Analyser ses conversations WhatsApp avec R

Enjeux de l'accès aux données et introduction au text mining

Vestin Hategekimana

08.11.2022

Table des matières I

Présentation

moodle

Classe: TO2022

Qui suis-je?

Vestin Hategekimana

- Assistant-doctorant en démographie (IDESO)
  - Institut de socioéconomie et de démographie (UNIGE)
- Migration et Mobilité en Suisse en temps de crise
- Passionné par les sciences des données (computational social sciences)

WeData

WeData

"Des stats et du code!"

Groupe étudiant ayant une passion pour le code et les statistiques: cours et contenu!

- Notre site: https://wedata.ch/
- Notre chaîne YouTube: WeData
- Instagram: @wedata_unige

Questions d'introduction

Lien votamatic

Code: GHCQ

Pour ce cours

Note

Pour le bon déroulement du cours, sachez que:

1. Je suis un amateur passionné
2. C'est mon premier cours à l'université
3. C'est la première fois que j'enseigne le sujet
4. Ce n'est pas un cours formel
5. Vous pouvez partir à n'importe quel moment
6. Vous pouvez m'interrompre si vous avez une question

WhatsApp et ses données

WhatsApp

Qu'est-ce que c'est WhatsApp? (1)

Qu'est-ce que c'est WhatsApp? (2)

En quelques points

1. Application de discussion (message, téléphone, vidéo)
2. Existe depuis 2009
3. Depuis 2020 plus de 2 milliards d'utilisateurs
4. Acheter par Meta (Facebook) en 2014

La panne 04.10.2021

Sécurité de WhatsApp

Polémique et chiffrement

- Les messages dans WhatsApp sont chiffrés
- Depuis 2021 données transférées à Meta (Facebook)
  - Avant ce n'était pas obligatoire pour utiliser l'application
  - Suscité une polémique
  - Seulement les Métadonnées (pas les conversations)

Les données que WhatsApp a sur moi

- Possible de faire la demande sur l'application
- Principalement:
  - Données utilisateur public, appareils et n° de téléphone, etc.
  - Moins grave que Facebook

Article du Monde

Mais, est-ce vraiment sûr?

Quels sont les risques?

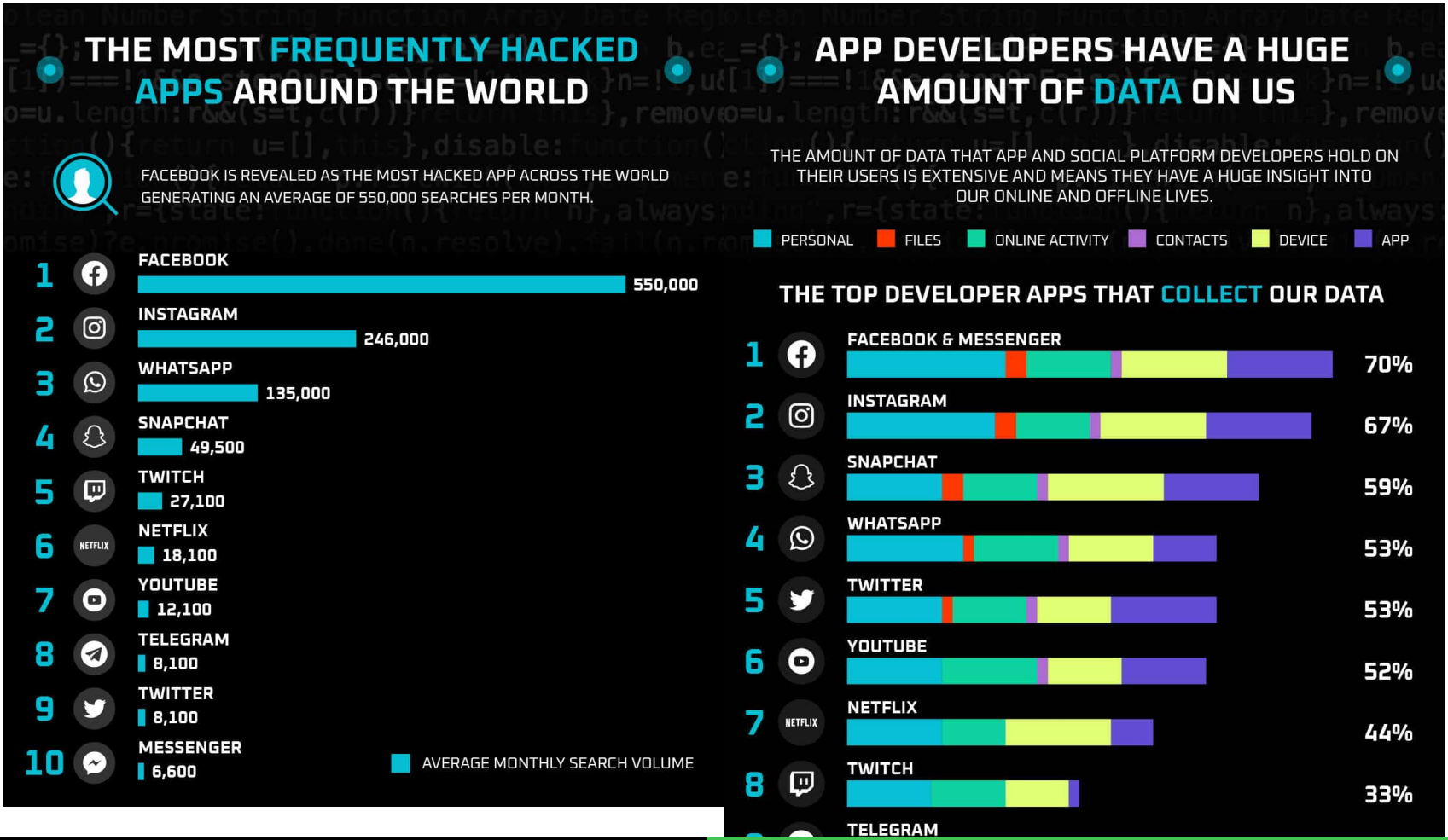
Un problème de consentement:

- Les messages signalés sont accessibles à l'entreprise
  - Après signalement les 5 derniers messages sont envoyés
  - La personne signalée n'est pas au courant
  - Les messages passent par un algorithme de trie et sont envoyé vers des humains s'ils sont difficiles à analyser
- Les métadonnées: Le cas de Natalie May Edward
- Il est possible d'exporter les conversations

Article de numerama

Article ProPublica

Article de TechSchielder sur les hackers



Méthodes utilisées par les hackers pour whatsapp

Sources pour information:

- Article TechSchielder (graphiques)
- 9 Ways Your WhatsApp Messages Can Be Hacked
- Can You Get Hacked Through WhatsApp?

Exercice: Obtenir ses données de conversation whatsapp

Exercice: Obtenir ses données de conversation whatsapp

Suivre les étapes du site: Étapes

1. Ouvrez la discussion individuelle ou de groupe.
2. Appuyez sur Plus d'options > Plus > Exporter discussion.
3. Choisissez d'exporter la discussion avec ou sans fichiers médias.

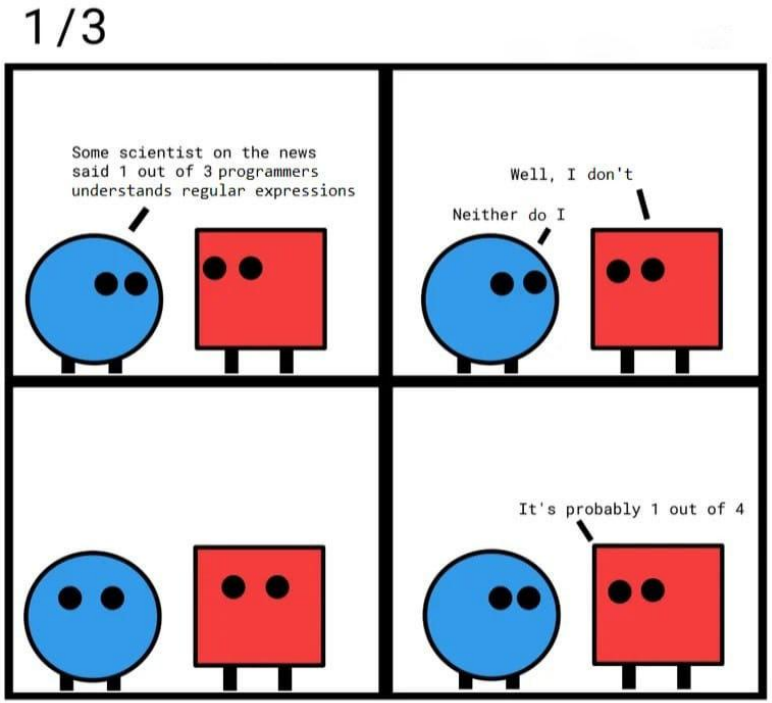
C'est tout!

Introduction au REGEX

Qu'est-ce que le regex?

Qu'est-ce que le regex?

Language utilisant un processus de sélection de chaînes de caractère basé sur la structure.



Pourquoi apprendre le regex

- Utiliser dans de nombreux langages et logiciels
- Très pratique dans la gestion de données textuelles
- Un super outil pour le text mining
  - Élargie les possibilités d'analyse
  - Permet un nettoyage et une recherche en profondeur
  - Automatise des processus

Dans ce cours le regex permet de nettoyer/formater nos données et de faire des recherches dedans. Les bases seront surtout enseignées pour comprendre le code ensuite.

Où trouve-t-on le regex?

- Dans les langages de programmation (JavaScript, R, Python, etc.)
- Dans les éditeurs de code (Notepad++, VS Code, RStudio, etc.)
- Dans les navigateurs avec extension (Firefox, Chrome, etc.)
- Etc.

Généralement la commande ctrl/cmd + F ou ctrl/cmd + H

Exemple de regex

Collecter

```
# Collecter les numéros de téléphonne Suisse
"^(\+41|0|00)\s?\d{2}\s?\d{3}\s?\d{2}\s?\d{2}$"
# Collecter les adresses emails
"\w{1,15}\.?\@\w{1,15}\.\w{1,3}"
```

Exemple de regex (1)

