Workshop 1

#### ***Paul Daigle et Marine Leroi***

#### ***24/01/2018***

* [À faire avant le workshop](#gjdgxs)
  + [Jeux de données](#30j0zll)
* [Présentation de R et Rmarkdown](#1fob9te)
  + [Fonctionnalités de R](#3znysh7)
  + [Fonctionnalités de Rmarkdown (textes, code chunk, recherche reproductible, différents outputs…)](#2et92p0)
* [Présentation de Rstudio](#tyjcwt)
  + [Tour de l’interface](#3dy6vkm)
* [TP 1 : Syntaxe de Rmarkdown](#1t3h5sf)
* [TP 2 : Les données](#4d34og8)
  + [Format des données](#2s8eyo1)
  + [Importation de données](#17dp8vu)
    - [CSV](#3rdcrjn)
    - [Google sheet](#26in1rg)
      * [Exemple](#lnxbz9)
      * [Erreurs habituelles](#35nkun2)
  + [Manipulation](#1ksv4uv)
    - [Wide to long](#44sinio)
    - [Colonnes](#2jxsxqh)
      * [Sélectionner une colonne](#z337ya)
      * [Créer une colonne](#3j2qqm3)
      * [Supprimer une colonne](#1y810tw)
      * [Changer le type de données](#4i7ojhp)
      * [Filtrer les données](#2xcytpi)
      * [Trier les données](#1ci93xb)
* [TP 3 : Visualisation des données](#3whwml4)
  + [Head, summary](#2bn6wsx)
  + [Librairie ggplot2](#qsh70q)
    - [Fonction ggplot()](#3as4poj)
    - [geom\_line()](#1pxezwc)
    - [geom\_bar()](#49x2ik5)
    - [geom\_point()](#2p2csry)
    - [Fonctions/Paramètres utiles](#147n2zr)
* [Autres trucs cools…](#3o7alnk)

À faire avant le workshop

* Remplir le [Google sheet](https://docs.google.com/spreadsheets/d/1w45_Gh3BdCErygi8oPU0CAdyFYGYLRAytHX6Hx9V12E/edit?usp=sharing) en choisissant un pseudo disponible (s’il n’y a plus de pseudo disponible, veuillez nous contacter).
* Ouvrir une session sur <https://hector.nuance-r.com/rstudio/> avec vos identifiants (ceux du Google sheet).
* Accéder au dossier [Google Drive](https://drive.google.com/drive/folders/1gYY8Mk7XHPDdxpJNZ_PXUO2eAL8Ya0rD) et télécharger le fichier csv “**CPI.csv**”
* Se familiariser (i.e. jeter un coup d’oeil aux jeux de données 1-3 suivant).
* Si quelque chose ne fonctionne pas, veuillez nous contacter.

Jeux de données

[Données 1 - Solar contractors in California](https://docs.google.com/spreadsheets/d/13PnLASB3pADTmqa31SvLZj9pDWd9tnS35hIrUf3Lc-s/edit#gid=0)

[Données 2 - Electricity providers in California](https://docs.google.com/spreadsheets/d/1bhDB8VLTJL_vA58jKd82N9W6JuMoqyvM_OlniSZGR-8/edit?usp=sharing)

[Données 3 - Cost per electricity source](https://docs.google.com/spreadsheets/d/1k6gpUbM-PhHSmLcJubonRBnKetUatdCrQ1A1OjCKrbw/edit?usp=sharing)

Présentation de R et Rmarkdown

Fonctionnalités de R

R est :

* Un langage informatique dédié aux statistiques et à la science des données pour permettre le traitement des mégadonnées.
* Similaire à Python et Matlab.
* Un langage *Open Source*.

Fonctionnalités de Rmarkdown (textes, code chunk, recherche reproductible, différents outputs…)

* Rmarkdown est une combinaison de R et de markdown. Il permet de produire des documents combinant texte et code, et ainsi permet de faire de la **recherche reproductible**. Par exemple, pour mettre à jour un document, il suffit simplement de mettre à jour les données.
* L’insertion de *code chunks* permet d’utiliser du code en **R** ou dans d’autres langages comme Python.
* Un même script .Rmd permet d’avoir différents *outputs* : .pdf, .docx, .html, .pptx, ioslides, beamer, flex\_dashboard.

Présentation de Rstudio

Tour de l’interface

* source
* console
* environnement
* fichiers

TP 1 : Syntaxe de Rmarkdown

* Le **YAML**: entête de script .Rmd, permet de définir le titre, l’auteur et la date, mais surtout l’**output(s)**. Il est généralement automatiquement généré.
* Le “**#**” en début de ligne permet de créer des titres et sous-titres. Plusieurs “#” permet de changer le niveau des titres. e.g. #Titre1, puis ##Sous-titre1, ###Sous-sous-titre1
* Le “\*” permet de mettre en \**italique*\* ou en \*\***gras**\*\* une partie du texte.
* Le “**$**” permet de créer des équations sous format LaTex. e.g. \(E=mc^2\), \(i\hbar\frac{\partial}{\partial t}\Psi = H\Psi\). [Liste de symboles en LaTex](https://oeis.org/wiki/List_of_LaTeX_mathematical_symbols)
* titre_image
* [lien](http://docs.google.com/drhector.org)
* code chunk –> montrer via interface
* library(nameOfLibrary) à mettre dans le code chunk.

Exercice 1

Trouver une équation célèbre sur Wikipédia, et écrire le code pour la présenter (pour les paresseux voici la célèbre [équation d’Einstein](https://en.wikipedia.org/wiki/Mass%E2%80%93energy_equivalence)). Cela doit inclure :

* Un titre (e.g. “Mass-energy equivalence”)
* Un sous-titre (e.g. “L’équation”)
* Une phrase simple introduisant l’équation (e.g. “Voici l’équation d’Einstein, tirée de Wikipédia”)
* Un mot en gras (e.g. “équation”)
* Un mot en italique (e.g. “Einstein”)
* Un lien (e.g. le lien de la page Wikipédia)
* Une équation

TP 2 : Les données

Format des données

Il existe plusieurs formats de données en R : vecteurs, matrices, dataframes et listes. Il est important de savoir que les différents formats ont différentes propriétés. En science des données, le **dataframe** est le plus utilisé. Pour ceux qui utilise *python*, le dataframe est équivalent à celui de la librairie *pandas*. Nous verrons plus loin comment fonctionne un dataframe. Pour connaître les fonctionnalités et utilités des autres, nous vous recommandons [ce lien](https://www.statmethods.net/input/datatypes.html).

