Eksploracja tekstu i wyszukiwanie informacji w mediach społecznościowych

LABORATORIUM 4

- Prawo Zipfa
- Prawo Heapa

Prawo Zipfa

Jak zostało wspomniane na Wykładzie 3 (http://www.if.pw.edu.pl/~julas/TEXT/pliki/TEXT3.pdf), istnieje szereg tzw. **praw lingwistycznych**, które, podobnie jak prawa fizyczne, mają postać kwantytatywną, wiążąc jedne wielkości z innymi (np liczbę słów i liczbę słów unikalnych) lub też określają jakiś rozkład prawdopodobieństwa.

Najpopularniejszym prawem, znanym szeroko również poza interdyscyplinarną dziedziną lingwistyki kwantytatywnej, jest **prawo Zipfa**, związane z częstością występowania słów. Zgodnie z tym prawem słowa o randze r występują z częstością f daną wzorem:

$f(r) \sim \frac{1}{r^\alpha}$

gdzie $\alpha \geq 1$.

Przeprowadzimy teraz analizę prawa Zipfa dla trzech książek Juliusza Verne'a:

```
# PRZYKŁAD 4.1
library(gutenbergr)
library(dplyr)
```

```
##
## Attaching package: 'dplyr'
```

```
## The following objects are masked from 'package:stats':
##
##     filter, lag
```

```
## The following objects are masked from 'package:base':
##
##     intersect, setdiff, setequal, union
```

```
library(tidytext)
library(magrittr)
library(ggplot2)

g <- gutenberg_works()

v <- gutenberg_download(c(83, 103, 1268))
```

```
## Determining mirror for Project Gutenberg from http://www.gutenberg.org/robot/harves
t
```

```
## Using mirror http://aleph.gutenberg.org
```

```
books <- g[g$gutenberg_id %in% c(83, 103, 1268),c("gutenberg_id","title")]

books
```

```
## # A tibble: 3 x 2
##   gutenberg_id title
##          <int> <chr>
## 1           83 From the Earth to the Moon; and, Round the Moon
## 2          103 Around the World in Eighty Days
## 3         1268 The Mysterious Island
```

Wykonujemy podobne operacje, jak na poprzednich zajęciach, tzn. zliczamy poszczególne słowa w każdej z trzech książek i dodatkowo dokładamy całkowita liczbę słów

```
# PRZYKŁAD 4.2

v %<>% left_join(books) %>%
  mutate(gutenberg_id = NULL)
```

```
## Joining, by = "gutenberg_id"
```

```
verne_words <- v %>%
  unnest_tokens(word, text) %>%
  count(title, word, sort = TRUE)

total_words <- verne_words %>%
  group_by(title) %>%
  summarise(total = sum(n))

verne_words %<>% left_join(total_words)
```

```
## Joining, by = "title"
```

```
verne_words
```

```
## # A tibble: 25,167 x 4
##    title                                          word      n  total
##    <chr>                                          <chr> <int>  <int>
##  1 The Mysterious Island                          the   17003 194300
##  2 From the Earth to the Moon; and, Round the Moon the    7794  92024
##  3 The Mysterious Island                          of     6708 194300
##  4 The Mysterious Island                          to     5403 194300
##  5 The Mysterious Island                          and    5183 194300
##  6 Around the World in Eighty Days                the    4713  63831
##  7 From the Earth to the Moon; and, Round the Moon of     3973  92024
##  8 The Mysterious Island                          a      3828 194300
##  9 The Mysterious Island                          was    3330 194300
## 10 The Mysterious Island                          in     2895 194300
## # ... with 25,157 more rows
```

