Xts

Jcb

19/07/2014

Contents

1 Comment utiliser eXtended Time Series (XTs)

1.1 Documentation

- ebook r_cookbook.pdf chapitre 14.
- shading and points with plot.xts from xtsExtra
- Quantitative Finance Applications in R – 3: Plotting xts Time Series
- R graph with two y axes: comment dessiner sur la même figure, 2 courbes ayant les mêmes abcisses et des ordonnées différentes (par exemple le nombre de passages et le nombre d'hospitalisations).
- Liste de ressources pour R en français
- Utilisation de XtsExta

1.2 Fichier utilisé pour la démo

Activité du SAMU 67 en 2014. Les données sont fournies par le serveur de veille et d'alerte SAGEC.

```
# dataframe pour la démo:
library(xts)
```

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

```
path <- "../RPU_2014/Analyse/Activite_SAMU/" # pour lancer depuis la console
path <- "../../RPU_2014/Analyse/Activite_SAMU/" # pour machine XPS maison
file <- "samu67_2014.csv"
d67 <- read.csv(paste0(path,file))
d67$date <- as.Date(d67$date, "%d/%m/%Y")
# on se limite à 2014
d67 <- d67[d67$date < "2015-01-01",]
```

1.3 R time series

On peut identifier trois classes:

- **ts** est la classe de base. En pratique elle est à éviter car trop restrictive et comprtant peu de fonctions utiles.
- **zoo** hérite de la précédente
- **xts** étend les capacité de zoo. Il est également optimisé pour la vitesse et doit être préféré pour de gros volumes de données.

On peut compléter avec deux documents de référence:

1.4 Créer un objet Time Series (ts)

```
vignette("zoo")
vignette("xts")
```

```
ts <- xts(x, dt)
```

```
zts <- zoo(x, dt)
```

```
ts <- as.xts(zoo)
```

- x un vecteur, une matrice ou un dataframe. x doit être de type **numeric**
- dt un vecteur de type *Date* ou équivalent. Ce vecteur est appelé *index*. Son implémentation diffère entre *zoo* et *xts*:
- zoo: l'index peut être n'importe quelles valeurs ordonnées comme des objets *Date*, *POSIXct*, integers ou des flottants.
- xts: sont atorisés Date , POSIXct , et les objets de type chron (yearmon , yearqtr , and dateTime objects). *xts* est plus restrictif que *zoo* car il implémente des fonctions avancées qui nécessitent impérativement des objets de type *date*.

Application: création d'un objet xts à partir du fichier SAMU 67

```r
# création d'un objet XTS
a <- xts(d67, order.by = d67$date)
```

ou plus simplement: a <- xts(d67, d67$date). L'*index* est une des colonnes du dataframe. On peut d'ailleurs supprimer cette colonne au moment de la création de l'objet xts:

```r
a <- xts(d67[,-2], d67$date)
```

Deux méthodes permettent de récupérer séparément l'index (**index**) et les données (**core.data**). La méthode *index* remplace la fonction *rownames* qui n'est pas implémenté pour *xts*.

```r
head(index(a))
```

```
## [1] "2014-01-01" "2014-01-02" "2014-01-03" "2014-01-04" "2014-01-05"
## [6] "2014-01-06"
```

```r
head(coredata(a))
```

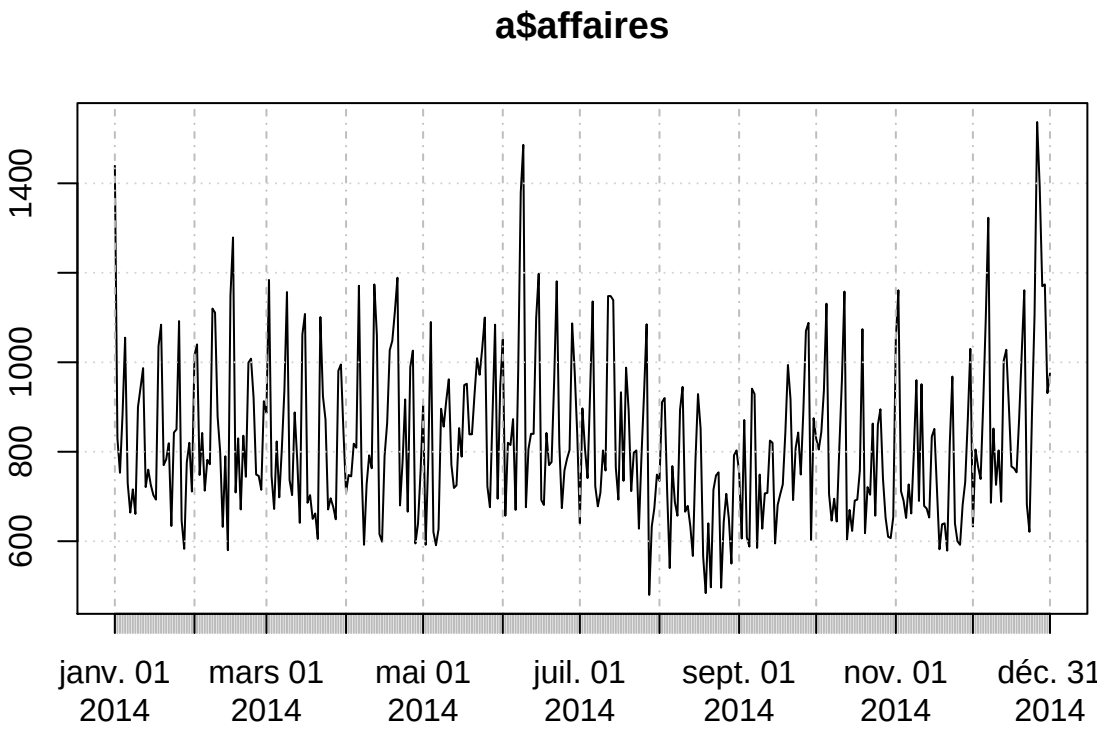
```
##      jour       affaires moyenne     ecart.type  St
## [1,] "mercredi" "1439"   "1063.0000" "254.23283" " 0.678959250"
## [2,] "jeudi"    " 830"   "1066.1429" "249.83090" "-1.745210787"
## [3,] "vendredi" " 753"   "1093.7143" "281.61600" "-2.009854148"
## [4,] "samedi"   " 884"   "1012.4286" "260.10886" "-1.293749315"
## [5,] "dimanche" "1055"   "1005.8571" "266.89226" "-0.615870052"
## [6,] "lundi"    " 730"   " 967.2857" "261.36738" "-1.707862790"
##      Cusum
## [1,] "0.678959250"
## [2,] "0.000000000"
## [3,] "0.000000000"
## [4,] "0.000000000"
## [5,] "0.000000000"
## [6,] "0.000000000"
```