Importation de données

CSV

Importer des données à travers un .csv est plus courant du fait que les données sont souvent sous ce format, mais est au final archaïque (Les Google sheets, c’est cool!). Il existe 2 méthodes pour importer un fichier .csv:

* Méthode manuelle à travers l’environnement RStudio
* Méthode par le code à travers la fonction read.csv()

dataFromCSV <- read.csv("CPI.csv", header=TRUE, sep=",")

Exercice 2

Importer les données du fichier csv via les 2 méthodes.

Google sheet

Pour importer des données d’un document google sheet, il faut :

* S’assurer que les paramètres de partage du fichier soient ouverts à tous ceux qui ont accès au lien
* Importer la librarie **gsheet**
* Appeler la fonction gsheet2tbl()

Exemple

library(gsheet)  
mydataset1 <- gsheet2tbl("https://docs.google.com/spreadsheets/d/13PnLASB3pADTmqa31SvLZj9pDWd9tnS35hIrUf3Lc-s/edit?usp=sharing")

Erreurs habituelles

* Il n’y a pas de “s” après gsheet (pour la librairie et pour la fonction)
* Il faut mettre le lien entre guillemets lorsque l’on utilise la fonction gsheet2tbl()

Exercice 3

Importer les données du Google sheet.

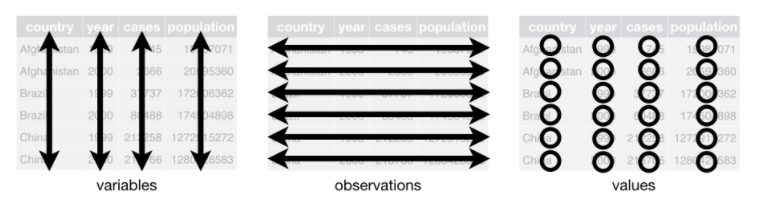
Manipulation

Wide to long

Il existe 2 formats de présentation/sauvegarde de données : **wide** et **long** (ou *narrow*). Voir définition de [Wikipédia](https://en.wikipedia.org/wiki/Wide_and_narrow_data)

Colonnes

Un dataframe en R est décomposé en observations (nombre de lignes) et en variables (colonnes). Les variables (colonnes) ont habituellement des noms (ceux donnés par le fichier de données). Il est évidemment possible de jouer avec les colonnes.



Sélectionner une colonne

Il suffit d’utiliser le symbole “$”, suivi du nom de la colonne.

print(mydataset1$COMPANY)

Créer une colonne

mydataset1$country <- "USA"  
mydataset1$employeesInThousands <- mydataset1$EMPLOYEES/1000

Supprimer une colonne

mydataset1$country <- NULL  
mydataset1$employeesInThousands <- NULL

Changer le type de données

Parfois, il est utile de spécifier le type de données dans le dataframe, notamment pour éviter des bugs.

mydataset1$CITY <- sapply(mydataset1$CITY, as.factor)  
mydataset1$total\_installed <- sapply(mydataset1$total\_installed, as.numeric)

Exercice 3

* Créer une colonne “Comité”, dont la valeur est “SDS”
* Créer une colonne “employeesPerMW”, dont la valeur est le nombre d’employés divisé par le nombre de MW installés en 2015 (colonne “installed\_2015”)
* Vérifier que les colonnes se sont bien créées
* Spécifier le type de données : as.factor pour la colonne “Comité” et as.numeric pour la colonne “employeesPerMW”
* Supprimer les colonnes

Filtrer les données

Filtrer des données est super utile lorsque l’on veut sélectionner qu’une partie des données, notamment selon un certain critère. Il suffit d’utiliser la fonction **filter()** de la library **dyplr**.

**Attention :** en argument de la fonction filter(), il faut inclure le nom du dataframe et la condition de filtrage. Pour la condition de filtrage, il ne faut mettre que le nom de la colonne (i.e. ne pas mettre le nom du dataframe + “$” + nom de la colonne).

library(dplyr)  
mydatasubset1 <- filter(mydataset1, EMPLOYEES > 1000)  
mydatasubset2 <- filter(mydataset1, Type == "Residential" | Type == "Commercial")

Trier les données

Il peut être aussi utile de trier les données par ordre croissant ou décroissant. Pour trier en ordre croissant, il suffit alors d’utiliser la fonction **arrange()** de la librarie **dyplr**. Les arguments à fournir sont le nom du dataframe et le nom de la colonne. Pour trier en ordre décroissant, il s’agit de la même fonction, à l’exception qu’il faut entrer le nom de la colonne à l’intérieur de la fonction **desc()**.

mydataSorted <- arrange(mydataset1, EMPLOYEES)  
mydataReversed <- arrange(mydataset1, desc(EMPLOYEES))

Exercice 4

* Filtrer les données pour avoir seulement les compagnies de *Type* “**Utility**”
* Filtrer les nouvelles données pour avoir seulement celles qui ont plus de 1000 MW d’installés au total (“*total\_installed*”)
* Ranger les nouvelles données dans l’ordre descendant en fonction du nombre d’employés (“*EMPLOYEES*”)

TP 3 : Visualisation des données

Head, summary

Pour avoir une idée générale des données, il existe 2 fonctions : **head()** et **summary()**. Il suffit de mettre en argument

* head() : affiche les 6 premières observations (lignes) du dataframe.
* summary() : affiche les statistiques de base (minimum, 1er quartile, médiane, moyenne, 3e quartile, maximum) pour chacune des variables.

head(mydataset1)

## # A tibble: 6 x 12  
## RANK COMPANY `OVERALL RANK` CITY STATE EMPLOYEES Service Type   
## <int> <chr> <int> <fct> <chr> <dbl> <chr> <chr>  
## 1 1 SolarCity 3 San Ma… CA 14000 Rooftop … Resi…  
## 2 2 Arraycon 4 Sacram… CA 200 Construc… Util…  
## 3 3 Cupertino … 5 San Jo… CA 1975 EPC Util…  
## 4 4 Swinerton … 7 San Fr… CA 2385 EPC Util…  
## 5 5 Rosendin E… 8 San Jo… CA 4500 EPC Util…  
## 6 6 Baker Elec… 9 Escond… CA 650 Electric… Util…  
## # ... with 4 more variables: total\_installed <dbl>, installed\_2015 <dbl>,  
## # lat <dbl>, long <dbl>

summary(mydataset1)

