dans le dossier *dataQ*

```
wd <- getwd()
setwd("~/Documents/Resural/Stat Resural/Archives_Sagec/dataQ")
system(paste0("mysql -u root -pmarion archives < ", file))
setwd(wd)
```

3. Lecture des données dans R

```
library("RMySQL")
con<-dbConnect(MySQL(),group = "archives")
rs<-dbSendQuery(con,paste("SELECT * FROM RPU__ ",sep=""))
dx<-fetch(rs,n=-1,encoding = "UTF-8")
max(dx$ENTREE)
min(dx$ENTREE)
```

4. nettoyage des données

- suppression de la colonne 16: dx<-dx[,-16]
- transcodage des FINESS (vérification nombre hôpitaux)
- transformation en facteurs
- création d'une colonne AGE (alertes age < 0 et age > 120)

5. sauvegarde des données

- jour à sauvegarder: jour <- as.Date(min(dx$ENTREE))
- dday <- dx[as.Date(dx$ENTREE) == jour,]
- fichier du jour: write.table(dday, paste0(date.jour,".csv"), sep=',', quote=TRUE, na="NA", row.names=FALSE,col.names=TRUE)
- fichier général: write.table(dday, "RPU2014.csv", sep=',', quote=TRUE, na="NA", append = TRUE, row.names=FALSE,col.names=TRUE)

6. fonctions helpers

source("quot_utils.R") ou source("Preparation/RPU Quotidiens/quot_utils.R") en mode console.

- **rpu_jour**: fonction principale. En entrée on donne la date ISO souhaitée et en sortie retourne un dataframe avec les données correspondantes. Le WD doit pointer sur le dossier contenant le fichier .sql correspondant. Ce fichier doit être dézippé.

- **finess2hop**: transforme le code FINESS en nom court d'hopital
- **parse_rpu**:

séquence:

- date.jour <- "2014_02"
- dx <- parse_rpu(date.jour)
- dx$FINESS <- $-as.factor(finess2hop(dx$FINESS))$
- summary(dx$FINESS) fait un décompte des RPU par établissement sur la période => permet de vérifier si anomalies quantitatives. Suppose de disposer d'un historique moyenne, écart-type par type de jour.
- dx <- rpu2factor(dx)

```
#' Méthode générale
#' Préalable: disposer d'une base de donnée MySql avec une table appelée "archives". Cette base doit êt
#'@ data date.jour nom du fichier. Pour une utilisation courante il s'agit de la date du jour au format
parse_rpu <- function(date.jour){
  library("RMySQL")
  file <- paste0("rpu_", date.jour, "_dump.sql")
  wd <- getwd()
  setwd("~/Documents/Resural/Stat Resural/Archives_Sagec/dataQ")
  system(paste0("mysql -u root -pmarion archives < ", file))
  con<-dbConnect(MySQL(),group = "archives")
  rs<-dbSendQuery(con,paste("SELECT * FROM RPU__ ",sep=""))
  dx<-fetch(rs,n=-1,encoding = "UTF-8")
  dx<-dx[,-16]
  dx$FINESS <- as.factor(finess2hop(dx$FINESS))

  dx$AGE<-floor(as.numeric(as.Date(dx$ENTREE)-as.Date(dx$NAISSANCE))/365)

  dx$EXTRACT <- as.Date(dx$EXTRACT)
  setwd(wd)
}

#' Transformation du code Finess et nom court d'hôpital
finess2hop <- function(a){
  # a<-dx$FINESS
  a[a=="670000397"]<-"Sel"
  a[a=="670017755"]<-"Sel" # GHSO depuis le 5/1/2016
  a[a=="680000684"]<-"Col"
  a[a=="670016237"]<-"Odi" # Finess géo. utilisé pour les RPU depuis le 1/1/2016
  a[a=="670780204"]<-"Odi" # Finess juridique
  a[a=="670000272"]<-"Wis"
  a[a=="680000700"]<-"Geb"
  a[a=="670780055"]<-"Hus"
  a[a=="670000025"]<-"Hus" # NHC
  a[a=="670783273"]<-"Hus" # HTP
  a[a=="680000197"]<-"3Fr"
  a[a=="680000627"]<-"Mul"
  a[a=="670000157"]<-"Hag"
  a[a=="680000320"]<-"Dia"
  a[a=="680000395"]<-"Alk"
  a[a=="670000165"]<-"Sav"
```

```
  a[a=="680000494"]<-"Ros"
  a[a=="670780162"]<-"Dts"
  a[a=="670780212"]<-"Ane"
  a[a=="680000601"]<-"Tan"
  a[a=="670009109"]<-"Ccm" # CCOM Ilkirch 2015-04-23
  a[a=="680000627"]<-"Her" # Hasenrain  2015-04-23
  return(a)
}
```

1.1.0.2 controles quotidiens

- nlevels(dx$FINESS) si différent de 14 => problème
- nb moyen et ecart-type de RPU par établissement et par jour

```
date1 <- "2014-03-01"
date2 <- "2014-03-05"
p <- seq(as.Date(date1), as.Date(date2), 1)
for(i in 1:length(p)){
  x <- parse_rpu(p[i])
  table(x$FINESS, as.Date(x$ENTREE))
}
```

1.2 Commentaires:

```
r <- table(as.Date(a$ENTREE), a$FINESS)
r <- r[,-13] # supprime la colonne 13 qui est totalement vide ?
r
```

- altkirch: toujours des trous inexpliqués: 1/1, 5/1, 11 et 12/1, 16/1, 18/1, 2/3
- mulhouse: 15/1, 7/2, 5-6-7/3 zéro rpu
- ste odile: 16 au 31/1 pas de rpu
- sélestat: 22 et 23/2 pas de RPU
- diaconat strasbourg: 1-2-3-4/3 puis plus rien
- roosvelt: depuis le 5/2 OK

