Stack-Overflow Analiza

Mateusz Pieszczek

14 02 2021

Contents

Biblioteki

W większości są standardowe. Testowałem różne rozwiązania przy szeregach czasowych oraz scrapingu. Finalnie nie wysztkich użyłem.

```
library(tidyverse)
library(lubridate)
library(dygraphs)
library(scales)
library(data.table)
library(tidytext)
library(rvest)
library(splitstackshape)
library(timetk)
library(janitor)
library(xts)
library(zoo)
```

Pobieranie i wstępna edycja danych

Pobieranie Danych

Pierwszy chunk przedstawia funkcje uzytą przy pobieraniu danych ze strony zapytań. Pewne dane zależnie od wielkości były zapisane w wielu zmiennych: jak reputacja. Finalnie nie użyłem wszystkich danych, częściowo z braku czasu.

```
zbieraj <- function(url){
  Sys.sleep(0.5)
  print(paste0("Processing ",url))
  strona <- url %>%
  read_html()
```

```r
import_time <- Sys.time()

questions <- strona %>%  html_nodes(".question-summary")

#tytul
title <- questions %>%
html_node("h3") %>%
html_node(".question-hyperlink") %>%
html_text(trim=TRUE)
#liczba wyświetleń
views <- questions %>%
html_node(".views") %>%
html_attr("title")
#czas
time <- questions %>%
html_node(".user-info") %>%
html_node(".relativetime ") %>%
html_attr("title")
#tagi
tag <- questions %>%
html_node(".tags") %>%
html_text(trim=TRUE)
#liczba glosow
vote_count <- questions %>%
html_node(".vote-count-post strong") %>%
html_text()
#liczba odpowiedzi
response_count <- questions %>%
html_node(".status strong") %>%
html_text()

#reputacja
#reputation <- questions %>%
#html_node(".user-info") %>%
#html_node(".-flair") %>%
#html_node(".reputation-score") %>%
#html_attr("title")


big_rep <- questions%>%
html_node(".user-info") %>%
html_node(".-flair") %>%
html_node(".reputation-score") %>%
html_attr("title")
small_rep <- questions%>%
html_node(".user-info") %>%
html_node(".-flair") %>%
html_node(".reputation-score") %>%
html_text()

#gold
gold <- questions %>%
html_node(".user-info") %>%
```

```
  html_node(".-flair") %>%
  html_node(".badge1+ .badgecount") %>%
  html_text()

  #silver
  silver <- questions %>%
  html_node(".user-info") %>%
  html_node(".-flair") %>%
  html_node(".badge2+ .badgecount") %>%
  html_text()

  #bronze
  bronze <- questions %>%
  html_node(".user-info") %>%
  html_node(".-flair") %>%
  html_node(".badge3+ .badgecount") %>%
  html_text()

  df <- tibble(tag,vote_count, response_count, views, time, big_rep,small_rep, gold,silver,bronze, titl
    mutate(import_date=import_time)
  df %>% save_csv()
  df
}
```

Funkcja do appendowania pliku z danymi. Pobieranie danych trwalo dosyć długo. Po pobraniu doszedłem do wniosku, że pare rzeczy można było poprawić, jak pobieranie tytułu. Można było formatować go od razu na miejscu niż puszczać do pliku tekstowego.

```
save_csv <-function(df){
 df %>% write.table("dane/stack/Stack.csv",sep = ',',col.names = !file.exists("dane/stack/Stack.csv"), a
}
```

Tutaj znajdue się chunk gdzie pobierałem same dane.

```
#fn <-"dane/stack/Stack.csv"
#if (file.exists(fn)) {
#  file.remove(fn)
#}
ostatnia <- read_html("https://stackoverflow.com/questions?tab=newest&pagesize=50&page=1") %>%
  html_nodes(".s-pagination--item__clear+ .js-pagination-item") %>%
  html_text() %>%
  as.numeric()

strony <- paste0("https://stackoverflow.com/questions?tab=newest&pagesize=50&page=",1:ostatnia)
stack_quest <- strony[1:ostatnia] %>% map_dfr(zbieraj)
stack_quest
```