## RANK COMPANY OVERALL RANK CITY   
## Min. : 1.00 Length:130 Min. : 3.0 San Diego :12   
## 1st Qu.: 33.25 Class :character 1st Qu.:106.0 Los Angeles : 9   
## Median : 65.50 Mode :character Median :208.0 San Jose : 5   
## Mean : 65.50 Mean :220.6 Escondido : 4   
## 3rd Qu.: 97.75 3rd Qu.:333.2 Fresno : 4   
## Max. :130.00 Max. :489.0 San Francisco: 3   
## (Other) :93   
## STATE EMPLOYEES Service   
## Length:130 Min. : 2.00 Length:130   
## Class :character 1st Qu.: 13.25 Class :character   
## Mode :character Median : 28.00 Mode :character   
## Mean : 251.99   
## 3rd Qu.: 78.75   
## Max. :14000.00   
##   
## Type total\_installed installed\_2015 lat   
## Length:130 Min. : 0.17 Min. : 0.120 Min. :32.89   
## Class :character 1st Qu.: 3.85 1st Qu.: 0.940 1st Qu.:33.67   
## Mode :character Median : 10.56 Median : 2.660 Median :33.99   
## Mean : 98.70 Mean : 39.852 Mean :35.51   
## 3rd Qu.: 34.28 3rd Qu.: 8.525 3rd Qu.:37.77   
## Max. :2184.00 Max. :870.000 Max. :40.51   
##   
## long   
## Min. :-123.8   
## 1st Qu.:-121.6   
## Median :-118.3   
## Mean :-119.3   
## 3rd Qu.:-117.0   
## Max. :-116.0   
##

Librairie ggplot2

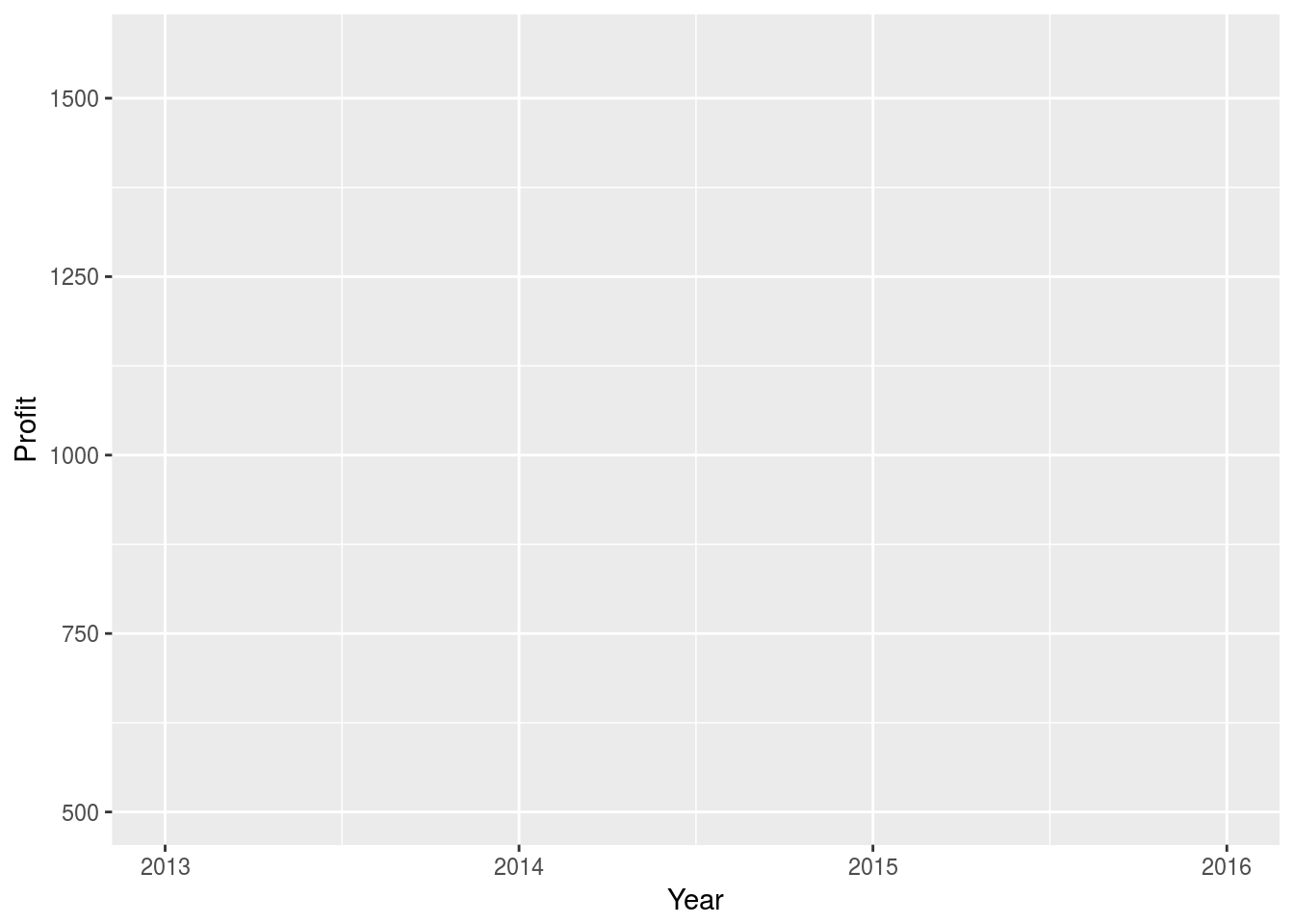
Les libraries ggplot2 et ggthemes sont extrêmement utiles pour la visualisation de données à travers de beaux graphiques. Elle est simple d’utilisation mais permet aussi un niveau de personalisation esthétique poussé.

Fonction ggplot()

La fonction **ggplot()** permet de manière générale de définir les données utilisées pour le graphique. L’argument **data =** permet d’appeler le **dataframe** qui contient les données. Il faut ensuite définir l’argument **aes()** pour savoir quels sont les axes x et y, ainsi que les couleurs (e.g. s’il y a plusieurs courbes, avec une couleur chacune). Ainsi, les arguments de **aes()** varient en fonction du type de graphique utilisé. Pour notre exemple, nous avons **x =**, **y =** et **color =**. Se référer à la documentation de [Rstudio](https://www.rstudio.com/wp-content/uploads/2015/03/ggplot2-cheatsheet.pdf) pour plus de détails.