2 En pratique

- dézipper le fichier du jour dans */home/jcb/Documents/Resural/Stat Resural/Archives_Sagec/dataQ*
- charger le fichier **quot_utils.R** pour disposer des routines
- répéter l'étape **rj** autant de fois qu'il y a de fichiers à analyser
- assembler les fichiers avec **assemble()**

```
source("Preparation/RPU Quotidiens/quot_utils.R")
```

2.1 si un seul fichier

NB: le fichier doit se trouver dans la dossier **dataQ**.

```
rj <- rpu_jour("2014-11-18")
dj <- assemble(comment = TRUE)

# Exhaustivité des RPU du jour:
table(as.Date(rj$ENTREE), rj$FINESS)
```

2.2 Si un seul fichier de rattrapage

NB: le fichier doit se trouver dans la dossier **dataQ**. La méthode **parse_rpu** complète le nom du fichier avec le préfixe "rpu_" et le suffixe "_dump.sql". par exemple pour le fichier"rpu_01_15_dump.sql", on utilisera rj <- parse_rpu("01_15")

```
wd <- getwd()
rj <- parse_rpu("2015-0607")
rpu.par.jour(rj)
# les RPU sont bruts sauf le FINESS qui est transcodé. Pour les rendre compatibles, il faut appeler rpu
rj <- rpu2factor(rj)
setwd(wd)
```

2.3 Si on a une collection de fichiers:

2.3.0.3 NB: SUPPRIMER LE FICHIER /home/jcb/Documents/Resural/Stat Resural/Archives_Sagec/dat

Il faut également, à mois échu, créer un dossier pour le mois dans le dossier **archivesCSV** et y ranger les fichiers *.csv* du mois échu (sinon ils sont repris dans les calcule).

```
# Prépare la zone de travail
# - effacement des variables
# - effacement des fichiers csv temporaires
rm(list=ls())
source("Preparation/RPU Quotidiens/quot_utils.R")
file.to.delete <- "/home/jcb/Documents/Resural/Stat Resural/Archives_Sagec/dataQ/archivesCsv/rpu2014.da
file.remove(file.to.delete)

date1 <- "2016-01-08"
date2 <- "2016-01-25"

mc <- substr(date1, 6, 7)
ac <- substr(date1, 1,4)

p <- seq(as.Date(date1), as.Date(date2), 1)
for(i in 1:length(p)){
  dx <- rpu_jour(p[i])
}
dx <- assemble(comment = TRUE) # assemble utilise les fichiers .csv présentas dans archivesCSV

min(as.Date(dx$ENTREE))
max(as.Date(dx$ENTREE))
```

2.3.0.4 Sélectionner une période particulière (ie. mai 2014)

num.mc <- 1 # numéro du mois courant mois.c <- paste("d",)

```
d01 <- dx[as.Date(dx$ENTREE) >= "2016-01-01" & as.Date(dx$ENTREE) < "2016-02-01",]
min(as.Date(d01$ENTREE))
max(as.Date(d01$ENTREE))

d01 <- normalise(d01)
save(d01, file="rpu2016d01_provisoire.Rda")
rm(dx)

# si le mois est complet:
# save(d11, file="rpu2015d11.Rda")
# rm(rpu2015d11_provisoire.Rda)

# uniquement le premier mois
# d16 <- d01
# save(d16, file="rpu2016d0112_provisoire.Rda")
# save(d16, file="rpu2016d0112.Rda")

#load("RPU_2014/rpu2016d0112_provisoire.Rda")
load("rpu2016d0112_provisoire.Rda") # si console
d16 <- d16[as.Date(d16$ENTREE) < "2015-01-01",]

# dx <- rbind(d01, d02, d03, d04, d05, d06, d07, d08)
dx <- rbind(d16, d01)

min(as.Date(dx$ENTREE))
max(as.Date(dx$ENTREE))

d16 <- dx

# Uniquement le CCOM
ccm2016 <- d16[d16$FINESS == "Ccm",]
save(ccm2016, file = "ccm2016.Rda")

# TOUS les rpu SAUF le CCOM
d16 <- d16[d16$FINESS != "Ccm",]
save(d16, file="rpu2016d0112_provisoire.Rda")

# Table des RPU par jour et par FINESS pour le mois en cours
# n <- tapply(as.Date(dsi$ENTREE), list(as.Date(dsi$ENTREE), dsi$FINESS), length)
rpu.par.jour(d01)
```

Total RPU par mois et par FINESS

```
t <- tapply(as.Date(d16$ENTREE), list(d16$FINESS, months(as.Date(d16$ENTREE))), length)
t <- t[, c("janvier","février","mars", "avril","mai","juin","juillet", "août", "septembre", "octobre",
# tableau mensuel brut
t

# Total brut par mois
apply(t, 2, sum, na.rm = TRUE)

# Sauvegarde
write.csv2(t, file = "rpu_2016_par_mois.xls")
```

3 Mois courant: projection du nb de RPU à mois constant de 30 jours. ATTENTION: modifier si nécessaire le n° du mois

n.jours <- as.numeric(max(as.Date(d01$ENTREE$)) − $min(as.Date($d01ENTREE)) + 1) n.rpu.mois.constant <- nrow(d01) * 30 / n.jours n.rpu.mois.constant