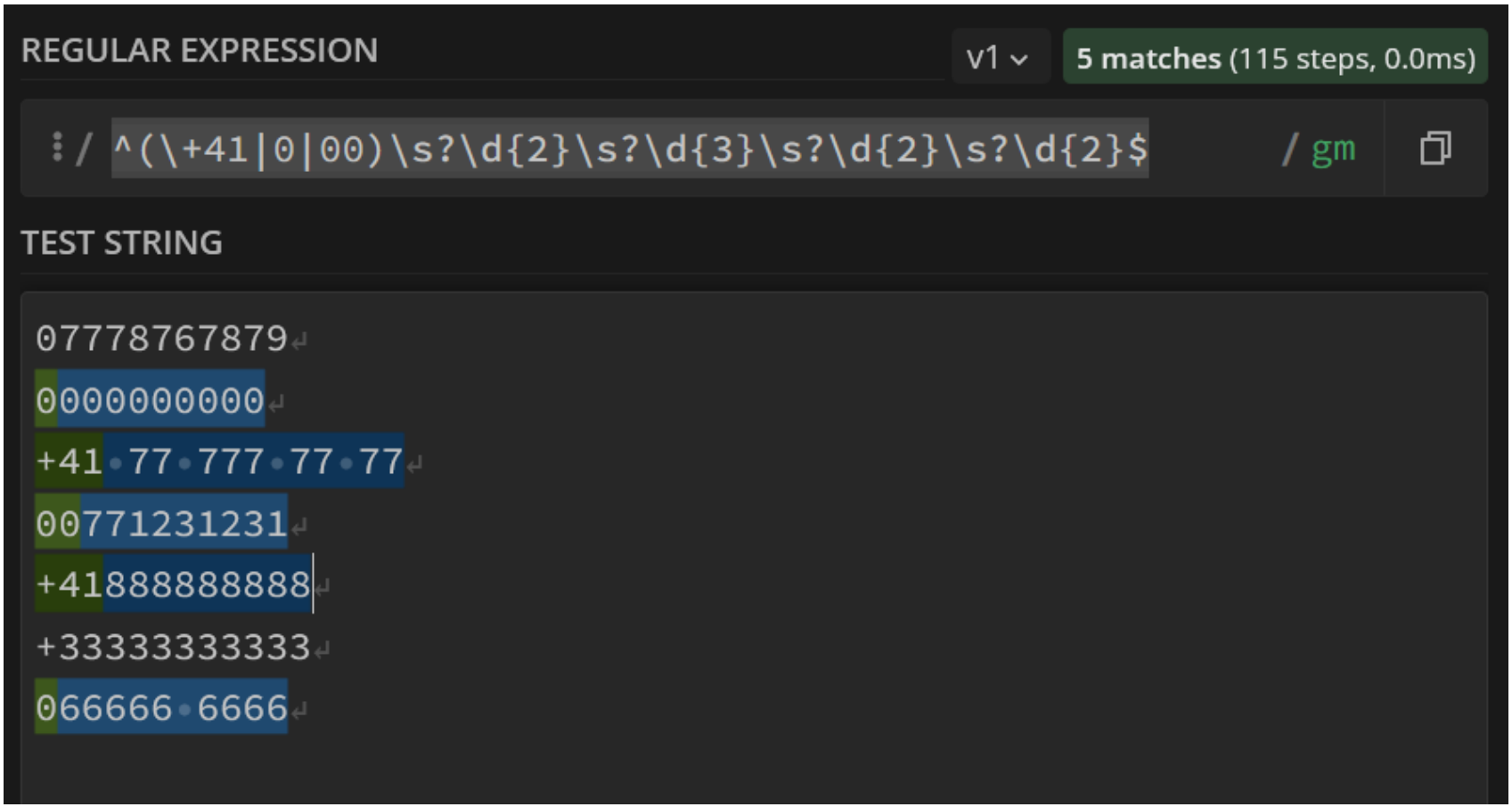


Figure 2: Source: https://regex101.com

Exemple de regex (2)

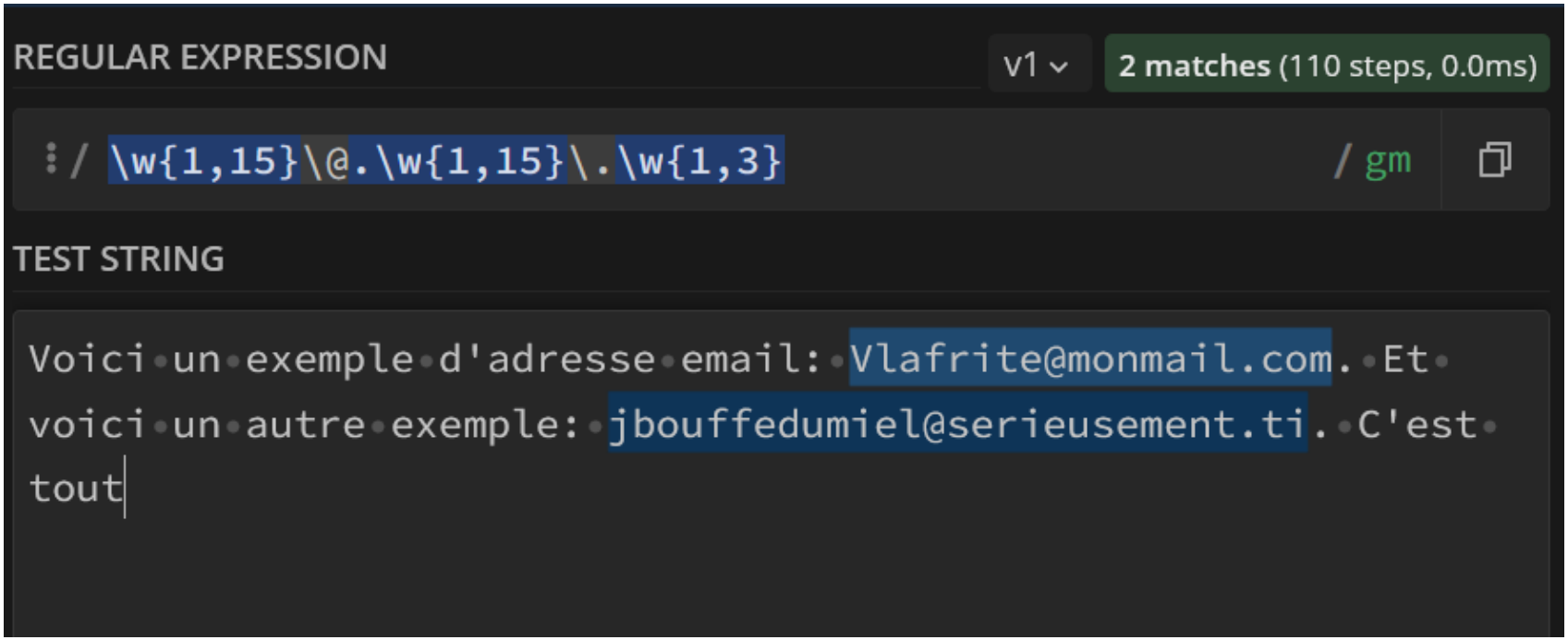


Figure 3: Source: https://regex101.com

Ressources pour le regex

Cheatsheets,

- quickref
- cheatography

Pratique:

- regexone
- regex101

Exercices

Exercices

1. Télécharger les données d'entrainement sur moodle
2. Aller sur regex101: https://regex101.com/
3. Copier-coller le texte dans regex101
4. Suivez le guide!

Sélectionner des caractères

Caractère: chiffre ou lettre

```
# N'importe quel caractère
"\w"

# N'importe quelle série de 2 caractères
"\w\w"
"\w{2}"

# N'importe que série de 3 caractères ou plus
"\w{3,}"

# Entre 3 et 5 caractères
"\w{3,5}"
```

Sélectionner des chiffres

```
# N'importe quelle chiffre
"\d"

# N'importe quelle série de 2 chiffres
"\d\d"
"\d{2}"

# N'importe que série de 3 chiffres ou plus
"\d{3,}"

# Entre 3 et 5 chiffres
"\d{3,5}"
```

Sélectionner des lettres

```
# Sélectionner n'importe quelle lettre
"[a-z]"

# Sélectionner n'importe quelle lettre de a à p
"[a-p]"

# Sélectionner toutes les lettres "a", "b", "c", "f", "t",
"[abcftw]"

# Sélectionner n'importe quelle série de 2 lettres
"[a-z]{2}"

# etc...
```

Sélectionner des chiffres bis

```
# Sélectionner les chiffres entre 4 et 8
"[4-8]"

# Sélectionner des chiffres spécifiques
"[4167]"

# Sélectionner des chiffres et des lettres
"[a-z0-9]"
```

Sélectionner n'importe quoi

```
# N'importe quoi (un élément!)
"."

# N'importe quelle série de 2
".."
".{2}"

# N'importe quel série de 3 chiffres ou plus
".{3,}"

# Entre 3 et 5 chiffres
".{3,5}"
```

Sélectionner des éléments particuliers

```
# Sélectionner tous les "5", "a", "t", ":"
"[5at:]"

# Sélectionner les espaces
"\s"

# Sélectionner les tabulations
"\t"

# Sélectionner les nouvelles lignes
"\n"

# On peut aussi compter tout ça (ex. entre 3 et 8 espaces)
"\s{3,8}"
```

Le dénombrement

exemple avec le "." pour n'importe quel caractère

```
# Zéro ou un élément
".?"

# Zéro ou plus d'éléments
".*"

# Un ou plus d'éléments
".+"

# Exactement 3 éléments
".{3}"
```

Le dénombrement

```
# Trois ou plus d'éléments
".{3,}"

# Entre 5 et 10 éléments
".{5,10}"
```

Sélectionner le contraire

```
# Les contraires
"\W"  "\D"  "\S"

# Cas spécifique, tout sauf ce qu'il y a entre crochet
"[^wert]"
```

Sélectionner les caractères spéciaux

Il existe des caractères qui sont utilisés pour la syntaxe du regex. Il est donc nécessaire d'ajouter un "" devant pour que le langage sache que c'est ce caractère spécial qu'on souhaite et non une syntaxe.

```
# Les caractères spéciaux
"\."  "\,"  "\?"  "\^"  "\$"
"\("  "\["  "\{"  "\|"  "\\"
```

Début/Fin d'une sélection

Pour préciser où la recherche doit commencer. C'est plus spécifique.

```
# Début
"^"

# Fin
"$"

# Début et fin
"^$"

# L'élément sélectionner doit absolument commencer et fini
"^s.*s$"
```

Le contexte

On peut décider de sélectionner des éléments à condition qu'ils soient suivis ou précédés par un élément spécifié. L'élément suivant/précédent n'est pas pris dans la sélection.

```
# Tous les "a" suivis par un "s"
"a(?=s)"

# Tous les "a" qui ne sont pas suivis par un "s"
"a(?!s)"

# Tous les "er" précédé par "mang"
"(?<=mang)er"
```

Le contexte

```
# Tous les "er" qui ne sont pas précédés par "mang"
"(?<!mang)er"
```

"ou" et les groupes

Un groupe permet de focaliser/préciser le regex pour une partie et/ou de réutiliser cette partie dans le remplacement.

```
# Choisir "chien" ou "chat"
"chien|chat"

# Créer un groupe avec le début de tous les mots finissant
"([a-z]*)er$"

# Dans le remplacement: remplacer "er" par "é" (on réutilis
"\1é"

# Mot commençant par "c" ou "v" et finissant par "e"
"(c|v)[a-z]*e"
```

Enfin au bout des bases!

Avec les premiers éléments que nous avons vus. Nous avons suffisamment de bases pour nous lancer dans le regex pour des analyses. Ce n'est que le début, vous pouvez encore aller plus loin.

Dans le cadre de ce cours, ce que nous avons vu sera suffisant.

Passons aux choses sérieuses!!!