1.5 Tracer le graphe d'une série temporelle

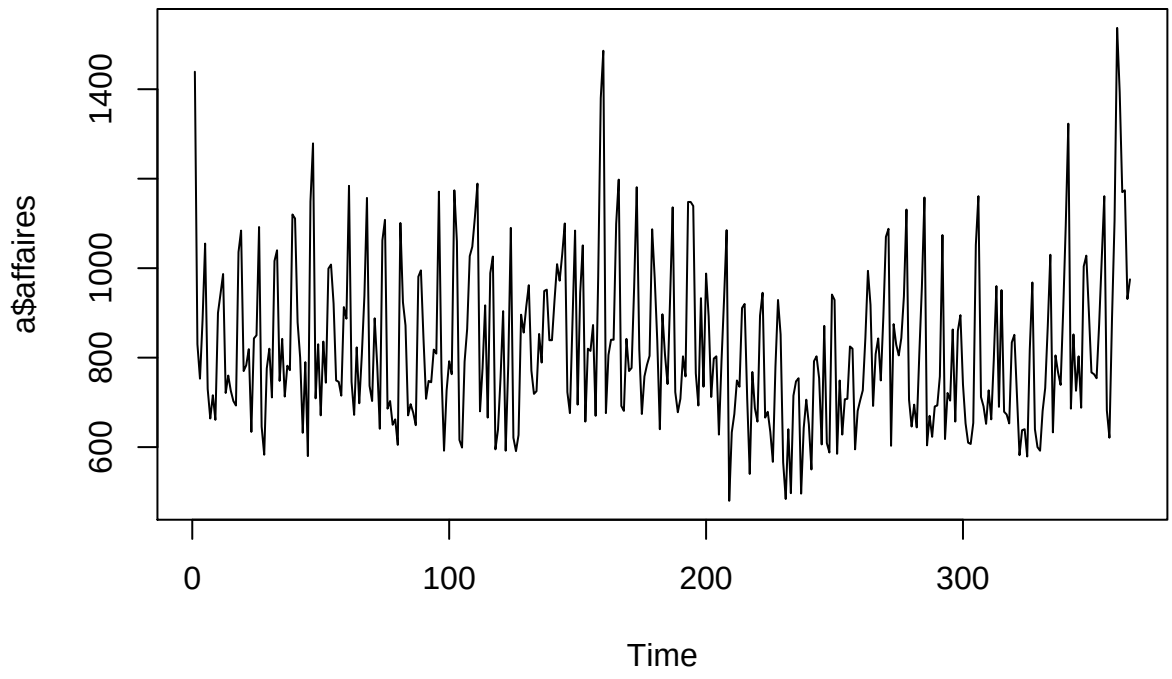
XTS étend la classe ZOO. En particulier on peut utilisser un dataframe comprenant plusieurs colonnes. Une de ces colonnes doit être du type **Date** et sera indiquée par la variable **order.by**.

La méthode **plot** permet de tracer des TS simples ou multipes:

```r
plot(a$affaires,type="l")
```