Total par mois et par mois constants

```
# Total par mois (non corrigé). Il vaut mieux utiliser la fonction 'month' de Lubridate qui conserve l'
# des mois plutôt que 'months' qui retourne les résultats en ordre dispersé. Par contre le nom des mois
t <- tapply(as.Date(d16$ENTREE), month(as.Date(d16$ENTREE), label = TRUE), length)
names(t) <- c("janvier","février","mars", "avril","mai","juin","juillet", "août", "septembre","octobre"

t

# nb de jours dans le mois (la séquence doit inclure le mois suivant. SOURCE: https://stat.ethz.ch/pipe
n.j <- as.integer(diff(seq(as.Date("2016-01-01"), as.Date("2016-12-01"), by = "month")))

# nb de RPU par mois constant de 30 jours. On supprime les valeurs de t égales à NA, correspondant aux
# par exemple décembre si la série s'arrête à novembre
t.corrige <- t[1:length(t[!is.na(t)])] * 30 / n.j

barplot(t.corrige, main = "Nombre de RPU par mois standards de 30 jours", col = "cornflowerblue")

barplot(t2, beside = TRUE, cex.names = 0.7)
```

Nombre de jour en 2016

```
anc <- "2016"
d1 <- as.Date(paste0(anc, "-01-01"))
d2 <- as.Date(paste0(as.integer(anc) + 1, "-01-01"))
n.days <- as.integer(d2-d1)
```

n.days

3.0.0.5 Nombre de RPU attendus pour l'année: { rpu_extrapoles} n <- nrow(d16) td <- max(as.Date(d16$ENTREE)) - min(as.Date(d16$ENTREE)) n * n.days / as.numeric(td)

3.0.0.6 Nb actuel de RPU par FINESS

```
t <- tapply(as.Date(dx$ENTREE), dx$FINESS, length)
t

# nb prévisionnel par FINESS
td <- max(as.Date(dx$ENTREE)) - min(as.Date(dx$ENTREE))
round(t * 365 / as.numeric(td), 0)
```

3.0.1 bilan des RPU à la date du 31/7/2015

```
# table des RPU depuis le début année transformé en dataframe
# la colonne HUS est vide depuis le 1/1/2015 (à supprimer) remplacée par NHC et HTP
```

```
# la colonne MUL est vide à partir du 1/9/2015 et remplacée par EMR (11/8/2015) et HSR (1/9/2015)
rpu.jour <- as.data.frame.matrix(rpu.par.jour(d16))


# sauvegarde de la matrice
# write.csv(rpu.jour, file = "rpu.jour_31-01-2016.csv")
write.csv(rpu.jour, file = paste0("rpu_jour_", anc, ".csv"))

# fonction qui compte le nombre de rpu = 0 dans le vecteur x
rpu.manquant <- function(x){length(which(x == 0))}

# Nombre de jours manquants: applique la fonction à la matrice rpu.jour
# ne plus tenir comte de Hus. Pour Mul, Hsr et Emr tenir compte de la remarque précédente.
apply(rpu.jour, 2, rpu.manquant)

3Fr Alk Ane Col Dia Dts Geb Hag Hus Mul Odi Ros Sav Sel Wis HTP NHC Emr Hsr
  0   0   1   1   0   0   0   0 264  25   1   0   0  13   0   0   0 222 243

moins de 20 rpu/jour
rpu.manquant <- function(x){length(which(x < 20))}
3Fr Alk Ane Col Dia Dts Geb Hag Hus Mul Odi Ros Sav Sel Wis HTP NHC
  0   0  46   1   0   7   0   0  61   5   5  80   0  26   2 151 151

# graphe

load("~/Documents/Stat Resural/RPU_2014/rpu2016d0112_provisoire.Rda")

rpu <- read.csv("rpu.jour.csv")
names(rpu)[1] <- "date"
names(rpu)[2] <- "3Fr"

library(xts)
xts <- xts(rpu, order.by = as.Date(rpu$date), frequency = TRUE)

# exemple de Sélestat:
a <- as.integer(xts$Sel)
s <- summary(a)
sd <- sd(a)

plot(xts$Sel, ylim = c(0, s["Max."]), las = 2, cex.axis = 0.6, main = paste0("Nombre de RPU par jour en

# Two new arguments here are the major.ticks and minor.ticks settings.
# The major.ticks argument represents the periods in which we wish to chop up the horizontal axis; it is

abline(h = s["Mean"], col = "red")
abline(h = s["Mean"]-sd, col = "red", lty = 2)
abline(h = s["Mean"]-sd*2, col = "red", lty = 2)

# Moyenne et sd corrigés: on recalcule les paramètres en ne tenant pas compte des jours aberrants (plus
# On retire de a les jours où le nb de RPU est plus petit que le nombre moyen de RPU - 2 sd:
b <- a[a > s["Mean"]-sd*2]

# on recalcule les paramètres de centralité et de dispersion:
s.b <- summary(b)
```

```
sd.b <- sd(b)

# on dessine les nouvelles limites:
abline(h = s.b["Mean"], col = "red")
abline(h = summary(b)["Mean"] - sd(b), col = "blue", lty = 2)
abline(h = summary(b)["Mean"] - sd(b) * 2, col = "blue", lty = 2)
abline(h = summary(b)["Mean"] - sd(b) * 3, col = "blue", lty = 2)
```

3.0.2 Recherche de doublons: CODE_POSTAL, COMMUNE, ENTREE, FINESS, NAISSANCE, SEXE

“‘{ doublons} a <- duplicated(d16[, c(2,3,6,8,13,16)]) sum(a) which(a) # quelle ligne d16[which(a), c(2,3,6,8,13,16)]
“‘

3.0.2.1 Eventuellement sauvegarder au format .Rda

```
#dx <- normalise(dx)
#dx$FINESS <- factor(dx$FINESS) # supprime les facteurs vides
#dx <- dx[dx$ENTREE >= "2014-04-01" & dx$ENTREE < "2014-05-01",]
#save(dx, file="rpu2014d04_provisoire.Rda")
```

et assembler le tout (d1 = fichier .Rda des mois précédents)

```
#a <- rbind(d1,dx)
#save(a, file="rpu2014d0103_provisoire.Rda")
```

Pour fabriquer les courbes interactives d'activité, voir le projet **dygraph**.