Exercice (0)

Télécharger et ouvrir les données d'entrainement (conversation WhatsApp)

0. Trouver les textes entre parenthèses
1. Trouver tous les numéros de téléphone (suisses et français)
2. Trouver toutes les adresses email
3. Trouver des mots de passe

Exercice (0) - Solution

Il n'existe pas qu'une seule solution parfaite!

```
# 0. Trouver les textes entre parenthèse
"\(.*\)" # avec les parenthèse
"(?<=\().*(?=\))" # sans les parenthèses

# 1. Trouver tous les numéros de téléphone (suisses et fran
"\+\d{2}\s(\d{3}|\d\s\d{2}\s\d{2})\s\d{2}\s\d{2}"

# 2. Trouver toutes les adresses emails
"\w+\.?\w+\@\w+\.\w+"

# 3. Trouver des mots de passe
"mdp|mot de passe|code"
```

Exercice (1)

Le format du texte est un peu "brouillon". Pourriez-vous supprimer tous les "sauts de lignes" et en ajouter seulement à la fin des phrases? - ***Ensemble***

Exercice (1) - Solution

Le format du texte est un peu "brouillons". Pourriez-vous supprimer tous les "sauts de lignes" et en ajouter seulement à la fin des phrases?

```
# Dans un premier temps
"\n+" # Sélection
"\s" # Remplacement

# Dans un deuxième temps
"(?<=(\.|\?|\!))\s" # Sélection
"\n" # Remplacement
```

Exercice (2)

Il y a une erreur dans le texte. En réalité Marie Jane et Billy Boy sont de la même famille et le nom de famille est "Wilke". Pouvez-vous corriger cela? - ***5 minutes***

Exercice (2)

Il y a une erreur dans le texte. En réalité Marie Jane et Billy Boy sont de la même famille et le nom de famille est "Wilke". Pouvez-vous corriger cela? - 5 minutes

```
# Sélection
"(Billy|Marie)\s(Boy|Jane)"

# Remplacement
"\1\sWilke"
```

Exercice (3)

Le format des dates dans le texte n'est pas le bon (jj.mm.aa). Pouvez-vous le transformer dans le format suivant: jj-mm-aaaa (ex. 12-03-2020)? - ***5 minutes***

Exercice (3)

Le format des dates dans le texte n'est pas le bon (jj.mm.aa). Pouvez-vous le transformer dans le format suivant: jj-mm-aaaa (ex. 12-03-2020)?

```
# Sélection
"(\d{2})\.(\d{2})\.(\d{2})"

# Remplacement
"\1-\2-20\3"
```

Exercice (4)

Il y a des numéros de téléphone avec des espaces et des zéros. Pouvez-vous vous assurer que les numéros de téléphone suisses commencent par +41 et les numéros français par +33 tous en retirant les espaces? (ex. +41777777777) - ***5 minutes***

Note: *ici tous les numéros qui ne commencent pas par "+33" sont considérés comme suisses.*

Exercice (4)

```
# Pour les numéros suisses
# Sélection
"\+[^3]{2}\s(\d{3})\s(\d{2})\s(\d{2})"

# Remplacement
"+41\1\2\3"

# Pour les numéros français
# Sélection
"\+[3]{2}\s(\d)\s(\d{2})\s(\d{2})\s(\d{2})"

# Remplacement
"+41\1\2\3\4"
```

Importer et nettoyer les données

Les projets dans R

Les projets dans R

Les avantages:

- Permets d'avoir des dossiers séparés avec leur propre dossier de travail, historique et source pour les documents.
- Contribue aussi à la reproductibilité de votre travail

Tutoriel

Mise en place du projet

Mise en place du projet

1. Commencer un projet sur R
2. Mettre les données d'entrainement dans le projet

Tidyverse

Tidyverse

Dans le cadre de ce cours, nous allons nous baser sur les packages du {tidyverse}. C'est un ensemble de packages partageant une philosophie commune "tidyformat" qui sont dédié au nettoyage et a la manipulation de données.

Sources: https://www.tidyverse.org/

> *Tidyformat: un dataframe avec en ligne des individus et en colonne des variables.*

dplyr / tidyr / stringr / ggplot2

Nous allons principalement utiliser les packages {dplyr} et {tidyr}.
Mais nous nous aiderons des d'autres packages plus spécifiques.

Installation des packages

Avec cette fonction, on installe tous les packages de tidyverse, les 8 de base plus des bonus (ex. rvest, lubridate, dtplyr, etc.).

```
install.packages("tidyverse")
```

Préférences

Chargement des packages

Vous pouvez charger les packages comme vous le souhaitez. Cependant, la première méthode est plus simple. Mais elle charge aussi d'autres packages qu'on n'utilise pas forcément et n'est pas informative sur les packages utilisés.

```
# Charger tout
library(tidyverse)
```

```
# Être sélectif
library(dplyr)
library(tidyr)
library(stringr)
library(ggplot2)
library(forcats)
```

Import/Nettoyage

Importer les données

Tip

Vous pouvez bénéficier de l'autocomplétion en appuyant sur la touche "TAB" à l'intérieur des guillemets.
Forcer l'encoding en UTF-8 est nécessaire pour la lisibilité du texte.

```
texte <- readLines("whatsapp_conv.txt",
                   encoding = "UTF-8")
class(texte)
```

```
[1] "character"
```

Contenu

Nous avons un vecteur de texte

```
# Contenu
head(texte)
```

```
[1] ""
[2] "14.05.20 à 16:58 - Les messages et les appels sont ch
[3] "14.05.20 à 16:38 - +00 000 00 00 a créé le groupe \"A
[4] "14.05.20 à 16:58 - +00 000 00 00 vous a ajouté(e)"
[5] "14.05.20 à 16:58 - +00 000 00 00: je taquinais"
[6] "14.05.20 à 16:58 - Jean Poutre: trop cool le group"
```

Longueur

Nous avons 240 éléments, donc 240 lignes. Cependant:

- Les lignes vides sont aussi comptées
- Il y a des messages qui sont coupés en morceaux à cause de saut à la ligne
- Il faut régler ça!

```
# Longueur
length(texte)
```

```
[1] 240
```

0. Se débarrasser de la ligne en trop

Il y a une ligne vide au départ en trop qu'on peut retirer:

```
# Retirer la première ligne
texte <- texte[-1]

head(texte)
```

```
[1] "14.05.20 à 16:58 - Les messages et les appels sont ch
[2] "14.05.20 à 16:38 - +00 000 00 00 a créé le groupe \"A
[3] "14.05.20 à 16:58 - +00 000 00 00 vous a ajouté(e)"
[4] "14.05.20 à 16:58 - +00 000 00 00: je taquinais"
[5] "14.05.20 à 16:58 - Jean Poutre: trop cool le group"
[6] "14.05.20 à 16:59 - +33 3 33 33 33 33: Oh ok, pourquoi
```

1. Joindre en un bloc de texte

! Important

Il y a un espace dans le paramètre "collapse", pour espacer chaque vecteur (donc ligne) par un espace dans le bloc final

```
texte <- paste(texte, collapse = " ")
```

Regex dans R

Nous allons séparer le bloc de texte en différentes parties. Chaque partie représentera un message. Afin de faire cela, nous devons faire la séparation en nous basant sur l'élément qui suit un espace (ceux que nous avons créés).

! Important

Lorsque nous utilisons du regex dans R il faut ajouter un \ supplémentaire à chaque \ utilisé.

2. Dernière correction

```
texte <-
  str_split(texte,"\\s(?=\\d{2}\\.\\d{2}\\.\\d{2})") %>%
  unlist() %>% # transformer en vecteur
  str_trim() # retirer les espaces en trop

head(texte)
```

```
[1] "14.05.20 à 16:58 - Les messages et les appels sont ch
[2] "14.05.20 à 16:38 - +00 000 00 00 a créé le groupe \"A
[3] "14.05.20 à 16:58 - +00 000 00 00 vous a ajouté(e)"
[4] "14.05.20 à 16:58 - +00 000 00 00: je taquinais"
[5] "14.05.20 à 16:58 - Jean Poutre: trop cool le group"
[6] "14.05.20 à 16:59 - +33 3 33 33 33 33: Oh ok, pourquoi
```

Pas assez sécurisé

Note

Ici nous n'avons pas précisé la séparation (seulement les espaces suivis d'une date) parce qu'il manquait de l'espace. Cela pourrait être un problème si au milieu d'un message il y avait un espace suivi d'une date (ce n'est pas rare). Nous pouvons préciser notre regex en ajoutant une plus longue condition.

Format: Dataframe

Tibble

Nous allons utiliser un format de data frame spécifique dans R {tibble} qui est assez informatif. Nous mettons le tout dans un objet que nous appelons “whatsapp”.

```
whatsapp <- tibble(texte)

whatsapp
```

```
# A tibble: 195 x 1
   texte
   <chr>
 1 "14.05.20 à 16:58 - Les messages et les appels sont chi
 2 "14.05.20 à 16:38 - +00 000 00 00 a créé le groupe \"Am
 3 "14.05.20 à 16:58 - +00 000 00 00 vous a ajouté(e)"
 4 "14.05.20 à 16:58 - +00 000 00 00: je taquinais"
```

Colonnes

Maintenant, nous désirons créer des colonnes avec les informations suivantes:

1. date (et heure)
2. nom
3. texte

Pour faire ça, nous allons utiliser la fonction `separate()` du package {tidyr}. Qui permet de séparer une colonne de texte en plusieurs colonnes. Il faut donc spécifier le(s) séparateur(s) à utiliser et les noms de colonnes désirés.

Séparation

```
whatsapp <-
  whatsapp %>%
  separate(texte, c("date", "nom", "texte"),
           sep = "(\\s\\-\\s|\\:\\s)")

glimpse(whatsapp)
```

```
Rows: 195
Columns: 3
$ date  <chr> "14.05.20 à 16:58", "14.05.20 à 16:38", "14.0
$ nom   <chr> "Les messages et les appels sont chiffrés de
$ texte <chr> NA, NA, NA, "je taquinais", "trop cool le gro
```

Valeurs manquantes

Nous avons des `NA` dans nos données. Ce sont les lignes qui ne respectent pas le format (annonces). Nous pouvons les retirer avec `na.omit()` (fonction de base) ou `drop_na()` du package {tidyr}.

```
# Base
whatsapp <-
  na.omit(whatsapp)
```

```
# Tidyr
whatsapp <-
  drop_na(whatsapp)
```

Format de date (1)

```
whatsapp
```

```
# A tibble: 187 x 3
   date                nom                 texte
   <chr>               <chr>               <chr>
 1 14.05.20 à 16:58    +00 000 00 00       "je taquinais"
 2 14.05.20 à 16:58    Jean Poutre         "trop cool le group"
 3 14.05.20 à 16:59    +33 3 33 33 33 33   "Oh ok, pourquoi pas'
 4 14.05.20 à 17:00    +11 111 11 11       "pourquoi on fait un
 5 14.05.20 à 17:01    +11 111 11 11       "Parce qu'il manquait
 6 14.05.20 à 17:02    Marie Jane          "l'autre c'était pour
 7 14.05.20 à 17:02    Jean Poutre         "zéro pointé"
 8 14.05.20 à 17:03    Billy Boy           "Top le groupe. Après
 9 14.05.20 à 17:10    Ziao                "Bonjour bonjour"
```

Format de date (2)

Nous pouvons encore faire quelque chose pour les dates en les mettant au format `datetime`. Nous utilisons le package {lubridate} pour ça (installé de base avec {tidyverse}).

```
library(lubridate)
```

Nous utiliserons la fonction, `dmy_hm()` car le format de date est dans cet ordre (day/month/year/hour/minute).

date

```
whatsapp <-
  whatsapp %>%
  mutate(date = dmy_hm(date))

whatsapp
```

```
# A tibble: 187 x 3
   date                nom                texte
   <dttm>              <chr>              <chr>
 1 2020-05-14 16:58:00 +00 000 00 00      "je taquinais"
 2 2020-05-14 16:58:00 Jean Poutre        "trop cool le grou
 3 2020-05-14 16:59:00 +33 3 33 33 33 33 "Oh ok, pourquoi p
 4 2020-05-14 17:00:00 +11 111 11 11      "pourquoi on fait
 5 2020-05-14 17:01:00 +11 111 11 11      "Parce qu'il manqu
```

Fin du nettoyage

Nous avons enfin terminé avec le nettoyage. Nous sommes prêts à passer à l'analyse de données!

