Zadanie zaliczeniowe 1

Hanna Kranas

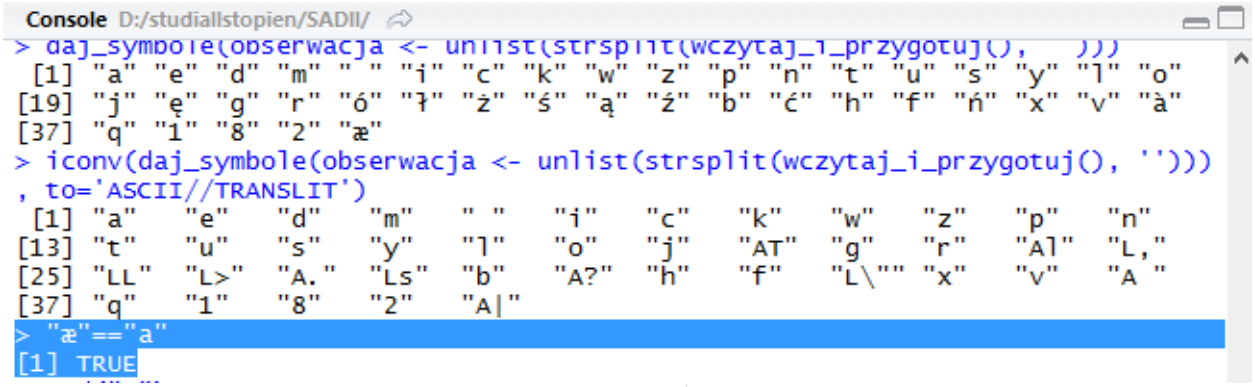
6 maja 2017

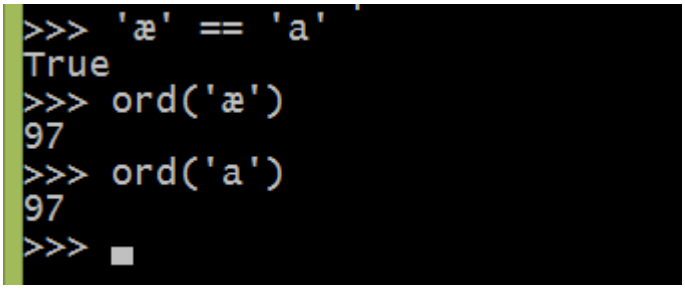
Contents

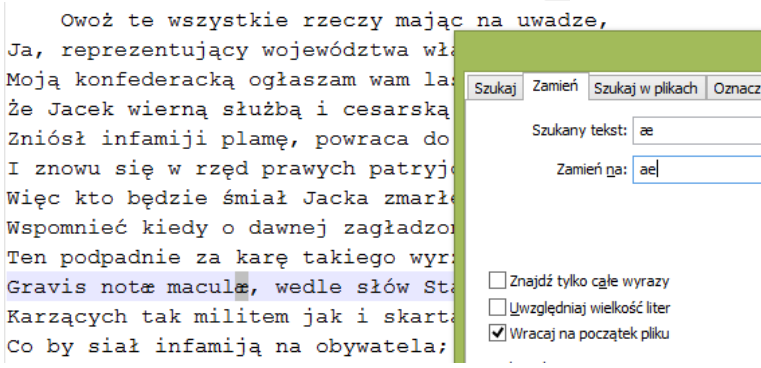
Przygotowanie danych

Usuwanie trudnych znaków

Ponieważ ani R ani Python nie radziły sobie z usuwaniem niektórych znaków, usunięte zostały w notepadzie.







Funkcja przygotowująca dane

Przygotowanie danych wykonane jednak zostało głównie w R:

```
wczytaj_i_przygotuj <- function(nazwa_pliku='pan-tadeusz.txt', polskie = TRUE){
  docs <- SimpleCorpus(DirSource("D:/studiaIIstopien/SADII/SAD-zal-1",encoding = 'UTF-8'),control = lis
docs <- tm_map(docs,removePunctuation)
docs <- tm_map(docs,removeNumbers)
docs <- tm_map(docs,stripWhitespace)
plik <- tolower(as.character(docs[[nazwa_pliku]]))
plik <- gsub ("[-....-\t-]", "", plik)
plik <- plik[plik!=""] #usuwam  puste
  if(polskie){
    cat('POLSKIE LITERY! ąę!')
  } else {
    plik <- iconv(plik,from="UTF-8",to="ASCII//TRANSLIT") #bez polskich
  }
  return(plik)
}
```

Kod i symulacje

Symbole

Po przygotowaniu pliku, potrzebna nam będzie możliwość wyciągnięcia symboli z obserwacji:

```
daj_symbole <- function(obserwacja){
  cat('dlugosc obserwacji: ')
```

```
  cat(length(obserwacja))
  #ta funkcja ekstrachuje wszystkie unikalne elementy - czyli dla nas symbole
  symbole <- sort(unique(obserwacja))
  return(symbole)
}
```

HMM i BaumWelch - model i trenowanie

Tworzenie HMM i trenowanie algorytmem Bauma-Welcha odbywa się w ten sposób:

```
prob <- function (x) {x / sum (x)} # funkcja pomocnicza

symuluj <- function(obserwacja,stany){
  symbole <- daj_symbole(obserwacja)
  macierz_emisji <- c() #szykujemy macierz emisji
  for(i in 1:length(stany)){
    x <- rep(1/length(symbole),length(symbole))
    x <- x +  rnorm(length(symbole),0,1/1000000) #dajemy małe zaburzenie
    x <- abs(x)/sum(abs(x)) #i poprawiamy zeby sie sumowaly do jedynki dalej
    macierz_emisji <- c(macierz_emisji,x)
  }
  #tworzymy hmm'a z nieco-losowymi prawdopodobieństwami początkowymi i przejść
  hmm <- initHMM (stany, symbole, startProbs=(prob(runif(length(stany)))),
                  transProbs=apply(matrix(runif(length(stany)*length(stany)),
                                          length(stany)),1,prob),
                  emissionProbs=matrix(macierz_emisji,nrow =length(stany),
                                       ncol=length(symbole),byrow=TRUE))
  #uczymy algorytmem Bauma-Welcha
  wynik <- baumWelch(hmm,obserwacja)
  return(wynik)
}
```

Pełna symulacja literowa

Pełna symulacja dla liter z przygotowanego pliku wygląda tak:

```
litery <- function(przygotowany_plik=wczytaj_i_przygotuj()){
  obserwacja <- unlist(strsplit(przygotowany_plik, ''))
  obserwacja <- obserwacja[!is.na(obserwacja)]
  stany <- c('1','2') # dwa stany
  wynik <- symuluj(obserwacja,stany)
  return(wynik)
}
```

Klastrowanie sufiksów

W przypadku słów konieczne było klastrowanie, ponieważ inaczej algorytm Bauma-Welcha nie dawał sobie rady. Sufiksy max-3literowe poklastrowane zostały po ich długości oraz umiejscowieniu samogłosek/spółgłosek, łącznie 14 klastrów. Wykonano też drugie klastrowanie, w którym sufiksy jednoliterowe połączono w jedną grupę (‘1’), i dwuliterowe też (‘2’).

```
klastruj <- function(obserwacja){
  obserwacja <- gsub ("^[aeyiou][^aeyiou][aeyiou]$", "ara", obserwacja)
  obserwacja <- gsub ("^[aeyiou][aeyiou][aeyiou]$", "aaa", obserwacja)
  obserwacja <- gsub ("^[aeyiou][^aeyiou][^aeyiou]$", "arr", obserwacja)
  obserwacja <- gsub ("^[aeyiou][aeyiou][^aeyiou]$", "aar", obserwacja)
  obserwacja <- gsub ("^[^aeyiou][^aeyiou][aeyiou]$", "rra", obserwacja)
  obserwacja <- gsub ("^[^aeyiou][aeyiou][aeyiou]$", "raa", obserwacja)
  obserwacja <- gsub ("^[^aeyiou][aeyiou][^aeyiou]$", "rar", obserwacja)
  obserwacja <- gsub ("^[^aeyiou][^aeyiou][^aeyiou]$", "rrr", obserwacja)
  obserwacja <- gsub ("^[^aeyiou]$", "r", obserwacja) #1
  obserwacja <- gsub ("^[aeyiou]$", "a", obserwacja) #1
  obserwacja <- gsub ("^[^aeyiou][^aeyiou]$", "rr", obserwacja) #2
  obserwacja <- gsub ("^[aeyiou][aeyiou]$", "aa", obserwacja) #2
  obserwacja <- gsub ("^[aeyiou][^aeyiou]$", "ar", obserwacja) #2
  obserwacja <- gsub ("^[^aeyiou][aeyiou]$", "ra", obserwacja) #2
  return(obserwacja)
}
```

