LES PIRES MOTS DE PASSES

22/11/2021

#Les packages que j'ai utilisé tout le long de mon analyse.

```
knitr::opts_chunk$set(echo = TRUE)
library(tidyverse)
library(regexplain)
library(glue)
library(distill)
library(ggthemes)
library(data.table)
library(janitor)
library(ggbeeswarm)
library(tinytex)
```

Visualisation des 500 mots des passes les plus utilisés

On a tout dû le faire plusieurs fois. Changer son mot de passe n'est jamais plaisant. Le nouveau mot de passe a comme possibilité d'être entièrement différent ou une variation de son précédant. Peut-être même y ajouter quelques chiffres ou symboles pour le rendre plus complexe ?

La banque de données tirée de Information is Beautiful sur les 500 mots de passe les plus utilisés de 2017 nous donnent de très bons exemples à ne pas répéter.J'ai décidé de l'analyser pour trouver les caractéristiques qui font que ces mots de passe ne sont pas sécuritaires au-delà d'être les plus utilisés.

Grâce aux différentes variables enregistrées, je me suis concentré sur les catégories, le temps de décryptage face à la longueur des caractères et la facilité d'être piraté en fonction du type de caractère utilisé.

Données

```
tuesdata <- tidytuesdayR::tt_load('2020-01-14')

dat_raw <- tuesdata$passwords

summary(dat_raw)
```

J'importe et regarde l'ensemble des variables. Il y a 507 observations dont 7 qui sont N.A. Rank indique la popularité du mot de passe, 1 étant le plus populaire. Password est le mot de passe tel quel. Category sépare les mots de passe en 10 catégories. Value me donne la valeur du temps de décryptage et time_unit associe la valeur à une unité de temps (secondes, minutes, heures, journées, semaines, mois, années). Offline_crack_sec représente le temps de décryptage en secondes si on utilise une attaque de décryptage hors-ligne. Rank_Alt est une alternative au premier rank (presque similaire). Strength représente la force/sécurité du mot de passe sur une échelle.(1 étant faible et 10 fort). Pour finir, font_size est une variable pour le fichier original qui sert à créer le graphique pour KIB.

Conversion temps

```
dat <- dat_raw %>%
  mutate(
    time = case_when(
      time_unit == "seconds" ~ value,
      time_unit == "minutes" ~ value * 60,
      time_unit == "hours" ~ value * 60 * 60,
      time_unit == "days" ~ value * 60 * 24,
      time_unit == "weeks" ~ value * 60 * 24 * 7,
      time_unit == "months" ~ value * 60 * 24 * 30,
      time_unit == "years" ~ value * 60 * 24 * 365,
      TRUE ~ NA_real_ ))

time_table <- tribble(
    ~ time_unit, ~ in_sec,
    "years", 60*60*24*365,
    "months", 60*60*24*30,
    "weeks", 60*60*24*7,
    "days", 60*60*24,
    "hours", 60*60,
    "minutes", 60,
    "seconds",  1 )
```

J'ai pris ce code à Cedric Scherer, 2020 pour être capable de donner le temps total de décryptage en seconde pour tous grâce à la conversion.

Modification données

```
dat <- dat_raw %>%
    na.omit() %>%
    mutate(password_len = str_length(password))%>%
    left_join(time_table, by = "time_unit") %>%
    mutate(guess_crack_sec = value * in_sec)


dat <- dat %>%
  mutate(type = case_when(
    grepl("[A-Za-z]+", password) & grepl("[0-9]+", password) ~ "Lettres & chiffres",
    grepl("[A-Za-z]+", password) ~ "Lettres",
    grepl("[0-9]+", password) ~ "Chiffres",
    TRUE ~ as.character(password))) %>%
  select(-c(in_sec, value, time_unit,rank_alt,offline_crack_sec, font_size))
```

Je crée un nouveau dataset avec les modifications qui conviennent à mon analyse. J'enlève tous les N.A et rajoute la variable password_len pour avoir la longueur de caractères des mots de passe. Je convertis le temps de décryptage en secondes pour tous les observations grâce à time_unit que j'ai créé plus haut.

Création d'une nouvelle variable "type" qui indique si le mot de passe contient seulement des lettres, seulement des chiffres ou s'il est composé des deux. Je termine par enlever les colonnes/variables qui ne seront pas inspectées cette fois-ci.

Analyse

```
count_cat <- dat %>%
  count(category, sort=TRUE)
#Je crée un nouveau data set à partir de "dat" qui contient seulement les
#différentes catégories ainsi que la somme des mots de passe qui les constitue.


just_numbers <- dat[dat$password %like% "^[0-9]*$", ]
#Je crée un sous-ensemble des mots de passe constitués seulement de chiffres.
#Il y en a 40 en tout donc 8% de mon ensemble de données.


just_letters <- dat[dat$password %like% "^[a-zA-Z]+$", ]
# Je crée un sous-ensemble des mots de passe constitués seulement de lettres.
#Il y en a 446 en tout donc 89.2% de mon ensemble de données.


both_letters_numbers <- dat[dat$password %like% '([0-9].*[a-zA-Z])|([a-zA-Z].*[0-9])', ]
# Je crée un sous-ensemble des mots de passes constitués de lettres et de
# chiffres. Il y en a 446 en tout donc 89.2% de mon ensemble de données.

mean(str_length(dat$password))
#moyenne de longueur en caractère des mots de passe est 6.2

length_tibble <- as_tibble(count(dat, password_len, sort = T))
# répartition de la distribution par rapport au nombre de caractères.
# on est moins étonné de la moyenne quand on voit que 50% de l'échantillon
# a 6 caractères.

sum(dat$strength > 10)
# 15 mots de passe qui ont pour force plus de 10.

sum(dat$strength == 0)
# 30 mots de passe qui ont pour force: 0
# En tout (15 + 30)/500 = 9% des mots de passe ont une force de sécurité ne
# correspondant pas au niveau de la variable (1 a 10) tel que décrite dans le
# dictionnaire des données sur Github. Je décide de ne pas les enlever de ma
# banque de données, car ceux a 0 sont les plus simples (avec répétition de
# caractère continue) et ceux au-dessus de 10 sont les plus
# complexes (avec lettres et chiffres).
```

