Utiliser l'environnement R pour créer ses visualisations de données issues d'Analytics (1ère partie)

Par Guillaume et Sylvain Peyronnet

Domaine :	Recherche	Référencement
Niveau :	Pour tous	Avancé

Connaissez-vous R ? Il s'agit d'un langage de programmation et un environnement de travail largement utilisé lorsqu'il s'agit d'analyser des données statistiques issues de sources diverses. Dans cet ensemble de deux articles, nous allons vous présenter cet outil et son utilisation pour proposer des graphiques et tableaux de bord plus ou moins simples ou sophistiqués en partant des données extraites de Google Analytics. Apprenons, ce mois-ci, à récupérer les informations d'Analytics et à créer un graphique simple affichant le nombre de visites sur un site web.

Google Analytics est probablement l'un des outils de récolte de données de sites web le plus utilisé au monde. Son interface web est sympathique et ergonomique, mais pas toujours très pratique, notamment pour faire des représentations réellement adaptées au besoin de son entreprise ou de son activité. Mais si l'interface web n'est pas toujours telle qu'on la désirerait, les données existent dans l'outil, et il serait dommage de s'en passer uniquement parce que les interfaces ne sont pas « au top ». Ce mois-ci nous allons commencer un diptyque sur l'utilisation de l'outil R pour créer des visualisations spécifiques à partir des données issues de Google Analytics.

Qu'est-ce que R ?

R (disponible sur le site [1]) est un langage de programmation et un environnement de travail utilisé pour un très grand nombre de tâches dont des analyses statistiques de données, ou encore la création de visualisations graphiques. R est open source et a été bâti autour d'un langage appelé S, developpé aux Bell Labs par John Chambers dans les années 70 (voir [2]). A l'origine de ce langage, on retrouve Ross Ihaka et Robert Gentleman, mais il existe aujourd'hui une communauté particulièrement dynamique qui contribue au développement de l'outil.

Pour installer R, rien de compliqué, il suffit de se rendre sur le site [1] et de télécharger l'archive qui convient à votre système d'exploitation. Vous pouvez également, si vous le souhaitez, utiliser une interface graphique plus fournie que celle de base en téléchargeant Rstudio qui est gratuit dans sa version open source sans support. Pour se servir de R, il existe deux modes de fonctionnement : en ligne de commande (mode interactif) ou en écrivant ces commandes dans des fichiers qui seront interprétés ensuite (scripts). Tout au

long de cet article, nous utiliserons le mode interactif, qui permet de saisir des commandes de manière simple, et de procéder par étapes tout en contrôlant le résultat en temps réel.

R est un système très riche qui possède de nombreuses fonctionnalités, notamment grâce à son système de packages. Un package est tout simplement une bibliothèque de fonctions utiles, créees par l'équipe de développement de R ou par des tiers. Pour installer un package, on peut passer *via* l'interface graphique ou *via* la commande install.packages. Vous pourrez ensuite utiliser le package en le chargeant :

```
> # On charge le package ggplot2
> library(ggplot2)
```

Au fil de nos deux articles sur le sujet, nous présenterons un certains nombre de packages, complétés de plus ou moins d'explications, mais Il existe une multitude de livres introduisant le langage R, et nous aiguillons le lecteur intéressé vers le site [4] qui présente des ressources gratuites et payantes.

Obtenir les données depuis Google Analytics

La première étape avant de pouvoir réaliser de belles visualisations, est d'obtenir les données pour les créer. Pour cela, il existe une procédure un peu laborieuse à suivre, que nous décrivons maintenant.

Vous devez d'abord obtenir des accès pour l'API de Google Analytics, pour cela, direction le site [5]. A partir de là, créez un nouveau projet, puis activez l'API Analytics pour ce projet. Il vous faut enfin vous diriger vers le gestionnaire d'API, rubrique identifiants, pour obtenir ces identifiants vous permettant de requêter les données.

Une fois que ceci est fait, vous pouvez lancer R, pour commencer le travail sérieux.
La première opération est d'installer et de charger le package pour gérer Google Analytics.

```
> # On charge le package RGoogleAnalytics
> library(RGoogleAnalytics)
```

Ensuite, le plus simple est de préparer des variables génériques pour stocker vos identifiants.

```
> # Gestion du token de connexion
> client.id <- "XXXXXXXXXXX.apps.googleusercontent.com"
> client.secret <- "XXXXXXXXXXXXXXXXX"
> token <- Auth(client.id,client.secret)
```

Une fois cette opération réalisée, il va falloir autoriser R a accéder à votre compte Google Analytics, pour cela il suffit d'utiliser la commande suivante :

```
> # On active via le navigateur
> ga <- RGoogleAnalytics()
```

Avec cette commande, une fenêtre va s'ouvrir dans votre navigateur pour vous demander une confirmation. Après acceptation, tout est prêt, vous pourrez accéder à vos données Google Analytics.