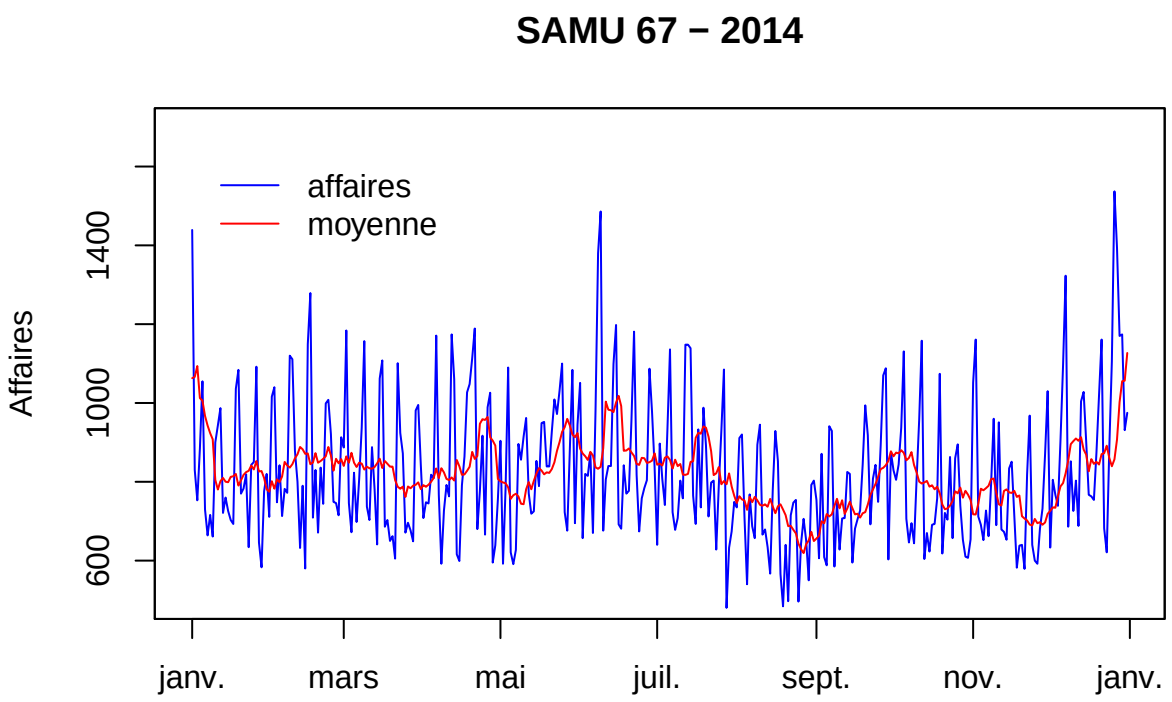

```
plot.ts(a$affaires)
```



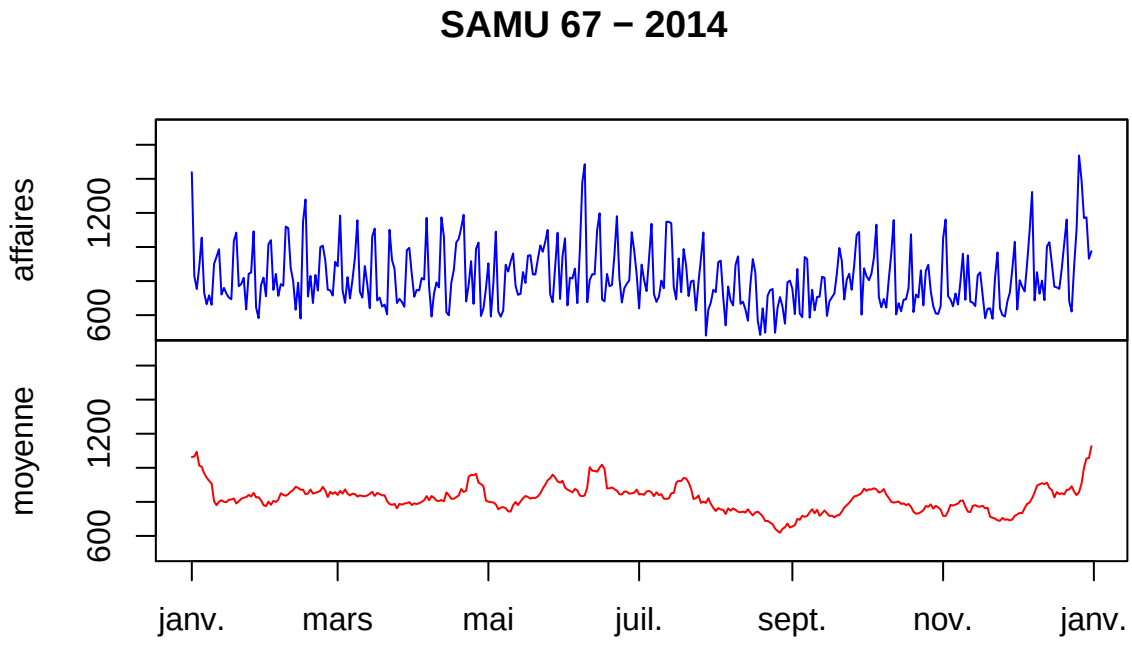
xts ne sait dessiner qu'une série temporelle à la fois. Pour en tacer deux simultannément, il faut utiliser **zoo**. On forme un objet **zoo** comportant 2 colonnes, affaires et moyenne. En jouant avec le paramètres *screens* on peut afficher les graphes sur le même graphique ou sur deux graphiques distincts

```
b <- zoo(a[, c(2:3)])

# Plot the two time series in one plot
```

```
plot(b, screens = 1, ylim = c(500,1700), col = c("blue", "red"), main = "SAMU 67 - 2014", xlab= "", ylab
legend(as.Date("2014-01-01"), 1650, c("affaires","moyenne"), lty=c("solid", "solid"), col = c("blue", "
```




```
# Plot the two time series in two plots
plot(b, screens = c(1,2), ylim = c(500,1700), col = c("blue", "red"), main = "SAMU 67 - 2014", xlab= ""
```

