Digraph

JcB

18/12/2014

# Préparation des fichiers pour Dygraph

**Dygraph** (DG) est une bibliothèque Javascript permettant d'afficher des courbes dynamiques. DG affiche des données de séries temporelles à partir d'une matrice dont la première colonne est obligatoirement une colonne **date** au format ISO. Chaque colonne qui suit sera affichée individuellement.

Deux tables sont préparées pour *Resural*: - **hop\_2014.csv** nombre de passages par jour pour l'ensemble des SU d'Alsace et nombre d'hospitalisations. - **SU\_2014.csv** nombre de passages par jour et par SU + total, moyenne et nb hospitalisation => fichier complet pour **Dygraph**.

L'application test se trouve dans le dossier DyGraph.

## Passages et taux d'hospitalisation

Fabrique et affichage d'un dataframe de 6 colonnes:

* date du jour
* nombre total de passages = mutation + transferts
* nombre d'hospitalisation
* nombre de mutation
* nombre de transferts
* taux d'hospitalisation = hospitalisation / passages

todo: tester le dataframe avec dygraph

Le dataframe s'appelle **devenir** et il s'enregistre dans *devenir.csv* [read.csv("devenir.csv")]

load("~/Documents/Resural/Stat Resural/RPU\_2014/rpu2014d0112\_c.Rda") # d14  
load("~/Documents/Resural/Stat Resural/RPU\_2014/rpu2015d0112\_provisoire.Rda")  
source("../../new\_functions.R") # f0nctopn mode.sotie()  
library(xts)

## Loading required package: zoo  
##   
## Attaching package: 'zoo'  
##   
## The following objects are masked from 'package:base':  
##   
## as.Date, as.Date.numeric

# création d'une table date, passages, hospitalisation, mutation, transfert  
# hospitalisation = mutation + transfert  
# remplacé par la fonction mode.sortie:  
# passages.jour <- tapply(as.Date(d14$ENTREE), as.Date(d14$ENTREE), length)  
# mut <- d14[d14$MODE\_SORTIE == "Mutation", "ENTREE"]  
# mutations.jour <- tapply(as.Date(mut), as.Date(mut), length)  
# trans <- d14[d14$MODE\_SORTIE == "Transfert", "ENTREE"]  
# transfert.jour <- tapply(as.Date(trans), as.Date(trans), length)  
# hospit.jour <- mutations.jour + transfert.jour  
# date <- unique(sort(as.Date(d14$ENTREE)))  
# devenir <- data.frame(date, passages.jour, hospit.jour, mutations.jour, transfert.jour)  
  
ms2014 <- mode.sortie(d14)  
ms2015 <- mode.sortie(d01)  
devenir <- rbind(ms2014, ms2015) # on lie les 2 années  
  
# paramètres dérivés  
apply(devenir[2:5], 2, mean, na.rm = TRUE)

## passages.jour hospit.jour mutations.jour transfert.jour   
## 1148.41406 227.02083 210.66406 16.35677

apply(devenir[2:5], 2, median, na.rm = TRUE)

## passages.jour hospit.jour mutations.jour transfert.jour   
## 1137 225 209 16

apply(devenir[2:5], 2, sd, na.rm = TRUE)

## passages.jour hospit.jour mutations.jour transfert.jour   
## 154.739073 27.061377 25.632860 4.905403

apply(devenir[2:5], 2, min, na.rm = TRUE)

## passages.jour hospit.jour mutations.jour transfert.jour   
## 768 162 155 5

apply(devenir[2:5], 2, max, na.rm = TRUE)

## passages.jour hospit.jour mutations.jour transfert.jour   
## 1689 341 321 49

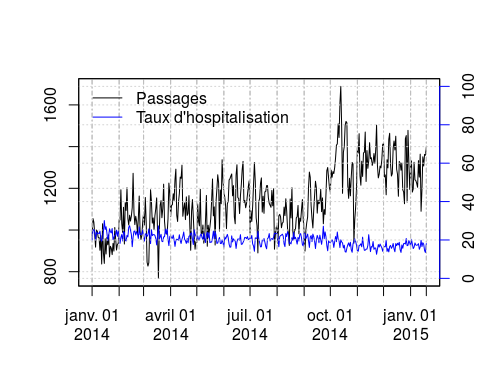
taux.hosp <- round(devenir$hospit.jour \* 100 / devenir$passages.jour, 2)  
summary(taux.hosp)

## Min. 1st Qu. Median Mean 3rd Qu. Max.   
## 12.57 17.85 20.20 20.05 22.32 30.12

sd(taux.hosp)

## [1] 3.075179

# création d'un ojet xts  
d.xts <- xts(devenir, order.by = devenir$date)  
# graphe avec 2 axes y pour les passages et le taux d'hospitalisation  
plot(d.xts$passages, minor.ticks = FALSE, main = "")  
par(new = T) # permet de dessiner un second graphique avec ses propres paramètres  
plot(d.xts$taux, axes = F, ylim = c(0, 100), col = "blue", main="")  
axis(4, ylim = c(0, 100), col = "blue" ) # utilise l'axe de droite. Prévoir plus de marge  
legend("topleft", legend = c("Passages","Taux d'hospitalisation"), col = c("black", "blue"), lty = 1, bty = "n")