Edycja pobranych danych

Tutaj starałem się oczyścić pobrane dane.

```
#Odczytanie z csv
stack_quest <- read.csv("dane/stack/Stack_no_title.csv",header=TRUE,sep = ",",quote = '"',dec = ".",na.s
stack_quest
```

Pierwszą rzecz jaką zrobiłem, to rozdzielenie pytań. Dla kazdego tag-u jaki zawiera pytanie stworzyłem

osobny wiersz. Pozwoliło to Spojrzenie na trendy przez wzgląd na tagi z osobna. Należy jedynie pamietać, że sumarycznie pytań jest mniej niż wynikało by to z wykresów. Np w późniejszych latach można zauważyć duży zbiór pytań, które posiadają równocześnie tag "R" oraz "python".

```
df2 <- cSplit(stack_quest, "tag", sep = " ", direction = "long")
setDF(df2) # parsowanie do data-frama
```

Następnie przetowrzyłem pola które pomocnicze które docelowo miały być zparsowane do innych zmiennych. W przypadku reputacji wartość dokładna w zależności od rzędu wielkości znajdowała się, albo w jednej albo w drugiej zmiennej.

```
df2 <- df2 %>%
  mutate(big_rep = as.integer(str_remove(str_replace(big_rep,"reputation score ","0"),","))) %>%
  mutate(reputation = ifelse(big_rep==0,as.integer(str_remove(small_rep,',')),big_rep)) %>%
  select(-big_rep,-small_rep,-import_date)
```

Tutaj przeprowadzam standardowe parsowanie. Wszystkie pobrane zmienne były zmiennymi tekstowymi.

```
#formating columns
df2 <- df2 %>%
  mutate(vote_count = as.integer(vote_count),
         response_count= as.integer(response_count),
         views = as.integer(str_remove(str_remove(views,pattern = " views"),pattern = ',')),
         reputation= as.integer(str_remove(reputation,pattern = ',')),
         gold= as.integer(gold),
         silver= as.integer(silver),
         bronze= as.integer(bronze),
         time = as.POSIXct(time, format = "%Y-%m-%d %H:%M:%SZ")
         )
```

Na koniec usunąłem pytania, które nie posiadały czasu(pytania z zewnątrz). W obecnej analizie były nieprzydatne.

```
no_time <- df2 %>%
  filter(is.na(time))
#Jeśli jest NA to trzeba by uzupełnic datą
df2 <- df2 %>%
  filter(!is.na(time))
```

Ładowanie danych

Po uzyskaniu danych powyżej zapisywałem je w mniejszej objetościowo formie, aby móc sprawniej pracować na nich. Finalnie w "finala_project.RData" znajdują się juz małe data_framy które użyłem przy tworzeniu wykresów.

```
#load("Shiny_projekt_stack/Stack/Data_sets.RData")
load("final_project.RData")
```

Finalnie zbiór nad którym tutaj pracowałem wygladał następująco:

```
temp6 <- head(df2)
```

```
temp6
```

```
##          tag vote_count response_count views                time gold silver
## 1     arrays          0              0     2 2021-01-27 13:19:12   NA     NA
## 2      numpy          0              0     2 2021-01-27 13:19:12   NA     NA
## 3     stream          0              0     2 2021-01-27 13:19:12   NA     NA
## 4        jsf          0              0     3 2021-01-27 13:19:12   NA     NA
```

```
## 5 primefaces          0             0     3 2021-01-27 13:19:12   NA     NA
## 6     python          0             0     3 2021-01-27 13:18:54   NA     NA
##   bronze reputation temp_num
## 1      9        105        1
## 2      9        105        1
## 3      9        105        1
## 4      1          1        1
## 5      1          1        1
## 6     NA          1        1
```

Najważniejsza rzecz jaką usunąłem to tytuły. Miałem w planie wykonać jakiąś pomniejszą analizę słów zawartych w tytułach. Niestety zauważyłem że dane zawarte w tytułach są dosyć problematyczne do sparsowania. Wiele pytań dotyczy np. Regex-ów czy też edycji tekstu w ogólności i zawiera w sobie znaki specjalne, separatory, znaki końca lini, itd. Uporządkowanie tych danych zabrało by mi więcej czasu i na ten moment odpuściłem ta analizę. W pythonie (ponieważ trochę łatwiej mi w nim edytowac plik tesktowy, jako ciąg znaków) usunąłem tytuły oraz duplikaty pytań (ładowałemn dane z przerwami, możliwe było że pare stron pytań załadowałem wielokrotnie)).