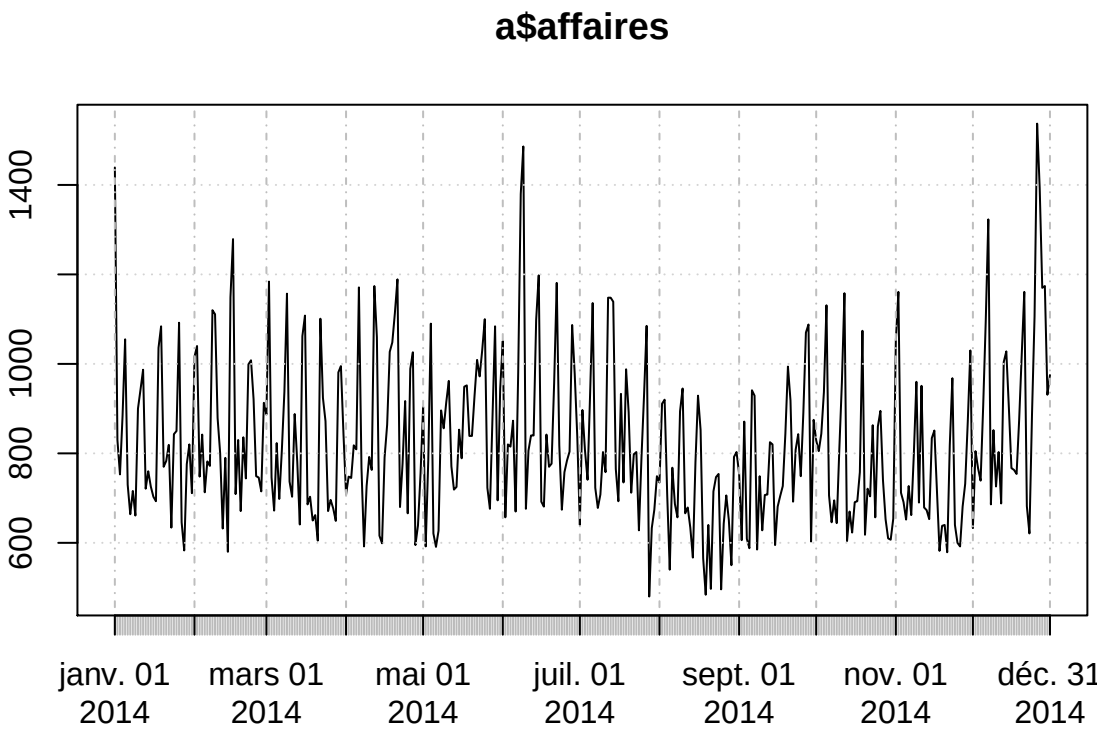



```
head(b)
```

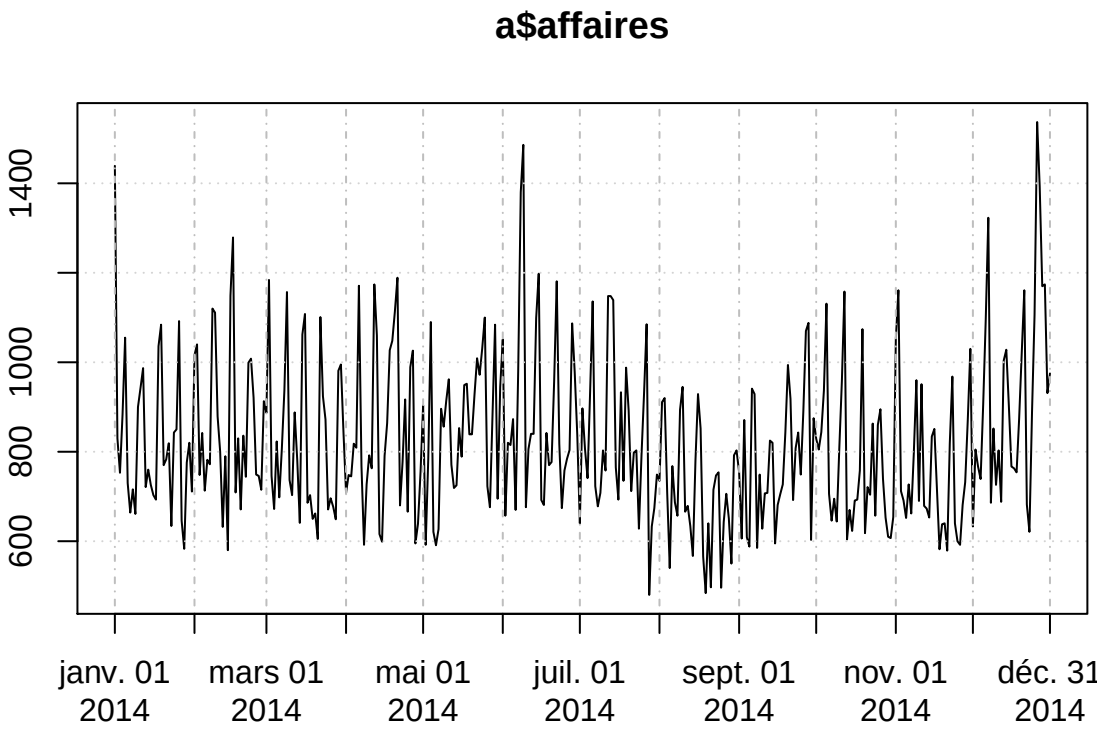
```
##            affaires moyenne
```

```
## 2014-01-01 1439     1063.0000
## 2014-01-02  830     1066.1429
## 2014-01-03  753     1093.7143
## 2014-01-04  884     1012.4286
## 2014-01-05 1055     1005.8571
## 2014-01-06  730      967.2857
```

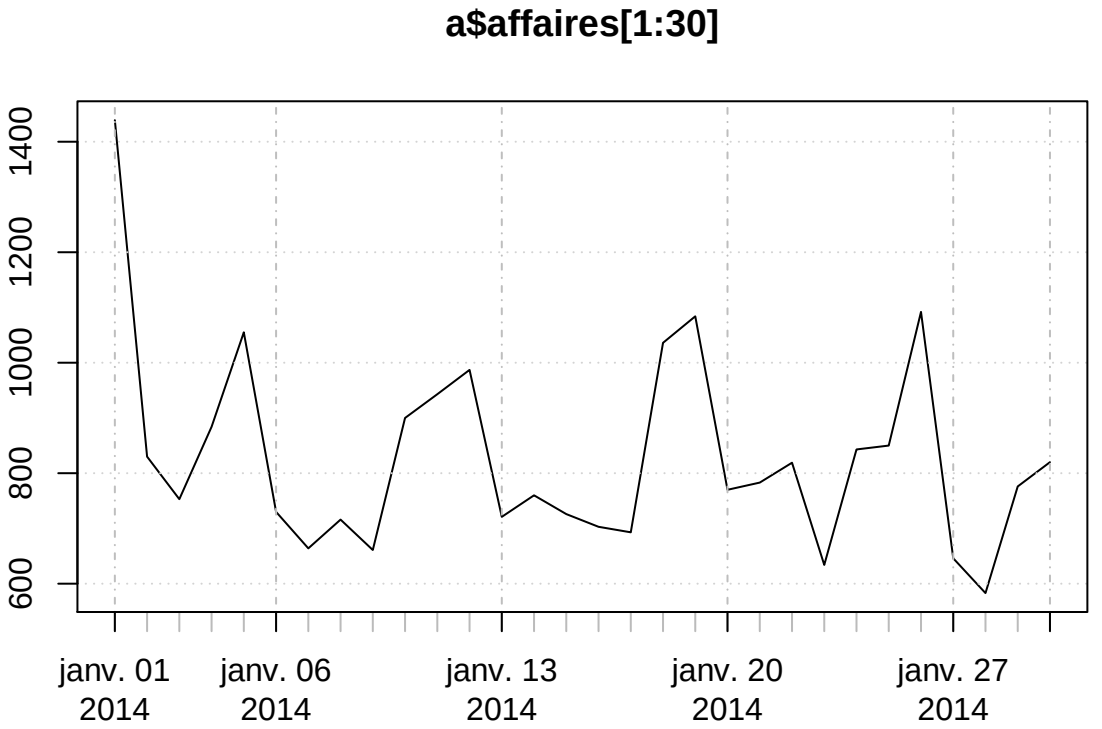
```r
plot(a$affaires) # Précise la colonne à plotter
```




```r
plot(a$affaires, minor.ticks = FALSE) # supprime les graduations mineures. Indiqué lorsque l'échelle de
```




```r
plot(a$affaires[1:30], minor.ticks = TRUE) # uniquement les 30 PREMIERS JOURS
# marquer les dimanches
points(c(a$date[5], a$affaires[5]), col = "red")
points(c(a$date[12], a$affaires[12]), col = "red")
```



Première et dernière observations

Les objets *ts* rangent les observations par ordre chronologique, on peut utiliser les méthodes **head** et **tail**

pour retrouver les observations extrêmes. *xts* propose également:

- **first**: "3 weeks", "5 days", "months", "years", "2 quarter". Voir help(first.xts)
- **last_**: vir help(last.xts)

```
first(as.xts(b), "3 weeks")
```

```
##            affaires moyenne
## 2014-01-01 "1439"   "1063.0000"
## 2014-01-02 " 830"   "1066.1429"
## 2014-01-03 " 753"   "1093.7143"
## 2014-01-04 " 884"   "1012.4286"
## 2014-01-05 "1055"   "1005.8571"
## 2014-01-06 " 730"   " 967.2857"
## 2014-01-07 " 664"   " 942.7143"
## 2014-01-08 " 716"   " 923.4286"
## 2014-01-09 " 661"   " 907.8571"
## 2014-01-10 " 900"   " 804.5714"
## 2014-01-11 " 943"   " 780.4286"
## 2014-01-12 " 987"   " 801.4286"
## 2014-01-13 " 721"   " 809.8571"
## 2014-01-14 " 760"   " 800.1429"
## 2014-01-15 " 726"   " 798.8571"
## 2014-01-16 " 703"   " 812.5714"
## 2014-01-17 " 693"   " 814.0000"
## 2014-01-18 "1036"   " 820.0000"
## 2014-01-19 "1084"   " 790.4286"
```

```
last(as.xts(b), "3 days")
```

```
##            affaires moyenne
## 2014-12-29 "1174"   "1055.2857"
## 2014-12-30 " 931"   "1056.5714"
## 2014-12-31 " 975"   "1126.8571"
```

1.6 Extraction de sous ensembles

xts rend possible l'extraction d'objets par années, semaines, jours ou même secondes. L'opérateur [] accepte des indices numériques habituels, mais également des *datetime* au format ISO:8601 time format — "CCYY-MM-DD HH:MM:SS". En pratique cela veut dire que pour extraire un mois particulier, il est nécessaire de spécifier l'année. Pour identifier une heure particulière, par exemple 8 heures le 1er janvier 2014, il faut inclure la date complète en plus de l'heure: "2014-01-01 08". Il est aussi possible de rechercher explicitement un intervalle de temps en utilisant le sépatateur "/" recommandé par la norme ISO. La forme de base est *"from/to"* où *from* et *to* sont optionnels. Si l'un des deux (ou les deux) est manquant, c'est interprété comme "rechercher depuis le début", ou "jusqu'à la fin" selon le cas. "/" équivaut à récupérer toutes les données. Cela permet également d'extraire de données sans savoir exactement quelles sont les limites.