# enregistrement du dataframe en csv:  
write.csv(devenir, file = "devenir.csv")

## Circuit rapide

NB: write.csv insère "" comme nom de la première colonne qui correspond aux dates.

dx <- d14  
  
# autr chose  
k <- table(as.Date(dx$ENTREE), factor(dx$FINESS))  
total <- apply(k,1,sum)  
hop <- cbind(k, total)  
mean <- apply(k,1,mean)  
hop <- cbind(hop, mean)  
# hospitalisations:  
a <- dx[dx$MODE\_SORTIE == "Mutation" | dx$MODE\_SORTIE == "Transfert", 'ENTREE']  
a <- as.Date(a)  
# hosp <- tapply(a, as.factor(yday(a)), length)  
hosp <- tapply(a, a, length)  
hop <- cbind(hop, hosp)  
write.csv(hop, file="SU\_2014.csv")  
  
date <- rownames(k)  
hop2 <- cbind(date, total, hop[,"hosp"])  
colnames(hop2) <- c("Date", "Passages", "Hospitalisations")  
write.csv(hop2, file="hop\_2014.csv", row.names=F)

## Création du fichier *hop*2014.csv\_

1. on décupère le fichier général de l'ensemble des RPU provisoires (**dx**) ou consolidés (**d1**). Les dates utilisées sont dans la colonne \_\_dxENTREE) permet de ne conserver que la date du jour et d'éliminer la partie horaire.
2. on crée la *table*  **k** en croisant les colonnes **dxFINESS**

* k <- table(as.Date(dx$ENTREE), dx$FINESS)  
    
   3Fr Alk Col Dia Geb Hag Hus Mul Odi Sav Sel Wis Dts Ros  
   2014-01-01 46 0 177 76 44 99 90 130 73 97 101 39 0 0 0  
   2014-01-02 34 31 181 77 33 101 134 189 71 83 84 37 0 0 0  
   2014-01-03 36 38 199 82 34 86 117 187 61 93 75 30 0 0 0  
   2014-01-04 38 17 169 92 31 89 95 177 71 84 87 32 0 0 0  
   2014-01-05 42 1 164 75 38 85 79 183 76 64 76 35 0 0 0  
   2014-01-06 40 11 160 90 51 87 128 179 77 78 76 23 0 0 0

1. on fait la somme de chaque ligne dans le vecteur **total** avec la méthode *apply*: total <- apply(k,1,sum). Puis on "colle" ce vecteur à la table *k* sous forme d'une colonne supplémentaire:

* total <- apply(k,1,sum)  
  hop <- cbind(k, total)  
  head(hop)  
    
   3Fr Alk Col Dia Geb Hag Hus Mul Odi Sav Sel Wis Dts Ros total  
  2014-01-01 46 0 177 76 44 99 90 130 73 97 101 39 0 0 0 972  
  2014-01-02 34 31 181 77 33 101 134 189 71 83 84 37 0 0 0 1055  
  2014-01-03 36 38 199 82 34 86 117 187 61 93 75 30 0 0 0 1038  
  2014-01-04 38 17 169 92 31 89 95 177 71 84 87 32 0 0 0 982  
  2014-01-05 42 1 164 75 38 85 79 183 76 64 76 35 0 0 0 918  
  2014-01-06 40 11 160 90 51 87 128 179 77 78 76 23 0 0 0 1000

1. De la même manière on calcule la **moyenne** (*mean*) et l'écart-type (*sd*):

mean <- apply(k,1,mean)  
sd <- apply(k,1,sd)  
hop.moy <- cbind(hop, mean, sd)  
head(hop.moy)  
  
 3Fr Alk Col Dia Geb Hag Hus Mul Odi Sav Sel Wis Dts Ros total mean sd  
2014-01-01 46 0 177 76 44 99 90 130 73 97 101 39 0 0 0 972 64.80000 53.07165  
2014-01-02 34 31 181 77 33 101 134 189 71 83 84 37 0 0 0 1055 70.33333 60.90234  
2014-01-03 36 38 199 82 34 86 117 187 61 93 75 30 0 0 0 1038 69.20000 61.72543  
2014-01-04 38 17 169 92 31 89 95 177 71 84 87 32 0 0 0 982 65.46667 56.31273  
2014-01-05 42 1 164 75 38 85 79 183 76 64 76 35 0 0 0 918 61.20000 55.66250  
2014-01-06 40 11 160 90 51 87 128 179 77 78 76 23 0 0 0 1000 66.66667 57.11350

1. transformation en objet de type **xts**

Many xts-sepcific methods have been written to better handle the unique aspects of xts. These include, ‘"["’, merge, cbind, rbind, c, Ops, lag, diff, coredata, head and tail. Additionally there are xts specific methods for converting amongst R's different time-series classes.