**Attention :** Les arguments de **aes()** sont le nom des variables du dataframe utilisé.

library(ggplot2)  
library(ggthemes)  
  
linePlotData <- gsheet2tbl("https://docs.google.com/spreadsheets/d/1bhDB8VLTJL\_vA58jKd82N9W6JuMoqyvM\_OlniSZGR-8/edit?usp=sharing")  
  
ggplot(data = linePlotData, aes(x = Year, y = Profit, color = Company))

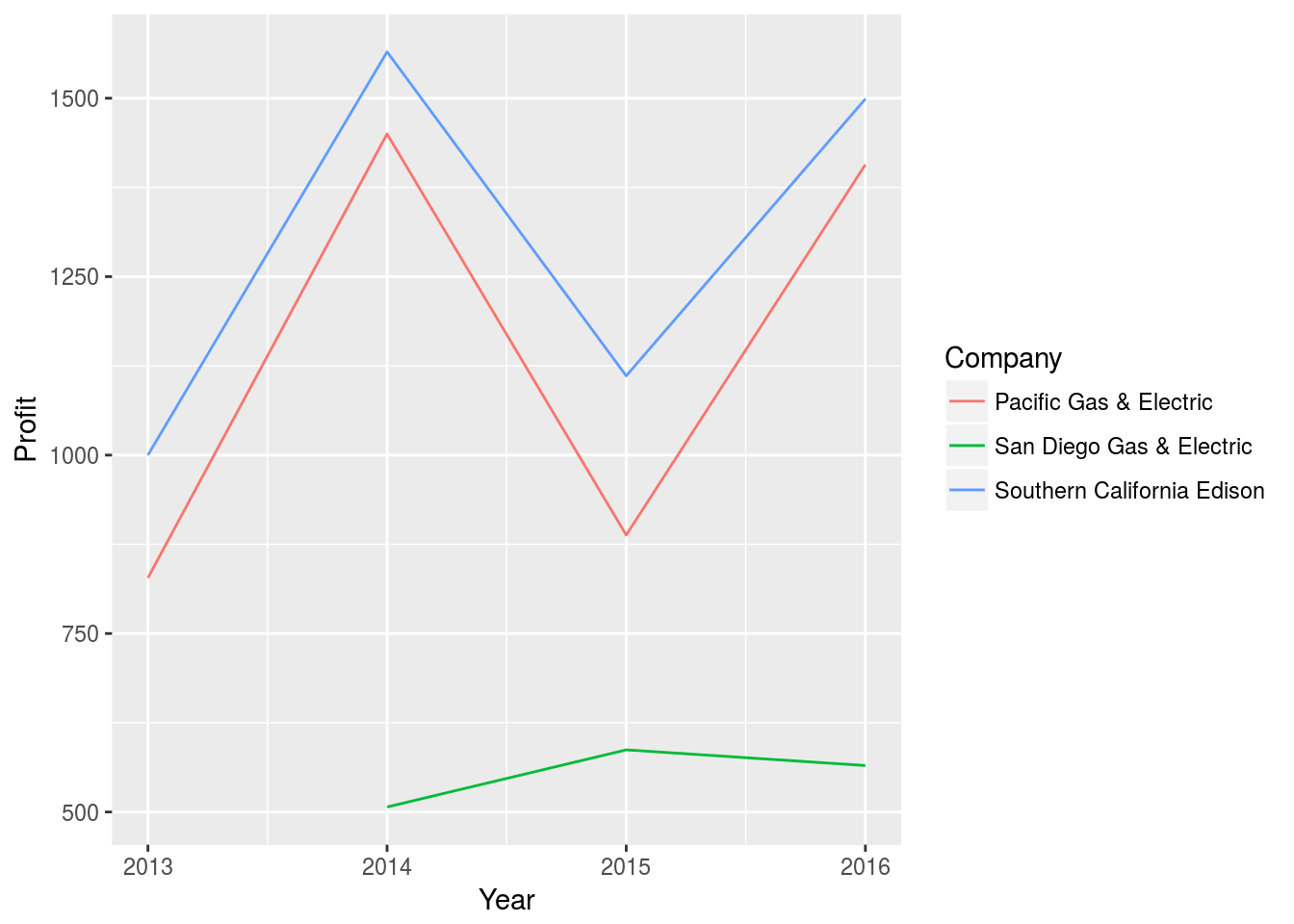


geom\_line()

La fonction **ggplot()** toute seule n’affichera rien. Il faut appeler le type de graphique souhaité. Voir la fiche de [Rstudio](https://www.rstudio.com/wp-content/uploads/2015/03/ggplot2-cheatsheet.pdf) pour connaitre les différents graphiques possibles. Ici, nous montrerons d’abord un exemple de lignes.

**NB:** Il faut mettre le signe “**+**” après la fonction **ggplot()**. Il est habituel de mettre le signe “**+**” sur la même ligne que la fonction.

ggplot(data = linePlotData, aes(x = Year, y = Profit, color = Company)) +  
 geom\_line()

