

RPU Quotidiens

JcB

10/01/2014

Contents

1	Fichier RPU quotidien	1
1.1	Descriptif	2
1.2	Commentaires:	4
2	En pratique	4
2.1	si un seul fichier	4
2.2	Si on a une collection de fichiers:	5
2.3	exhaustivité des données	6
3	Résumé activité	7
4	Tracé de la courbe des RPU avec PLOT.XTS	7
5	Tracé de la courbe des RPU avec PLOT.ZOO	7
6	Activités quotidienne	7
7	Focus mains	9
8	Nombre de DP par jour	9
8.1	Tableau de codeurs	10

1 Fichier RPU quotidien

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 est 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'hôpital
- **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 être

#' @ data.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=="680000684"]<-"Col"
  a[a=="670016237"]<-"Odi"
  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 inexplicables: 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

```

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 on a une collection de fichiers:

2.2.0.3 NB: SUPPRIMER LE FICHIER /home/jcb/Documents/Resural/Stat Resural/Archives_Sagec/dataQ/archivesCsv/rpu2014.da

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).

```
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 <- "2015-06-08"
date2 <- "2015-06-21"
```

```
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)
```

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

2.2.0.4 Sélectionner une période particulière (ie. mai 2014) num.mc <- 1 # numéro du mois courant mois.c <- paste("d",)

```
d06 <- dx[as.Date(dx$ENTREE) >= "2015-06-01" & as.Date(dx$ENTREE) < "2015-07-01",]
max(as.Date(d06$ENTREE))
min(as.Date(d06$ENTREE))
d06 <- normalise(d06)
save(d06, file="rpu2015d06_provisoire.Rda")
rm(dx)
```

```
# si le mois est complet:
# save(d05, file="rpu2015d05.Rda")
# rm(rpu2015d05_provisoire.Rda)
```

```
# uniquement le premier mois
# d15 <- d01
# save(d15, file="rpu2015d0112_provisoire.Rda")
# save(d15, file="rpu2015d0112.Rda")
```

```
load("rpu2015d01.Rda") # mois précédent
load("rpu2015d02.Rda") # mois précédent
load("rpu2015d03.Rda") # mois précédent
load("rpu2015d04.Rda") # mois précédent
load("rpu2015d05.Rda") # mois précédent
```

```
dx <- rbind(d01, d02, d03, d04, d05, d06)
min(as.Date(dx$ENTREE))
max(as.Date(dx$ENTREE))
```

```
d15 <- dx
save(d15, file="rpu2015d0112_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$FINISS), length)
```

```
rpu.par.jour(d06)
```

2.2.0.5 Nombre de attendus pour l'année: “{ rpu_extrapolés} n <- nrow(d15) td <- max(as.Date(d15*ENTREE*)) – min(as.Date(dx*ENTREE*)) n * 365 / as.numeric(td)

“

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

```
“{ doublons} a <- duplicated(d15[, c(2,3,6,8,13,16)]) sum(a) which(a) # quelle ligne d15[which(a), c(2,3,6,8,13,16)]
```

“

2.2.1.1 Eventuellement sauvegarder au format .Rda

```
#dx <- normalise(dx)
#dx$FINISS <- factor(dx$FINISS) # 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**.

2.3 exhaustivité des données

On forme une table en croisant FINESS et ENTREE:

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

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

3 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** créée précédemment, on ajout 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.

```
“{ activité}
s <- rowSums(rpu) b <- rownames(rpu) a <- as.data.frame(cbind(b,s)) m <- rowMeans(rpu)
colnames(a) <- c("Date","RPU") aDate <- as.Date(aDate) aRPU <- as.numeric(as.character(aRPU))
# transforme les facteurs en nombre
library("xts") library("lubridate")
plot(aDate,aRPU, type="l", ylab="nombre de RPU", xlab=paste0("Année", year(Sys.Date())),
main="Activité des SU d'Alsace en nombre de RPU")
```