Il faut savoir que la connexion n'est pas maintenue *ad vitam aeternam*, il faudra donc périodiquement la renouveller. Pour cela le mieux est de sauver le token de connexion dans un fichier, et de le charger et valider à nouveau quand nécessaire.

```
> # Sauvergarde du token de connexion dans un fichier
> save(token,file="./mon_token_de_connexion")
> # réactivation du token
> load(« ./ mon_token_de_connexion")
> validateToken(token)
```

Nous allons maintenant pouvoir requêter Google Analytics pour obtenir quelques données à visualiser. Pour ce premier article, nous allons rester dans la simplicité et nous contenter de prendre les visites et les transactions d'un site web durant l'année 2015. Pour savoir ce que vous pouvez obtenir, n'hésitez pas à vous référer au site [6].

Il va nous falloir tout d'abord définir les dimensions et les métriques qui nous intéressent, pour cela, nous allons stocker l'information dans deux variables spécifiques.

```
> # Choix des dimensions et métriques d'interêt
> ga_dim <- 'ga:date'
> ga_met <- 'ga:visits'
```

Nous allons donc obtenir la dimension date et la métrique visits, cela signifie que l'on aura une vision du nombre de visites par date.

On va également définir les dates de début et de fin de période que l'on souhaite visualiser, choisir un critère de tri (ici il est en fait inutile car il existea une seule métrique), et enfin on va préciser l'identifiant du site que l'on souhaite analyser. Pour trouver ce dernier identifiant, il faut se rendre dans votre compte Analytics, puis choisir dans le panneau d'administration le site qui vous intéresse. L'identifiant de profil est disponible en cliquant sur « paramètres de la vue » sous le nom d'identifiant de la vue.

```
> # Choix des dimensions et métriques d'interêt
> startdate  <- '2015-01-01'
> enddate <- '2015-12-01'

> # Choix pour le tri
> sort <- 'ga:visits'
```

```
> # identifiant de la vue
> profileid <- 'ga :XXXXXXX'
```

On peut ensuite préparer la requête, selon le format préconisé par l'API (voir [7]).

```
> query.list <- Init(start.date = startdate,
                     end.date = enddate,
                     dimensions = ga_dim,
                     metrics = ga_met,
                     max.results = 10000,
                     sort = sort,
                     table.id = profileid)

> ga.query <- QueryBuilder(query.list)
```

Une fois la requête enregistrée sous le nom de ga.query, il ne reste qu'à la transmettre à Google pour récupérer les données, en format data frame de R, ici sous le nom de ga.data.

```
> ga.data <- GetReportData(ga.query, token)
```

Si vous avez travaillé un peu lentement, votre token devra probablement être réactivé à ce moment là (voir plus haut).

Arrivé à ce stade, nous avons donc initialisé une méthode de connexion, validé les identifiants, récupéré les visites par date d'un site web donné.
Voici à quoi ressemble les premières lignes de ga.data :

```
      date visits
1 20151016     18
2 20150905     20
3 20150910     21
4 20151009     22
5 20150619     23
```

On voit tout de suite que le codage de la date ne permet pas de travailler de manière efficace en matière de visualisation, on va donc travailler les données pour avoir un format jour-mois-année lisible.

```
> # On charge des packages pour travailler sur les dates et pour splitter les
> #données
> library(quantmod)
> library(plyr)

> # format année, mois, journée
```

```
> ga.data$date <- as.Date(as.character(ga.data$date),format="%Y%m%d")

> # on isole l'année
> ga.data$year <- as.numeric(as.POSIXlt(ga.data$date)$year+1900)

> # On isole le mois
> ga.data$month <- as.numeric(as.POSIXlt(ga.data$date)$mon+1)

> # On crée des labels textuels pour les mois
> ga.data$monthf <- factor(ga.data$month,levels=as.character(1:12),
labels=c("Jan","Fev","Mar","Avr","Mai","Juin","Juil","Août","Sep","Oct","Nov","Dec")
,
                ordered=TRUE)

> # On isole les jours de la semaine
> ga.data$weekday <- as.POSIXlt(ga.data$date)$wday
> ga.data$weekday[ga.data$weekday==0] <- 7

> # On crée des labels textuels pour les jours de la semaine
> ga.data$weekdayf <- factor(ga.data$weekday,levels=rev(1:7),
labels=rev(c("Lun","Mar","Mer","Jeu","Ven","Sam","Dim")),
                  ordered=TRUE)

> # Manipulations pour compter les semaines, la position dans le mois, etc.
> ga.data$yearmonth <- as.yearmon(ga.data$date)
> ga.data$yearmonthf <- factor(ga.data$yearmonth)
> ga.data$week <- as.numeric(format(as.Date(ga.data$date),"%W"))
> ga.data <- ddply(ga.data,.(yearmonthf),transform,monthweek=1+week-min(week))
```