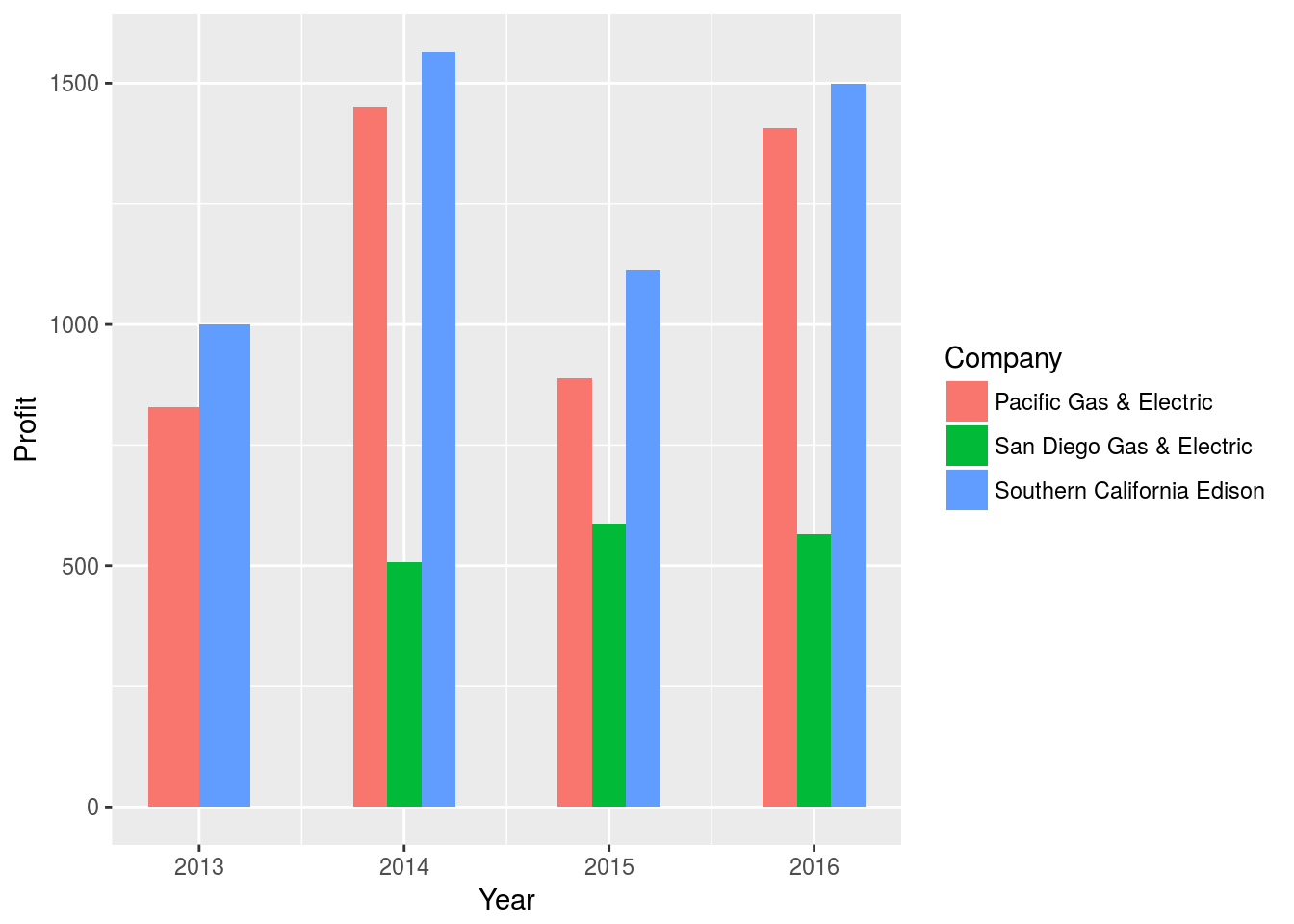


geom\_bar()

De la même façon, on utilise **geom\_bar()**. Pour cette fonction, il est cependant nécessaire de spécifier l’argument **stat =**.

**Attention :** Ici il faut utiliser l’argument **fill =** pour la fonction **aes()**, sinon c’est moche…

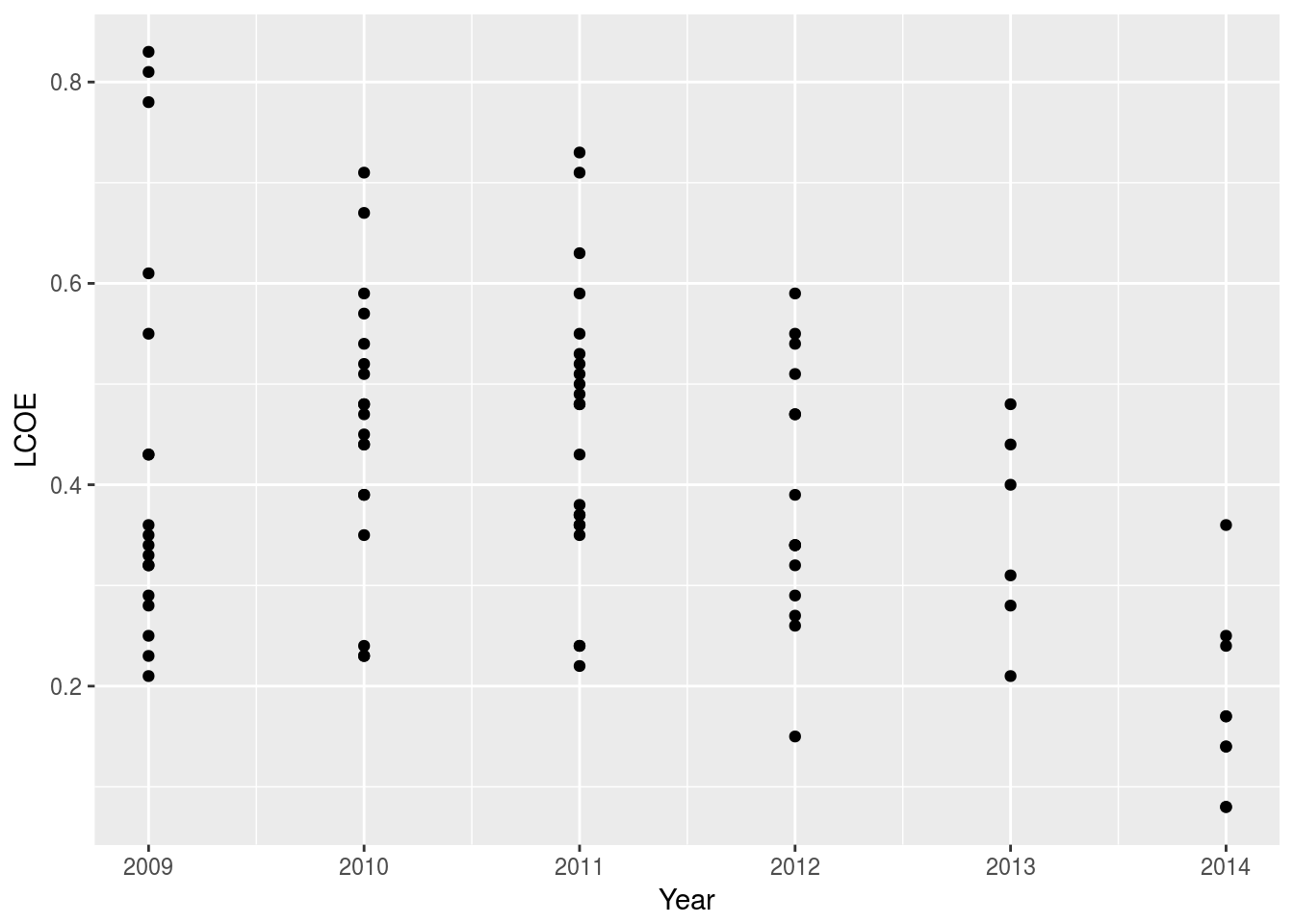
ggplot(data = linePlotData, aes(x = Year, y = Profit, fill = Company)) +  
 geom\_bar(stat="identity", position = "dodge", width = 0.5)



geom\_point()

Pour faire des beaux scatter plots!