Zakres danych

Udało mi się pobrać wszystkie dane ze strony. Podczas analiz zauważyłem że warto pominąć lub przynajmniej zwrócić uwagę na to że pierwsze 2 lata mają wiele archiwalnych pytan które nie posiadają np. daty(w takim formacie jak reszt pytań) czy informacji o użytkowaniku który zapostował. Dodatkowo więcej znajdue się tam pytań z zewętrznych forów. Obecnie większośc pytań jest zadawanych bezpośrednio na stronie "Stack-a" i do takich pytań dostępne są wszystkie informacje. Oczywiście też dane z roku 2021 są zebrane jedynie ze stycznia(nie całego).

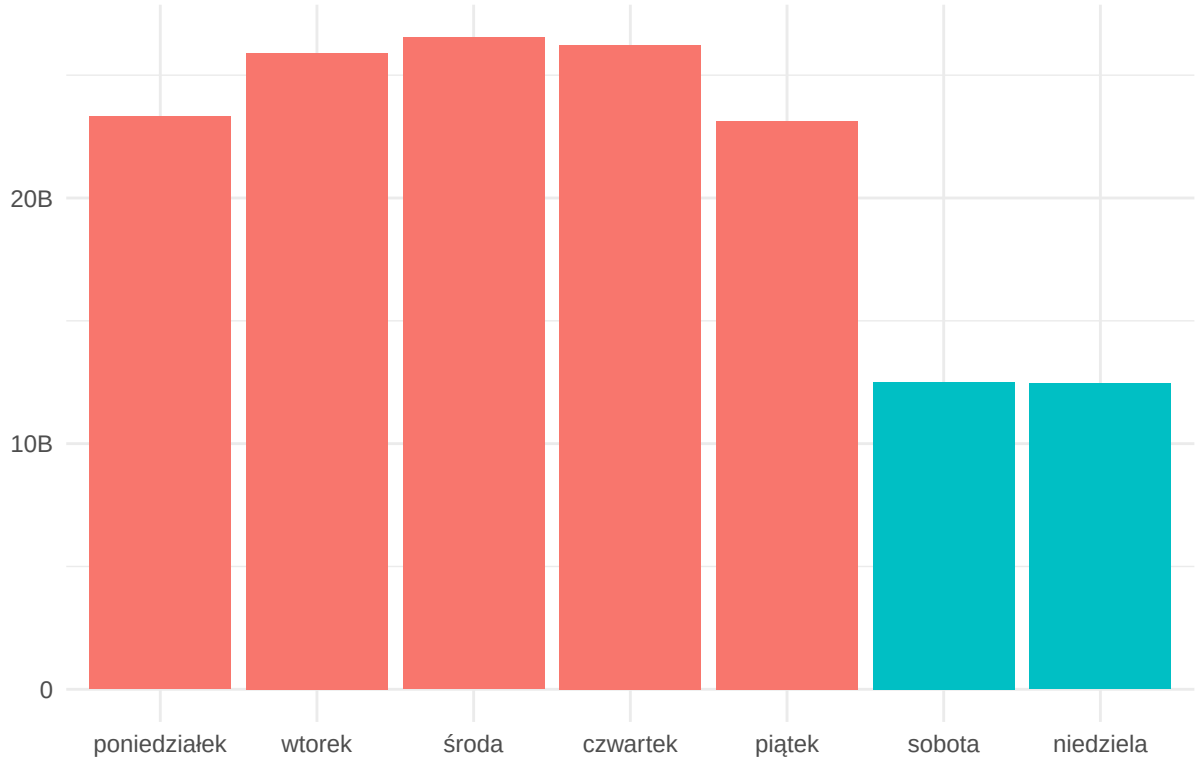
Tendencje tygodniowe

Pierwszą rzecz jaką chciałem sprawdzić to czy istnieją jakieś tendencje co do pytań w zależności od dni tygodnia.

```
temp1 <- df2 %>%
  filter(!is.na(views)) %>%
  select(time, views) %>%
  mutate(day_of_week = factor(weekdays(time),
                              levels = c("poniedziałek","wtorek","środa","czwartek","piątek","sobota","
  group_by(day_of_week) %>%
  summarise(aggr = sum(views))
```