Bonus

```
# En une fois
whatsapp <-
  readLines("whatsapp_conv.txt", encoding = "UTF-8") %>%
  paste(collapse = " ") %>%
  str_split("\\s(?=\\d{2}\\.\\d{2}\\.\\d{2})") %>%
  unlist() %>%
  tibble(texte=.) %>%
  separate(texte, c("date", "nom", "texte"),
           sep = "(\\s\\-\\s|\\:\\s)") %>%
  drop_na() %>%
  mutate(date = dmy_hm(date))
```

Analyser les données

Introduction à ggplot2

Le package ggplot2

Qu'est-ce que c'est?

- L'un des packages de visualisation sur R les plus populaires
- Permets de créer des graphiques de manière modulaire (par couche)
- ggplot a permis la création d'autres extensions
- Il existe une interface graphique si vous débutez {esquisse}

Source:

- ggplot2
- Extensions
- Esquisse

Autres ressources ggplot2

R Graphics Cookbook, 2nd edition

Data Visualization

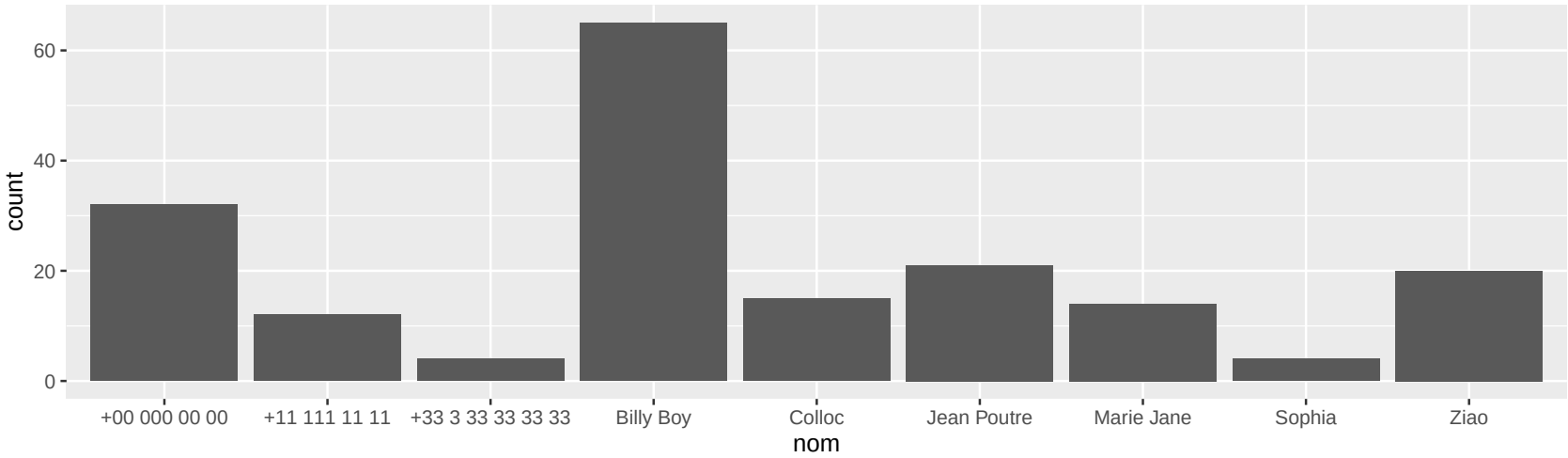
Fundamentals of Data Visualization

R for Data Science

Exemple (1)

Qui est la personne qui a écrit le plus de messages?

```
ggplot(whatsapp, aes(nom)) +
  geom_bar()
```



Graphique

Pour un graphique il faut au minimum:

- un jeu de donnée
- Une variable
- une géométrie (geom_)

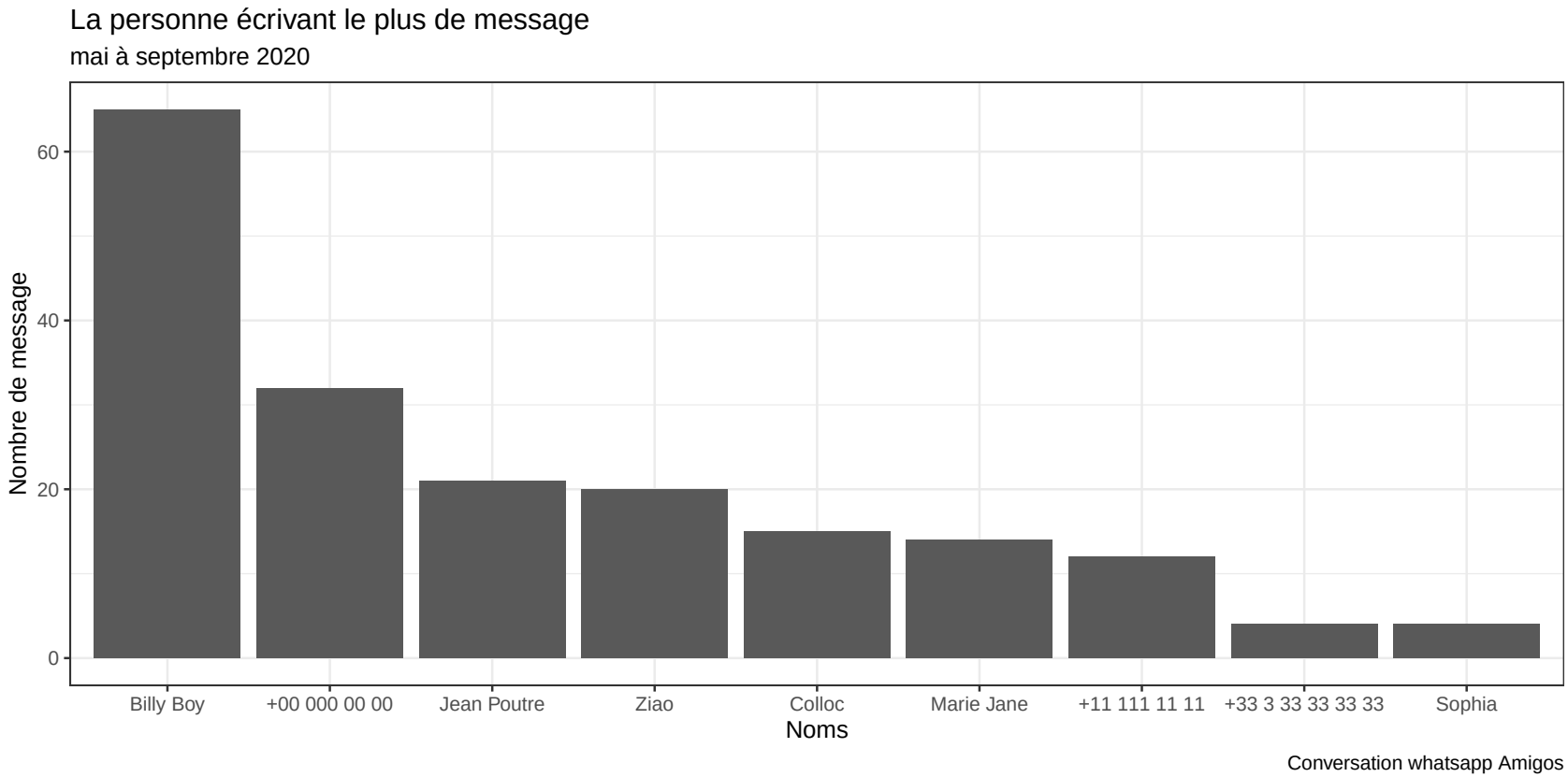
```
ggplot(data = *,
       mapping = aes(x, y, group, color, fill, etc.)) +
  geom_*(...)
```

Exemple (1) suite

On peut améliorer le graphique

```
ggplot(whatsapp, aes(fct_infreq(nom))) +
  geom_bar() +
  labs(title = "La personne écrivant le plus de message",
       subtitle = "mai à septembre 2020",
       x = "Noms",
       y = "Nombre de message",
       caption = "Conversation whatsapp Amigos") +
  theme_bw()
```

Exemple (1) suite



Esquisse (1)

On peut reproduire ce graphique avec esquisse.

Premièrement on installe le package

```
install.packages("esquisse")
```

Puis, on lance la fonction `esquisser()`

```
esquisse::esquisser()
```

Esquisse (2)

Aspect de l'interface visuelle esquisse

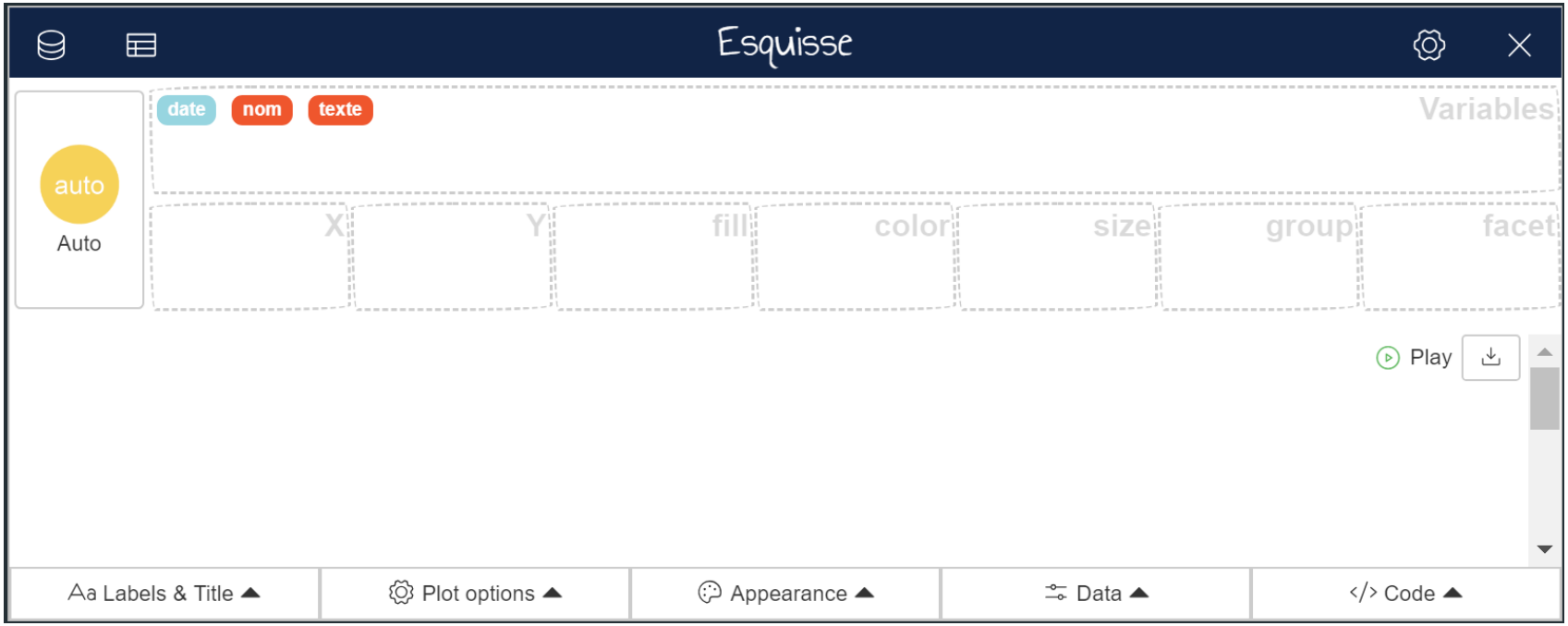


Figure 4: esquisse

Thème

Nous pouvons aussi fixer le thème en avance avec la fonction `theme_set()` du package ggplot2:

```
theme_set(theme_bw())
```

Mon thème favori est, `theme_bw()` mais vous êtes libre de choisir celui qui vous plait: liste des thèmes

Une fois que c'est fait, nous n'avons plus besoin de le préciser dans nos graphiques.