Exemples:

```
mars <- a['2014-03']
head(mars)
```

1.6.0.1 Toutes les données du mois de mars

```
##            jour       affaires moyenne     ecart.type  St
## 2014-03-01 "samedi"   " 887"   " 840.2857" "130.39902" "-0.441758924"
## 2014-03-02 "dimanche" "1184"   " 864.4286" "125.13973" " 1.753716753"
## 2014-03-03 "lundi"    " 746"   " 848.4286" "111.48073" "-1.718800717"
## 2014-03-04 "mardi"    " 672"   " 873.5714" "161.90929" "-2.044965161"
## 2014-03-05 "mercredi" " 823"   " 848.5714" "166.80214" "-0.953303964"
## 2014-03-06 "jeudi"    " 698"   " 837.5714" "176.70772" "-1.589843403"
##            Cusum
## 2014-03-01 "0.000000000"
## 2014-03-02 "1.753716753"
## 2014-03-03 "0.034916036"
## 2014-03-04 "0.000000000"
## 2014-03-05 "0.000000000"
## 2014-03-06 "0.000000000"
```

```
tail(mars)
```

```
##            jour       affaires moyenne     ecart.type  St
## 2014-03-26 "mercredi" " 696"   " 788.1429" "182.18802" "-1.305756937"
## 2014-03-27 "jeudi"    " 677"   " 783.5714" "185.05932" "-1.375877118"
## 2014-03-28 "vendredi" " 649"   " 790.1429" "180.28073" "-1.582905940"
## 2014-03-29 "samedi"   " 981"   " 792.2857" "178.58491" " 0.256720200"
## 2014-03-30 "dimanche" " 995"   " 798.5714" "171.52926" " 0.345160749"
## 2014-03-31 "lundi"    " 842"   " 781.4286" "139.22028" "-0.364923806"
##            Cusum
## 2014-03-26 "0.000000000"
## 2014-03-27 "0.000000000"
## 2014-03-28 "0.000000000"
## 2014-03-29 "0.256720200"
## 2014-03-30 "0.601880948"
## 2014-03-31 "0.236957143"
```

```
mars <- a['/2014-02-15']
```

1.6.0.2 Toutes les données depuis le début jusqu'au 15 février

1.7 Format de date

Décrit au paragraphe: ?strptime

1.7.0.3 N° de la semaine française (de 00 à 53), symbole **%V** est défini par la norme ISO 8601. La semaine commence le lundi. Si la semaine contenant le 1er janvier comporte 4 jours ou plus dans la nouvelle année, alors elle est considérée comme la semaine 1. Sinon il s'agit de la dernière semaine de l'année précédente et la semaine suivante est la semaine 1.

1.7.0.4 N° de la semaine US (de 00 à 53), symbole %U commence le *dimanche*.

1.8 Compléments

Pour fournir certains résultats, XTs a besoin d'une bibliothèque complmentaire **xtsExtra** qui ne fait pas partie de cran et qui se charge à l'adresse:

```
install.packages("xtsExtra", repos="http://R-Forge.R-project.org")
```

Une fois téléchargé, on trouve le dossier dans */home/jcb/R/i686-pc-linux-gnu-library/3.1*. Le dossier *Doc* contient un pdf consacré à plot.xts.

Cette extension toujours activement dévelopée est née pendant le Google Summer Camp en 2012. Les auteurs ont pris grand soins de rester parfaitement compatible avec la version XTS de base.

1.9 barplot.xts

Nécessite xtsExtra.

1.10 Divers

Exploration de diverses fonctions de date:

- month.abb: constantes listant les noms de mois abbrégés
- month.names: noms des mois en anglais et méthode de francisation via la fonction format qui utilise la variable **locale**

Norme ISO pour les semaines. La première semaine de l'année correspond à celle contenant le 1er jeudi de l'année civile cad la semaine du **4 janvier**.

