

R-project-2020

Kilian DEBRAUX Anthony QUERE

17/11/2020



Figure 1: logo

Stack overflow est un site web de type forum permettant de poser ses questions à une communauté. Celle-ci est principalement conçue pour les développeurs. Nous avons obtenu nos données depuis **Les archives de stackexchange** (<https://archive.org/details/stackexchange>). Notre jeu de données a été réduit pour être utilisable à 300 000 données réparties sur 3 mois entre Août et Septembre 2020.

Nous avons rencontré des problèmes avec notre jeu de données. Nous avons commencé à explorer un jeu de données de la même source mais il s'est révélé incohérent et peu réaliste. Nous avons hélas mis beaucoup de temps à le remarquer et nous avons alors été confronté à un second problème. Le fichier que nous voulions était bien sur cette plateforme mais sous un autre nom et hélas un autre poids. Nous nous sommes donc retrouvé avec une archive de 15.6Go à télécharger avec de faibles connexions. Le premier téléchargement a échoué à la toute fin après de nombreuses heures mais nous avons réessayé en passant cette fois-ci par un *curl* et nous avons enfin eu les données. Nous avons eu très peu de temps entre ce moment et l'heure de rendu, nous espérons que vous pourrez en prendre considération pendant votre correction. Nous tenons à nous excuser pour la gêne occasionnée.

Les technologies populaires

Cette fonction permet de récupérer des posts en fonction des tags mis sous forme de vecteur de regex

```
get_posts_by_tag <- function(dataset, tags) {  
  regexes = paste("<", tags, ">", sep = "")  
  posts <- c()  
  for (r in regexes) {  
    p <- subset(dataset, grepl(r, Tags))$Id  
    posts <- union(posts, p)  
  }  
  return(posts)  
}
```

```
# get only questions posts
questions <- subset(posts, PostTypeId == "1")
```

Les tags le plus populaires

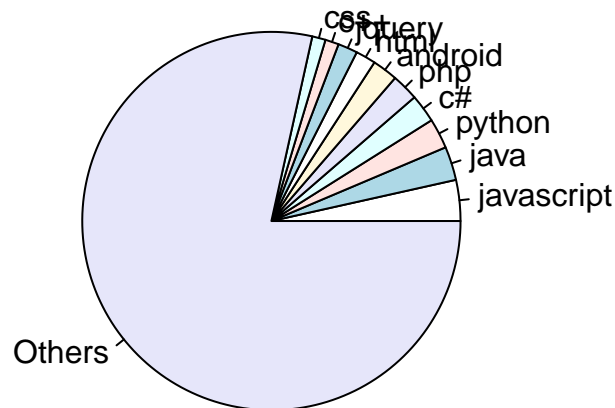
```
sortedTags <- tags[order(tags$Count, decreasing = TRUE),]

TAGS_NUMBER = 10

mostPopularTags = head(sortedTags, n = TAGS_NUMBER)
lessPopularTags = tail(sortedTags, n = nrow(sortedTags) - TAGS_NUMBER)

otherValue = sum(lessPopularTags$Count)

pie(c(mostPopularTags$Count, otherValue), c(mostPopularTags$TagName, "Others"))
```



Nous avons pu récupérer le contenu de ce jeu de données car bien plus léger que le fichier principal. Nous pouvons constater par ce graphique que les tags les plus présents concernent essentiellement les technologies du web (javascript, php, html et css; les autres technologies peuvent aussi servir en web comme le java avec le framework Spring ou le c# avec .net). Nous avons choisi de n'en nommer que 10 pour des soucis de clarté. Ce diagramme nous permet de voir aisément les problématiques principales de la plateforme.

Les frameworks front-end

Les technologies évoluant continuellement en raison des besoins différents de chaque entreprise, plusieurs d'entre elles rendent disponibles des frameworks. Côté Front-End, on en retrouve 5 principaux : Angular, Angular.js

(il est important de les distinguer car ils n'ont pas du tout le même fonctionnement), React.js, Vue.js et JQuery. Nous voulons maintenant définir un “leader du marché” et montrer leurs différences de popularité.

```
angularRegex = c("angular", "angular[2-9]", "angular10")

angularjsRegex = c("angularjs", "angular1.6")

reactRegex = c("reactjs")

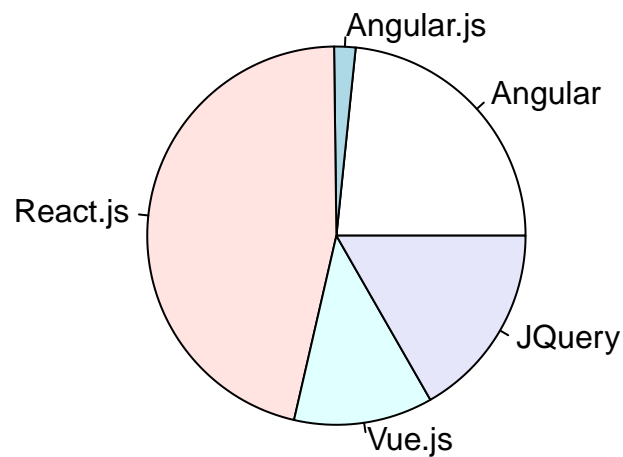
vueRegex = c("vue.js", "vuejs2")

jqueryRegex = c("jquery")

angularPosts <- get_posts_by_tag(questions, angularRegex)
angularjsPosts <- get_posts_by_tag(questions, angularjsRegex)
reactPosts <- get_posts_by_tag(questions, reactRegex)
vuePosts <- get_posts_by_tag(questions, vueRegex)
jqueryPosts <- get_posts_by_tag(questions, jqueryRegex)

i <- c(length(angularPosts), length(angularjsPosts), length(reactPosts), length(vuePosts), length(jqueryPosts))
l <- c("Angular", "Angular.js", "React.js", "Vue.js", "JQuery")

pie(i, l)
```