Exemple (2)

On peut faire la même chose, mais en utilisant cette fois {dplyr}. Dans un premier temps on agrège le nombre de messages par personne:

```
whatsapp %>%
  count(nom, sort = TRUE)
```

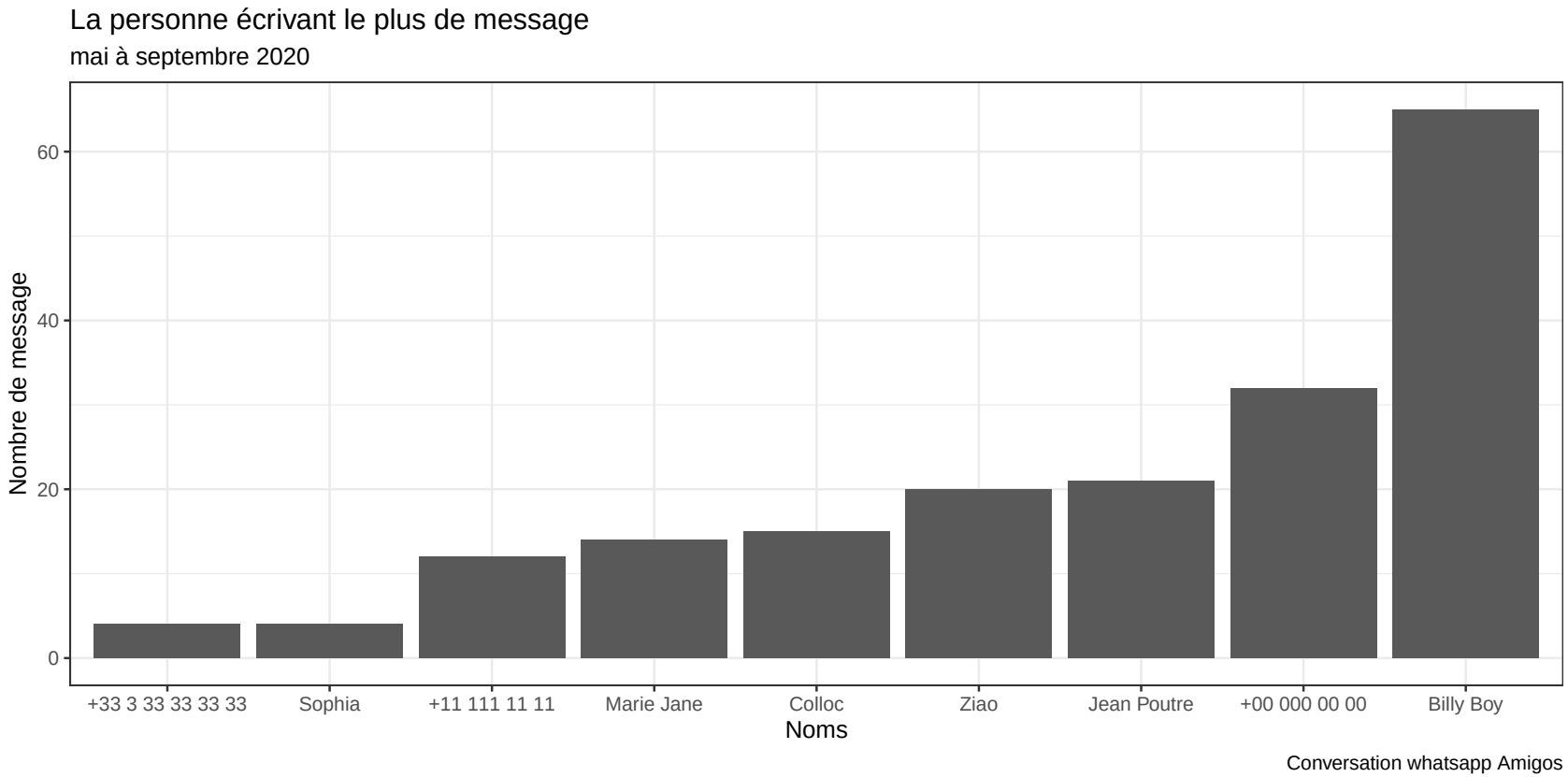
```
# A tibble: 9 x 2
  nom                    n
  <chr>              <int>
1 Billy Boy             65
2 +00 000 00 00         32
3 Jean Poutre           21
4 Ziao                  20
5 Collec                15
```

Exemple (2) suite

Puis on utilise `geom_col()` à la place de `geom_bar()`. Nom sera placé en x et n (le compte) en y. Pour réordonner, on peut utiliser soit la fonction `fct_reorder()` soit la fonction `reorder()`:

```
whatsapp %>%
  count(nom, sort = TRUE) %>%
  ggplot(aes(fct_reorder(nom, n), n)) +
  geom_col() +
  labs(title = "La personne écrivant le plus de message",
       subtitle = "mai à septembre 2020",
       x = "Noms",
       y = "Nombre de message",
       caption = "Conversation whatsapp Amigos")
```

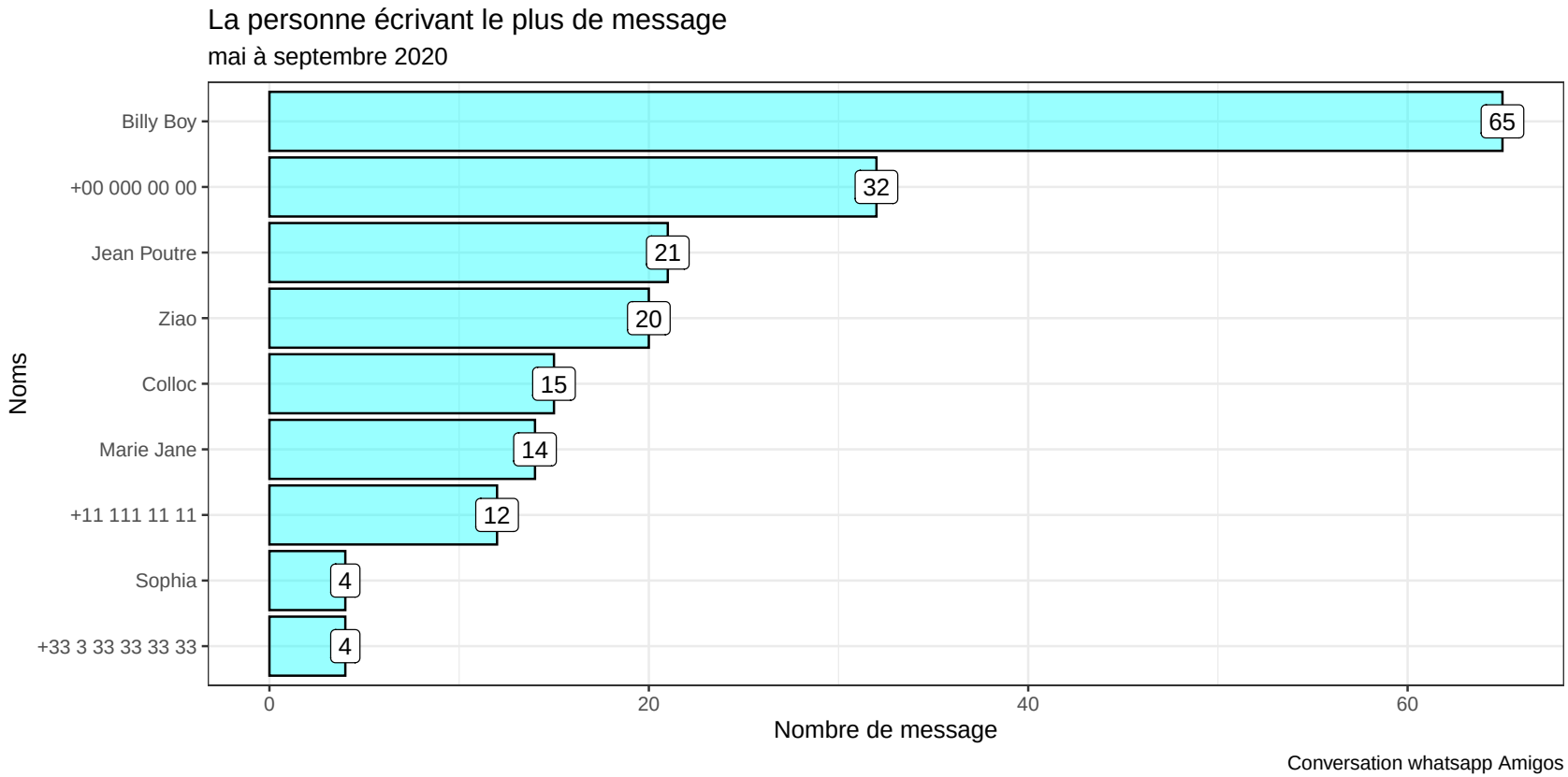
Exemple (2) suite



Bonus

```
whatsapp %>%
  count(nom, sort = TRUE) %>%
  ggplot(aes(fct_reorder(nom, n), n)) +
  geom_col(fill = "cyan", color = "black", alpha = 0.4) +
  geom_label(aes(label = n)) +
  labs(title = "La personne écrivant le plus de message",
       subtitle = "mai à septembre 2020",
       x = "Noms",
       y = "Nombre de message",
       caption = "Conversation whatsapp Amigos") +
  coord_flip()
```

Bonus


La personne écrivant le plus de message
mai à septembre 2020
Billy Boy
+00 000 00 00
Jean Poutre
Ziao
Colloc
Marie Jane
+11 111 11 11
Sophia
+33 3 33 33 33 33
65
32
21
20
15
14
12
4
4
Noms
0
20
40
60
Nombre de message
Conversation whatsapp Amigos

Exemple (3)

Nous pouvons aussi utiliser les données temporelles que nous avons. Il nous suffit de faire exactement la même chose avec `date`à la place de `nom`:

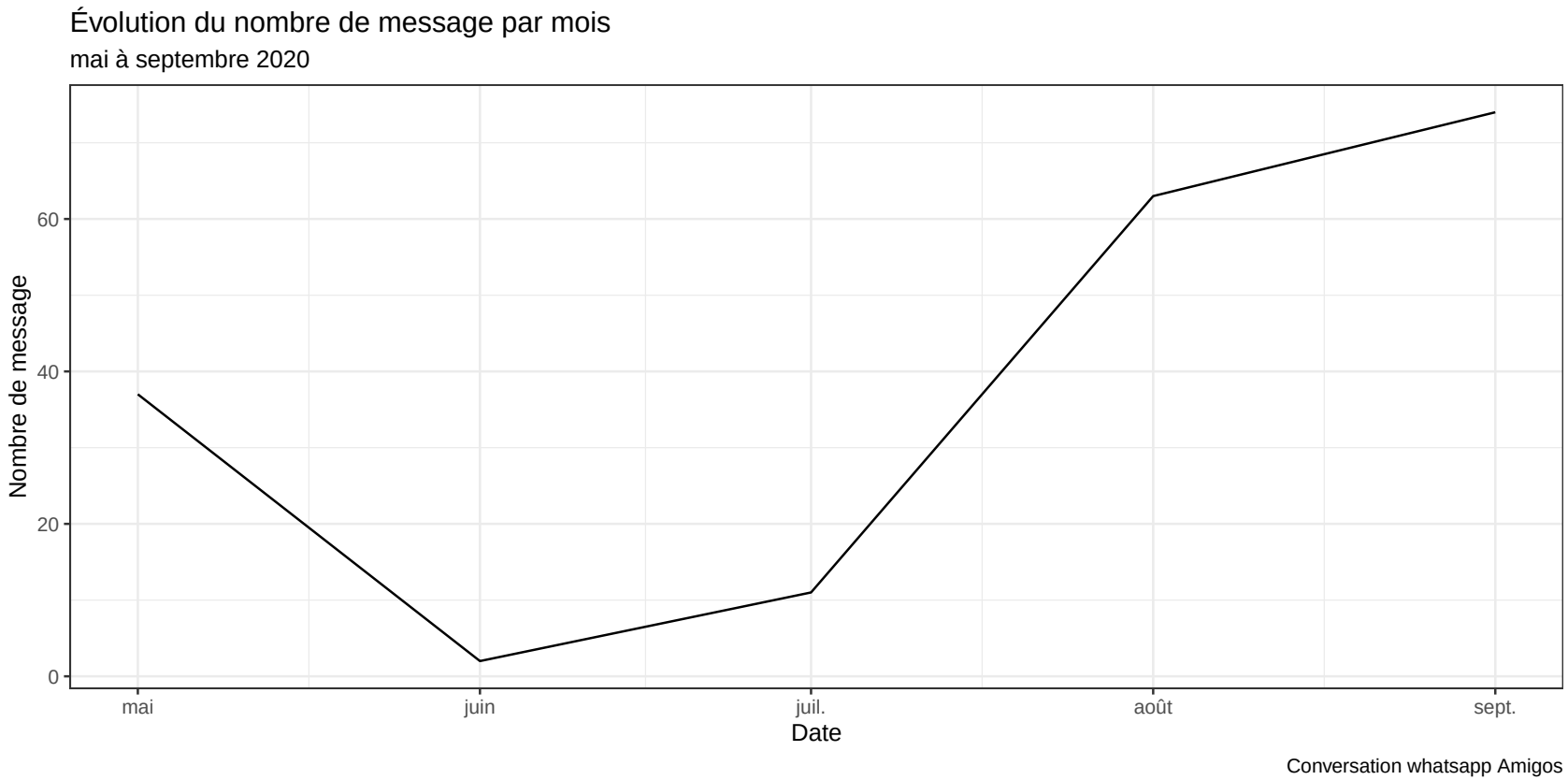
```
whatsapp %>%
  count(date, sort = TRUE) %>%
  ggplot(aes(date, n)) +
  geom_line() +
  labs(title = "Évolution du nombre de message",
       subtitle = "mai à septembre 2020",
       x = "Date",
       y = "Nombre de message",
       caption = "Conversation whatsapp Amigos")
```

Exemple (3)

Nous allons grouper les messages par mois avec la fonction `floor_date()` du package {lubridate}:

```
whatsapp %>%
  mutate(date = floor_date(date, unit="month")) %>%
  count(date, sort = TRUE) %>%
  ggplot(aes(date, n)) +
  geom_line() +
  labs(title = "Évolution du nombre de message",
       subtitle = "mai à septembre 2020",
       x = "Date",
       y = "Nombre de message",
       caption = "Conversation whatsapp Amigos")
```

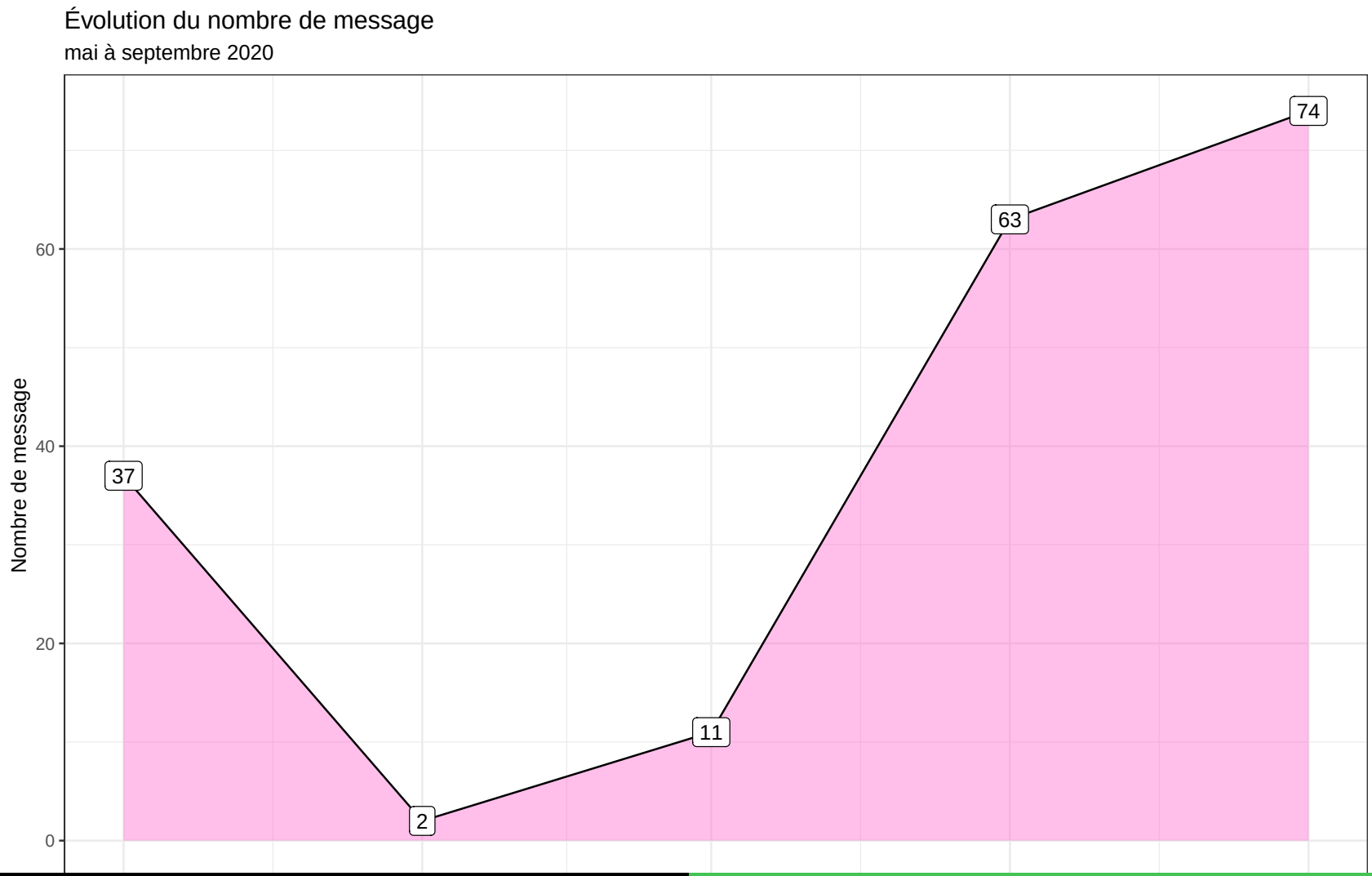
Exemple (3)


Évolution du nombre de message par mois
mai à septembre 2020
Nombre de message
0
20
40
60
mai
juin
juil.
août
sept.
Date
Conversation whatsapp Amigos

Bonus

```
whatsapp %>%
  mutate(date = floor_date(date, unit="month")) %>%
  count(date, sort = TRUE) %>%
  ggplot(aes(date, n)) +
  geom_area(fill = "#FF61CC", color = "black", alpha = 0.4)
  geom_label(aes(label = n)) +
  labs(title = "Évolution du nombre de message par mois",
       subtitle = "mai à septembre 2020",
       x = "Date",
       y = "Nombre de message",
       caption = "Conversation whatsapp Amigos")
```

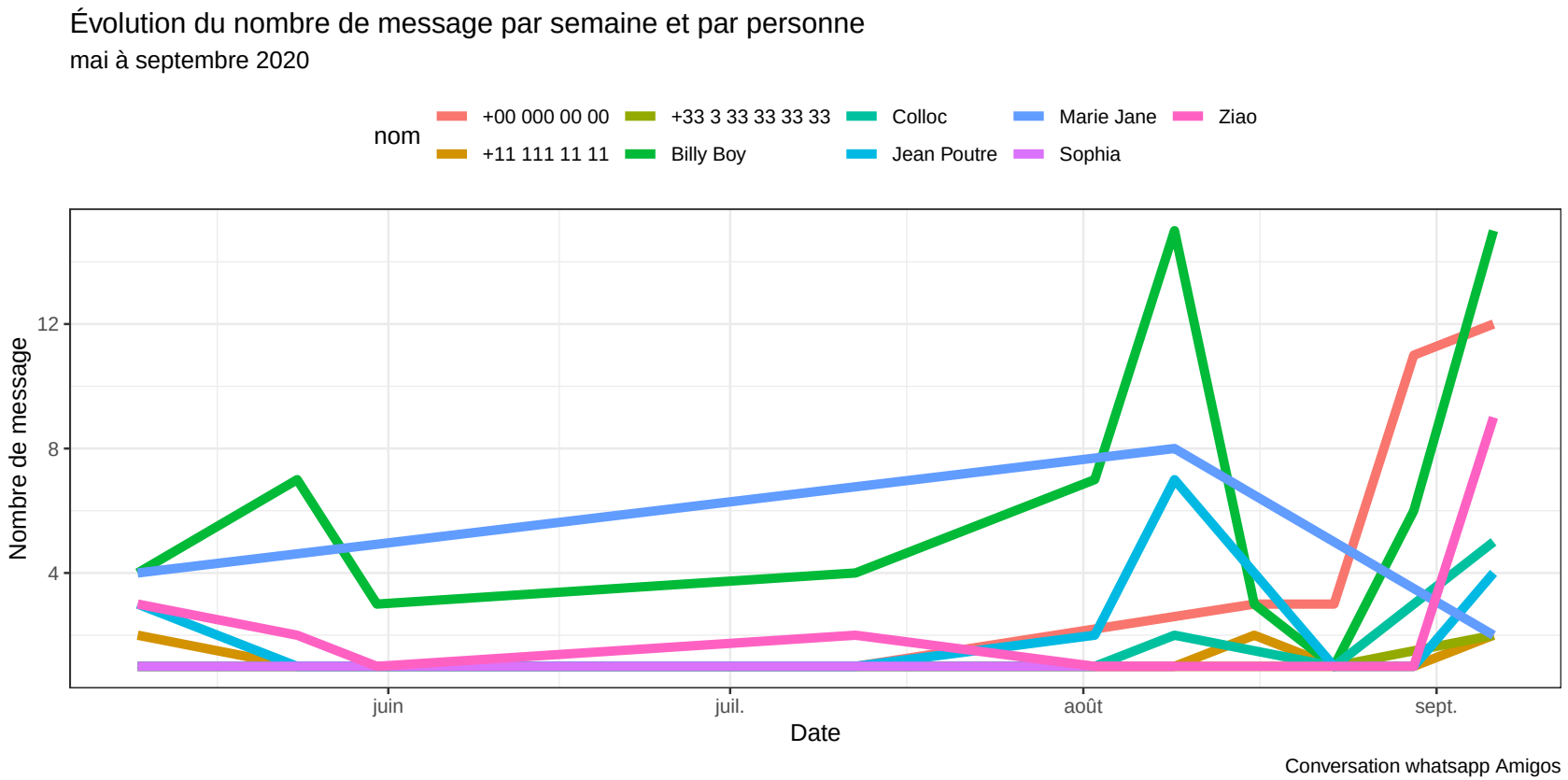
Bonus


Évolution du nombre de message
mai à septembre 2020
Nombre de message
0
20
40
60
37
2
11
63
74