Pełna symulacja sufiksowa

Sposób symulacji dla słów jest następujący:

```
slowa <- function(przygotowany_plik=wczytaj_i_przygotuj()){
  #słowa wymagają trochę więcej przygotowań
  plik <- unlist(strsplit(przygotowany_plik," "))
  obserwacja <- unlist(str_sub(plik[plik!=""],start=-3)) #wyciągamy max-3literowe sufiksy
  obserwacja <- obserwacja[!is.na(obserwacja)]
  obserwacja <- klastruj(obserwacja)
  print(table(obserwacja)) #wypisuję dla informacji o tym ile jakich sufiksów
  stany <- c('1','2','3','4','5') # 5 stanów
  wynik <- symuluj(obserwacja,stany)
  return(wynik)
}
```

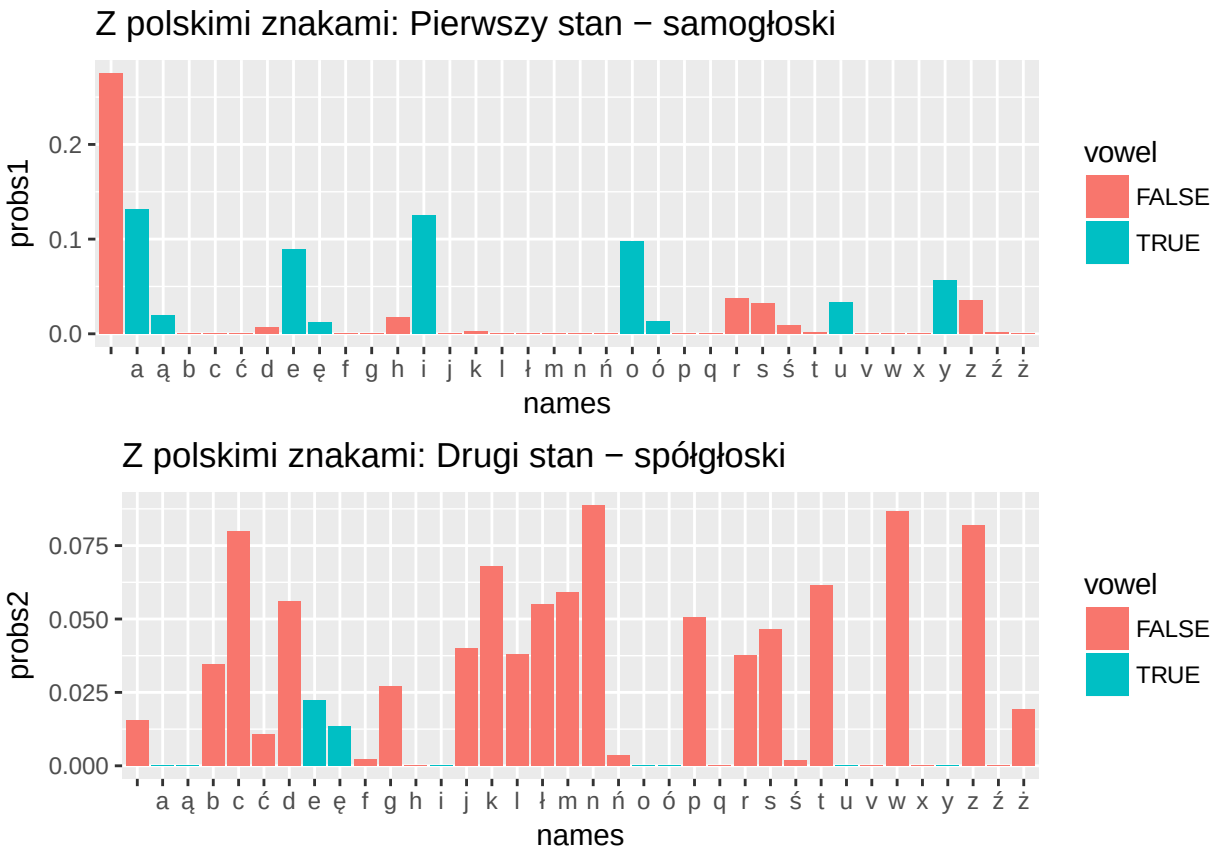
Wyniki

W poniższych wynikach można zobaczyć analizy dla liter (znaki polskie lub bez) z dwoma stanami, liter ze znakami polskimi z trzema stanami (tylko 20000 znaków) oraz sufiksów z pięcioma.

Litery z polskimi znakami

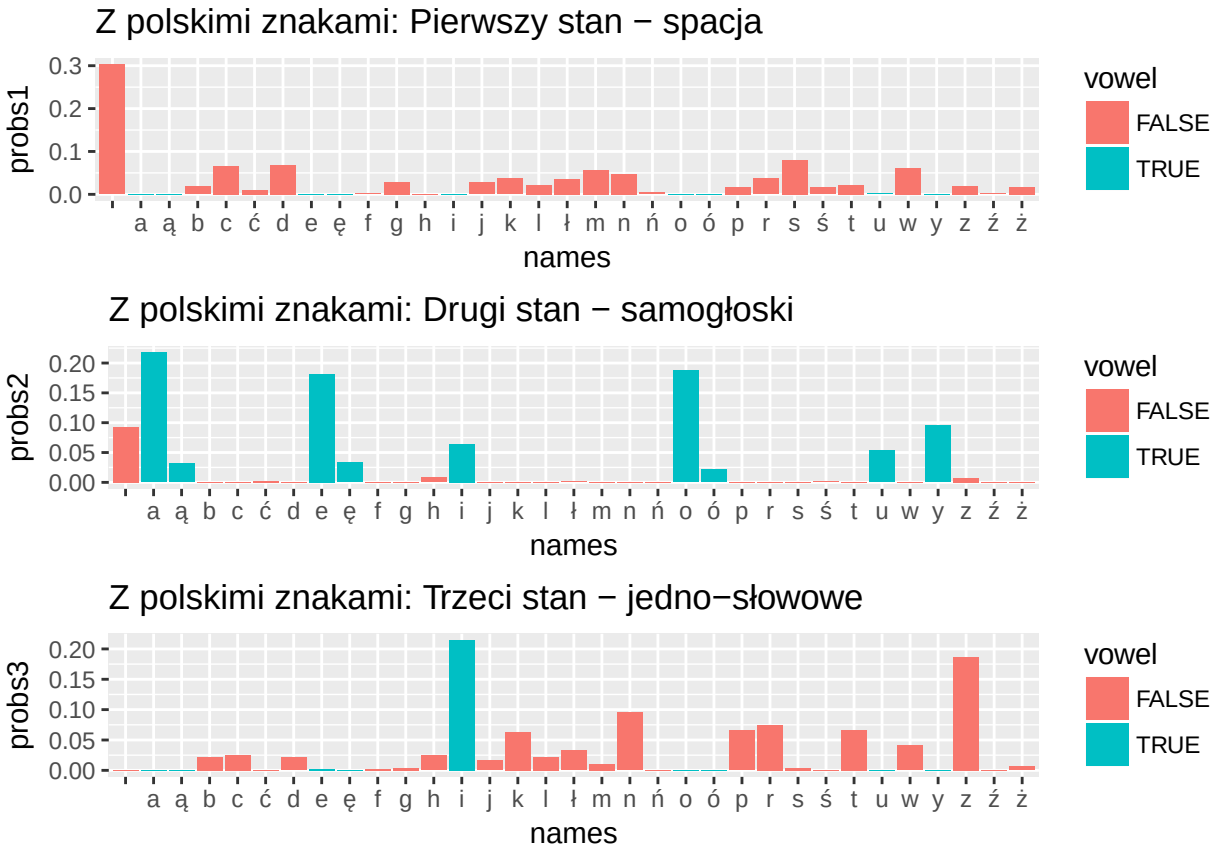
Widać wyraźnie że pierwsza grupa jest samogłoskowa. Na wykresach przedstawione są prawdopodobieństwa emisji poszczególnych symboli z wytrenowanego modelu HMM. Przy polskich znakach ładnie dzielą się stany, ale widać że są problemy przy literach które rzadko występują w tekście.

```
## [1] "To jest grupa pierwsza:"
##   a ą e h i o ó ś u y ź
## [1] "To jest grupa druga:"
## b c ć d ę f g j k l ł m n ń p q r s t v w x z ż
```



Trzy stany

Z ciekawości wykonano analizę dla większej (3) liczby stanów. Dla liter z polskimi znakami i 3 stanami widać stan "spacjowy", stan samogłoskowy, oraz stan trzeci, który na pierwszy rzut oka nie jest jasny, ale może jest to związane z często występującymi literami "jedno-słowowymi" (i,z)? Ponieważ nie była jasna interpretacja (oraz ograniczone możliwości obliczeniowe) nie stosowano analiz na większych liczbach stanów.