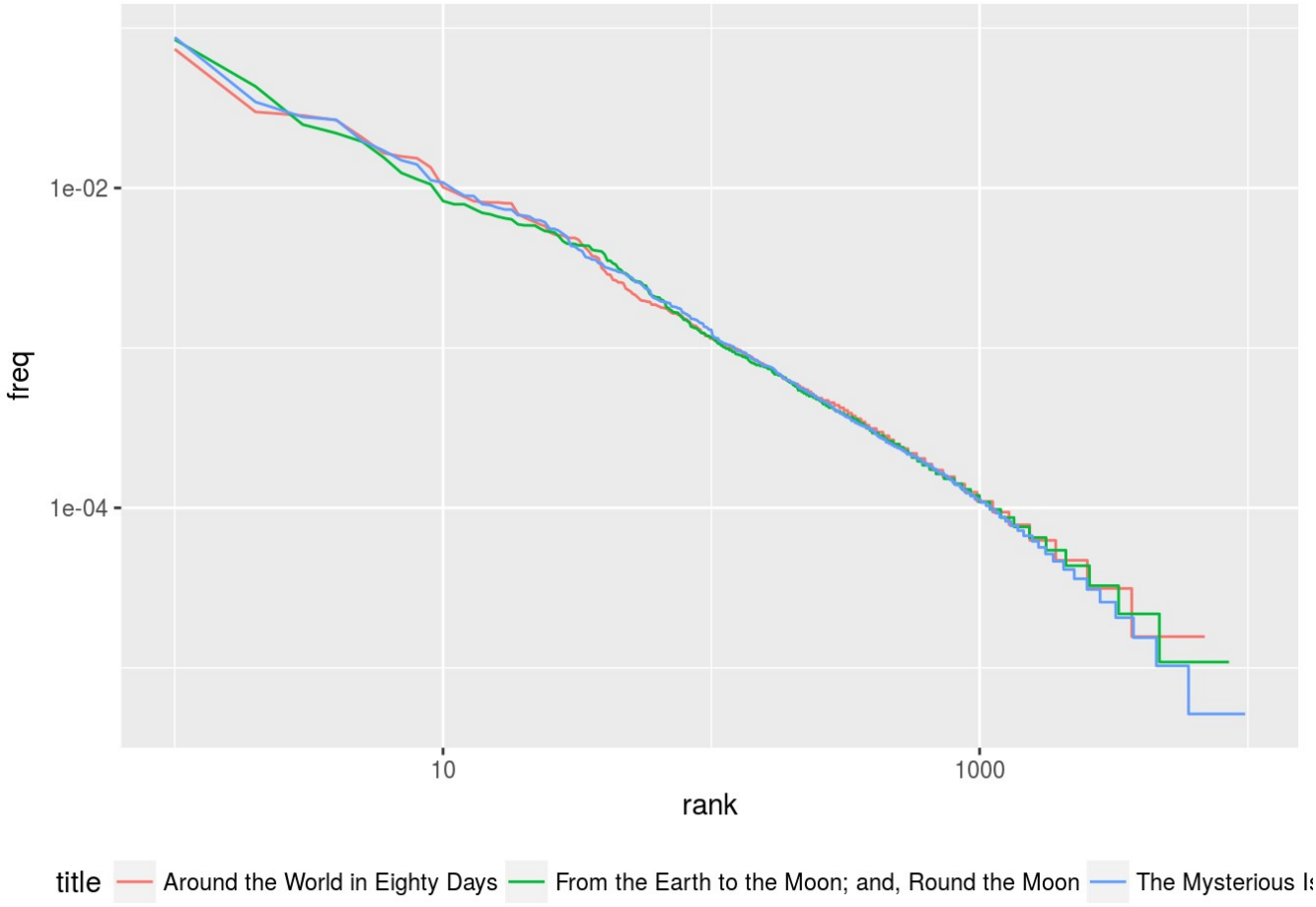
Mając te dane, w każdej grupie (tzn w dla każdej książki) możemy otrzymać zarówno rangę słów - od najczęsciej do najrzadziej występujących oraz ich częstośc:

```
# PRZYKŁAD 4.3

verne_freq <- verne_words %>%
  group_by(title) %>%
  mutate(rank = row_number(), freq = n / total)

verne_freq
```

```
## # A tibble: 25,167 x 6
## # Groups:   title [3]
##    title                                  word      n  total  rank   freq
##    <chr>                                  <chr> <int>  <int> <int>  <dbl>
##  1 The Mysterious Island                  the   17003 194300     1 0.0875
##  2 From the Earth to the Moon; and, Round. the    7794  92024     1 0.0847
##  3 The Mysterious Island                  of     6708 194300     2 0.0345
##  4 The Mysterious Island                  to     5403 194300     3 0.0278
##  5 The Mysterious Island                  and    5183 194300     4 0.0267
##  6 Around the World in Eighty Days        the    4713  63831     1 0.0738
##  7 From the Earth to the Moon; and, Round. of     3973  92024     2 0.0432
##  8 The Mysterious Island                  a      3828 194300     5 0.0197
##  9 The Mysterious Island                  was    3330 194300     6 0.0171
## 10 The Mysterious Island                  in     2895 194300     7 0.0149
## # ... with 25,157 more rows
```