4 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 = "brown") # {"years",
"months", "weeks", "days", "hours", "minutes", "seconds"} lines(rollmean(x, 7), col="red", lwd = 3)
```

5 Tracé de la courbe des RPU avec PLOT.ZOO

```
z <- as.zoo(x) plot(z, col="blue", 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")
x <- as.xts(aRPU,aDate)
z <- as.zoo(x) plot(z, col="blue", 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**

6 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

```
<<<<<< HEAD
d <- parse_rpu("2015-05-31")
=====
d <- parse_rpu("2015-06-15")
>>>>>> f3563fb10689def9b28448fd2e8b594712ced1b5
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$ENTREE), list(d$FINESSE, 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$ENTREE) > max(as.Date(d15$ENTREE)),]
d <- rpu2factor(d) # transforme les chiffres en facteurs cohérent avec d01
```

```
d15.p <- rbind(d15, d)
max(as.Date(d15.p$ENTREE))
```

```
# sauvegarde
save(d15.p, file = "d15_p.Rda")
```

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

```
library(xts)
a <- tapply(as.Date(d15.p$ENTREE), as.Date(d15.p$ENTREE), length)
xts.a <- xts(a, order.by = unique(as.Date(d15.p$ENTREE)))
```

```
svg("activite SU alsace 2015.svg")
plot(xts.a, las = 2, main = "Activité des SU d'Alsace", ylab = "nombre de RPU créés par jour", ylim = c(
mean2014 <- 1141.734
sd2014 <- 154.7858
abline(h = mean2014, col = "red") # nb moyende RPU en 2014
abline(h = mean2014 + sd2014, col = "red", lty = 2) # 1 écart-type
abline(h = mean2014 + sd2014 * 2, col = "red", lty = 2) # 2 écart-type
legend("topleft", legend = c("moyenne 2014", "écart-type 2014", "moyenne lissée"), col = c("red","red",
lines(rollmean(xts.a, 7), col="blue", lwd = 3) # moyenne lissée
copyright()
dev.off()
```



```
svg("activite SU alsace 2015.svg") plot() dev.off()
```

7 Focus mains

En 2014:

```
mains <- d14[d14$FINESS %in% c("Dts","Ros"),]
nrow(mains)/nrow(d14)
nrow(mains)*100/nrow(d14)

ros <- d14[d14$FINESS == "Ros",]
nrow(ros)

dts <- d14[d14$FINESS == "Dts",]
nrow(dts)
min(as.Date(dts$ENTREE))
max(as.Date(dts$ENTREE))
c <- tapply(as.Date(dts$ENTREE), as.Date(dts$ENTREE), length)
cbind(c)
```

8 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$ENTREE) < as.Date("2015-04-01"),]
# nombre de RPU
n <- nrow(dp)
# diagnostic non codés par jour: renvoie un array de n listes. n est égal au nombre de jours couverts p
dp.non.code <- tapply(dp$DP, as.Date(dp$ENTREE), is.na)
# en nombre: on calcule la somme de chaque liste, puis on délisle. On obtient pour chaque jour, le nomb
n.dp.non.code <- unlist(lapply(dp.non.code, sum))
# en pourcentages
p.dp.non.code <- unlist(lapply(dp.non.code, mean))
# nombre total de non codé
n.dp.non.code <- sum(n.dp.non.code)
```

Pour chaque établissement

```
# 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, chaq
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))

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

8.1 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), ylab = "% de DP c

text(1:15, 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 p
boxplot(b, las = 2, outline = FALSE, ylab = "% de DP non codés", main = "Complétude du diagnostic prin
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 du diagnosti

# 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))
text(1:15, a + 0.1, names(a), cex = 0.8)
```

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 de RPU transmis est anormalement bas. On calcule la moyenne et 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$FINISS), length)
m <- apply(n, 2, mean, na.rm = TRUE)
s <- apply(n, 2, sd, na.rm = TRUE)
seuil <- m - 1.96 * s

```