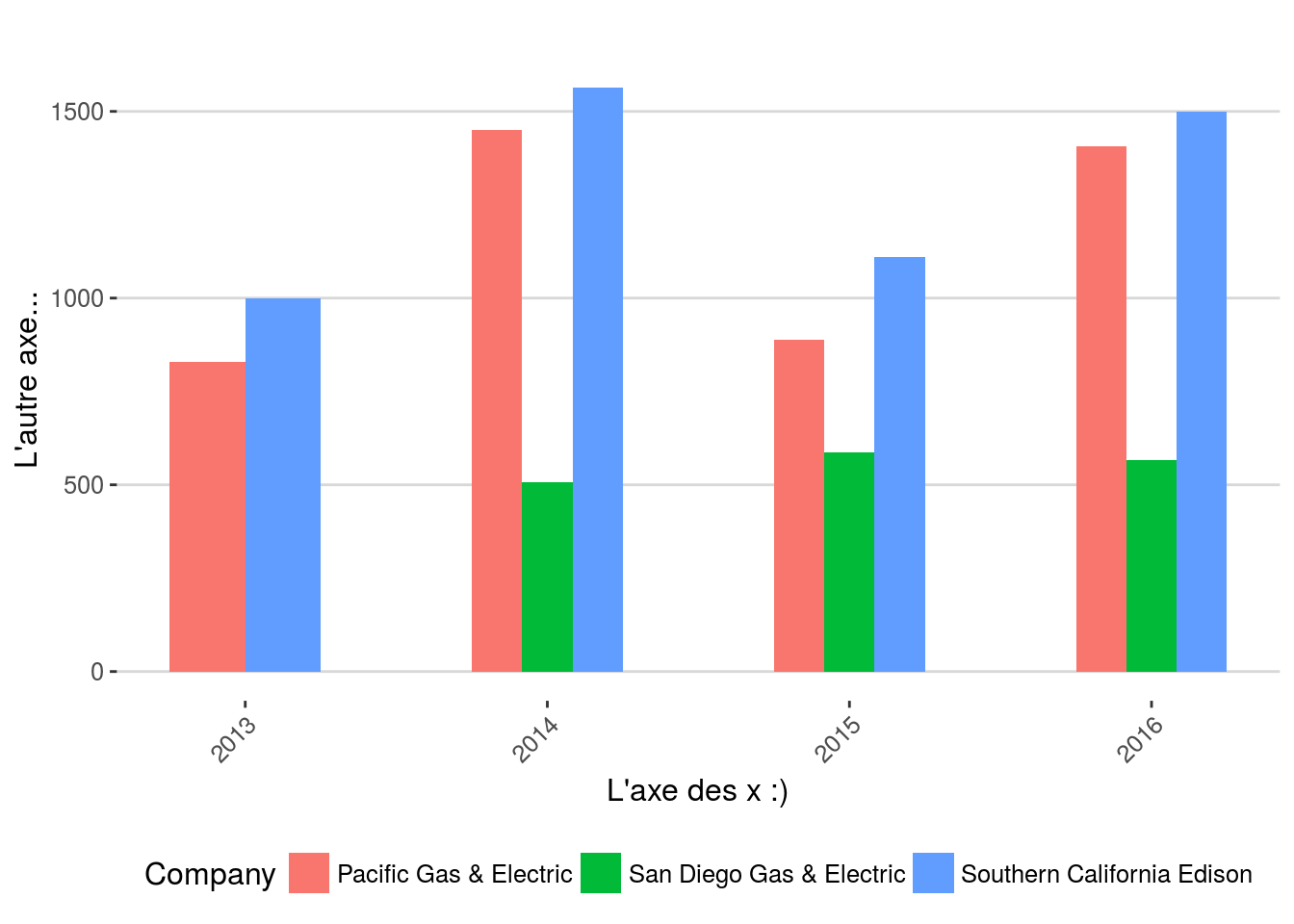
scatterPlotData <- gsheet2tbl("https://docs.google.com/spreadsheets/d/1k6gpUbM-PhHSmLcJubonRBnKetUatdCrQ1A1OjCKrbw/edit?usp=sharing")  
ggplot(data = filter(scatterPlotData, Source == "Solar, Photovoltaic"), aes(x = Year, y = LCOE)) +  
 geom\_point()