Si on regarde les premières ligne de ga.data, on obtient maintenant quelque chose de beaucoup plus complet.

	date	visits	year	month	monthf	weekday	weekdayf	yearmonth	yearmonthf	week	monthweek
1	2015-01-10	42	2015	1	Jan	6	Sam	jan 2015	jan 2015	1	2
2	2015-01-14	43	2015	1	Jan	3	Mer	jan 2015	jan 2015	2	3
3	2015-01-20	45	2015	1	Jan	2	Mar	jan 2015	jan 2015	3	4
4	2015-01-07	46	2015	1	Jan	3	Mer	jan 2015	jan 2015	1	2

Nous avons maintenant toutes les données nécessaires, au bon format, pour faire une belle visualisation du nombre de visites au long de l'année 2015.

Créer une heatmap représentant les visites de l'année 2015

R est un formidable outil de visualisation, notamment grâce à l'excellent package **ggplot2** de Hadley Wickham. Ce dernier est l'un des gourous de R, auteur de très nombreux packages, son « vrai » métier est statisticien à l'Université de Rice (au Texas). Le package

ggplot2 a réellement offert une nouvelle manière de concevoir les visualisations, en séparant ces dernières en trois aspects principaux :

• **Data**. Le premier point dans uns visualisation est bien évidemment les données qui vont être représentées. Dans notre exemple d'aujourd'hui, il s'agit de toute l'information qui se trouve dans le *data frame* ga.data.
• **Aesthetics**. Sous ce terme générique on retrouve tout les éléments qui vont concerner les couleurs, les formes, les labels des axes, le choix des données qui vont être affichées, etc. Bref, on va encoder dans les *aesthetics* tout se qui va donner le « look », la forme de la figure.
• **Geometry**. Il s'agit là des options de configurations du type de figure choisie par l'utilisateur (histogramme, courbe, nuage de points, heatmap, etc.).

Pour ce premier article sur R et Google Analytics, nous allons réaliser une heatmap des visites de l'année sur un site particulier. Une heatmap est un type de tableau qui va contenir la valeur d'une variable donnée dans chaque case, cette valeur étant représentée par un code couleur.

Voici le code qui permet de générer la figure, nous l'expliquerons ensuite. Ce code est largement inspiré de celui disponible sur le site [8].

```
> # On charge ggplot2
> library(ggplot2)

> graphique <- ggplot(ga.data, aes(monthweek, weekdayf, fill = visits)) +
   geom_tile(colour = "white") +
   facet_grid(year~monthf) +
   scale_fill_gradient(high="pink",low="grey") +
   labs(title = "Visites sur le site") +
   xlab("Semaine dans le mois") +
   ylab("")
```

Expliquons en détail le fonctionnement de cette commande.
Nous allons créer un object « graphique » qui va définir la figure que nous sommes en train de réaliser.
Pour cela, nous appelons tout d'abord la fonction **ggplot** sur les données contenues dans le *data frame* ga.data que nous avons défini précedemment. Nous invoquons ensuite les *aesthetics* (commande aes) pour signifier que nous souhaitons représenter la valeur du nombre de visites pour chaque jour de chaque semaine de chaque mois de l'année.

La couleur de base du pavage sera le blanc (geom_tile(colour = "white")) et nous allons mettre en légende des blocs avec les labels textuels des mois et l'année.

Le code couleur pour le nombre de visites sera un gradient entre le gris et le rose. Quand la case sera coloriée en gris cela voudra dire que le nombre de visites est le minimum (zéro ici) et lorsque cela sera en rose on atteindra le maximum (un peu plus de 300 visites pour

ce site spécifique). Enfin, on précise le titre de la figure et le type des éléments de l'axe des abscisses.