Litery bez polskich znaków

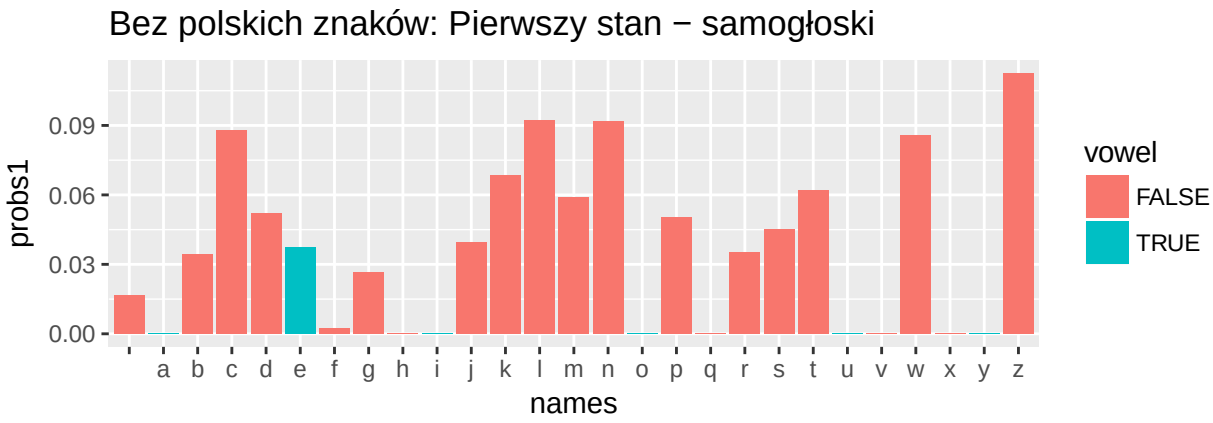
Widać wyraźnie że druga grupa jest samogłoskowa. Na wykresach przedstawione są prawdopodobieństwa emisji poszczególnych symboli z wytrenowanego modelu HMM.

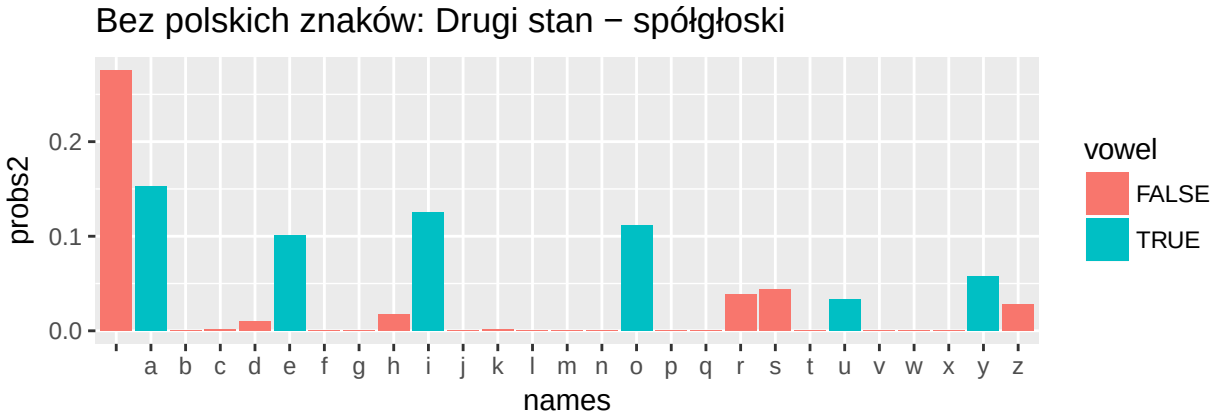
```
## [1] "To jest grupa pierwsza:"
```

```
## b c d f g j k l m n p q s t v w x z
```

```
## [1] "To jest grupa druga:"
```

```
##   a e h i o r u y
```

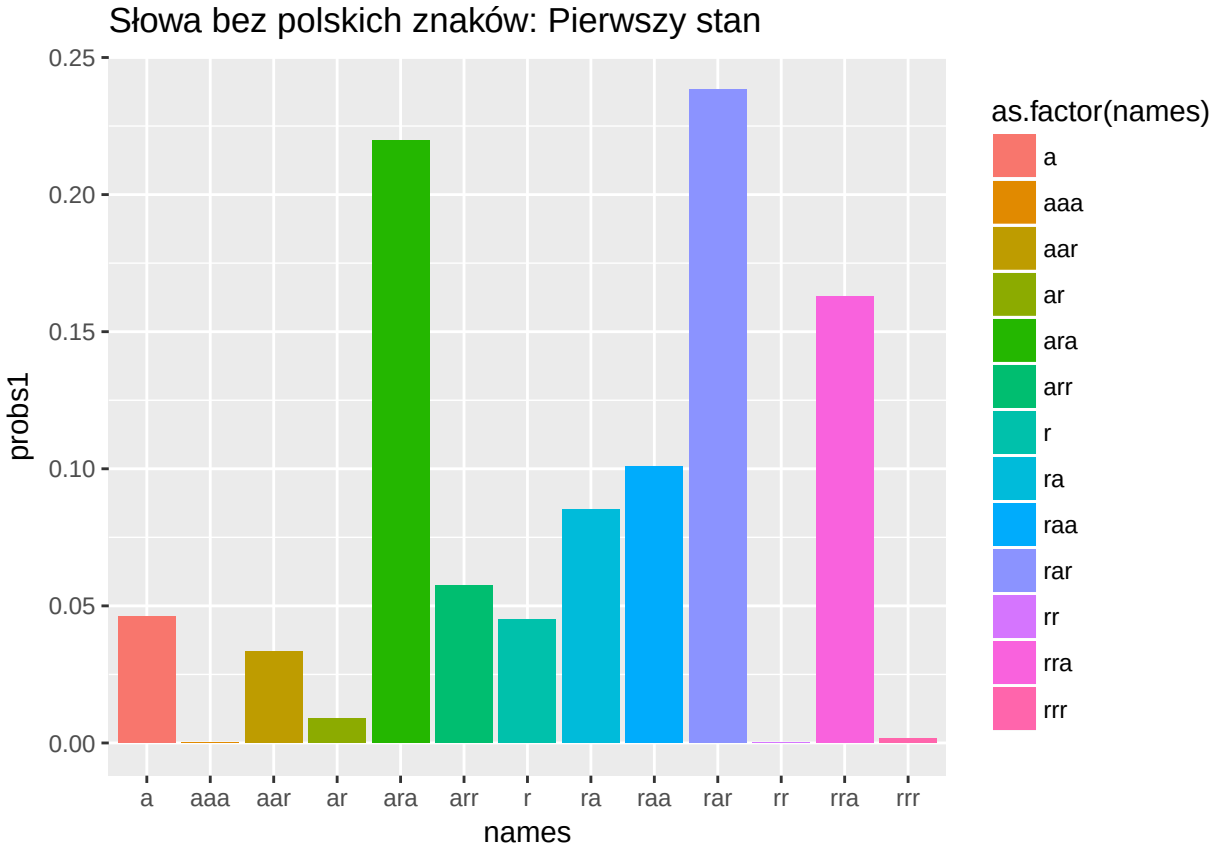

Bez polskich znaków: Pierwszy stan − samogłoski
probs1
0.09
0.06
0.03
0.00
a b c d e f g h i j k l m n o p q r s t u v w x y z
names
vowel
FALSE
TRUE

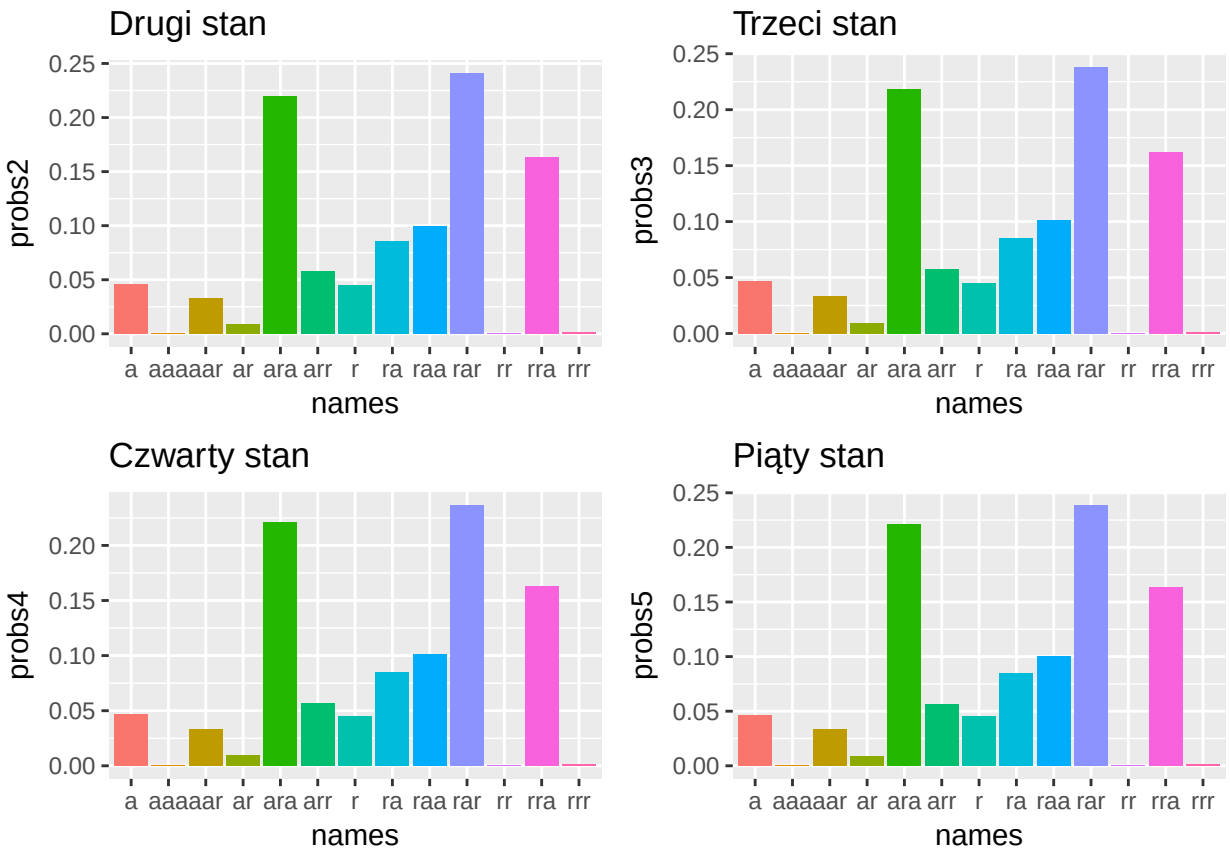

Bez polskich znaków: Drugi stan − spółgłoski
probs2
0.2
0.1
0.0
a b c d e f g h i j k l m n o p q r s t u v w x y z
names
vowel
FALSE
TRUE