Aby w łatwy sposób ocenić, czy dla poszczególnych książek faktycznie obserwujemy skalowanie Zipfa, dokonamy proste wizualizacji za pomocą pakietu **ggplot**

```
# PRZYKŁAD 4.4

ggplot(verne_freq) +
  geom_line(aes(x = rank, y = freq, color = title)) +
  scale_x_log10() + scale_y_log10() +
  theme(legend.position = "bottom")
```



Widać, że nie ma wiekszych różnic pomiędzy poszczególnymi pozycjami - we wszystkich przypadkach obserwujemy potęgowe skalowanie f w fukcji r.

Aby upodobnić prawo Zipfa do innych praw, ktore ``pracują'' zwykle na rozkładach prawdopodobieństwa, można jest przekształcić do następującej postaci:

$$P(f) \sim \frac{1}{r^{\alpha^*}}$$

przy czym zalezność pomiędzy α a α^* jest dana jako

$$\alpha^* = 1 + \frac{1}{\alpha}$$

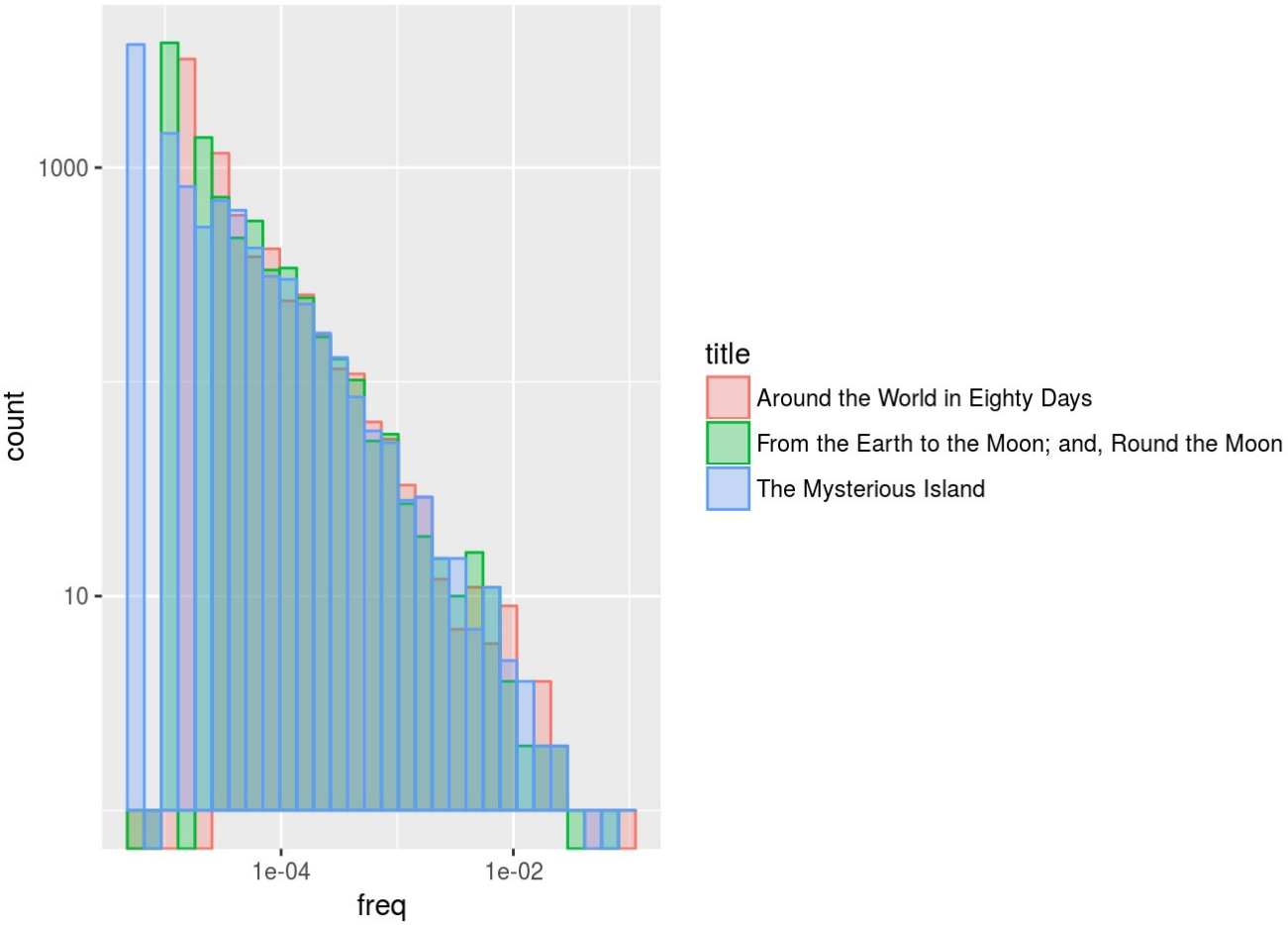
co w praktyce oznacza, że $1 < \alpha^* \leq 2$.