Voir aussi le programme IsoWeek

```
nth <- paste0(1:12, c("st", "nd", rep.int("th", 10)))
nth
```

```
##  [1] "1st"  "2nd"  "3th"  "4th"  "5th"  "6th"  "7th"  "8th"  "9th"  "10th"
## [11] "11th" "12th"
```

```
month.abb
```

```
##  [1] "Jan" "Feb" "Mar" "Apr" "May" "Jun" "Jul" "Aug" "Sep" "Oct" "Nov"
## [12] "Dec"
```

```
format(ISOdate(2000, 1:12, 1), "%B")
```

```
##  [1] "janvier"   "février"   "mars"      "avril"     "mai"
##  [6] "juin"      "juillet"   "août"      "septembre" "octobre"
## [11] "novembre"  "décembre"
```

```r
month.name
```

```
##  [1] "January"   "February"  "March"     "April"     "May"
##  [6] "June"      "July"      "August"    "September" "October"
## [11] "November"  "December"
```

```r
month.name <- format(ISOdate(2000, 1:12, 1), "%b")
month.name
```

```
##  [1] "janv." "févr." "mars"  "avril" "mai"   "juin"  "juil." "août"
##  [9] "sept." "oct."  "nov."  "déc."
```

```r
month.name <- format(ISOdate(2000, 1:12, 1), "%B")
month.name
```

```
##  [1] "janvier"   "février"   "mars"      "avril"     "mai"
##  [6] "juin"      "juillet"   "août"      "septembre" "octobre"
## [11] "novembre"  "décembre"
```

```r
format(Sys.time(), "%a %d %b %Y %X %Z")
```

```
## [1] "dim. 01 févr. 2015 18:51:46 CET"
```

```r
# ISOdate(année, mois, jours)
format(ISOdate(2000, 1, 1:7), "%a")
```

```
## [1] "sam." "dim." "lun." "mar." "mer." "jeu." "ven."
```

```r
format(ISOdate(2000, 1, 1:7), "%A")
```

```
## [1] "samedi"   "dimanche" "lundi"    "mardi"    "mercredi" "jeudi"
## [7] "vendredi"
```

```r
# format utilise les indications LOCALE pour convertir les données date-time en chaine de caractères. P
format(as.Date("2015-01-01"), "%a")
```

```
## [1] "jeu."
```

```r
format(as.Date("2015-01-01"), "%A")
```

```
## [1] "jeudi"
```

```r
format(as.Date("2015-01-01"), "%b")
```

```
## [1] "janv."
```

```r
format(as.Date("2015-01-01"), "%B")
```

```
## [1] "janvier"
```

```r
format(as.Date("2015-01-01"), "%A %B")
```

```
## [1] "jeudi janvier"
```

```r
format(as.Date("2015-01-01"), "%e")
```

```
## [1] " 1"
```

```r
format(as.Date("2015-01-01"), "%c")
```

```
## [1] "jeu. 01 janv. 2015 00:00:00 CET"
```

```r
format(as.Date("2015-01-01"), "%C")
```

```
## [1] "20"
```

```r
format(as.Date("2015-01-01"), "%d")
```

```
## [1] "01"
```

```r
format(as.Date("2015-01-01"), "%D")
```

```
## [1] "01/01/15"
```

```r
format(as.Date("2015-01-01"), "%e")
```

```
## [1] " 1"
```

```r
format(as.Date("2015-01-01"), "%E")
```

```
## [1] "%E"
```

```r
format(as.Date("2015-01-01"), "%F")
```

```
## [1] "2015-01-01"
```

```r
format(as.Date("2015-01-01"), "%g")
```

```
## [1] "15"
```

```r
format(as.Date("2015-01-01"), "%G")
```

```
## [1] "2015"
```

```r
format(as.Date("2015-01-01"), "%h")
```

```
## [1] "janv."
```

```r
format(as.Date("2015-01-01"), "%H")
```

```
## [1] "00"
```

```r
format(as.Date("2015-01-01"), "%I")
```

```
## [1] "12"
```

```r
format(as.Date("2015-01-01"), "%j")
```

```
## [1] "001"
```

```r
format(as.Date("2015-01-01"), "%m")
```

```
## [1] "01"
```

```r
format(as.Date("2015-01-01"), "%M")
```

```
## [1] "00"
```

```r
format(as.Date("2015-01-01"), "%n")
```

```
## [1] "\n"
```

```r
format(as.Date("2015-01-01"), "%p")
```

```
## [1] ""
```

```r
format(as.Date("2015-01-01"), "%r")
```

```
## [1] "12:00:00 "
```

```r
format(as.Date("2015-01-01"), "%R")
```

```
## [1] "00:00"
```

```r
format(as.Date("2015-01-01"), "%S")
```

```
## [1] "00"
```

```r
format(as.Date("2015-01-01"), "%t")
```

```
## [1] "\t"
```

```r
format(as.Date("2015-01-01"), "%T")
```

```
## [1] "00:00:00"
```

```r
# Weekday as a decimal number (1-7, Monday is 1).
format(as.Date("2015-01-01"), "%u")
```

```
## [1] "4"
```

```r
# Week of the year as decimal number (00-53) using Sunday as the first day 1 of the week (and typically
format(as.Date("2015-01-01"), "%U")
```

```
## [1] "00"
```

```r
format(as.Date("2015-01-01"), "%V")
```

```
## [1] "01"
```

```r
format(as.Date("2015-01-01"), "%w")
```

```
## [1] "4"
```

```r
format(as.Date("2015-01-01"), "%W")
```

```
## [1] "00"
```

```r
format(as.Date("2015-01-01"), "%x")
```

```
## [1] "01/01/2015"
```

```r
format(as.Date("2015-01-01"), "%X")
```

```
## [1] "00:00:00"
```

```r
format(as.Date("2015-01-01"), "%y")
```

```
## [1] "15"
```

```r
format(as.Date("2015-01-01"), "%Y")
```

```
## [1] "2015"
```

```r
format(as.Date("2015-01-01"), "%z")
```

```
## [1] "+0000"
```

```r
format(as.Date("2015-01-01"), "%Z")
```

```
## [1] "UTC"
```

```r
format(as.Date("2015-01-01"), "%k")
```

```
## [1] " 0"
```

```r
format(as.Date("2015-01-01"), "%l")
```

```
## [1] "12"
```

```r
format(as.Date("2015-01-01"), "%s")
```

```
## [1] "1420066800"
```

```r
format(as.Date("2015-01-01"), "%+")
```

```
## [1] "%+"
```

```r
format(as.Date("2015-01-01"), "%O")
```

```
## [1] "%O"
```

```r
format(as.Date("2015-01-01"), "%E")
```

```
## [1] "%E"
```

```r
format(as.Date("2015-01-01"), "%OS")
```

```
## [1] "00"
```

```r
format(Sys.time(), "%H:%M:%OS3")
```

```
## [1] "18:51:46.091"
```

```
Sys.getlocale("LC_TIME")
```

```
## [1] "fr_FR.UTF-8"
```

1.11 Nombre de jours par mois

Pb: déterminer le nombre de jours par mois sur une période donnée.

source: Number of days in each month Gabor Grothendieck ggrothendieck at gmail.com

```
date1 <- as.Date("2014-01-01")
date2 <- as.Date("2015-01-01")
jour.mois <- ts(diff(seq(date1, date2, by="month")), start = c(2014, 01), freq = 12)
jour.mois
```

```
##      Jan Feb Mar Apr May Jun Jul Aug Sep Oct Nov Dec
## 2014  31  28  31  30  31  30  31  31  30  31  30  31
```

```
as.numeric(jour.mois)
```

```
##  [1] 31 28 31 30 31 30 31 31 30 31 30 31
```

1.12 Appliquer une fonction sur une période calendaire

Le package xts fournit des fonctions pour traiter les séries temporelles par jour, semaine, mois, trimestre ou années:

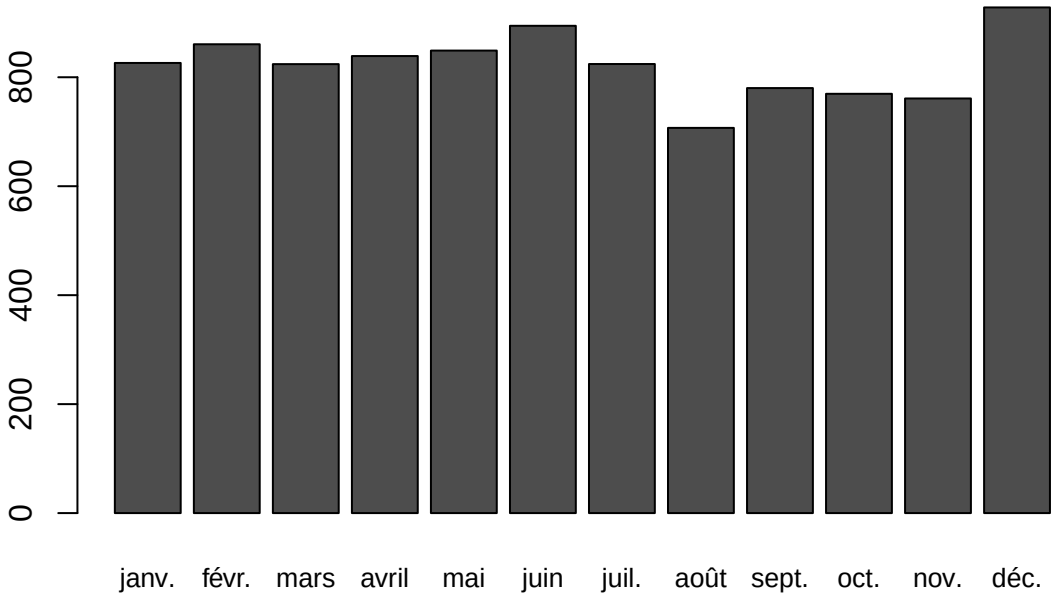
```
apply.daily(ts, f)
apply.weekly(ts, f)
apply.monthly(ts, f)m <- apply.monthly(a[,3], mean)

apply.quarterly(ts, f)
apply.yearly(ts, f)
```

Où *ts* est une série temporelle de type xts et *f* est une fonction à appliquer à la période. Si ts est un objet de type,*zoo* il faut d'abord le convertir en *xts*.

```
apply.monthly(as.xts(ts), f)
```

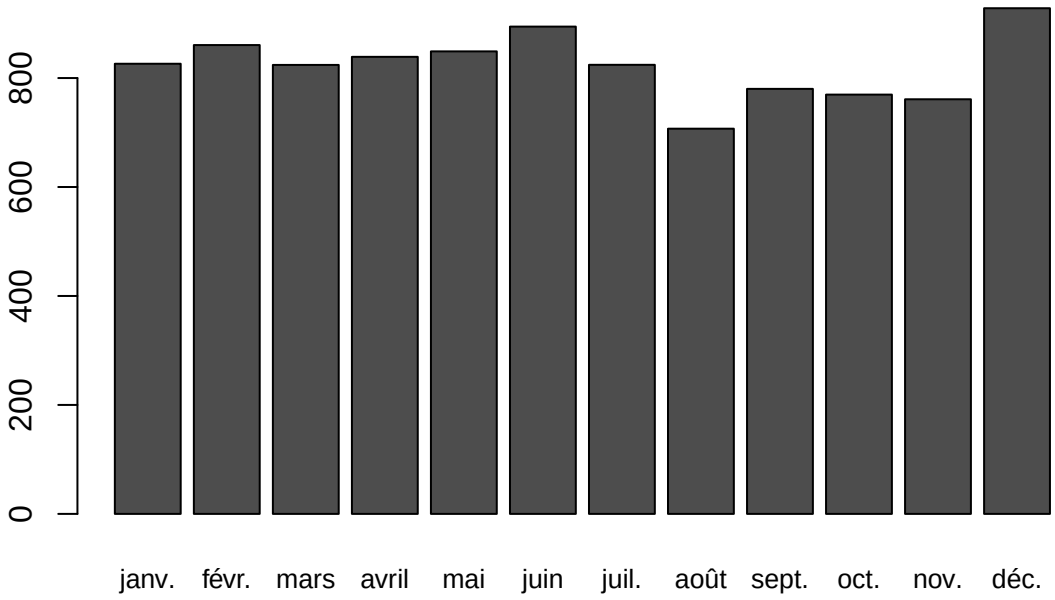
exemple: nombre moyen d'affaires par mois

```
a <- xts(d67, order.by = d67$date)
mean.affaires.mois <- apply.monthly(a$affaires, mean)
barplot(mean.affaires.mois, names.arg = format(ISOdate(2014, 1:12, 1), "%b"), main = "SAMU 67 - Nombre m
```



méthode alternative:

```
a <- xts(d67, d67$date) # écriture plus simple
mean.affaires.mois <- apply.monthly(a$affaires, mean) # idem
# on utilise index pour retrouver le nom des colonnes
barplot(mean.affaires.mois, names.arg = format(index(mean.affaires.mois), "%b"), main = "SAMU 67 - Nombi
```

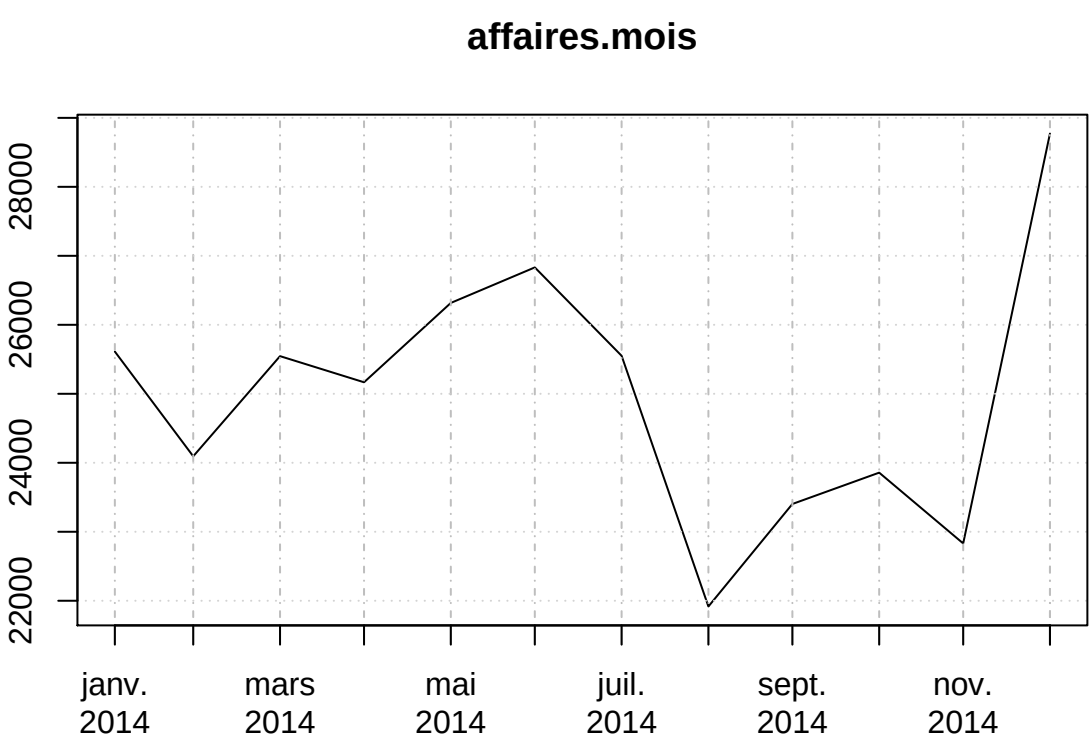


La fonction **sum** ne fonctionne pas ? On peut contourner le problème en créant une fonction *somme* :