La commande print(graphique) permet d'afficher l'objet « graphique » que nous venons de réaliser, et représente la figure 1.

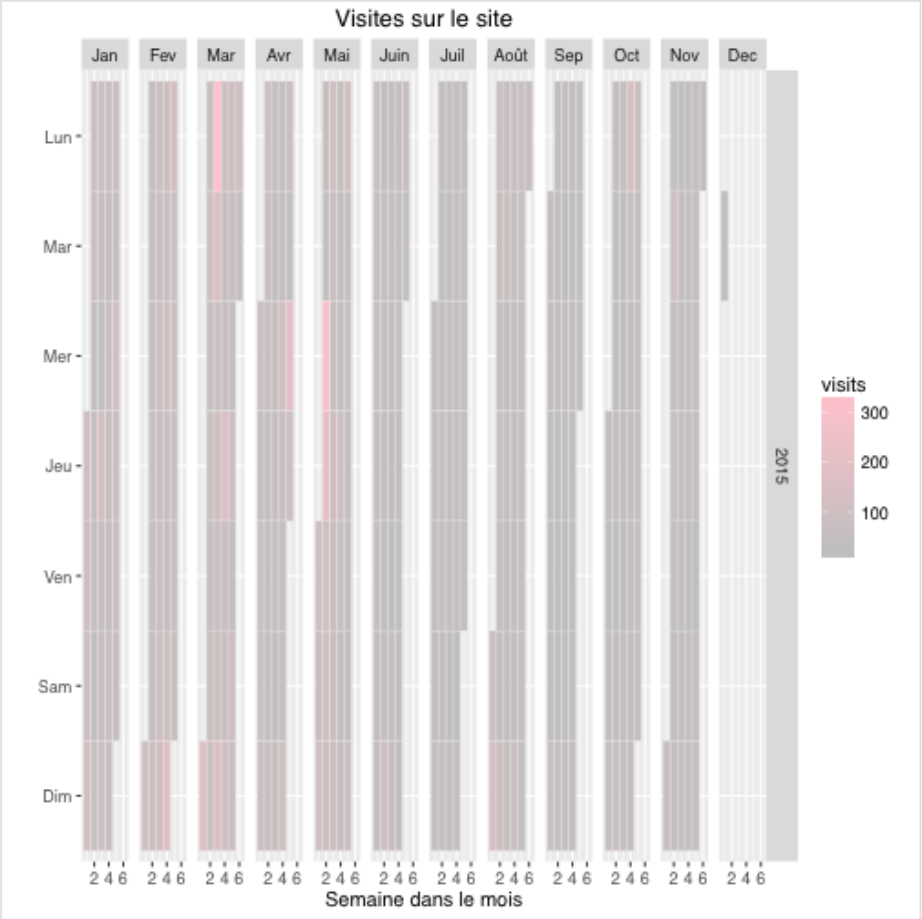

Fig.1. Représentation graphique des visites sur un site web grâce à R.

Pour sauver cette visualisation, par exemple au format pdf, il suffit d'utiliser la commande ggsave, en prenant soin de bien préciser l'extension de fichier pour définir le format pdf. Si on souhaite plutôt une image png, il suffit de sauver avec l'extension .png.

```
> print(graphique)
> ggsave(file="graphique.pdf")
```

Vous savez maintenant récupérer des données depuis Google Analytics et réaliser une visualisation simple. Le mois prochain, nous verrons des visualisations plus évoluées qui croisent plus de dimensions et de métriques.

Références

[1] https://www.r-project.org/

[2] John Chambers, Richard Becker, « *S: An Interactive Environment for Data Analysis and Graphics* », Éditions Chapman & Hall, 1988.

[3] https://www.rstudio.com/

[4] http://www.statmethods.net/about/books.html

[5] https://console.developers.google.com/project

[6] https://developers.google.com/analytics/devguides/reporting/core/dimsmets

[7] https://developers.google.com/analytics/devguides/reporting/core/v3/reference

[8] http://www.r-bloggers.com/calender-heatmap-with-google-analytics-data-2/

Guillaume Peyronnet *est gérant de Nalrem Médias.* **Sylvain Peyronnet** *est co-fondateur et responsable des ix-labs, un laboratoire de recherche privé. Ensemble, ils font des formations, pour en savoir plus : http://www.peyronnet.eu/blog/*