3.1 exhaustivité des données

On forme une table en croisant FINESS et ENTREE:

```
dx$FINESS <- factor(dx$FINESS) # supprime les facteurs vide
rpu <- table(as.Date(dx$ENTREE), dx$FINESS) # tous les RPU de l'année par FINESS
write.csv(rpu, file="exaustivite_rpu.csv") # enregistre la table au format .csv mais à la différence de
```

3.1.0.2 Préparation des fichiers pour Dygraph -> voir *Préparation des fichiers pour Dygraph.Rmd*

-> explorer cart.js qui fait des diagrammes interactifs en étoile.

4 Résumé activité

On crée un tableau Finess x Jour de l'année permettant de voir rapidement où sont les “trous”. A partir du tableau **rpu** on crée un dataframe **a** comportant deux colonnes: la date du jour et le nombre de RPU correspondants pour l'ensemble des SU d'Alsace. Ce dataframe peut être utilisé pour **Dygraph**.

On peut aussi l'utiliser pour tracer le graphe correspondant.

A partir de la table **rpu** crée précédamment, on ajoute une colonne avec la somme de la ligne, ce qui correspond au nombre de RPU créés quotidiennement. Puis on transforme le tableau *rpu* en dataframe appelé **a** où ne sont conservée que deux colonnes, la date du jour et le nombre de RPU. Le dataframe *a* est transformé en objet *xts* appelé **x**, ce qui permet de l'afficher sous forme de graphe d'une série temporelle.

“‘{ activite}

s <- rowSums(rpu[, 2:ncol(rpu)]) b <- rownames(rpu) a <- as.data.frame(cbind(b,s)) m <- rowMeans(rpu[, 2:ncol(rpu)])

colnames(a) <- c(“Date”,“RPU”) a$Date < -as.Date(a$Date) a$RPU < -as.numeric(as.character(a$RPU)) # transforme les facteurs en nombre

library(“xts”) library(“lubridate”)

plot(a$Date, a$RPU, type=“l”, ylab=“nombre de RPU”, xlab=paste0(“Année”, year(Sys.Date())), main=“Activité des SU d’Alsace en nombre de RPU”)

5 Tracé de la courbe des RPU avec PLOT.XTS

x <- as.xts(aRPU, aDate) plot(x , major.ticks= “weeks”, las = 2, minor.ticks = FALSE, major.format = “%d %b”, cex.axis = 0.8, main = “RPU 2015”, ylab = “Nombre de RPU par jour”, col = “cornflowerblue”) # {“years”, “months”, “weeks”, “days”, “hours”, “minutes”, “seconds”} lines(rollmean(x, 7), col=“red”, lwd = 3)

6 Tracé de la courbe des RPU avec PLOT.ZOO

z <- as.zoo(x) plot(z, col=“cornflowerblue”, ylab=“nombre de RPU”, xlab=paste0(“Année”, year(Sys.Date())), main=“Activité des SU d’Alsace en nombre de RPU”) lines(rollmean(z, 7), col=“red”, lwd = 3)

x <- as.xts(aRPU, aDate)

z <- as.zoo(x) plot(z, col=“cornflowerblue”, ylab=“nombre de RPU”, xlab=paste0(“Année”, year(Sys.Date())), main=“Activité des SU d’Alsace en nombre de RPU”) lines(rollmean(z, 7), col=“red”) “‘ Activité 2013-2014 ==========

Voir **Activités_2013-2014.Rmd**

7 Activités quotidienne

Objet: mesure de l’activité au jour le jour avant consolidation. En pratique revient à analyser le fichier du jour. Ce dernier contient les RPU de la veille et ceux des 7 derniers jours.

1. récupérer le fichier source, le décompacter.
2. appeler la fonction **parse_rpu** avec la date du jour, qui le transforme en dataframe “‘{}

d <- parse_rpu(“2016-01-25”)

min(as.Date(d$ENTREE))max(as.Date(d$ENTREE)) 3. puis la méthode __analyse_rpu__. Normalement on doit obtenir __16 valeurs__: analyse_rpu_jour(d) 4. on peut obtenir une matrice établissement/nb rpu par date avec la formule suivante: t <- tapply(as.Date(d$ENTREE), list(d$FINESS as.Date(d$ENTREE)), length) t(t)

```
3Fr Alk Col Dia Dts Geb Hag Hus Mul Odi Ros Sav Sel Wis
```

2015-01-01 48 51 190 59 29 52 129 306 220 15 9 83 85 28 2015-01-02 45 52 210 102 27 43 115 292 200 10 25 94 81 30 2015-01-03 31 42 203 85 30 64 125 323 NA NA 12 106 103 42 2015-01-04 43 39 175 79 21 48 102 291 186 2 12 81 76 31 2015-01-05 45 50 183 74 29 34 155 277 196 72 2 99 71 34 2015-01-06 37 37 153 82 31 48 131 283 176 61 12 84 76 22 2015-01-07 42 41 NA 66 35 39 125 296 156 64 8 69 84 39 “‘

On peut fabriquer un fichier provisoire constitué par la fusion des jours consolidés (d01) et des 7 dernier jours (d). Il faut supprimer dans le fichier d le premier jour qui correspond au dernier jour consolidé.

```
d <- d[as.Date(d$ENTREE) > max(as.Date(d16$ENTREE)),]
d <- rpu2factor(d) # transforme les chiffres en facteurs cohérent avec d01

d16.p <- rbind(d16, d)
max(as.Date(d16.p$ENTREE))

rpu.par.jour(d16.p)
rpu.jour.nc <- as.data.frame.matrix(rpu.par.jour(d16.p)) # nc = non consolidé

# sauvegarde
save(d16.p, file = "d16_p.Rda")
write.csv(rpu.jour.nc, file = paste0("rpu_jour_nc_", anc, ".csv"))
```

et l'afficher sous forme de série temporelle appelée **xts.a** avec 2 colonnes (date du jour + nb de RPU).

```
# Moyenne et SD 2014
mean2014 <- 1141.734
sd2014 <- 154.7858

# Moyenne et SD 2015
mean2015 <- 1383.074
sd2015 <- 113.3575

plot_rpu_quot(d16$ENTREE, mean2015, sd2015) # version Xts classique
# version uptodate:
plot_rpu_quot(d16.p$ENTREE, mean2015, sd2015)

# Archivage SVG
svg("activite SU alsace 2016.svg")
plot_rpu_quot(d16$ENTREE, mean2015, sd2015)
dev.off()
```

