Graphique de série temporelle

Astrid Deschênes

2015-04-13

Table des matières

L'objectif de cette fiche est d'expliquer, à l'aide d'un jeu de données, comment utiliser les librairies R afin de générer des graphiques de séries temporelles. L'utilisation de fonctions graphiques de base en R sera comparée à l'utilisation de fonctions provenant du package ggplot2.

Définition d'une série temporelle

Une série temporelle (aussi appelée série chronologique) est une suite d'observations chiffrées d'un même phénomène, ordonnées dans le temps. Ce type de série permet de décrire l'évolution d'un phénomène au cours du temps. Dans certaines situations, elle permet même d'expliquer, ainsi que de prévoir, ce phénomène à court ou à long terme. L'économie, la météorologie ainsi que l'épidémiologie sont quelques exemples de domaines où les séries temporelles sont souvent utilisées.

Exemple d'une série temporelle

À titre d'exemple, le jeu de données provenant d'un système de location de vélos dans la ville de Washington D.C., aux États-Unis, sera utilisé. Ce jeu de données contient le nombre de locations de vélos par heure pour les années 2011 et 2012. De plus amples informations concernant ce jeu de données sont disponibles sur le site web Bike Sharing Dataset Data Set.

Le jeu de données initial est chargé à partir du fichier *hour.csv* disponible sur le site web mentionné plus haut. Voici un aperçu d'une sous-section des données :

```
# Charger le package permettant l'affichage des données
library(knitr)
# Charger le fichier hour.csv qui contient les données
hour <- read.csv(file = "hour.csv", sep = ",", header = TRUE)
```

```r
# Afficher une sous-section des données
kable(hour[1:5,c("dteday", "season", "yr", "hr", "weathersit", "cnt")],
      caption = "Sous-section du jeu de données initial")
```

Table 1: Sous-section du jeu de données initial

dteday	season	yr	hr	weathersit	cnt
2011-01-01	1	0	0	1	16
2011-01-01	1	0	1	1	40
2011-01-01	1	0	2	1	32
2011-01-01	1	0	3	1	13
2011-01-01	1	0	4	1	1

Ce jeu de données contient 17379 observations et 17 variables. Voici une brève description des variables utilisées dans les exemples qui suivront :

- `dteday` : un facteur représentant la date de la prise de données
- `season` : un entier représentant la saison (1 : printemps, 2 : été, 3 : automne, 4 : hiver)
- `yr` : un entier représentant l'année (0 : 2011, 1 : 2012)
- `hr` : un entier représentant l'heure (entre 0 et 23)
- `weathersit` : un entier représentant les conditions atmosphériques au moment de la prise de données
  - 1 : Ciel clair, Quelques nuages, Partiellement nuageux
  - 2 : Brume + Nuageux, Brume + Nuages fragmentés, Brume + Quelques nuages, Brume
  - 3 : Faible neige, Faible pluie + Orage + Nuages épars, Faible pluie + Nuages épars
  - 4 : Forte pluie + Grésil + Orage + Brume, Neige + Brouillard
- `cnt` : un entier représentant le nombre total de vélos loués

Les données utilisées dans les exemples de cette fiche sont le résultat de l'agrégation du jeu de données initial. Cette agrégation, effectuée à l'aide du package dplyr permet d'obtenir la moyenne et son erreur type pour chaque heure du jour (variable `hr`) en fonction de l'année (variable `yr`) et des conditions atmosphériques (variable `weathersit`). Les conditions atmosphériques extrêmes (`weathersit` = 4) sont éliminées du jeu de données final.

```r
# Charger le package permettant l'agrégation des données
library(dplyr)
# Grouper les données en fonction de l'heure, des conditions atmosphériques et de l'année
hour_group <- group_by(.data = hour, hr, weathersit, yr)
# Calculer la moyenne ainsi que l'écart-type
hour_agg <- summarise(.data = hour_group, mean = mean(cnt, na.rm = TRUE), # moyenne
              n = sum(!is.na(cnt)),se = sd(cnt)/sqrt(n)) # se = erreur type de la moyenne
hour_agg <- subset(hour_agg,
                   weathersit != 4) # pour enlever les lignes avec weathersit == 4
# Afficher les premières lignes du jeu de données agrégées
kable(hour_agg[1:5,], caption = "Jeu de données agrégées")
```

Table 2: Jeu de données agrégées

hr	weathersit	yr	mean	n	se
0	1	0	47.44400	250	2.241871
0	1	1	71.41841	239	3.074518
0	2	0	36.33721	86	3.110136
0	2	1	56.69697	99	4.495364

hr	weathersit	yr	mean	n	se
0	3	0	22.16000	25	4.528311

Ce jeu de données agrégées contient 144 observations et 6 variables.