Bonus

```
whatsapp %>%
  mutate(date = floor_date(date, unit="weeks")) %>%
  count(date, nom, sort = TRUE) %>%
  ggplot(aes(date, n, color = nom)) +
  geom_line(size = 2) +
  labs(title = "Évolution du nombre de message par semaine
       subtitle = "mai à septembre 2020",
       x = "Date",
       y = "Nombre de message",
       caption = "Conversation whatsapp Amigos") +
  theme(legend.position = "top")
```

Bonus



Exemple (4)

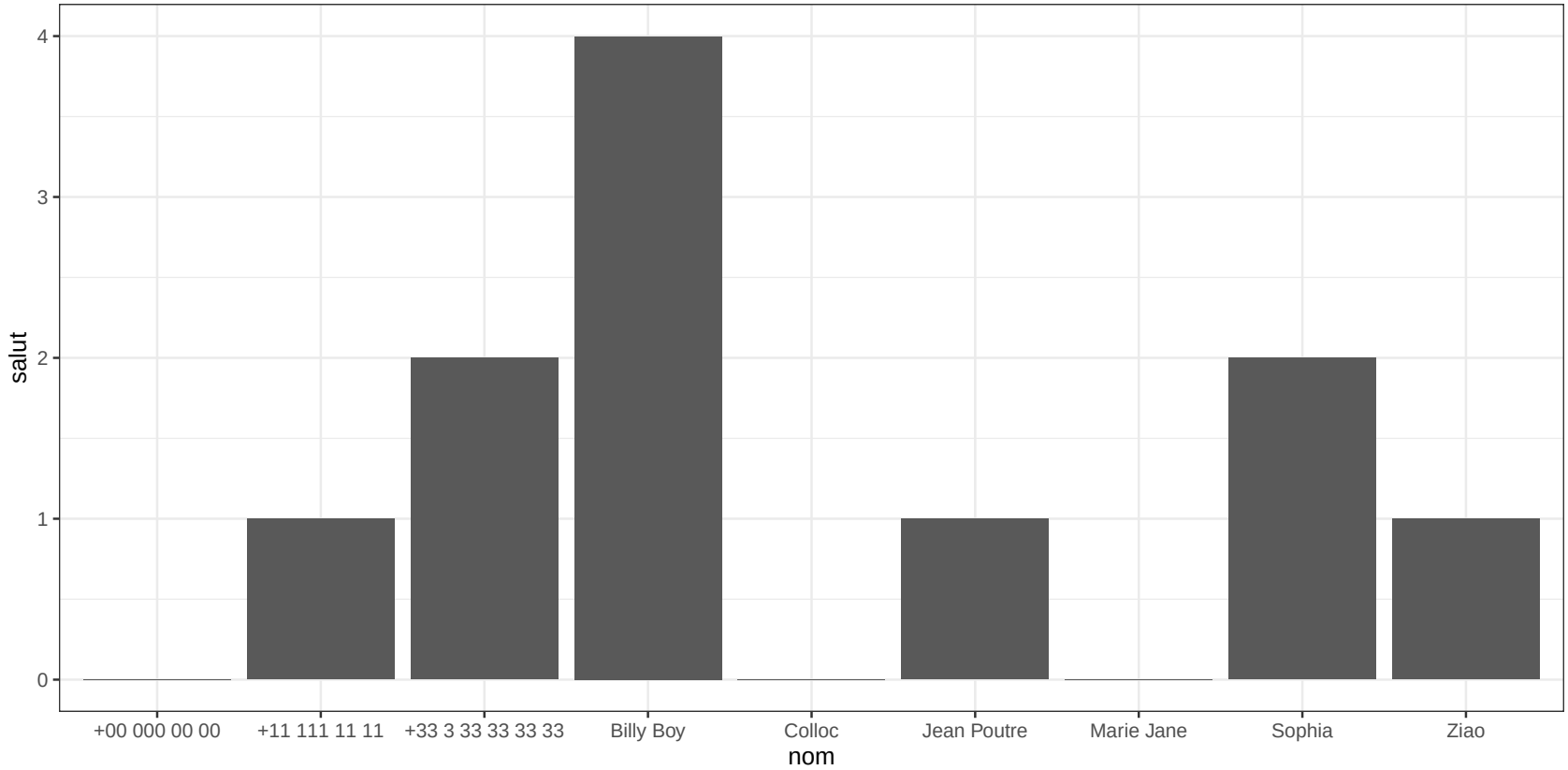
La fonction `stat_summary()` du package {ggplot2} peut aider à

Exemple (4)

Mais on peut faire la même chose avec les fonctions `group_by()` et `summarise()` du package {dplyr}:

```
whatsapp %>%
  mutate(texte = str_to_lower(texte),
         salut = str_count(texte, "(hi|hello|salut|hey)")) %>%
  group_by(nom) %>%
  summarise(salut = sum(salut)) %>%
  ggplot(aes(nom, salut)) +
  geom_col()
```

Exemple (4)


salut
0
1
2
3
4
+00 000 00 00
+11 111 11 11
+33 3 33 33 33 33
Billy Boy
Colloc
Jean Poutre
Marie Jane
Sophia
Ziao
nom

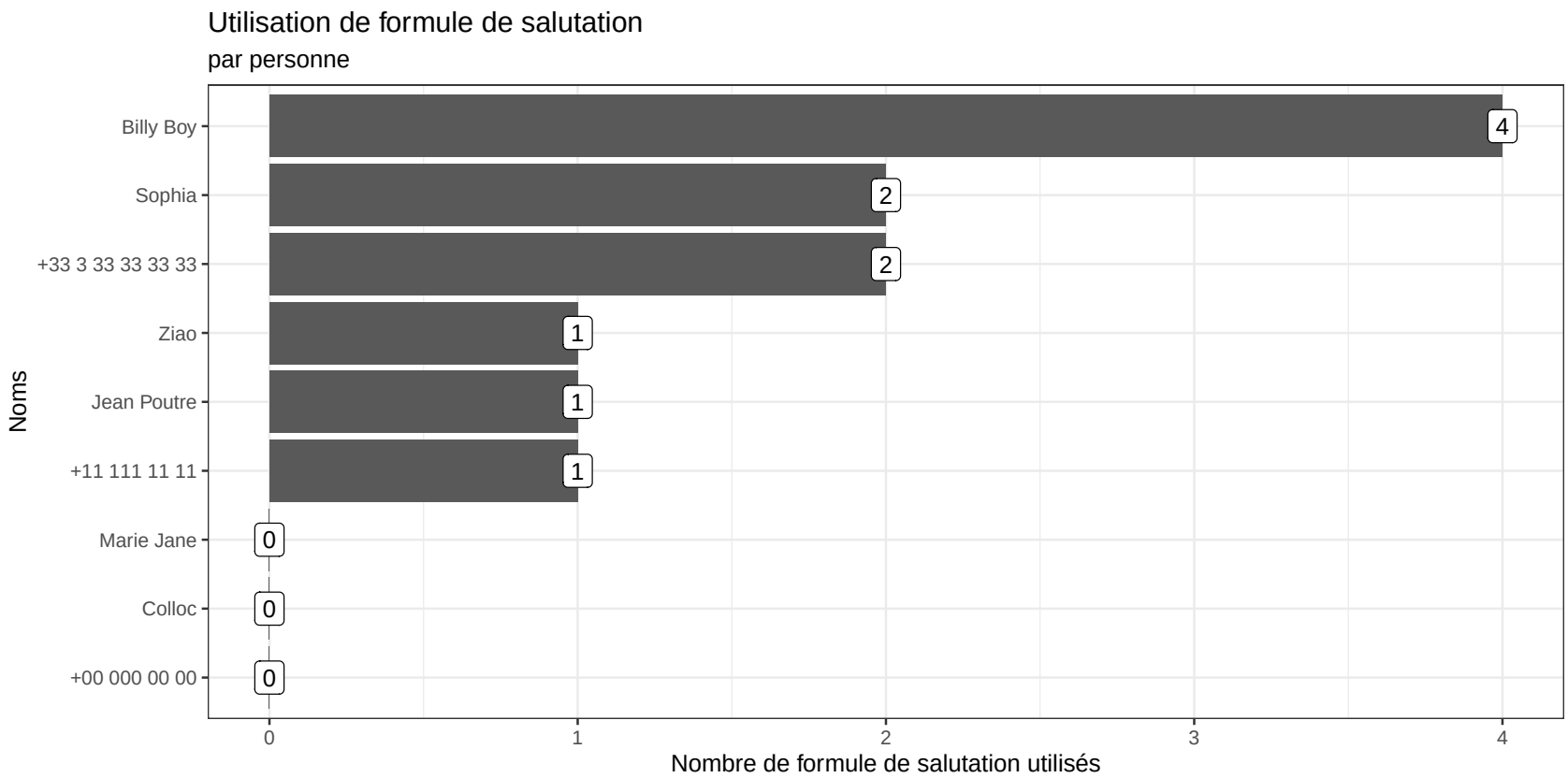
Améliorations (1)

```
whatsapp %>%
  mutate(texte = str_to_lower(texte),
         salut = str_count(texte, "(hi|hello|salut|hey)")) %>%
  ggplot(aes(fct_reorder(nom, salut, sum), salut)) +
  stat_summary(fun="sum", geom="bar", fun.y = "sum") +
  stat_summary(aes(label = ..y..), fun.y = "sum", geom = "t
  labs(title = "Utilisation de formule de salutation",
       subtitle = "par personne",
       y = "Nombre de formule de salutation utilisés",
       x = "Noms") +
  coord_flip()
```

Amélioration (2)

```
whatsapp %>%
  mutate(texte = str_to_lower(texte),
         salut = str_count(texte, "(hi|hello|salut|hey)")) %>%
  group_by(nom) %>%
  summarise(salut = sum(salut)) %>%
  ggplot(aes(fct_reorder(nom, salut), salut)) +
  geom_col() +
  geom_label(aes(label = salut)) +
  labs(title = "Utilisation de formule de salutation",
       subtitle = "par personne",
       y = "Nombre de formule de salutation utilisés",
       x = "Noms") +
  coord_flip()
```

Améliorations (3)



Autres types d'analyses possibles

Nous n'avons fait que quelques exemples d'analyses, mais on peut en faire d'autres. La seule limite est notre imagination.

- L'utilisation de certains termes/expressions
- Habitude d'utilisation de whatsapp (temps, volume, etc.).
- etc.

Cheatsheets utile pour aller plus loin,

- dplyr
- tidyr
- ggplot2
- lubridate
- forcats
- stringr

Suite: analyse textuelle

Nous avons vu une utilisation intéressante des données textuelles avec {stringr}, mais nous n'avons fait qu'effleurer l'univers des possibilités que nous offre l'analyse textuelle (text mining).

Voilà pourquoi nous introduisons deux autres packages intéressants:

- tidytext
- quanteda

Introduction à tidytext

Tidyverse

Comme son nom l'indique {tidytext} a pour objectif de travailler avec les données textuelles en gardant la philosophie du tidyverse: travailler avec des dataframes.

Lorsque nous travaillons avec tidytext, nous gardons des formats tabulaires ce qui n'est pas commun dans l'analyse textuelle qui utilise d'autres structures de données (matrice de terme document par exemple).