Słowa - sufiksy, klastrowane

Pięć stanów - więcej sufiksów:

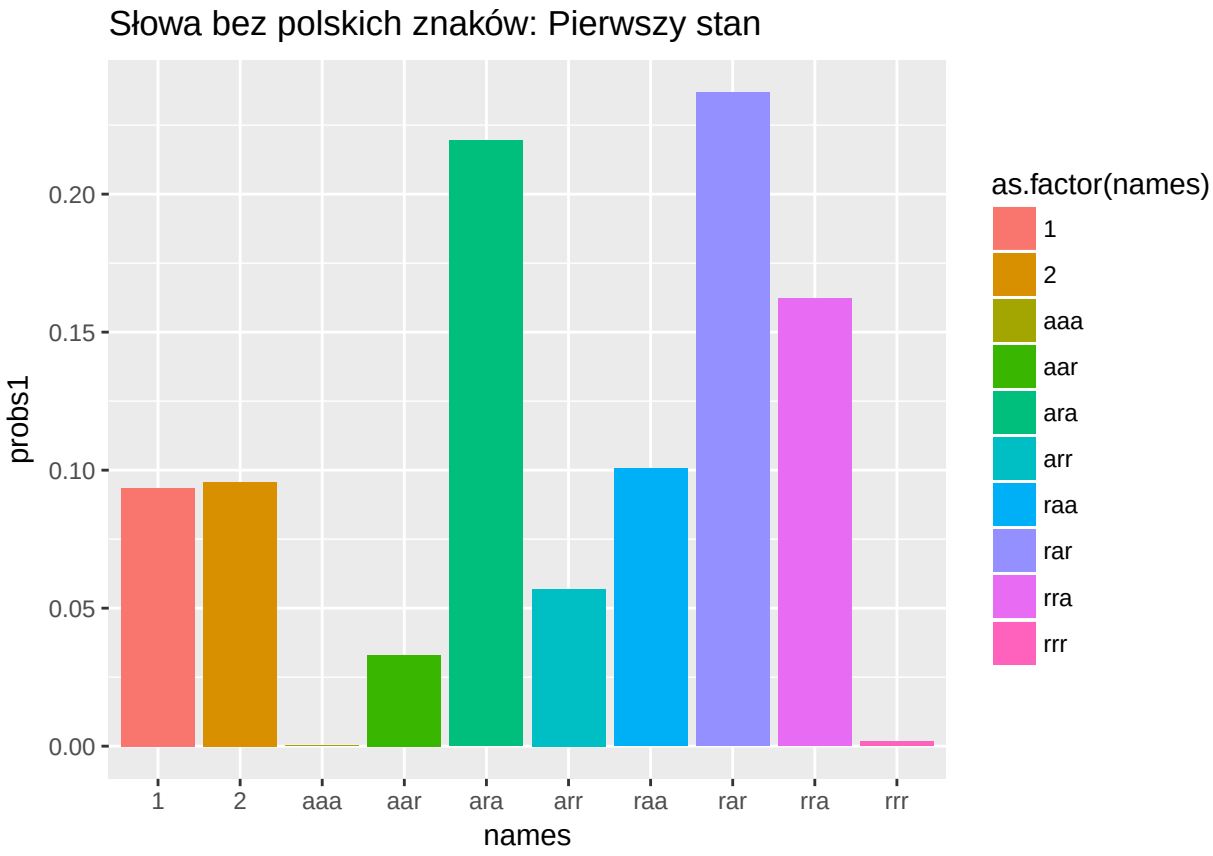
W każdym ze stanów prawdopodobieństwa emisji danych symboli są prawie identyczne.

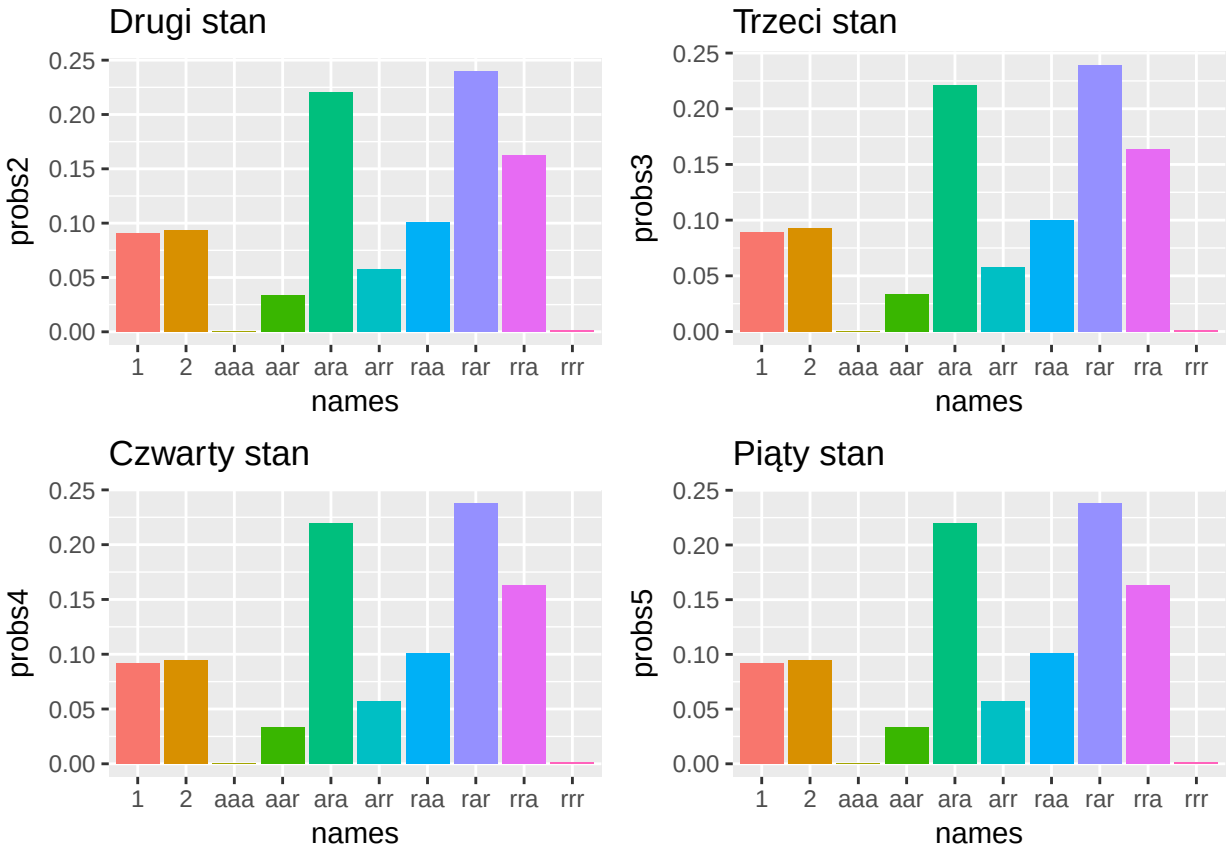




Pięć stanów - mniej sufiksów:

Wykonano dodatkowe klastrowanie, z mniejszą liczbą sufiksów, żeby sprawdzić czy coś to zmieni. Niestety znów w każdym ze stanów prawdopodobieństwa emisji danych symboli są prawie identyczne, można więc uznać że badanie sufiksów nie dało efektu.