Jak widać pytania które zostały zadane w weekend otrzymują mniejwięcej o połowę mniej wyświetleń. To sugeruje, że większość użytkowników storny odpowiada i wyświetla jedynie najnowsze pytania. Oczywiście są użytkownicy który patrzą na archiwalne pytania. W przypadku tych uzytkowników (tutaj moja hipoteza nie jestem w stanie zweryfikować tego na tych danych) otrzymują oni przy wyszukiwaniu np. za pomocą Googla pytania najpopularniejsze, czyli te z dni roboczych. Czyli dochodzi do sprzężenia zwrotnego.

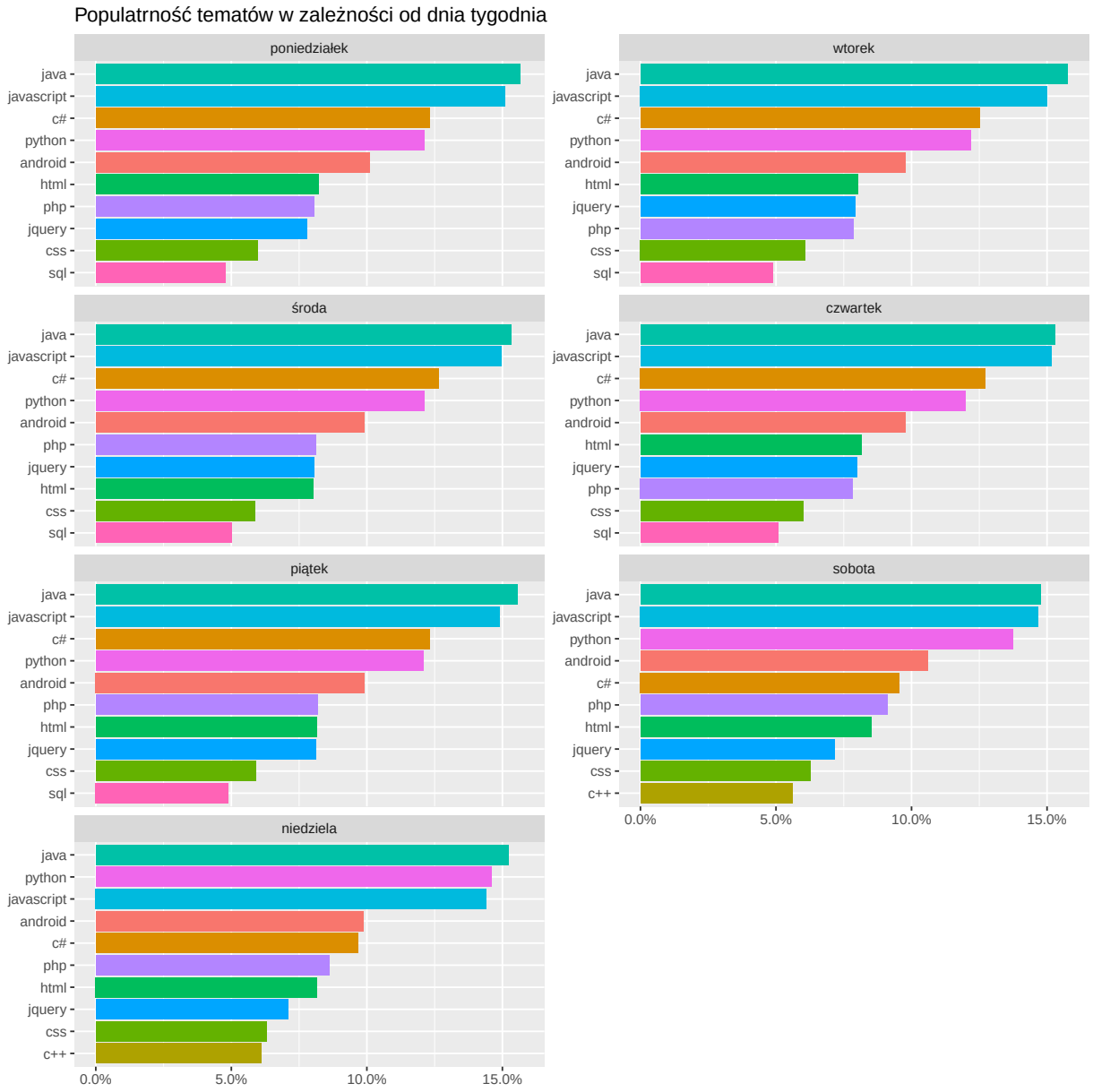
```
temp1 %>%
  ggplot(aes(day_of_week,aggr,fill=ifelse(day_of_week %in% c("sobota","niedziela"), 'orange', 'cornflow
  geom_col() +
  labs(x ="",y="",title = "Liczba wyświetleń pytania, w zależności od dnia utworzenia pytania",color=NU
  scale_y_continuous(labels =scales::label_number_si()) +
  theme_minimal() +
  theme(legend.position="none")
```




```
temp2 <- df2 %>%
  filter(!is.na(views)) %>%
  select(tag, time, views) %>%
  mutate(day_of_week = factor(weekdays(time),
                              levels = c("poniedziałek","wtorek","środa","czwartek","piątek","sobota","
  group_by(day_of_week, tag) %>%
  summarise(aggr = sum(views)) %>%
  slice_max(order_by = aggr, n = 10) %>%
  mutate(procent = aggr/sum(aggr))
```