Graphique de série temporelle

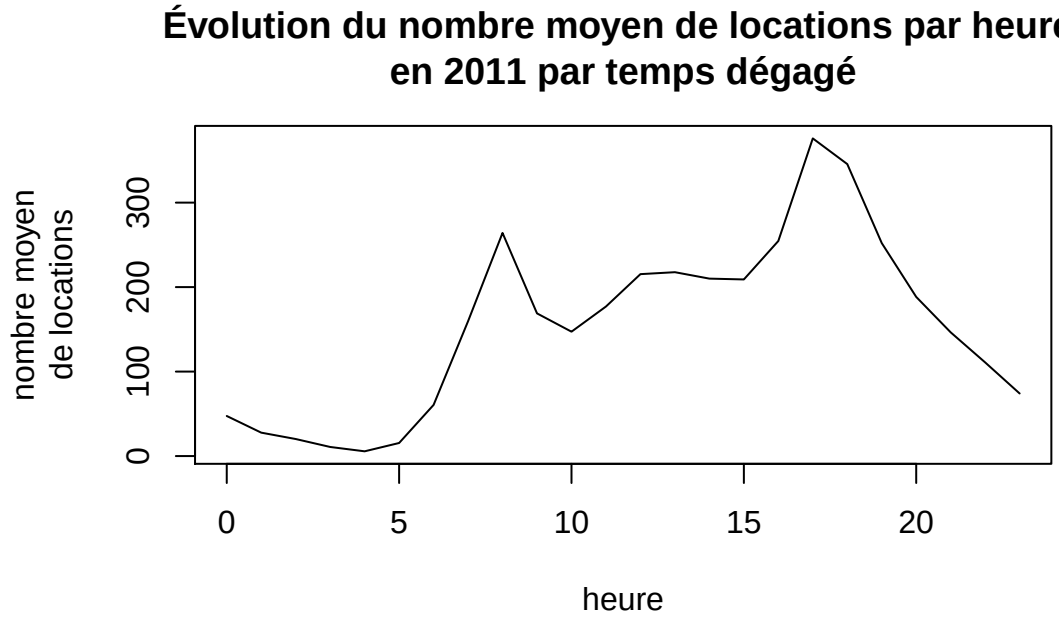
Graphique généré en utilisant les fonctions graphiques de base

La librairie `stats` offre des fonctionnalités de base permettant de manipuler les séries temporelles. La fonction `ts()` permet de convertir un vecteur numérique (objet `vector`) en objet `time-series`. Les objets `time-series` ont l'avantage d'être reconnus par la fonction graphique `plot()` et d'ainsi être affichés graphiquement dans un format standard aux séries temporelles.

Les paramètres de la fonction `ts(data, start=, end=, frequency=)` sont, dans l'ordre, l'objet contenant les données temporelles de la série (un object `vector` ou `matrix`), le temps de la première observation, le temps de la dernière observation ainsi que le nombre de fréquences par unités de temps.

Dans l'exemple qui suit, la librairie de base `stats` sera utilisée pour générer un graphique montrant l'évolution du nombre moyen de locations par heure en 2011 par temps dégagé. La fonction `ts` est donc utilisée sur le jeu de données agrégé en s'assurant de n'utiliser que les données liées à l'année 2011 (`yr` == 0) et qu'au temps dégagé (`weathersit` == 1). L'objet `time-series` est passé en paramètre à la fonction `plot()` dont les paramètres par défaut assurent un affichage standard pour les séries temporelles. Seuls les paramètres des axes (paramètres `xlab` et `ylab`) ainsi que du titre (paramètre `main`) ont besoin d'être personnalisés.

```
# Créer un objet time-series qui utilise uniquement les données de l'année
# 2011 (yr == 0) en temps dégagé (weathersit == 1)
# La variable d'intérêt est le nombre moyen de locations par heure (mean)
time_serie <- ts(hour_agg[hour_agg$weathersit==1 & hour_agg$yr==0, c("mean")],
                 start = 0, frequency = 1)
# Afficher le graphique en utilisant la fonction de base plot()
# en personnalisant le titre (main), l'axe des y (ylab) ainsi que l'axe des x (xlab)
plot(time_serie, ylab="nombre moyen\nde locations", xlab="heure",
     main="Évolution du nombre moyen de locations par heure \nen 2011 par temps dégagé")
```