Słowa bez polskich znaków: Pierwszy stan
probs1
0.20
0.15
0.10
0.05
0.00
1
2
aaa
aar
ara
arr
raa
rar
rra
rrr
names
as.factor(names)
1
2
aaa
aar
ara
arr
raa
rar
rra
rrr



Symulowanie na ulubionej poezji wytrenowanymi modelami

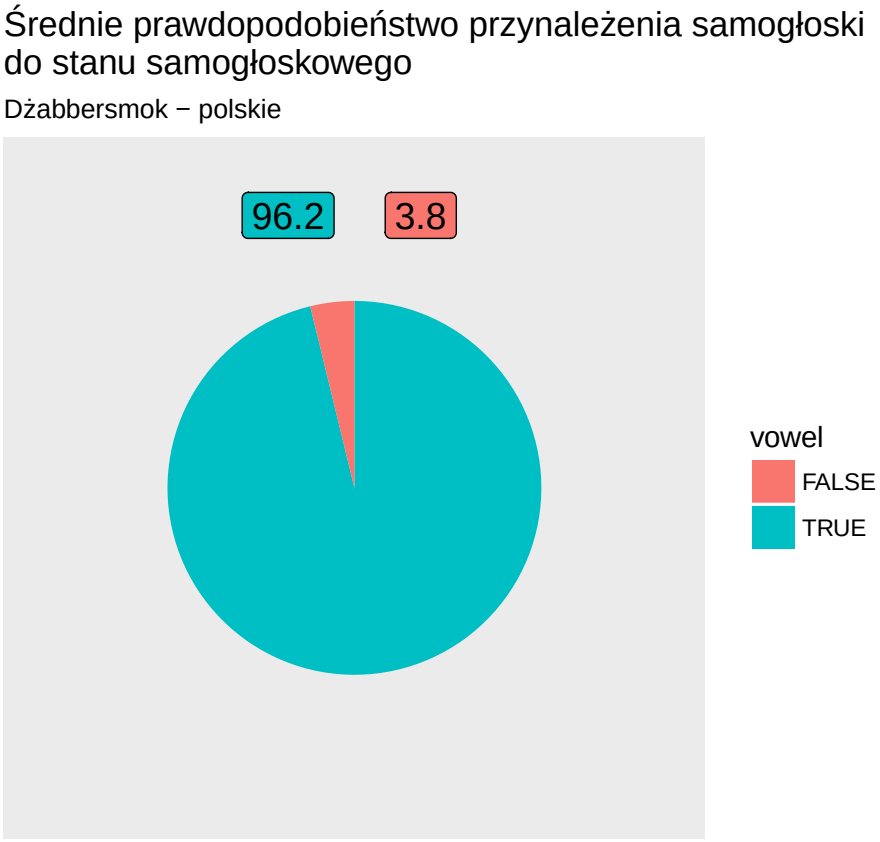
Wybrano 3 różne wiersze:
1. Jabberwocky, Lewis Caroll - przekład Macieja Słomczyńskiego (nie do końca polski język, ale uznano to za tym bardziej ciekawy test)
2. Dziewczyna, Bolesław Leśmian
3. Do prostego człowieka, Julian Tuwim

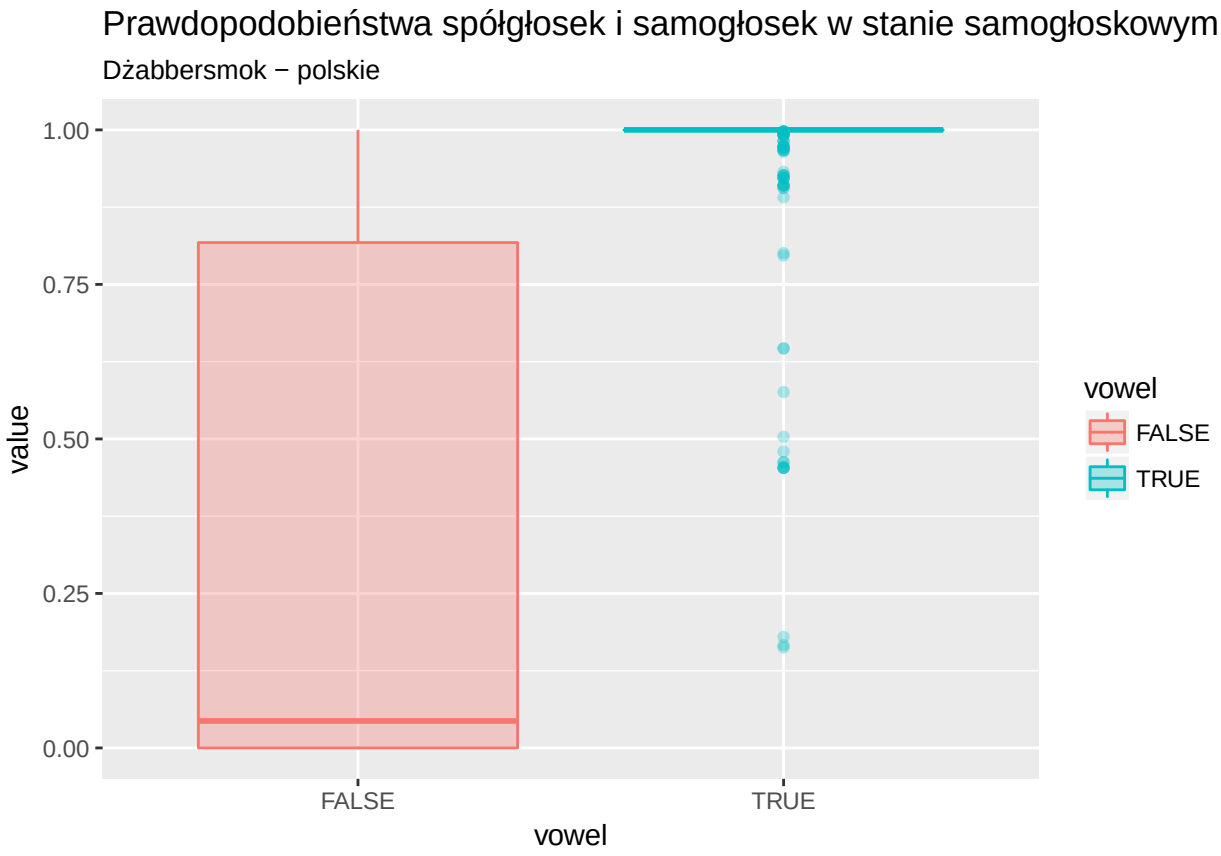
Dla każdego wiersza i rodzaju analizy literowej (polskie/bez polskich) wykonano dwa wykresy - średniego prawdopodobieństwa wystąpienia danej grupy do samogłoskowego stanu oraz prawdopodobieństw spółgłosek i samogłosek w stanie samogłoskowym. Jak można zauważyć prawdopodobieństwo należenia samogłosek do stanu samogłoskowego we wszystkich trzech wierszach jest bardzo wysoka, co sugeruje że zastosowany HMM został dobrze wytrenowany.

Z powodu słabych wyników dla dzielenia sufiksów do stanów nie wykonano tych analiz na wierszach.

Wiersze załączono do paczki z raportem.