Subsetting via "[" methods offers the ability to specify dates by range, if they are enclosed in quotes. The style borrows from python by creating ranges with a double colon “"::"” or “"/"” operator. Each side of the operator may be left blank, which would then default to the beginning and end of the data, respectively. To specify a subset of times, it is only required that the time specified be in standard ISO format, with some form of separation between the elements. The time must be ‘left-filled’, that is to specify a full year one needs only to provide the year, a month would require the full year and the integer of the month requested - e.g. '1999-01'. This format would extend all the way down to seconds - e.g. '1999-01-01 08:35:23'. Leading zeros are not necessary. See the examples for more detail.

library("xts")  
xts <- as.xts(hop.moy, descr ="Fréquentation des SU")  
head(xts)  
  
Un jour donné: xts["2014-01-01"]   
une plage: xts["2014-01-01/2014-01-03"] ou xts["2014-01-01::2014-01-03"]   
toutes les données: xts["/"]   
toutes les données depuis mars: xts['2014-03/']   
Toutes les données entre mars et juin: xts['2014-03/2014-06']   
toutes les données depuis mars jusqu'à la fin de l'année: xts['2014-03/2014']   
toutes les données jusque 2014 inclu: xts['/2014']

#### Plotting

L'objet *xts* ne peut dessiner que des séries temporelles univariées, en pratique une seule colonne à la fois. Si on passe toute la matrice à la fonction **plot** seule la dernière colonne est prise en compte et un message d'avis est affiché (In plot.xts(xts) : only the univariate series will be plotted).

plot(xts[,'mean'], main="Fréquentation moyenne des SU", ylab="Nombre de passages", minor.ticks = FALSE, col="gray")  
  
avec moyenne mobile:  
  
moy.mobile <- rollmean(xts$mean,7)  
lines(moy.mobile, col="blue", lwd=3)

On fabrique l'objet **cus** de type xps, qui ne conserve que les colonnes *total*, *mean* et *sd*:

cus <- xts[,c('total','mean','sd')]  
head(cus)  
 total mean sd  
2014-01-01 972 64.80000 53.07165  
2014-01-02 1055 70.33333 60.90234  
2014-01-03 1038 69.20000 61.72543  
2014-01-04 982 65.46667 56.31273  
2014-01-05 918 61.20000 55.66250  
2014-01-06 1000 66.66667 57.11350  
  
La classe \_xts\_ hérite de la classe \_zoo\_. A ce titre, elle possède plusieurs fonctions mobiles:  
  
moy.mobile <- rollmean(cus$total,7)  
colnames(moy.mobile) <- "moy.mob"  
sd.mobile <- rollapply(cus$total, 7, sd)  
colnames(sd.mobile) <- "sd.mob"  
cus <- cbind(cus, moy.mobile, sd.mobile)  
  
source("")  
plot(cus$total, minor.ticks = FALSE, main="Fréquentation moyenne des SU d'Alsace", ylab="Nombre de RPU", col="gray")  
lines(moy.mobile, col="blue", lty=2)  
lines(moy.mobile+sd.mobile, col="blue")  
lines(moy.mobile-sd.mobile, col="blue")  
legend("topleft", legend=c("moyenne mobile","écart-type"), col=c("blue","blue"), lty=c(2,1))

On refait la même manip mais en ajoutant les hospitalisations = somme mutation + transferts:

d <- read.table("../../data2.csv", header=TRUE, sep=",")  
library("lubridate")  
  
load("../../rpu2013-2014.Rda") # charge d2  
dx <- d2  
rm(d2)  
hosp <- dx[dx$MODE\_SORTIE == "Mutation" | dx$MODE\_SORTIE == "Transfert", 6] # 6 est la colonne ENTREE  
hospitalisation <- tapply(hosp, as.Date(hosp), length) # nb hospit par jour  
  
hop <- data.frame(cbind(hop, hospitalisation))

## Warning in cbind(hop, hospitalisation): number of rows of result is not a  
## multiple of vector length (arg 2)

head(hop)

## X3Fr Alk Col Dia Geb Hag Hus Mul Odi Sav Sel Wis Dts Ros Ane  
## 2014-01-01 46 0 177 76 44 99 90 130 73 97 101 39 0 0 0  
## 2014-01-02 34 31 181 77 33 101 134 189 71 83 84 37 0 0 0  
## 2014-01-03 36 38 199 82 34 86 117 187 61 93 75 30 0 0 0  
## 2014-01-04 38 17 169 92 31 89 95 177 71 84 87 32 0 0 0  
## 2014-01-05 42 1 164 75 38 85 79 183 76 64 76 35 0 0 0  
## 2014-01-06 40 11 160 90 51 87 128 179 77 78 76 23 0 0 0  
## X670780204 total mean hosp hospitalisation  
## 2014-01-01 0 972 60.7500 194 222  
## 2014-01-02 0 1055 65.9375 265 204  
## 2014-01-03 0 1038 64.8750 264 184  
## 2014-01-04 0 982 61.3750 237 194  
## 2014-01-05 0 918 57.3750 184 167  
## 2014-01-06 0 1000 62.5000 242 143

t <- data.frame(rownames(hop), hop$total, hop$hospitalisation)  
colnames(t)<-c("date","passages","hospitalisations")  
write.csv(t, file="data2.csv", row.names = FALSE)