W związku z powyższą dużą różnicą w ruchu na stronie w tygodniu, postanowiłem sprawdzić czy tematyka pytań różni się. Poniższe wykresy pokazuje najbardziej popularne (procentowo) tematy pytań w daniach. Pomimo różnicy w ilości pytań, tematyka jest podobna. Chociaż można zauważyć że python jest minimalnie bardziej poplurarny w weekendy, tak samo C++. Te minimalne przesunięcia pokazują czym programiści zajmują się zawodowo. Przypuszczenie, że bardziej niszowe języki będą bardziej popularne w weekendy ma tu swoje odzwierciedlenie. Jeśli przejrzeli byśmy pierwsze 100 tagów (nie ma miejsca na to w tym dokumencie) to zauważymy np. wzrost języków funkcyjnych w weekendy.

```
temp2 %>%
  ggplot(aes(reorder_within(tag, procent, day_of_week),procent,fill = tag))+
  geom_col()+
  coord_flip()+
  scale_x_reordered() +
  facet_wrap(~day_of_week,scales = "free_y",drop = FALSE,ncol = 2)+
  labs(x ="",y = "",title="Populatrność tematów w zależności od dnia tygodnia")+
  scale_y_continuous(labels = percent) +
  theme(legend.position="none")
```

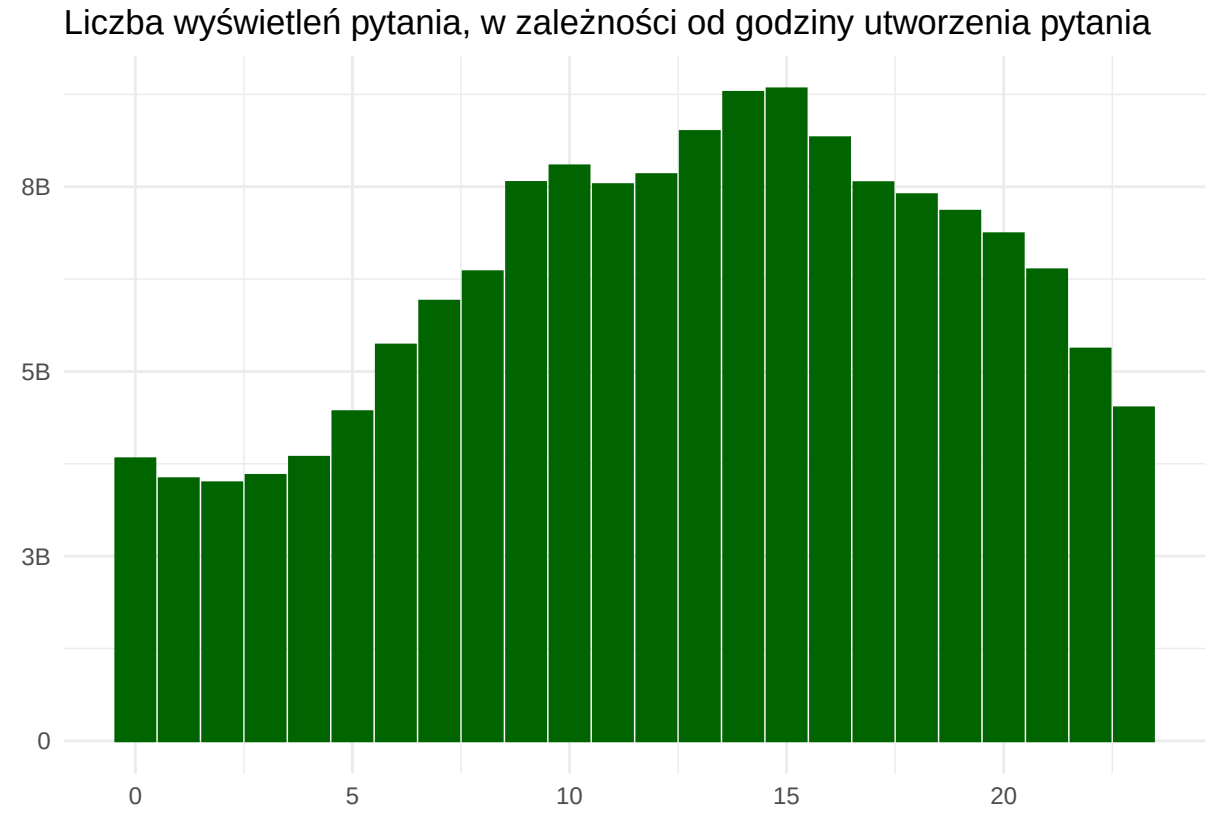


Tendencje w ciągu dnia

Kolejną rzeczą było sprawdzenie tendencji w ciągu dnia. Wynik chyba nie jhest specjalną niespodzinką. Strona i pytania są w języku angielskim. Czas był pobrany według poslkiej strefy czasowej. Nie mniej jeżeli uwzględnimy 6-9 godzin różnicy dla USA(9-dla zachodniego wybrzeża,6-dla wschodniego). To można zauważyć że szczyt ruchu jest skupiony wokół środka dnia dla krajów zachodu.

```
temp3 <- df2 %>%
 filter(!is.na(views)) %>%
 select(time, views) %>%
 mutate(hour = hour(time)) %>%
 group_by(hour) %>%
 summarise(aggr = sum(views))
```

```r
temp3 %>%
  ggplot(aes(hour,aggr)) +
  geom_col(color="darkgreen",fill="darkgreen") +
  labs(x ="",y="",title = "Liczba wyświetleń pytania, w zależności od godziny utworzenia pytania") +
  scale_y_continuous(labels =scales::label_number_si()) +
  theme_minimal() +
  theme(legend.position="none")
```