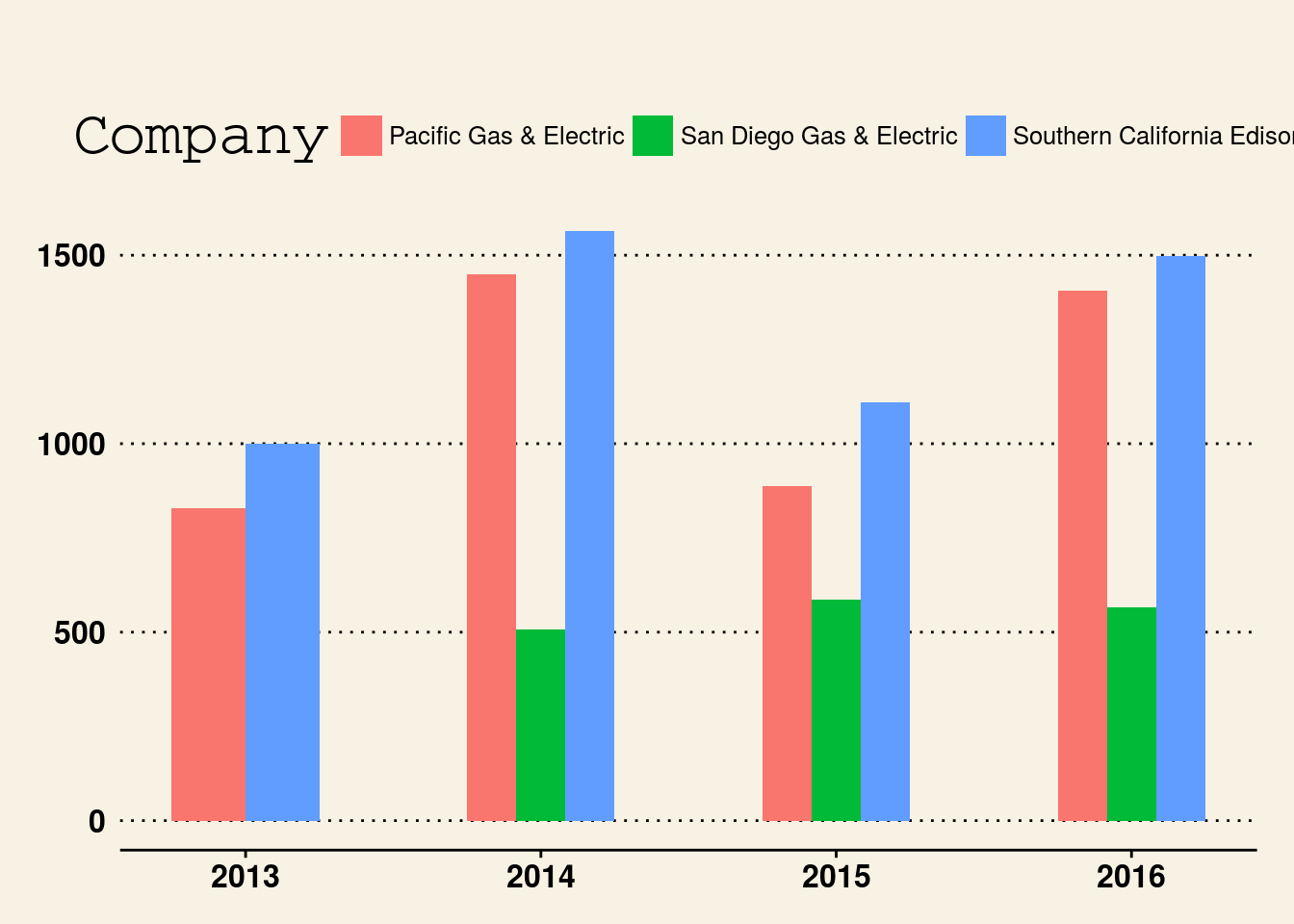
Fonctions/Paramètres utiles

* **ylab()** : nom de l’axe des y (mettre **“”** pour ne rien avoir)
* **xlab()** : on vous laisse deviner…
* **ggtitle()** : titre du graphique
* **theme\_XYZ()** : format selon des thèmes prédéfinis. [Thèmes de ggthemes](https://cran.r-project.org/web/packages/ggthemes/vignettes/ggthemes.html) [Thèmes de ggplot2](https://www.rstudio.com/wp-content/uploads/2015/03/ggplot2-cheatsheet.pdf)
* **coord\_flip()** : Inverser les axes
* **theme(axis.text.x = element\_text(angle = 45, hjust = 1))** : Faire pivoter les noms sur l’axe des x

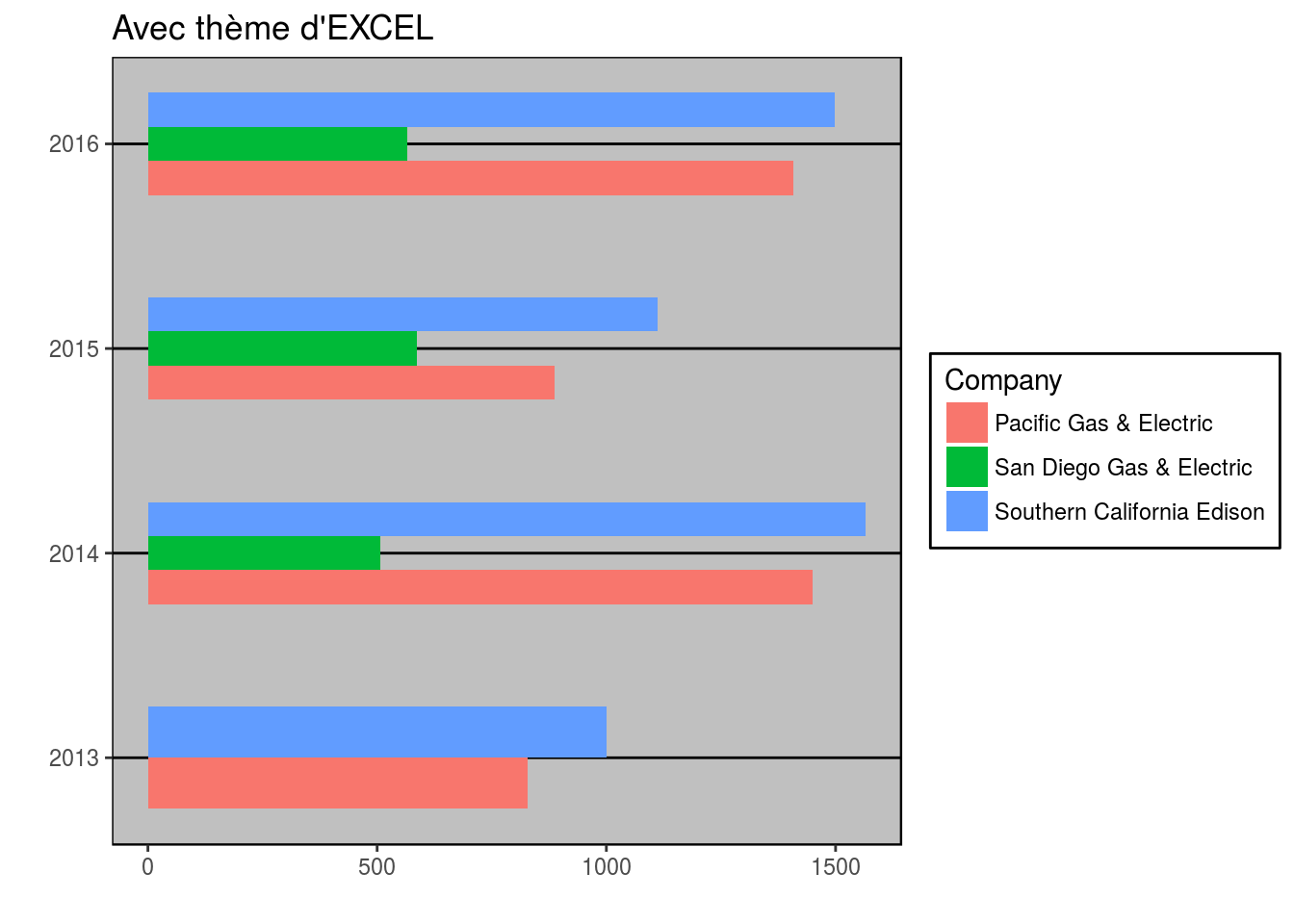
ggplot(data = linePlotData, aes(x = Year, y = Profit, fill = Company)) +  
 geom\_bar(stat="identity", position = "dodge", width = 0.5) +   
 xlab("L'axe des x :)") +  
 ylab("L'autre axe...") +  
 ggtitle("") +  
 theme(axis.text.x = element\_text(angle = 45, hjust = 1)) +  
 theme\_hc()



ggplot(data = linePlotData, aes(x = Year, y = Profit, fill = Company)) +  
 geom\_bar(stat="identity", position = "dodge", width = 0.5) +   
 ylab("") +  
 xlab("") +  
 ggtitle("") +  
 theme\_wsj()



ggplot(data = linePlotData, aes(x = Year, y = Profit, fill = Company)) +  
 geom\_bar(stat="identity", position = "dodge", width = 0.5) +   
 ylab("") +  
 xlab("") +  
 ggtitle("Avec thème d'EXCEL") +  
 coord\_flip() +  
 theme\_excel()