Graphique 1

```
graph1 <- ggplot(count_cat, aes(x = reorder(category, -n), y = n)) +
          geom_bar(stat = "identity", fill = "#FFBF71")+
          labs(x = " ", y = " ",
               title = "Les catégories de mauvais mot de passe") +
          theme_stata() +
          theme(plot.title = element_text(size = 20L, face = "bold"),
```

```
                axis.text.x = element_text(angle = 35,vjust = 1, hjust=1),
                axis.text.y = element_blank(),
                axis.ticks.y = element_blank())+
          geom_text(aes(label = n), vjust = -0.1)

suppressMessages(ggsave("img/graph1.png"))
```

Pour mon premier graphique, j'ai décidé de garder ça simple avec un graphique à barres qui représente la somme des mots de passe dans chaque catégorie. J'ai décidé d'arranger les catégories en ordre croissant par rapport à la somme pour une meilleure représentation visuelle qui permet de mieux les comparer. J'ai supprimé la description des axes puisque c'est déjà bien expliquer grâce au titre, au label(n) sur les barres et avec le nom des catégories. J'ai aussi donné un angle au catégories pour qu'elles soient lisible à la place de les poser à la verticale. Pour finir j'ai ajouté un geom_text qui indique la somme des mots de passe accordés à chaque catégorie.

Graphique 2

```
graph2 <- ggplot(data = dat , aes(x = type, y = strength))+
              geom_boxplot(fill = "#FFBF71")+
              labs(x = " ", y = "Niveau de sécurité",
                    title = "Type de caractère VS sa force")+
              ggthemes::theme_stata()+
            theme(plot.title = element_text(size = 20L, face = "bold"),
                axis.text.y = element_text(angle = 0,vjust = 1, hjust=1))

suppressMessages(ggsave("img/graph2.png"))
```

J'ai pris un geom_boxplot pour mieux illustrer l'écart de force/sécurité du mot de passe entre les différents types. Encore une fois j'ai supprimé la description des variables puisque je trouvais cela assez clair avec le titre et les axes.

Graphique 3

```
class(dat$type)
# Je constate que type est une variable "character".

dat <- dat %>%
      mutate(type=as.factor(type))
# J'ai convertis la variable type en facteur pour pouvoir
# faire le graphique suivant.


graph3 <- dat %>%
          ggplot(aes(x=(password_len),
                     y=guess_crack_sec,
                     color = type == "Lettres & chiffres"))+
          geom_point()+
```

```
          scale_color_manual(values = c("black", "orange"))+
          geom_smooth(formula = y ~ log(x), method = "lm", color = "#FFBF71")+
          scale_y_continuous(trans = "log2",
                             labels = c("0","8 min", "18 heures", "13 semaines", "34 ans"))+
          labs(x = "Nombre de caractère",
               y = "Temps sur échelle logarithmique",
          title = "Longueur VS temps de décryption en ligne (échelle log2)",
          color='Composé de chiffres et lettres')+
          ggthemes::theme_stata() +
          theme(plot.title = element_text(size = 14L, face = "bold"),
                axis.text.y = element_text(angle = 0,vjust = 0.2, hjust=1))

suppressMessages(ggsave("img/graph3.png"))
```

J'ai commencé par comparer la longueur des mots de passe au temps de décryptage avec un geom_point. J'ai remarquer que les temps de décryptage étaient dispropotionnés les un aux autres comme je l'indique au graphique 4 plus bas. C'est pour cette raison que j'ai positionné l'axe Y sur une échelle logarithmique à fin mieux voir l'effet du nombre de caractère sur le temps de décryption. J'ai même changé l'équivalence du temps en secondes pour des unités plus représentatives et faciles à comprendre. Le texte de l'axis Y a aussi légèrement été ajusté pour agencer l'esthétique.

J'ai aussi ajouté un geom_smooth avec un modèle linéaire correspondant à la modification de l'axe Y en log pour mieux démontrer la tendance. Je n'ai pas choisi les autres types de modèle comme "glm", "gam" ou "loess" à cause de la taille de mon échantillon et des observations incluses. De plus, j'ai ajouté une troisième variable correspondant aux mots de passe qui sont constitués de chiffres et de lettres (puisqu'ils sont plus sécuritaires). Ces observations ont été coloriée par des points orange pour garder les couleurs que j'ai choisies depuis le début.

Pour finir j'ai décidé de bien indiquer la description de l'axe des X, Y(avec échelle) et la légende puisqu'il y a beaucoup d'information sur le graphique.

Graphique sans échelle logarithmique du temps

```
graph4 <- dat %>%
          ggplot(aes(x=(password_len), y=guess_crack_sec))+
                   geom_point()

suppressMessages(ggsave("img/graph4.png"))
```

Le graphique 4 a été créé pour expliquer mon raisonnement derrière le choix de l'échelle logarithmique pour l'axe des Y. Je pense que ce graphique est encore moins clair et moins intuitif que le graph3.