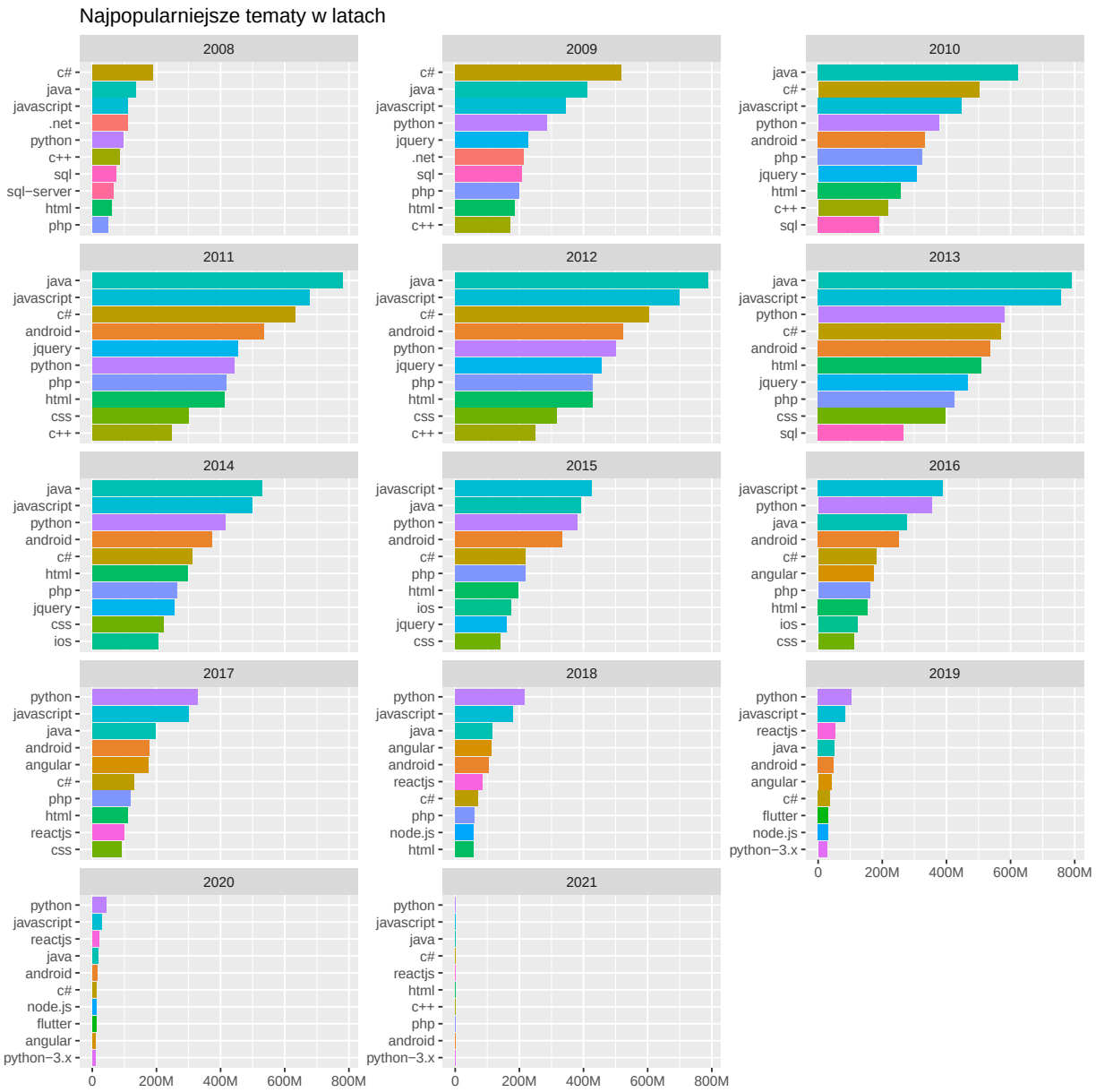


Tendencje w latach

Kolejną anlizą którą zrobiłem było sprawdzenie trendów w przeciągu lat. Wydało mi się sensowne zestawienie napjpopularniejszych tagów w poszczególnych latach. Widać pewne zmiany C# w początkowych latach był bardzo popularny a następnie spadł do połowy popularności Javy. Tak samo widać wzrost popularności języka Python w ostatnich 4 latach. Pozostawiłem skalę ilościową, aby zobrazować jeszcze jedną rzecz. W ostatnich latach jest znacznie większe zróżnicowanie tematyczne. Jeżeli zobaczymy liczbe wyświetleń w latach(kolejny wykres), to widać nieznaczny spadek popularności(i wzrost w 2020-najpewniejh związany z Covidem). Ale tak czy inaczej najpopularniejsze temty pytań stanowią znacznie mniejszą część pytań na stornie.

```r
temp4 <- df2 %>%
  filter(!is.na(views)) %>%
  select(tag, time, views) %>%
  mutate(year = year(time)) %>%
  group_by(year, tag) %>%
  summarise(aggr = sum(views)) %>%
  slice_max(order_by = aggr, n = 10)
```