8 Comparaison 2016-2015

8.1 Nombre quotidien RPU

Techniques pour créer un axe des abcisse temporel:

- http://stackoverflow.com/questions/15575625/how-to-replace-numbers-on-x-axis-by-dates-when-using-plot-in-r
- dossier Statistiques/Electric

min <- min(as.Date(d16.p$ENTREE$))max < −max(as.Date(d16.pENTREE)) cal <- seq(min, max, 1) # calendrier pour l'axe des x

9 activité par jour en 2016

rpu.jour.2016 <- tapply(as.Date(d16.p$ENTREE$), $as.Date(d16.p$ENTREE), length)

10 activité par jour en 2015 (à la même date)

d15.p <- d15[as.Date(d15$ENTREE) <= max - 365,] rpu.jour.2015 <- tapply(as.Date(d15.p$ENTREE$), $yday(as.Date(d15.p$EN
length)

11 graphe

plot(cal, rpu.jour.2016, type = “l”, col = “blue”, lwd = 3, main = “Activité 2015 - 2016 en nombre de RPU”, ylab = “nombre de RPU”, xlab = “Jours”, xaxt = “n”) # axis.Date(1, at = seq(min, max, by = “day”), format = “%d/%m”, las = 2, cex.axis = 0.8)

12 pour aligner la courbe 2015 on utilise le mˆeme vecteur pour les abcisses (cal = dates 2016) et pour les valeurs d’ ordonnées, le nombre de RPU produits en 2015 par jour de l’année exprimés en n° du jour de l’année (1er janvier = 1, etc.). Cette astuce permet de superposer plusieurs années.

lines(cal, rpu.jour.2015, type = “l”, col = “green”, lwd = 1)

legend(“bottomleft”, legend = c(“2016”,“2015”), col = c(“blue”,“green”), bty = “n”, lty = 1, lwd = c(3,1))

13 moyenne année précédente

mean.rpu.2015 <- 1383.074 abline(h = mean.rpu.2015, lty = 2, col = “red”) text(min + 1, 1400, “moyenne 2015”, cex = 0.8, col = “red”)

copyright()

13.1 Comparaisons hebdomadaires

rpu.tot <- c(d15$ENTREE$, $d16ENTREE) # matrice 52 x 2 rpu.semaine <- tapply(as.Date(rpu.tot), list(year(as.Date(rpu.tot)), week(as.Date(rpu.tot))), length) rpu.semaine[, 1:4] barplot(rpu.semaine, main = “RPU hebdomadaires 2015-2016”, beside = TRUE, xlab=“semaines”, ylab=“nombre de RPU”, col = c(“green”, “blue”))

plot(rpu.semaine[2,], type = “l”, ylim = c(7000,11000), col=“blue”, xlab=“semaines”, ylab=“nombre de RPU”, main = “RPU hebdomadaires 2015-2016”) lines(rpu.semaine[1,], col = “green”) legend(“bottomleft”, legend=c(“2015”, “2016”), col=c(“green”,“blue”), bty=“n”, lty=1)

13.2 Hospitalisation

14 RPU dont le MODE_SORTIE est renseigné + date du jour

mode_sortie_renseigne <- rbind(d15[!is.na(d15$MODE_SORTIE), c("ENTREE", "MODE_SORTIE")], d16[!is.na(d16$MODE_SORTIE), c("ENTREE", "MODE_SORTIE")])

15 hosp = mutation + transfert

hosp <- mode_sortie_renseigne[mode_sortie_renseigne$MODE_SORTIE == "Mutation"|mode_sortie_renseigne$MODE_SOR == "Transfert", 'ENTREE']

16 RPU dont le MODE_SORTIE est renseigné par semaines

mode_sortie_renseigne.sem <- tapply(as.Date(mode_sortie_renseigne$ENTREE), list(year(as.Date(mode_sortie_renseigne$ENTREE)), week(as.Date(mode_sortie_renseigne$ENTREE))), length) mode_sortie_renseigne.sem[, 1:4]

17 hospitalisé par semaine

hosp.semaine <- tapply(as.Date(hosp), list(year(as.Date(hosp)), week(as.Date(hosp))), length) hosp.semaine[, 1:4]

barplot(hosp.semaine[, 1:4], main = "Hospitalisations hebdomadaires 2015-2016", beside = TRUE, xlab="semaines", ylab="nombre de RPU")

Ratio hospitalisation:

r <- hosp.semaine[, 1:4] / mode_sortie_renseigne.sem[, 1:4] r # Astuce: en augmentant artificiellement la dimension de l'axe des y on ménage de la place pour la légende barplot(r, beside = TRUE, main = "Taux d'hospitalisation 2015-2016", col = c("green","blue"), ylim = c(0,0.3)) legend("topright", legend = c("2015", "2016"), fill = c("green", "blue"), bty = "n")

18 Focus mains

En 2015:

```
mains <- d15[d15$FINESS %in% c("Dts","Ros","Ccm"),]
nrow(mains)/nrow(d15)
nrow(mains)*100/nrow(d15)

ros <- d15[d15$FINESS == "Ros",]
nrow(ros)

dts <- d15[d15$FINESS == "Dts",]
nrow(dts)

ccm <- d15[d15$FINESS == "Ccm",]
nrow(ccm)
```

```
min(as.Date(dts$ENTREE))
max(as.Date(dts$ENTREE))

c <- tapply(as.Date(dts$ENTREE), as.Date(dts$ENTREE), length)
cbind(c)
```