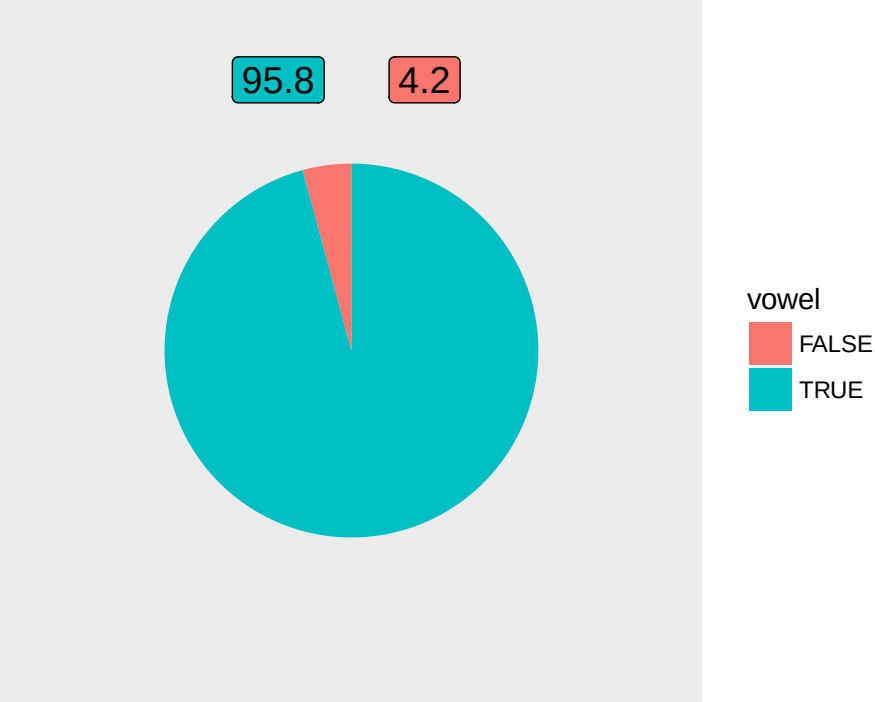
Dżabbersmok

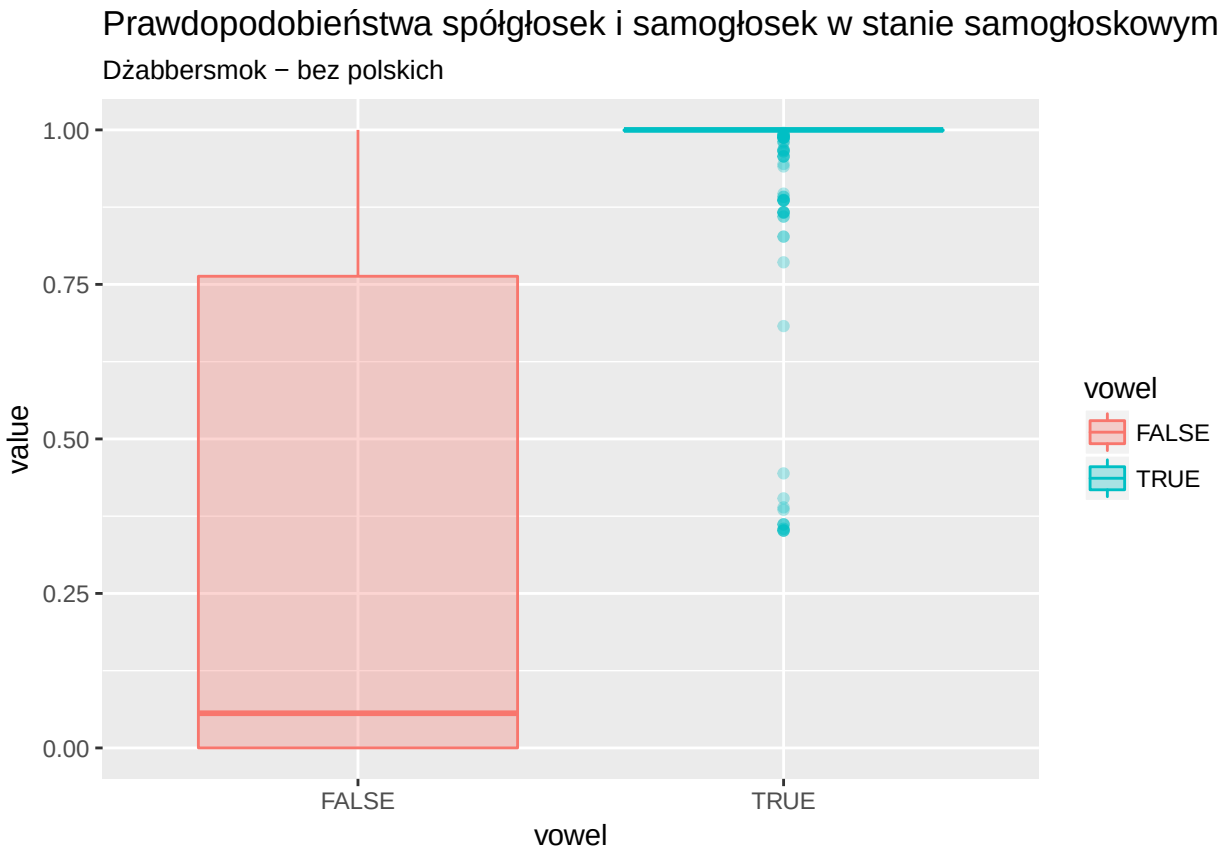



Prawdopodobieństwa spółgłosek i samogłosek w stanie samogłoskowym
Dżabbersmok − polskie
1.00
0.75
0.50
0.25
0.00
value
FALSE
TRUE
vowel
vowel
FALSE
TRUE

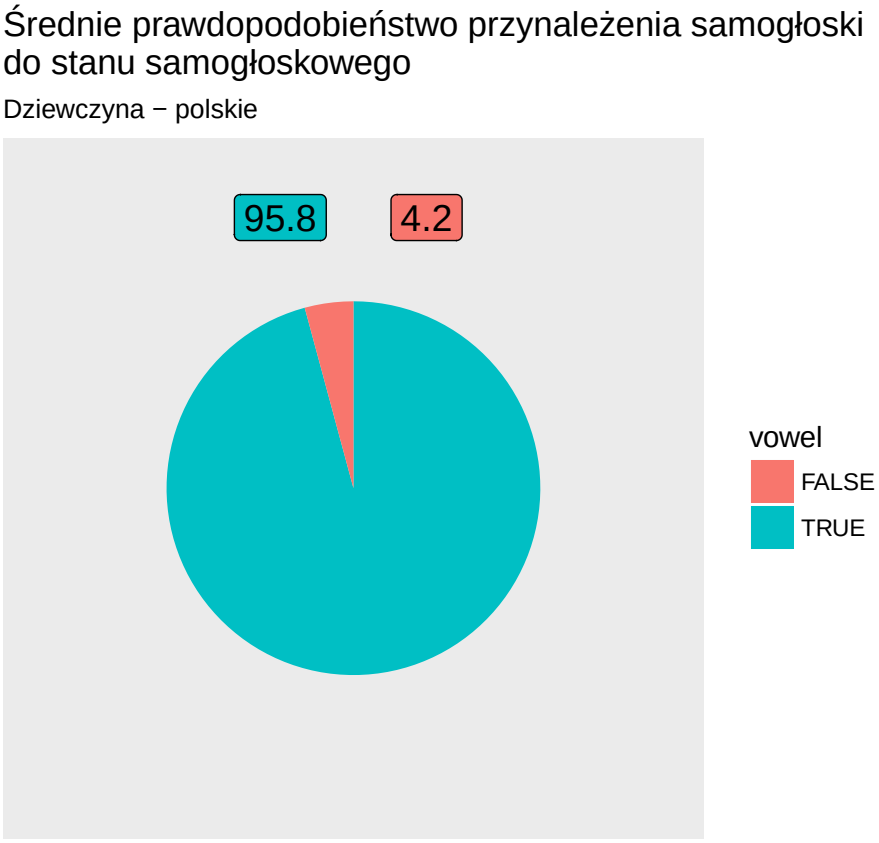
Średnie prawdopodobieństwo przynależenia samogłoski do stanu samogłoskowego

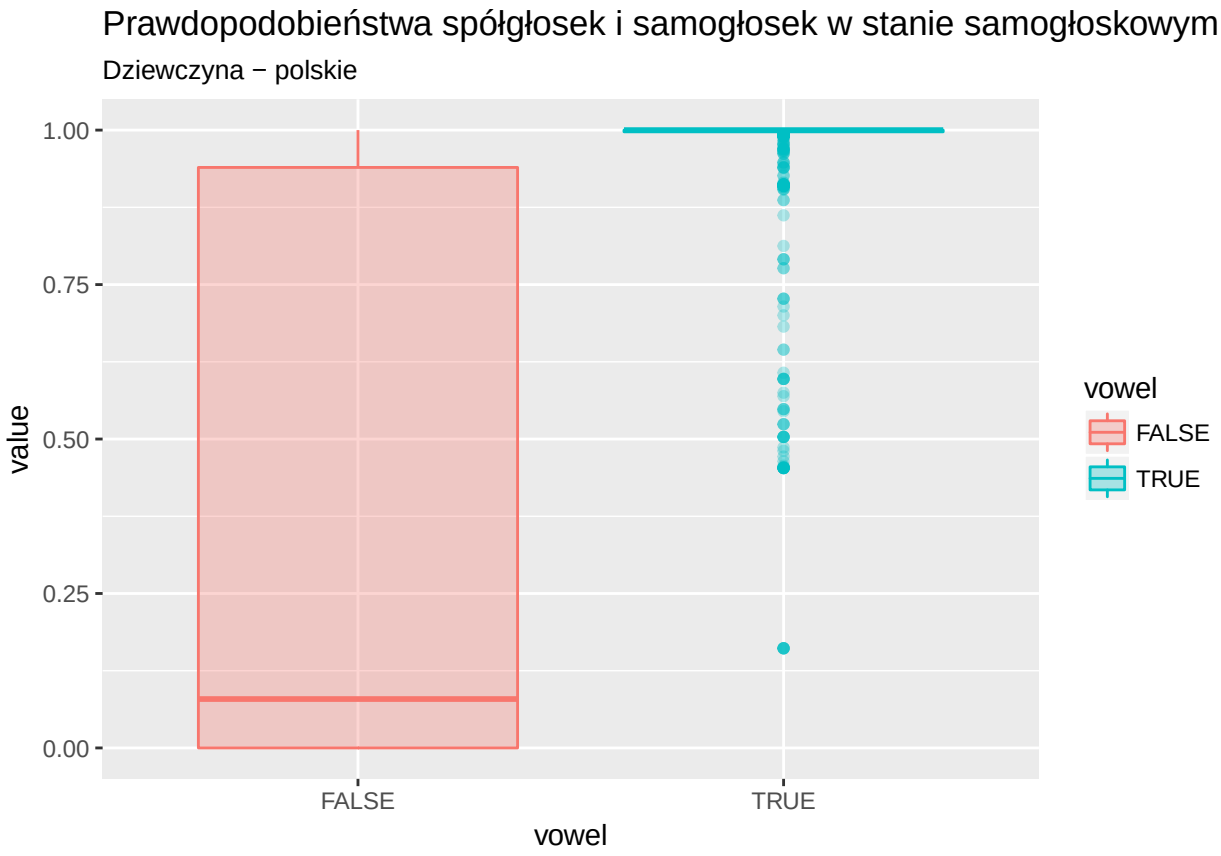
Dżabbersmok − bez polskich




Prawdopodobieństwa spółgłosek i samogłosek w stanie samogłoskowym
Dżabbersmok − bez polskich
1.00
0.75
0.50
0.25
0.00
value
FALSE
TRUE
vowel
vowel
FALSE
TRUE