Graphique généré en utilisant les fonctions du package `ggplot2`

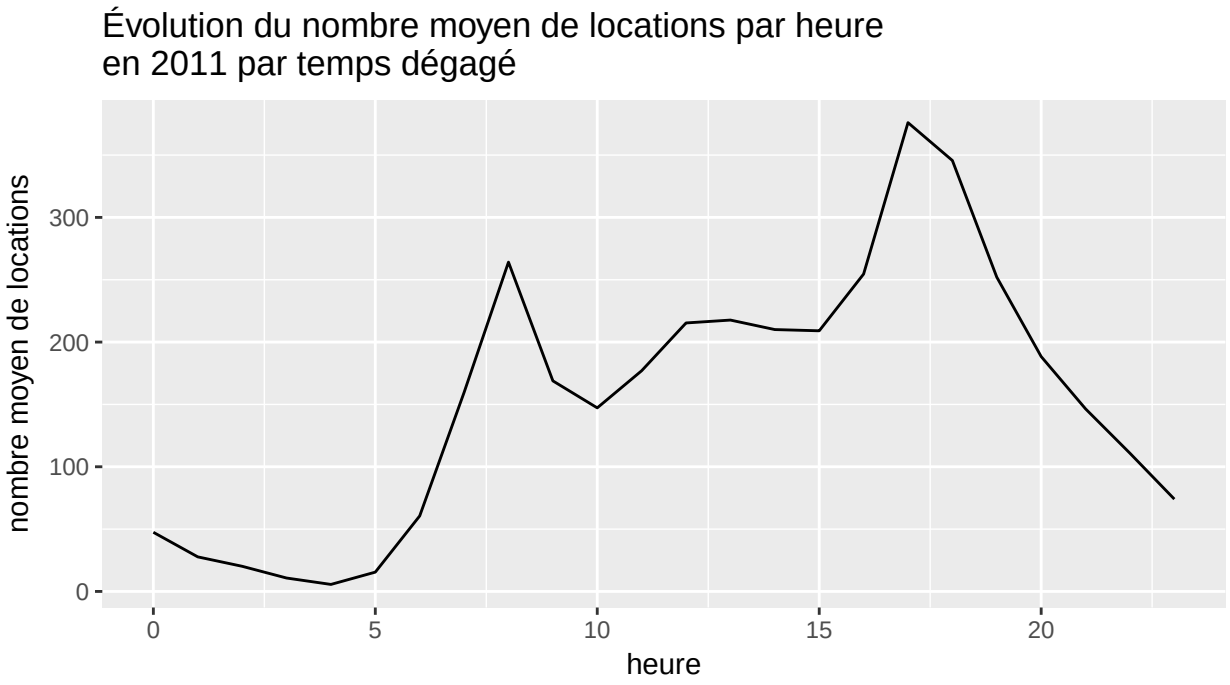
Le package ggplot2 est un package élaboré par Hadley Wickham pour la conception des graphiques avancés qui utilise une "grammaire graphique" spécifique. Cette grammaire est formée d'un ensemble de composants indépendants qui peuvent être assemblés d'une multitude de façons, d'où la grande flexibilité du package.

Le package `ggplot2` ne fait pas partie des packages de base, il doit donc être chargé afin de pouvoir l'utiliser.

```
# Charger le package ggplot2
library(ggplot2)
```

Dans le package `ggplot2`, la fonction `qplot()` est la fonction graphique qui permet de créer des graphiques de séries temporelles ressemblant à ceux générés à l'aide des packages de base. Les paramètres importants de la fonction `qplot(x=, y=, data=, geom=)` sont, dans l'ordre, le nom de la colonne du `data.frame` qui contient les valeurs des `x`, le nom de la colonne du `data.frame` qui contient les valeurs des `y`, l'objet de type `data.frame` qui contient les valeurs `x` et `y`, un vecteur contenant le ou les noms des types de graphiques qui seront produits. Dans le cas des séries temporelles, le paramètre `geom` prend la valeur de `line`, ce qui indique que les observations doivent être connectées par une ligne, selon l'ordre des `x`. Les fonctions `ggtitle`, `xlab` et `ylab` sont des fonctions qui viennent se greffer à la fonction `qplot()` et qui permettent la paramétrisation du titre, de nom de l'axe des `x` ainsi que du nom de l'axe des `y`.

```
# Créer un graphique de série temporelle qui utilise uniquement les données de
# l'année 2011 (hour_agg$yr==0) en temps dégagé (hour_agg$weathersit==1)
# La variable d'intérêt est le nombre moyen de locations par heure (mean)
# L'axe des x représente l'heure de la journée (hr)
# Les observations doivent être connectées par une ligne (geom = "line")
qplot(hr, mean, data = hour_agg[hour_agg$weathersit==1 & hour_agg$yr==0,],
  geom = "line") +
  ggtitle("Évolution du nombre moyen de locations par heure \nen 2011 par temps dégagé") +
  xlab("heure") + ylab("nombre moyen de locations")  # nom des axes
```



Graphique généré en utilisant les fonctions avancées du package `ggplot2`

Le package ggplot2 permet aussi de générer des graphiques beaucoup plus complexes que ceux de base pour les séries temporelles. Comme exemple, un graphique contenant un sous-panneau par année et une courbe par type de conditions atmosphériques sera créé à partir du même jeu de données agrégées.