Nous pouvons constater que les frameworks **React** et **Angular** sont très appréciés par la communauté aujourd'hui, d'autant plus react qui a pour avantage d'être très efficace et maitrisable en peu de temps, Angular a plus de fonctionnalités de base (comme *RxJs*) mais est sur certains points plus complexe. On

retrouve ensuite le **Jquery** ce qui n'est pas très étonnant étant donné que c'est un framework simple qui est encore beaucoup utilisé dans de nombreux sites même s'il utilise aujourd'hui certaines fonctionnalités dépréciées. **Vue.js** est aussi très présent, c'est un framework simple et léger mais bien moins populaire que React et Angular. Nous avons choisi d'ajouter également **Angular.js** car il existe un débat Angular contre **Angular Js**. Angular.js, aussi appelé *Angular 1* est la version de base du projet Angular qui a connu un changement de langage en arrivant en version 2; **Angular** utilise du *Typescript* et **Angular.js** du *Javascript*, une partie de la communauté préfère rester sur du javascript.

Les distributions linux

```
ubuntuRegex = c("ubuntu", "ubuntu-[1-2][0-9]-[0-9][0-9]")

centosRegex = c("centos", "centos[0-9]")

redhatRegex = c("redhat")

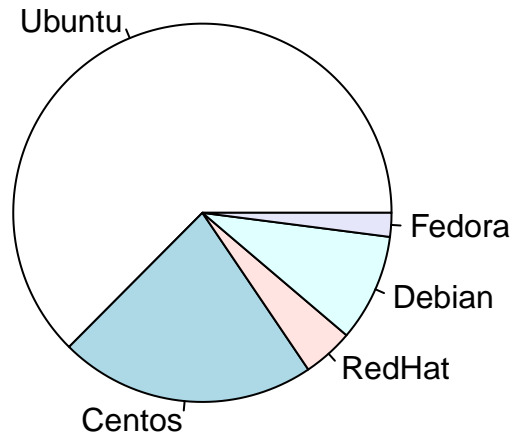
debianRegex = c("debian")

fedoraRegex = c("fedora", "fedora-[0-9]", "fedora-[0-9][0-9]")

ubuntuPosts <- get_posts_by_tag(questions, ubuntuRegex)
centosPosts <- get_posts_by_tag(questions, centosRegex)
redhatPosts <- get_posts_by_tag(questions, redhatRegex)
debianPosts <- get_posts_by_tag(questions, debianRegex)
fedoraPosts <- get_posts_by_tag(questions, fedoraRegex)

i <- c(length(ubuntuPosts), length(centosPosts), length(redhatPosts), length(debianPosts), length(fedoraPosts))
l <- c("Ubuntu", "Centos", "RedHat", "Debian", "Fedora")

pie(i, l)
```



Malgré notre choix de proposer 2 distributions basées sur Debian et 3 sur Centos, nous constatons une très grande préférence pour les distributions basées sur Debian à savoir **Debian** et **Ubuntu**. C'est le résultat que nous attendions car ces distributions linux sont très utilisées par la communauté pour leur utilisation personnelle. Les distributions issues de **Centos** (sauf **Fedora**) sont bien plus utilisées en leur version server, donc avec uniquement la *command line*, dans un contexte d'entreprise, c'est pourquoi on le retrouve en moins grande majorité.

Les langages de programmation

```
cRegex = c("c")
csRegex = c("c#")
cppRegex = c("c\\+\\+")
pythonRegex = c("python")
javaRegex = c("java")
dartRegex = c("dart")
jsRegex = c("javascript")
tsRegex = c("typescript")
rubyRegex = c("ruby")
```

```

phpRegex = c("php")

cPosts <- get_posts_by_tag(questions, cRegex)
csPosts <- get_posts_by_tag(questions, csRegex)
cppPosts <- get_posts_by_tag(questions, cppRegex)
pythonPosts <- get_posts_by_tag(questions, pythonRegex)
javaPosts <- get_posts_by_tag(questions, javaRegex)
dartPosts <- get_posts_by_tag(questions, dartRegex)
jsPosts <- get_posts_by_tag(questions, jsRegex)
tsPosts <- get_posts_by_tag(questions, tsRegex)
rubyPosts <- get_posts_by_tag(questions, rubyRegex)
phpPosts <- get_posts_by_tag(questions, phpRegex)

i <- c(length(cPosts), length(csPosts), length(cppPosts), length(pythonPosts), length(javaPosts), length(dartPosts), length(jsPosts), length(tsPosts), length(rubyPosts), length(phpPosts))
l <- c("C", "C#", "C++", "Python", "Java", "Dart", "Javascript", "Typescript", "Ruby", "Php")

pie(i, l)

```