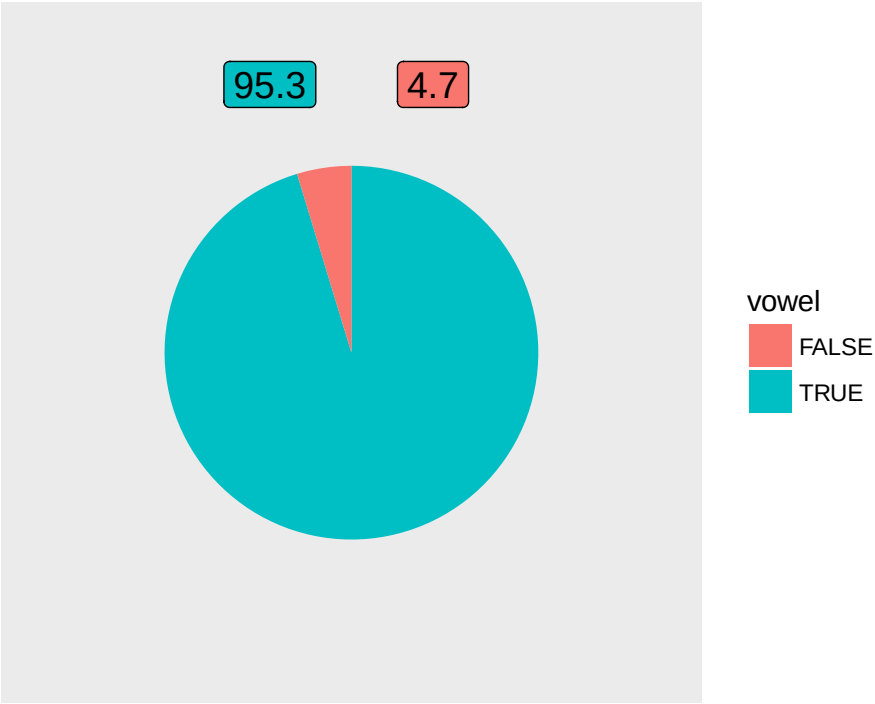
Dziewczyna

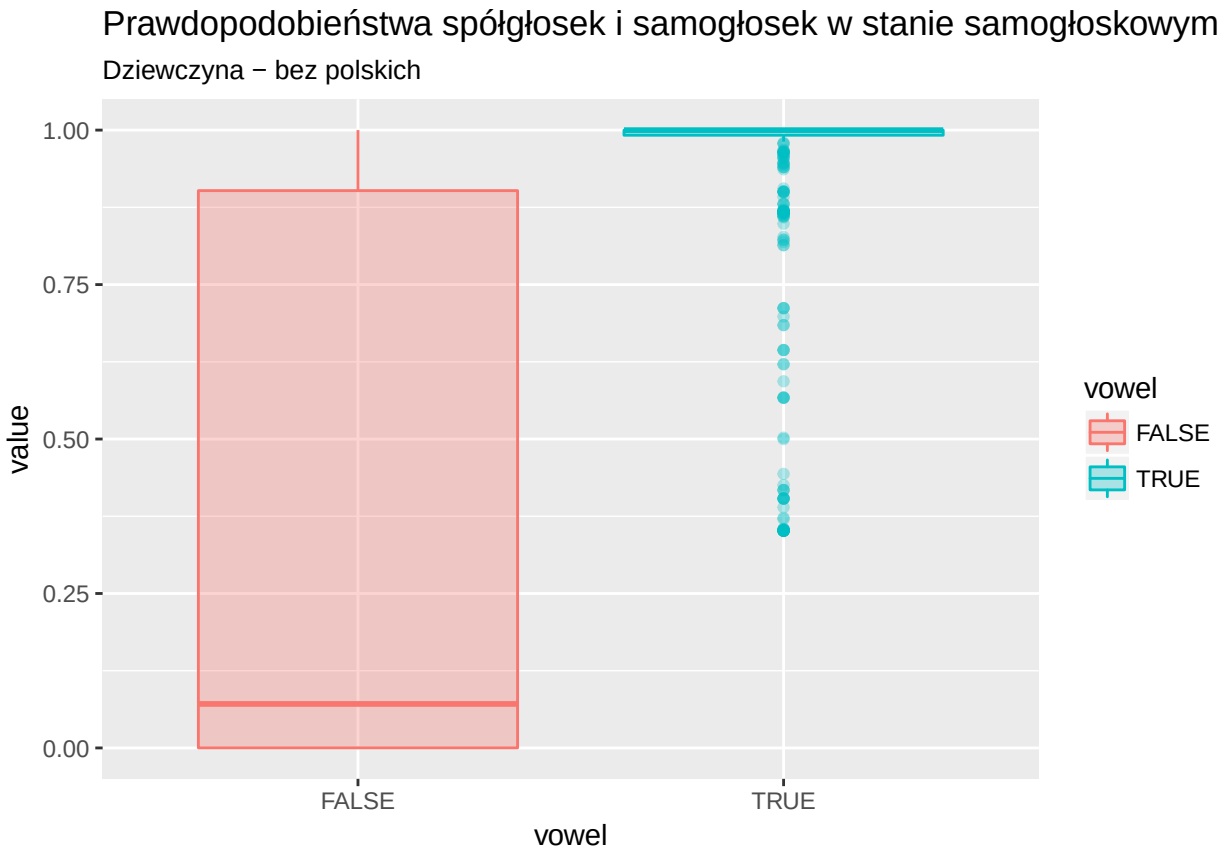

Średnie prawdopodobieństwo przynależenia samogłoski do stanu samogłoskowego
Dziewczyna − polskie
95.8
4.2
vowel
FALSE
TRUE


Prawdopodobieństwa spółgłosek i samogłosek w stanie samogłoskowym
Dziewczyna − polskie
1.00
0.75
0.50
0.25
0.00
value
FALSE
TRUE
vowel
vowel
FALSE
TRUE

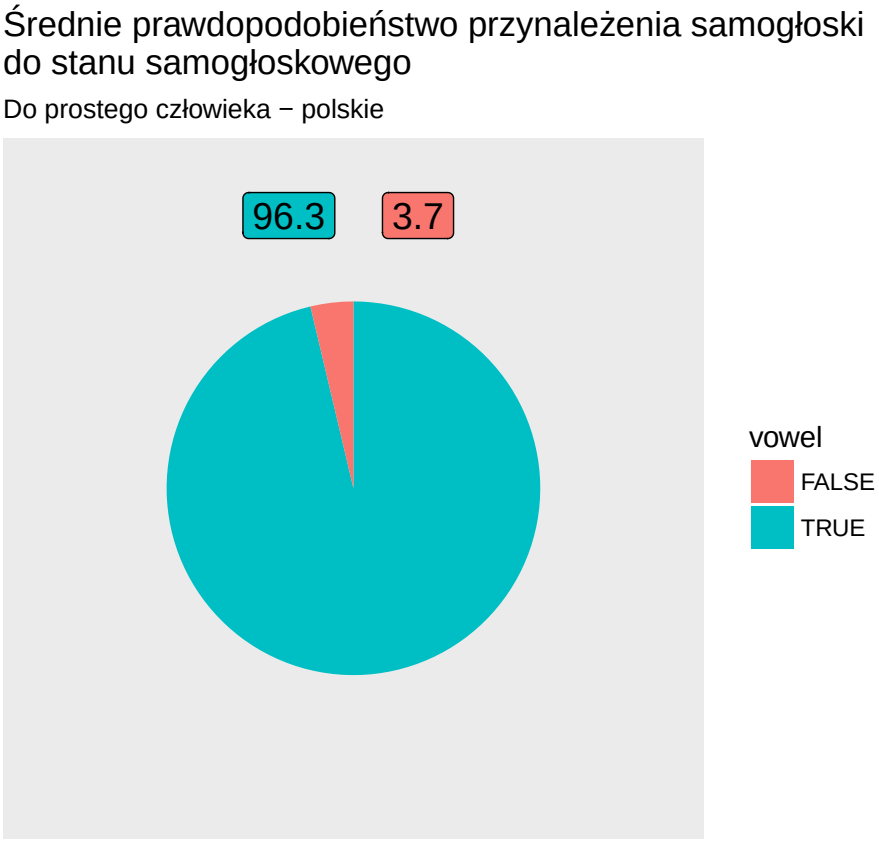
Średnie prawdopodobieństwo przynależenia samogłoski do stanu samogłoskowego