Julia Silge et David Robinson (travaillant à posit) ont publié un livre disponible gratuitement en ligne (sous forme de bookdown) qui introduit l'analyse textuelle avec {tidytext}

Source: tidytext

Charger le package

Il faut premièrement installer le package:

```
install.packages("tidytext")
```

Ensuite nous pouvons charger le package:

```
library(tidytext)
```

Nous pouvons commencer!

Nettoyage: Étapes

Avant de pouvoir exploiter nos données textuelles, nous devons passer par quelques étapes de nettoyage:

1. Tokenisation / Lemmatisation
2. Retrait des stopwords (mots courants)
3. Résumé numérique (principalement fréquence et tf_idf)

Tokenisation

Le but de la tokenisation est de diviser le texte en sous-groupes cohérents. Cela peut être des mots, des pairs de mots, des phrases ou des paragraphes.

Dans le cours d'aujourd'hui, nous ne nous intéresserons qu'aux mots.

```
whatsapp %>%
  unnest_tokens(mots, texte, token = "words")
```

```
# A tibble: 1,312 x 3
   date                nom                 mots
   <dttm>              <chr>               <chr>
 1 2020-05-14 16:58:00 +00 000 00 00       je
 2 2020-05-14 16:58:00 +00 000 00 00       taquinais
```

Tokenisation

```
# A tibble: 1,312 x 3
   date                nom                 mots
   <dttm>              <chr>               <chr>
 1 2020-05-14 16:58:00 +00 000 00 00       je
 2 2020-05-14 16:58:00 +00 000 00 00       taquinais
 3 2020-05-14 16:58:00 Jean Poutre         trop
 4 2020-05-14 16:58:00 Jean Poutre         cool
 5 2020-05-14 16:58:00 Jean Poutre         le
 6 2020-05-14 16:58:00 Jean Poutre         group
 7 2020-05-14 16:59:00 +33 3 33 33 33 33   oh
 8 2020-05-14 16:59:00 +33 3 33 33 33 33   ok
 9 2020-05-14 16:59:00 +33 3 33 33 33 33   pourquoi
10 2020-05-14 16:59:00 +33 3 33 33 33 33   pas
# ... with 1,302 more rows
```

Lemmatisation

La lemmatisation est un processus qui va plus loin que la tokenisation lorsqu'il s'agit de mot. En plus de diviser le text en mot, elle indique d'autres informations comme le rôle que le mot joue par exemple, appelées communément en anglais Part of Speech (POS).

Nous ne prendrons pas le temps de présenter des exemples de lemmatisation dans le cours puisque c'est un processus qui peut prendre du temps et demande de prendre en compte une autre approche.

Si vous souhaitez plus d'information, il y a les packages suivants qui pourraient vous intéresser:

- udpipe,
- spacyr (dépendance à python)

Stop words (1)

Nous pouvons retirer les stop words qui sont des mots très courants, mais qui ne servent pas forcément à l'analyse.

Il faut premièrement installer le package {stopwords}:

```
install.packages("stopwords")
```

Puis récupérer le style de stopwords que l'on souhaite (iso et snowball sont bien). Il y a deux méthodes.

```
stop <- stopwords::data_stopwords_stopwordsiso$fr
```

ou

```
stop <- stopwords::stopwords("fr", "stopwords-iso")
```

Stop words (2)

Puis nous pouvons utiliser cette nouvelle liste pour retirer ces mots dans nos données:

```
wa_token <-
  whatsapp %>%
  unnest_tokens(mots, texte, token = "words") %>%
  anti_join(tibble(stop), by=c("mots" = "stop"))

wa_token
```

stop words (3)

```
# A tibble: 604 x 3
   date                nom                 mots
   <dttm>              <chr>               <chr>
 1 2020-05-14 16:58:00 +00 000 00 00       taquinais
 2 2020-05-14 16:58:00 Jean Poutre         cool
 3 2020-05-14 16:58:00 Jean Poutre         group
 4 2020-05-14 16:59:00 +33 3 33 33 33 33   ok
 5 2020-05-14 17:00:00 +11 111 11 11       36ème
 6 2020-05-14 17:00:00 +11 111 11 11       groupe
 7 2020-05-14 17:00:00 +11 111 11 11       déjà
 8 2020-05-14 17:00:00 +11 111 11 11       châlet
 9 2020-05-14 17:01:00 +11 111 11 11       qu'il
10 2020-05-14 17:01:00 +11 111 11 11       manquait
# ... with 594 more rows
```

Fréquences

Nous n'allons simplement compter la fréquence de chaque mot soit simplement, soit par personne, soit par date, soit par personne et par date. Le choix dépend de l'analyse que l'on souhaite réaliser. Ici nous allons simplement faire le nombre de mots par personne:

```
wa_freq <-
  wa_token %>%
  count(nom, mots)

wa_freq
```

```
# A tibble: 491 x 3
   nom           mots          n
   <chr>         <chr>     <int>
 1 +00 000 00 00 00              2
```

Exemple avec les fréquences

Nous pouvons réaliser l'analyse des 10 mots les plus utilisés:

```
wa_freq %>%
  group_by(mots) %>%
  summarise(n = sum(n)) %>%
  slice_max(n, n = 10) %>%
  ggplot(aes(fct_reorder(mots, n), n)) +
  geom_col() +
  geom_label(aes(label = n)) +
  labs(title = "Les 10 mots les plus utilisés",
       x = "mots",
       y = "Nombre d'utilisation") +
  coord_flip()
```

tf_idf (1)

Le `tf_idf()` permet de montrer à quel point un terme est particulier à un document/corpus en comparant sa fréquence interne à celles générales. Plus un terme a un tf_idf élevé, plus il est particulier à un corpus:

```
wa_tfd_idf <-
  wa_freq %>%
  bind_tf_idf(term = mots , document = nom, n = n)

wa_tfd_idf
```

tf_idf (2)

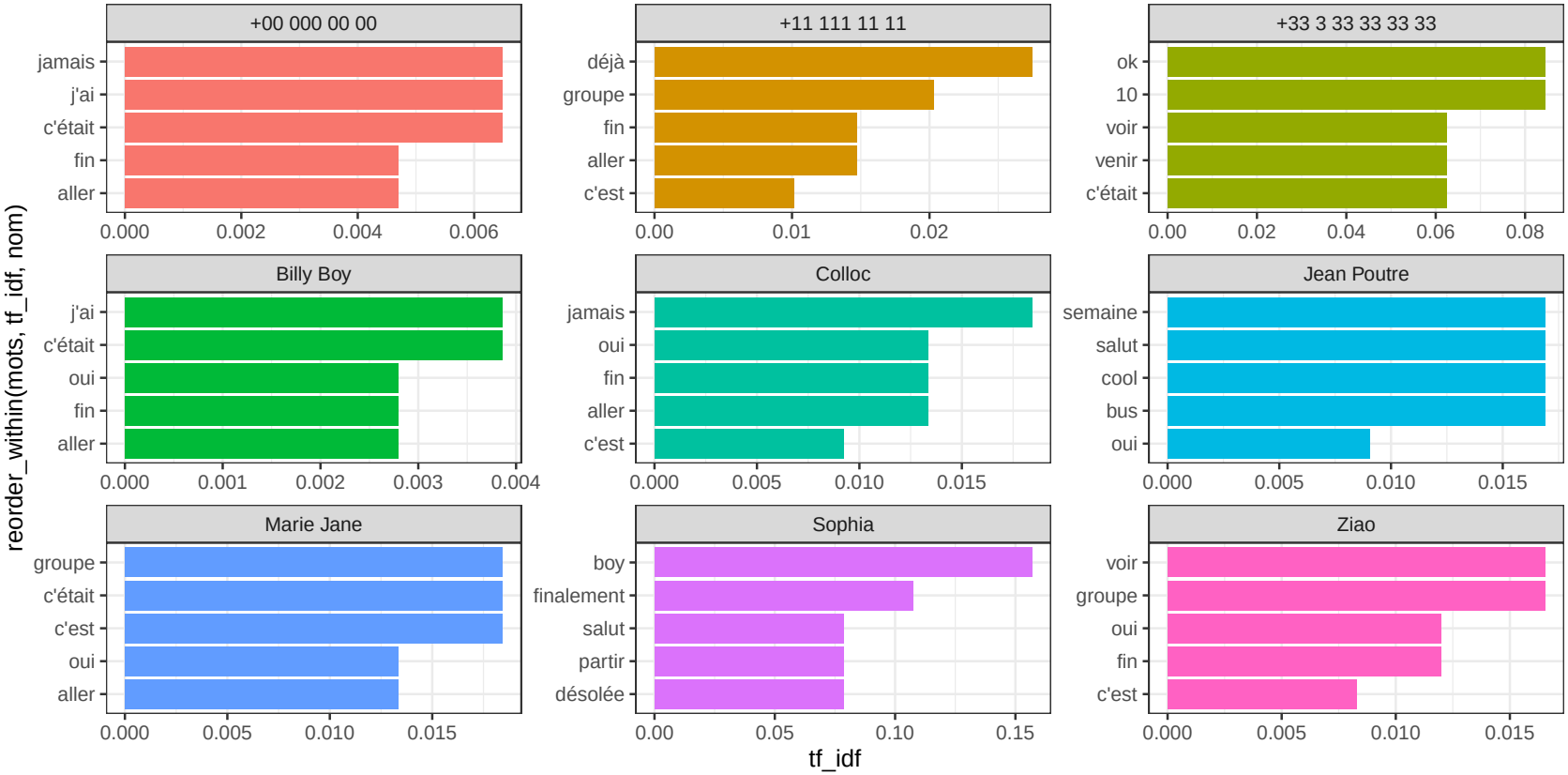
```
# A tibble: 491 x 6
   nom           mots          n    tf   idf tf_idf
   <chr>         <chr>     <int> <dbl> <dbl>  <dbl>
 1 +00 000 00 00 00            2 0.016  2.20 0.0352
 2 +00 000 00 00 0001          1 0.008  2.20 0.0176
 3 +00 000 00 00 10            2 0.016  1.10 0.0176
 4 +00 000 00 00 16            2 0.016  2.20 0.0352
 5 +00 000 00 00 28            1 0.008  2.20 0.0176
 6 +00 000 00 00 36            1 0.008  2.20 0.0176
 7 +00 000 00 00 776           1 0.008  2.20 0.0176
 8 +00 000 00 00 9730          1 0.008  2.20 0.0176
 9 +00 000 00 00 accessoire    1 0.008  2.20 0.0176
10 +00 000 00 00 agendé        1 0.008  2.20 0.0176
# ... with 481 more rows
```

Exemple avec le tf_idf

Nous pouvons essayer de voir qu'es sont les 5 termes les plus spécifiques de tous nos participants:

```
wa_tfd_idf %>%
  group_by(nom) %>%
  arrange(nom, tf_idf) %>%
  slice_head(n = 5) %>%
  ggplot(aes(reorder_within(mots, tf_idf, nom),
             tf_idf, fill = nom)) +
  geom_col() +
  scale_x_reordered() +
  facet_wrap(~nom, scales = "free") +
  coord_flip() +
  theme(legend.position = "none")
```

Exemple avec le tf_idf



tidytext

Il y a bien plus de possibilités d'analyse avec {tidytext}. Nous n'avons vu qu'une petite partie de ce qu'il était possible de faire {tidytext}. Vous pouvez vous documenter en lisant le livre sur la question: {tidytext}.

Feedback

Lien votamatic

Code: FRKR

Merci pour votre attention!