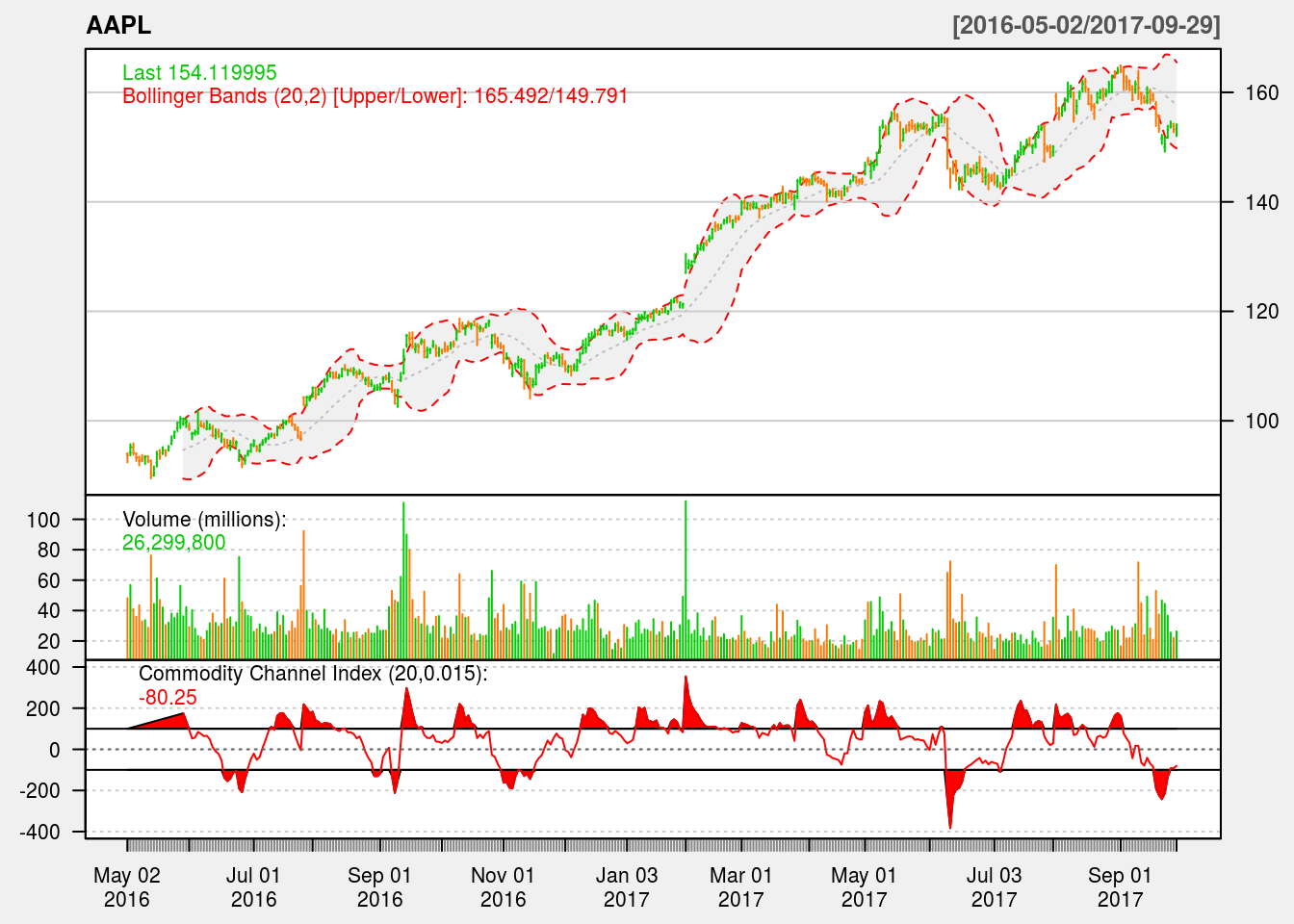


Autres trucs cools…

library(quantmod)  
getSymbols("AAPL",src="yahoo", from="2016-05-02", to="2017-10-02")

## [1] "AAPL"

barChart(AAPL, theme="white", TA="addVo();addBBands();addCCI()")



library(leaflet)  
data <- gsheet2tbl("https://docs.google.com/spreadsheets/d/13PnLASB3pADTmqa31SvLZj9pDWd9tnS35hIrUf3Lc-s/edit?usp=sharing")  
map2 <- addMarkers(clusterOptions = markerClusterOptions(), addTiles(leaflet(data = data)))  
map2

library(MASS)  
library(ISLR)  
library(stargazer)  
  
model1 <- lm(LCOE ~ Year, data = filter(scatterPlotData, Source == "Solar, Photovoltaic"))  
stargazer(model1, out="models1.htm")

##   
## % Table created by stargazer v.5.2 by Marek Hlavac, Harvard University. E-mail: hlavac at fas.harvard.edu  
## % Date and time: Thu, Feb 01, 2018 - 09:35:20 PM  
## \begin{table}[!htbp] \centering   
## \caption{}   
## \label{}   
## \begin{tabular}{@{\extracolsep{5pt}}lc}   
## \\[-1.8ex]\hline   
## \hline \\[-1.8ex]   
## & \multicolumn{1}{c}{\textit{Dependent variable:}} \\   
## \cline{2-2}   
## \\[-1.8ex] & LCOE \\   
## \hline \\[-1.8ex]   
## Year & $-$0.042$^{\*\*\*}$ \\   
## & (0.010) \\   
## & \\   
## Constant & 84.042$^{\*\*\*}$ \\   
## & (20.999) \\   
## & \\   
## \hline \\[-1.8ex]   
## Observations & 89 \\   
## R$^{2}$ & 0.154 \\   
## Adjusted R$^{2}$ & 0.145 \\   
## Residual Std. Error & 0.152 (df = 87) \\   
## F Statistic & 15.864$^{\*\*\*}$ (df = 1; 87) \\   
## \hline   
## \hline \\[-1.8ex]   
## \textit{Note:} & \multicolumn{1}{r}{$^{\*}$p$<$0.1; $^{\*\*}$p$<$0.05; $^{\*\*\*}$p$<$0.01} \\   
## \end{tabular}   
## \end{table}