19 Nombre de DP par jour

Exemple avec 2015 (étude SI 2015). TODO: RETIRER LES DP OU 'ORIENTATION' NE PERMET PAS DE CODER LE DP (SCAM,PSA,FUGUE, ETC.)

```
# dataframe date, DP, FINESS
dp <- d15[, c("ENTREE", "DP", "FINESS")]

# on se limite au 3 premiers mois de l(année)
# dp <- dp[as.Date(dp$ENTREE) < as.Date("2015-04-01"),]
# nombre de RPU
n <- nrow(dp)

# diagnostic non codés par jour: renvoie un array de n listes. n est égal au nombre de jours couverts pa
dp.non.code <- tapply(dp$DP, as.Date(dp$ENTREE), is.na)

# en nombre: on calcule la somme de chaque liste, puis on déliste. On obtient pour chaque jour, le nombr
n.dp.non.code <- unlist(lapply(dp.non.code, sum))
# nb moyen de DP non codés par jour
mean(n.dp.non.code)
sd(n.dp.non.code)

# en pourcentages
p.dp.non.code <- unlist(lapply(dp.non.code, mean))
mean(p.dp.non.code)
sd(p.dp.non.code)

# nombre total de non codé
n.dp.non.code <- sum(n.dp.non.code)
```

En moyenne 38% des RPU du jour n'ont pas de DP codés

Pour chaque établisement

```
# nombre de RPU sur la période par établissement:
n.rpu <- tapply(as.Date(dp$ENTREE), dp$FINESS, length)

# % de DP non codés par établissement:
# on forme une matrice de n lignes et c colonnes. Chaque ligne correspond à un jour de la période, chaqu
x <- function(x){mean(is.na(x))}
b <- tapply(dp$DP, list(as.Date(dp$ENTREE), dp$FINESS), x)

# on transforme la matrice des % de DP non codés en dataframe
c <- data.frame(unlist(b))
```

```
#nb moyen de RPU non codés
mean(b[,"Wis"])
m <- round(mean(b[,"Wis"])*100, 2)
plot(b[,"Wis"], type = "l", ylab = "% de RPU sans DP", xlab = "Jours", main = "Exemple d'un bon codeur"
text(x = 160, y = 0.6, labels = paste0("Taux de codage moyen = ", 100-m, " %"))


plot(b[,"Mul"], type = "l")
mean(b[,"Mul"], na.rm = TRUE)
```

19.1 Tableau de codeurs

```
p.codage <- 1 - apply(b, 2, mean, na.rm = TRUE)

plot(p.codage, pch = 16, col= ifelse(p.codage > 0.8,"blue","green"), ylim = c(0,1.1), xlim = c(0,20), yl

text(1:length(p.codage), p.codage + 0.05, names(p.codage))
abline(h = 0.8, lty = 2, col = "red")

# pour être considéré comme acceptable, un rectangle doit se situer sous la ligne pointillé rouge. De pl
# boxplot(b, las = 2, outline = FALSE, ylab = "% de  DP non codés", main = "Complétude du diagnostic pri
boxplot(b, las = 2, outline = FALSE, ylab = "% de  DP non codés", main = "Complétude et variabilité du c

abline(h = 0.2, lty = 2, col = "red")


# idem mais on calcule le nombre de DP non codé:
x <- function(x){sum(is.na(x))}
b2 <- tapply(dp$DP, list(as.Date(dp$ENTREE), dp$FINESS), x)
c2 <- data.frame(unlist(b2))
boxplot(b2, las = 2, outline = FALSE, ylab = "nombre de  DP non codés", main = "Complétude et variabili

# somme des DP non codés par établissement
n.non.code <- apply(c2, 2, sum, na.rm = TRUE)

# % non codage par établissement
n.non.code/n.rpu

# % exhaustivité
round(1 - n.non.code/n.rpu, 2)

# graphe
a <- sort(round(1 - n.non.code/n.rpu, 2))
plot(a, ylim = c(0, 1.1), ylab = "% de complétude du DP", xlab = "SU")
text(1:18, a + 0.1, names(a), cex = 0.8)
abline(h = 0.8, lty = 2, col = "red")
```

Nombre de RPU CCMU1 aux horaires de PDS (qs posée par Ste Anne: - nécessite lubridate - on forme un dataframe à 2 colonnes: ENTREE et CCMU = 1 - on ajoute une colonne JOUR pour le type de jour (dimanche = 1) - on ajoute une colonne HEURE (heure entière) - on fabrique un vecteur pour découper en tranches horaires - on calcule le nb de RPU dans chaque classe

```
library(lubridate)
```

```
ane <- ane2104[ane2104$GRAVITE == 1 & !is.na(ane2104$GRAVITE), c("GRAVITE","ENTREE")]
ane$JOUR <- wday(as.Date(ane$ENTREE))
ane$HEURE <- hour(ane$ENTREE)
h <- c(0, 8, 20, 23)
t <- cut(ane$HEURE, h, right = FALSE)
table(t)
# uniquement en semaine
t <- cut(ane$HEURE[ane$JOUR %in% 2:6], h, right = FALSE)
table(t)
# le samedi de 12 à 20h
t <- ane[ane$JOUR == 7 & ane$HEURE %in% 12:19,]
nrow(t)
# le dimanche de 8h à 20h
t <- ane[ane$JOUR == 7 & ane$HEURE %in% 12:19,]
nrow(t)
```