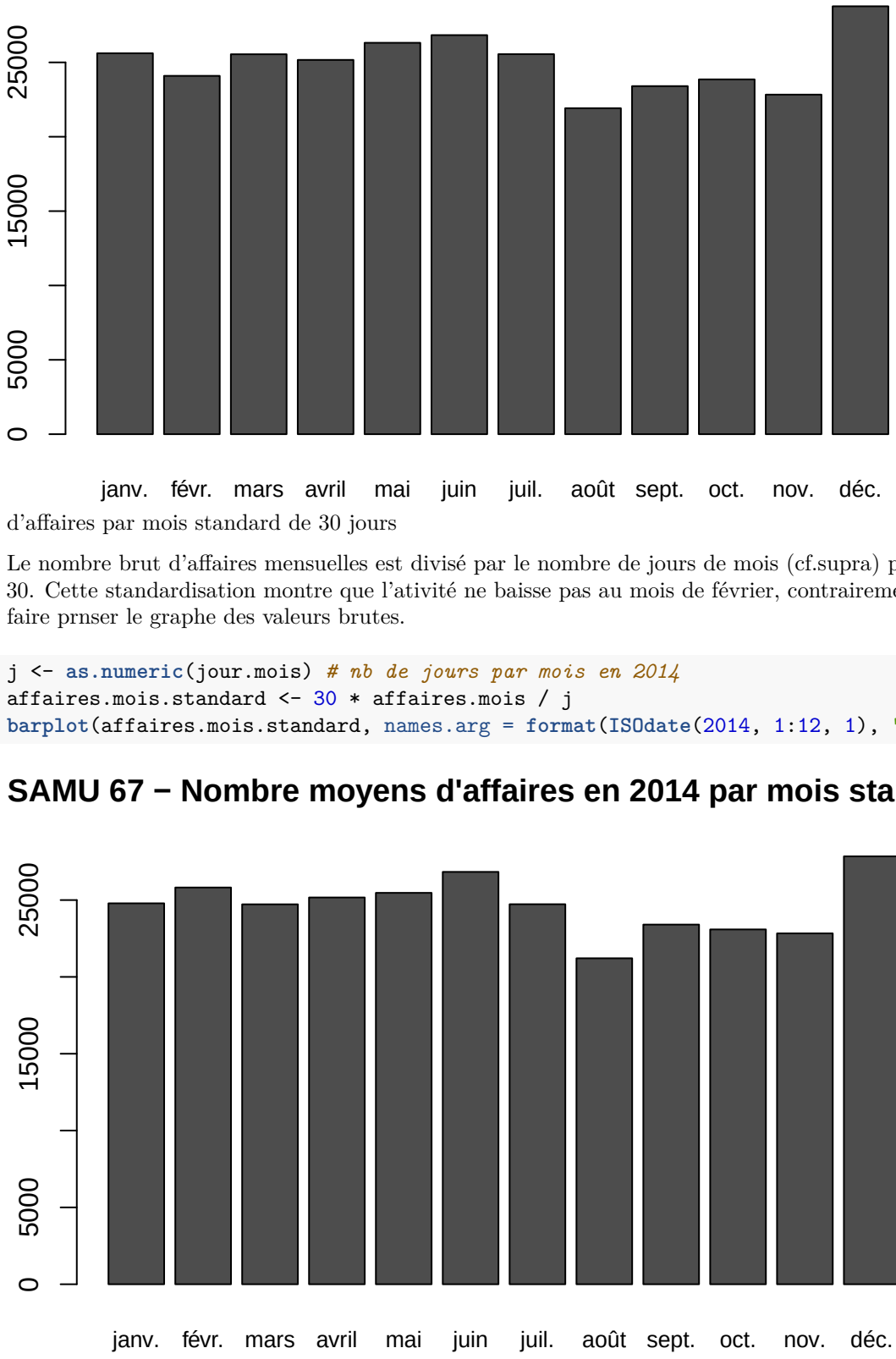
```
somme <- function(x){sum(as.numeric(x))}
affaires.mois <- apply.monthly(a$affaires, somme)
affaires.mois
```

```
##            affaires
## 2014-01-31    25612
## 2014-02-28    24094
## 2014-03-31    25546
## 2014-04-30    25167
## 2014-05-31    26317
## 2014-06-30    26832
## 2014-07-31    25551
## 2014-08-31    21917
## 2014-09-30    23403
## 2014-10-31    23857
## 2014-11-30    22831
## 2014-12-31    28772
```

```
plot(affaires.mois)
```




```
barplot(affaires.mois, names.arg = format(ISOdate(2014, 1:12, 1), "%b"), main = "SAMU 67 - Nombre moyens
```



exemple: nombre d'affaires par mois standard de 30 jours

Le nombre brut d'affaires mensuelles est divisé par le nombre de jours de mois (cf.supra) puis multiplié par 30. Cette standardisation montre que l'ativité ne baisse pas au mois de février, contrairement à ce que peut faire prnser le graphe des valeurs brutes.

```
j <- as.numeric(jour.mois) # nb de jours par mois en 2014
affaires.mois.standard <- 30 * affaires.mois / j
barplot(affaires.mois.standard, names.arg = format(ISOdate(2014, 1:12, 1), "%b"), main = "SAMU 67 - Nomb
```

2 Xts et Dygraphs

“‘{ xts_dygraphs}

library(dygraphs) path <- “../../RPU_2014/Analyse/Activite_SAMU/” # pour lancer depuis la console file <- “samu67_2014.csv” d67 <- read.csv(paste0(path,file)) d67$date <- as.Date(d67$date, “%d/%m/%Y”) a <- xts(d67, order.by = d67$date) d <- dygraph(a[,3:4], main = “SAMU 67 - 2014”, ylab = “Affaires”) d

“‘