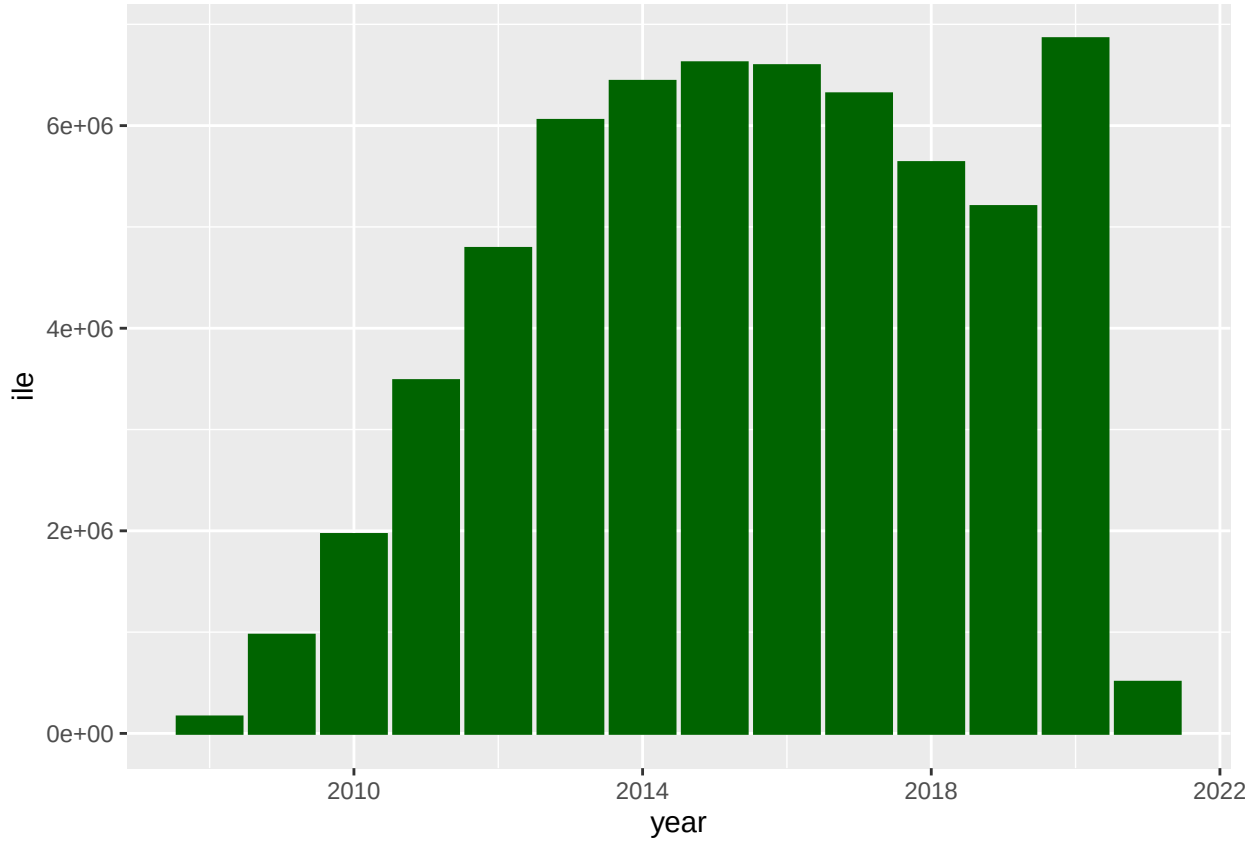
```
temp4 %>%
  ggplot(aes(reorder_within(tag, aggr, year),aggr,fill = tag))+
  geom_col()+
  coord_flip()+
  scale_x_reordered() +
  facet_wrap(~year,scales = "free_y",drop = FALSE,ncol = 3)+
  labs(x ="",y = "",title="Najpopularniejsze tematy w latach")+
  scale_y_continuous(labels =scales::label_number_si()) +
  theme(legend.position="none")
```




```
temp5 <- df2 %>%
  mutate(year = year(time)) %>%
  select(temp_num,year) %>%
  group_by(year) %>%
```

```
  summarise(ile = sum(temp_num))
```

```
temp5 %>%
  ggplot(aes(year,ile)) +
  geom_col(color="darkgreen",fill="darkgreen")
```



Analizy nie wykonane

Dużą ilość czasu zajeło mi pobranie samych danych, a nastęnie nie zdążyłem zrobić pewnych analiz, które planowłem, jak analiza treści tytuów pytań. Oraz analiza zalezności reputacji użytkownika od ilości odpowiedzi na jego pytania. Mimo ot mam nadzieję że sam projekt ma wartość pomimo tych braków.