Seuil d'alerte lorsque le nombre deRPU transmis est anormalement bas. On calcule la moyenne er l'écart-type du nb de RPU par jour et par établissement au cours des 3 premiers mois de 2015. Le seuil est fixé à m - 1.96 * sd:

```
n <- tapply(as.Date(dsi$ENTREE), list(as.Date(dsi$ENTREE), dsi$FINESS), length)
m <- apply(n, 2, mean, na.rm = TRUE)
s <- apply(n, 2, sd, na.rm = TRUE)
seuil <- m - 1.96 * s
```

20 Correctif ste Anne

Au mois d'août 2015, ste Anne fournit un correctif rattrapant tout les RPU depuis le 1/1/2015 au 22/8/2015 inclu. Les données sont fournies sous forme s'un fichier .sql et concertie en .csv selon la méthode habituelle.

```
file <- "/home/jcb/Documents/Resural/Stat Resural/Archives_Sagec/dataQ/SteAnne2015/Ane_2015_01_01-2015_
ane2015 <- read.csv(file)
min(as.Date(ane2015$ENTREE))
max(as.Date(ane2015$ENTREE))
# on supprime la 1ère colonne qui est exédentaire (numérotation)
ane2015 <- ane2015[, -1]
# la répartition des types de colonnes n'est pas la même que celle de d15 car les string ont été automa
f <- function(x){as.character(x)}
a <- lapply(ane2015, f)
a <- as.data.frame(a, stringsAsFactors = FALSE)
a$AGE <- as.numeric(a$AGE)
```

On récupère l'année 2015

```
load("~/Documents/Resural/Stat Resural/RPU_2014/rpu2015d0112_provisoire.Rda")
```

Enfin on replace les enregistrement correspondant à Ste Anne par les nouveaux

```
new2015 <- rbind(d15[d15$FINESS != "Ane",], a)

d15 <- new2015
save(d15, file="rpu2015d0112_provisoire.Rda")
```

21 Correctif HUS (19/7/2015)

L'objectif est de remplacer les RPU intitulé Hus par les RPU HTP et NHC entre le **1er janvier 2015** et le **31/5/2015** inclus dans le fichier **rpu2015d0112_provisoire.Rda**.

1. transfert dans le dossier dataQ d'une copie des fichier .sql du 8/8/2015 au 8/06/2015.
2. traitement des fichiers sql

```
source("Preparation/RPU Quotidiens/quot_utils.R")
file.to.delete <- "/home/jcb/Documents/Resural/Stat Resural/Archives_Sagec/dataQ/archivesCsv/rpu2014.da
file.remove(file.to.delete)

date1 <- "2015-01-08"
date2 <- "2015-06-08"

p <- seq(as.Date(date1), as.Date(date2), 1)
  for(i in 1:length(p)){
  dx <- rpu_jour(p[i])
}
dx <- assemble(comment = TRUE)

min(as.Date(dx$ENTREE))
max(as.Date(dx$ENTREE))


dx <- dx[as.Date(dx$ENTREE) >= "2015-01-01" & as.Date(dx$ENTREE) < "2015-06-01",]

min(as.Date(dx$ENTREE))
max(as.Date(dx$ENTREE))

rpu.jour <- rpu.par.jour(dx)
head(rpu.jour)
```

Il y a u trou dans les données, fin avril-début mai (du 30/4 inclus au 5/5/2015 inclus). il faut compléter les données avec **rpu_2015-04-30au5_dump.sql** qui se trouve dans **dataQ/Fichiers_correction**. On suppose que l'on est dans le directory: /home/jcb/Documents/Resural/Stat Resural/RPU_2014

Il faut récupérer le fichier dans: /home/jcb/Documents/Resural/Archives_Sagec/dataQ/Fichiers_correction/

puis l'utiliser avec la méthode *parse_rpu* de *quot_utils.R* ne garder les données que du 30/4 au 5/5 inclus.

```
file <- "../../Archives_Sagec/dataQ/Fichiers_correction/rpu_2015-04-30au5_dump.sql"
manquant <- parse_rpu("", file)
min(as.Date(manquant$ENTREE))
max(as.Date(manquant$ENTREE))
manquant <- manquant[as.Date(manquant$ENTREE) < "2015-05-06",]

dx <- rbind(dx, manquant)
```

Sauvegarde

```
d15.hus <- dx
save(d15.hus, file = "d15.hus.Rda")
```

3. on extrait les dataframe **htp** et **nhc** contenant tous les enregidytrements pour HTP et NHC

```
htp <- dx[dx$FINESS == "HTP",]
nhc <- dx[dx$FINESS == "NHC",]
nrow(htp) + nrow(nhc)
```

4. préparation de d15

A partir du fichier d15:

- on isole les RPU allant du 1er janvier au 31 mai 2015, où HTP et NHC sont confondus sous le vocable Hus: d15.prov
- on isole les RPU allant du 1er juin à la date de point où HTP et NHC sont différenciés: d15.prov2
- on supprime de d15.prov tous les RPU correspondant aux HUS: d15.prov1

```
load("~/Documents/Resural/Stat Resural/RPU_2014/rpu2015d0112_provisoire.Rda")
d15.prov <- d15[as.Date(d15$ENTREE) < "2015-06-01",]
d15.prov2 <- d15[as.Date(d15$ENTREE) > "2015-05-31",]
d15.prov1 <- d15.prov[d15.prov$FINESS != "Hus",]
```

Dans d15.prov, on isole les RPU correspondant aux HUS et on vérifie que leur nombre est bien égal à la somme NHC + HTP calculée à patir de dx (cf. supra)

```
hus <- d15.prov[d15.prov$FINESS == "Hus",]
nrow(hus)
```

5. réunification des données

dans d15.prov1 on ajoute ___ htp___ et **nhc** qui vont remplacer les RPU hus.

```
d15.prov3 <- rbind(d15.prov1, htp, nhc)
```

d15.prov3 a bien le même nombre de lignes que **d15.prov**, le fichier de départ.