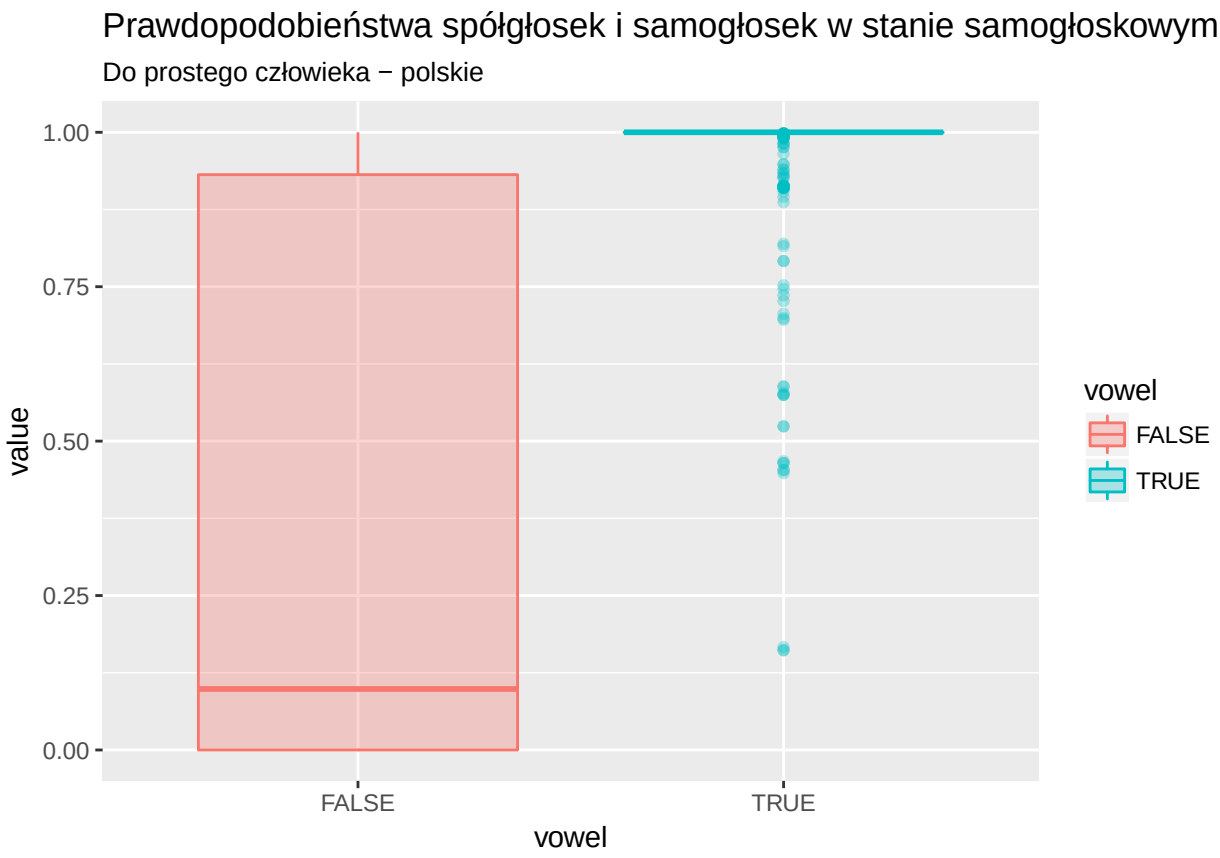
Dziewczyna − bez polskich




Prawdopodobieństwa spółgłosek i samogłosek w stanie samogłoskowym
Dziewczyna − bez polskich
1.00
0.75
0.50
0.25
0.00
value
FALSE
TRUE
vowel
vowel
FALSE
TRUE

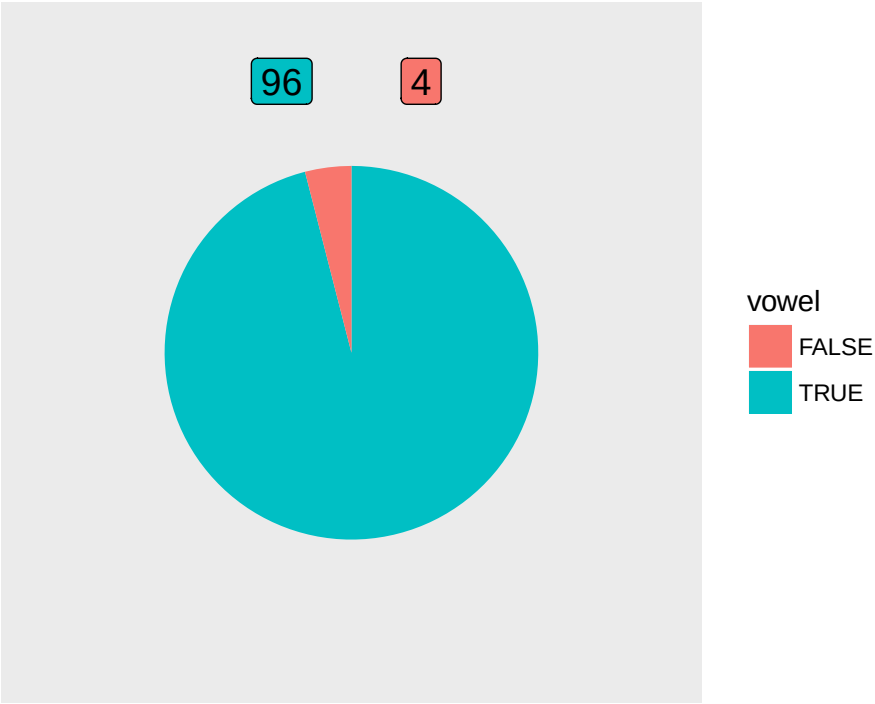
Do prostego człowieka

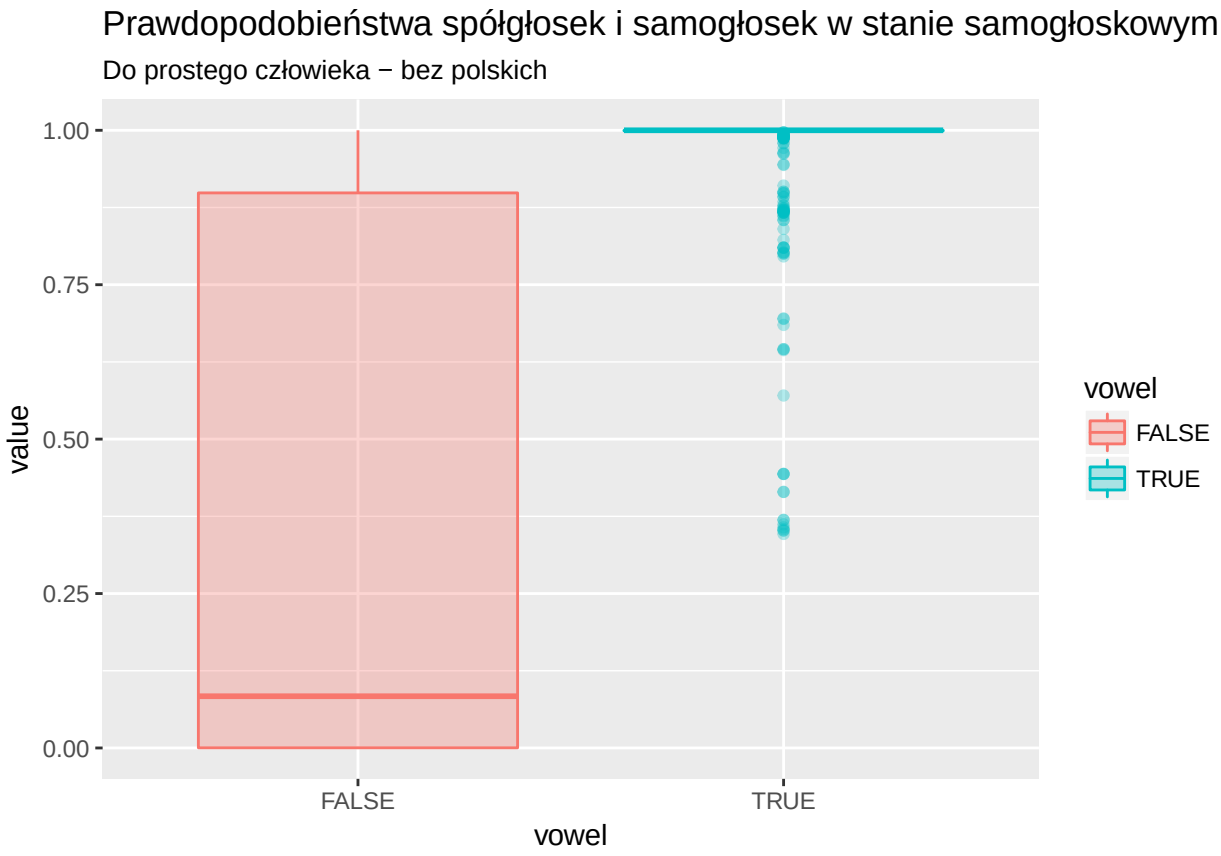



Prawdopodobieństwa spółgłosek i samogłosek w stanie samogłoskowym
Do prostego człowieka − polskie
1.00
0.75
0.50
0.25
0.00
value
FALSE
TRUE
vowel
vowel
FALSE
TRUE

Średnie prawdopodobieństwo przynależenia samogłoski do stanu samogłoskowego

Do prostego człowieka − bez polskich




Prawdopodobieństwa spółgłosek i samogłosek w stanie samogłoskowym
Do prostego człowieka − bez polskich
1.00
0.75
0.50
0.25
0.00
value
FALSE
TRUE
vowel
vowel
FALSE
TRUE