Cependant, afin de permettre l'affichage du nom des années dans chacun des sous-panneaux qui seront créés, il est nécessaire d'ajouter, au jeu de données agrégées, une colonne contenant le nom de l'année tel que nous désirons qu'il apparaisse sur le panneau.

```
# Ajouter une colonne qui contient le nom de l'année au jeu de données agrégées
hour_agg$year<-factor(hour_agg$yr, labels=c("2011", "2012"))
# Afficher les premières lignes
kable(hour_agg[1:5,],
      caption = "Jeu de données agrégées contenant une nouvelle colonne")
```

Table 3: Jeu de données agrégées contenant une nouvelle colonne

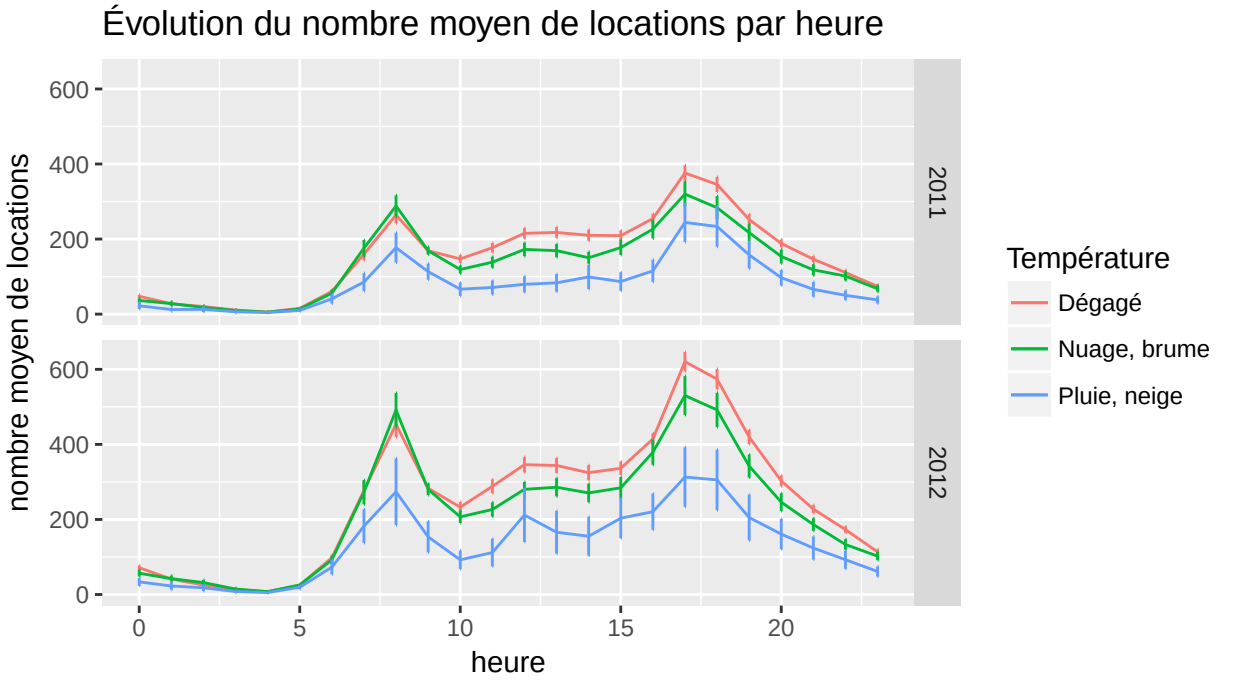
hr	weathersit	yr	mean	n	se	year
0	1	0	47.44400	250	2.241871	2011
0	1	1	71.41841	239	3.074518	2012
0	2	0	36.33721	86	3.110136	2011
0	2	1	56.69697	99	4.495364	2012
0	3	0	22.16000	25	4.528311	2011

Comme pour les graphiques plus simples, c'est la fonction `qplot()` qui est utilisée pour générer les graphiques complexes. La complexité est ajoutée par l'utilisation de paramètres ainsi que par la greffe de fonctions plus avancées. Pour l'exemple qui suit, les paramètres utilisés de la fonction `qplot(x=, y=, data=, geom=, color=)` sont identiques à l'exemple précédent à l'exception du dernier paramètre qui représente la variable

dont les niveaux seront associés à une ligne de couleur différente sur le graphique. Le graphique généré contiendra donc autant de lignes que de niveaux présents dans la variable.