La dernière étape consiste à fusionner **d15.prov3** avec **d15.prov2** pour reconstituer le fichier d15 courant:

```
d15.prov4 <- rbind(d15.prov3, d15.prov2)
```

d15.prov4 et d15 ont bien le même nombre de lignes

```
tapply(as.Date(d15.prov4$ENTREE), d15.prov4$FINESS, length)
min(as.Date(d15.prov4$ENTREE))
max(as.Date(d15.prov4$ENTREE))
```

6. Sauvegarde finale

```
d15 <- d15.prov4
save(d15, file="rpu2015d0112_provisoire.Rda")
```

22 Nombre de RPU par mois

Objet: alimenter le site RESURAL

```
library(lubridate)
rpu.mois <- tapply(as.Date(dx$ENTREE), month(as.Date(dx$ENTREE)), length)
rpu.mois

# nb de jours dans le mois (la séquence doit inclure le mois suivant. SOURCE: https://stat.ethz.ch/pipe
n.j <- as.integer(diff(seq(as.Date("2015-01-01"), as.Date("2015-11-01"), by = "month")))
# nb de RPU par mois constant de 30 jours
rpu.mois.cst <- rpu.mois * 30 / n.j
barplot(rpu.mois.cst, main = "Nombre de RPU par mois standards de 30 jours", col = "cornflowerblue")
```

23 Analyse par semaine

23.1 Nombre de RPU par semaine

```
s <- tapply(as.Date(d15$ENTREE), week(as.Date(d15$ENTREE)), length)
s
tot <- sum(s) # nombre total de RPU
p = s/tot # % de RPU par semaine
summary(p)
```

s est un vecteur d'entier égal au nomdre de semaines écoulées de puis le début de l'année. Chaque élément correspond au nombre de RPU de la semaine.

23.2 Nombre de RPU par semaine et par Finess

s2 est une matrice (transformable en dataframe) contenant le nombre de RPU par semaine pour chaque ES. Par soustractions successives on établit la variation du nombre de RPU par semaines, et sa représentation graphique.

```
s2 <- tapply(as.Date(d15$ENTREE), list(week(as.Date(d15$ENTREE)), d15$FINESS), length)
class(s2)
colnames(s2)
rownames(s2)
s3 <- as.data.frame(s2)
barplot(s3[,"3Fr"], xlab = "semaines")

# calcul de la variation du nb de RPU d'une semaine à l'autre pour un ES
d3 <- s3[,"3Fr"]

# x compte une unité de moins que d3. Le 1er chiffre de d3 correspond à la semaine 2
x <- diff(d3)

# ajout de 0 en tête du vecteur pour remplacer la première semaine
x <- c(0, x)

# pour supprimer la denière semaine qui est souvent incomplète (option)
```

```
x <-x[-length(x)]

# barplot sauf la dernière semaine qui est souvent incomplète
b <- barplot(x, col = ifelse(x > 0, "blue", "red"), names.arg = 1:length(x), cex.names = 0.8, las = 2, r
text(b, ifelse(x > 0, x + 2,  x - 2), x, cex = 0.8)

#  en pourcentages: diff(x)/x
a <- x[-1] # on enlève le 0 initial
b <- d3[1:(length(d3)-2)] # ou -1 si on a pas supprimé la dernière semaine pour x
p <- round(a*100/b, 2)
p
```

24 Comparaison 2014-2015-2016

On compare novembre et décembre 2014 à novembre-décembre 2015 puis 2016 à 2015

24.1 Récupération des données 2014-2015

```
# novembre 2014
load("~/Documents/Resural/Stat Resural/DATA/RPU_2014/Archives 2014/rpu2014d11.Rda")
nov2014 <- d11
rm(d11)
# décembre 2014
load("~/Documents/Resural/Stat Resural/DATA/RPU_2014/Archives 2014/rpu2014d12.Rda")
dec2014 <- d12
rm(d12)
# novembre 2015
load("~/Documents/Resural/Stat Resural/DATA/RPU_2015/rpu2015d11.Rda")
nov2015
rm(d11)
# décembre 2015
dec2015 <- d15.p[as.Date(d15.p$ENTREE) > as.Date("2015-11-30"),]

# Pour 2015 il faut retirer le CCOM
nov2015 <- nov2015[nov2015$FINESS != "Ccm",]
dec2015 <- dec2015[dec2015$FINESS != "Ccm",]
```

Combinaison:

```
a <- matrix(c(nrow(nov2014), nrow(dec2014), nrow(nov2015), nrow(dec2015)), nrow = 2)
colnames(a) <- c("2014", "2015")
rownames(a) <- c("nov","dec")
a
barplot(t(a), beside = TRUE, main = "Activité 2014-2015")
text(b[2,1], 35000, paste(round((a[1,2]/a[1,1] - 1)*100, 2), "%"))
text(b[2,2], 35000, paste(round((a[2,2]/a[2,1] - 1)*100, 2), "%"))
```

25 Pétards 2015-2016

Exploité à partir de d15.p pour un résultat rapide

```
library(stringr)
pattern <- "[W][3][9]"
p <- d15.p[!is.na(d15.p$DP) & str_detect(d15.p$DP, pattern) == TRUE,] # il faut éliminer les NA
n.p <- nrow(p)

# Age
summary(p$AGE)
hist(p$AGE, col = "cornflowerblue", border = "white", ylab = "Fréquence", xlab = "Age (années)", main =

# Sexe
summary(as.factor(p$SEXE))

# nb par jour
t <- tapply(as.Date(p$ENTREE), as.Date(p$ENTREE), length)
t
barplot(t, main = "Lésions par pétards (2015-2016)")

# par établisement
s <- summary(p$FINESS)
s[s > 0]

# lieu d'habitation
c <- cbind(table(p$COMMUNE))
colnames(c) <- "Nombre"
c
```