Trois nouvelles fonctions viennent se greffer à l'ensemble : `facet_grid()`, `geom_errorbar()` et `scale_colour_hue()`. La première fonction `facet_grid()` permet de partitionner un graphique en une matrice de panneaux selon la variable passée en argument et de personnaliser les noms donnés à chacun des panneaux. La fonction `geom_errorbar()` permet d'ajouter une barre d'erreur à chacune des observations du graphique. Les valeurs maximales et minimales que prennent ces barres sont assignées par les paramètres `ymax` et `ymin`. La fonction `scale_colour_hue()` permet de personnaliser la légende ainsi que les couleurs des niveaux présents dans le graphique.

```
# Créer un graphique de série temporelle contenant 1 panneau pour chaque année
# (facet_grid(yr ~ .)) et une courbe de couleur différente par type de condition
# atmosphérique (colour=as.character(weathersit))
# La variable d'intérêt est le nombre moyen de locations par heure (mean)
# L'axe des x représente l'heure de la journée (hr)
# Les observations sont connectées par une ligne (geom = "line")
qplot(hr, mean, data = hour_agg, geom = "line", colour=as.character(weathersit)) +
    facet_grid(year ~ .,              # génère 1 panneau par année avec nom exact
        labeller = label_parsed ) +  # de l'année comme identifiant
    geom_errorbar(aes(ymin=mean - 1.96*se, ymax=mean + 1.96*se),
        width=0) + # ajoute une barre d'erreur à chaque observation
    scale_colour_hue("Température",  # légende avec description personnalisée des niveaux
        labels = c("Dégagé", "Nuage, brume", "Pluie, neige")) +
    ggtitle('Évolution du nombre moyen de locations par heure') +  # titre
    xlab("heure") + ylab("nombre moyen de locations")   # nom des axes
```



Conclusion

Les séries temporelles sont utilisées dans plusieurs domaines scientifiques. En R, l'utilisation de l'objet `times-series`, par le biais de la fonction `ts()`, permet de faire afficher les graphiques de séries tempo-

relles selon un format standard prédéfini en utilisant la fonction `plot()`. La fonction `plot()` est, en effet, polymorphique, c'est-à-dire que son comportement est spécifique à l'objet passé en paramètre à celle-ci.

Le package ggplot2, élaboré par Hadley Wickham, permet lui aussi de générer des graphiques de séries temporelles par le biais de la fonction `qplot()`. De par sa flexibilité, ce package permet de recréer des graphiques ressemblant à ceux générés par les packages de base mais aussi de créer des graphiques beaucoup plus complexes.

Il existe donc une diversité d'options permettant de générer les graphiques de séries temporelles. Libre à vous d'utiliser celles qui correspondent le mieux à vos besoins.

Références

Références sur les graphiques de séries temporelles en R

R-Bloggers : Ce que je sais sur les séries temporelles

Références sur le package `ggplot2` et ses fonctions

Wickham, H. ggplot2 : elegant graphics for data analysis. Springer New York, 2009.

ggplot2 - Diviser un graphique en plusieurs panneaux avec facet_grid()

ggplot2 - Utilisation de qplot() (en anglais)

ggplot2 - Utilisation de la fonction scale_colour_hue() (en anglais)

Références sur les packages utilisés

Wickham, H et Francois, R (2015). dplyr : A Grammar of Data Manipulation. R package version 0.4.1. http://CRAN.R-project.org/package=dplyr

Yihui, X. (2015). knitr : A General-Purpose Package for Dynamic Report Generation in R. R package version 1.9.

Yihui, X. (2014). knitr : A Comprehensive Tool for Reproducible Research in R. In Victoria Stodden, Friedrich Leisch and Roger D. Peng, editors, Implementing Reproducible Computational Research. Chapman and Hall/CRC. ISBN 978-1466561595

Yihui, X. (2013). Dynamic Documents with R and knitr. Chapman and Hall/CRC. ISBN 978-1482203530

Références sur le jeu de données

Fanaee-T, Hadi, and Gama, Joao (2013). Event labeling combining ensemble detectors and background knowledge. Progress in Artificial Intelligence (2013) : pp. 1-15, Springer Berlin Heidelberg, doi:10.1007/s13748-013-0040-3.