Dla naszego zbioru możemy łatwo wykonać histogramy częstości i przedstawić ja na wspólnym wykresie:

```
# PRZYKŁAD 4.5

ggplot(verne_freq) +
  geom_histogram(aes(x = freq, fill = title, color = title), alpha = 0.3, position="id
entity") +
  scale_x_log10() + scale_y_log10()
```

```
## `stat_bin()` using `bins = 30`. Pick better value with `binwidth`.
```

```
## Warning: Transformation introduced infinite values in continuous y-axis
```



Należy jednak zwrócić uwagę, że te zabiegi są jedynie jakościowym przedstawieniem problemu: owszem widzimy potęgową zależność pomiędzy zmiennymi, ale warto byłoby wyznaczyć konkretne wartości wykładników α i α^*.

W tym celu będziemy musieli na chwilę opuścić ggplot i wrócić do zwykłej funkcji **plot()**. Poza tym, przyda nam się funkcja **stats.bin()** z pakietu **fields**. Ograniczmy się w naszych badaniach do jednego zbioru:

```
# PRZYKŁAD 4.6

library(fields)
```

```
## Loading required package: spam
```

```
## Loading required package: dotCall64
```

```
## Loading required package: grid
```

```
## Spam version 2.1-2 (2017-12-21) is loaded.
## Type 'help( Spam)' or 'demo( spam)' for a short introduction
## and overview of this package.
## Help for individual functions is also obtained by adding the
## suffix '.spam' to the function name, e.g. 'help( chol.spam)'.
```

```
##
## Attaching package: 'spam'
```

```
## The following objects are masked from 'package:base':
##
##     backsolve, forwardsolve
```

```
## Loading required package: maps
```

```
## See www.image.ucar.edu/~nychka/Fields for
##  a vignette and other supplements.
```

```
data <- verne_freq %>% ungroup() %>%
  filter(title == "The Mysterious Island") %>%
  select(rank, freq)
```

Następnie stworzymy funkcję **log.scale()**, która będzie miała za zadanie równomiernie rozłozyc wartości pomiędzy dwiema skrajnymi, ale w sposób logarytmiczny. W kolejnym kroku dokonamy binowania danych przy udziale funkcji **stats.bin()** oraz regresji liniowej (funkcja **lm()**, w zmiennych poddanych transformacji logarytmicznej). Ostatecznie wykreślimy zależność i dopasowanie.

```
# PRZYKŁAD 4.7
log.scale <- function(x, n) exp(seq(log(x[1]), log(x[length(x)]), length.out = n))

ranks.scale <- with(data, seq(min(rank), max(rank), length.out = 20))
ranks.scale <- log.scale(ranks.scale, 20)
ranks.scale
```

```
##  [1]    1.000000    1.621728    2.630002    4.265149    6.916912
##  [6]   11.217352   18.191496   29.501661   47.843676   77.589439
## [11]  125.828981  204.060406  330.930514  536.679346  870.348029
## [16] 1411.467939 2289.017357 3712.163991 6020.121015 9763.000000
```

```
freq.sb <- stats.bin(data$rank, data$freq, breaks = ranks.scale)
freq.sb
```

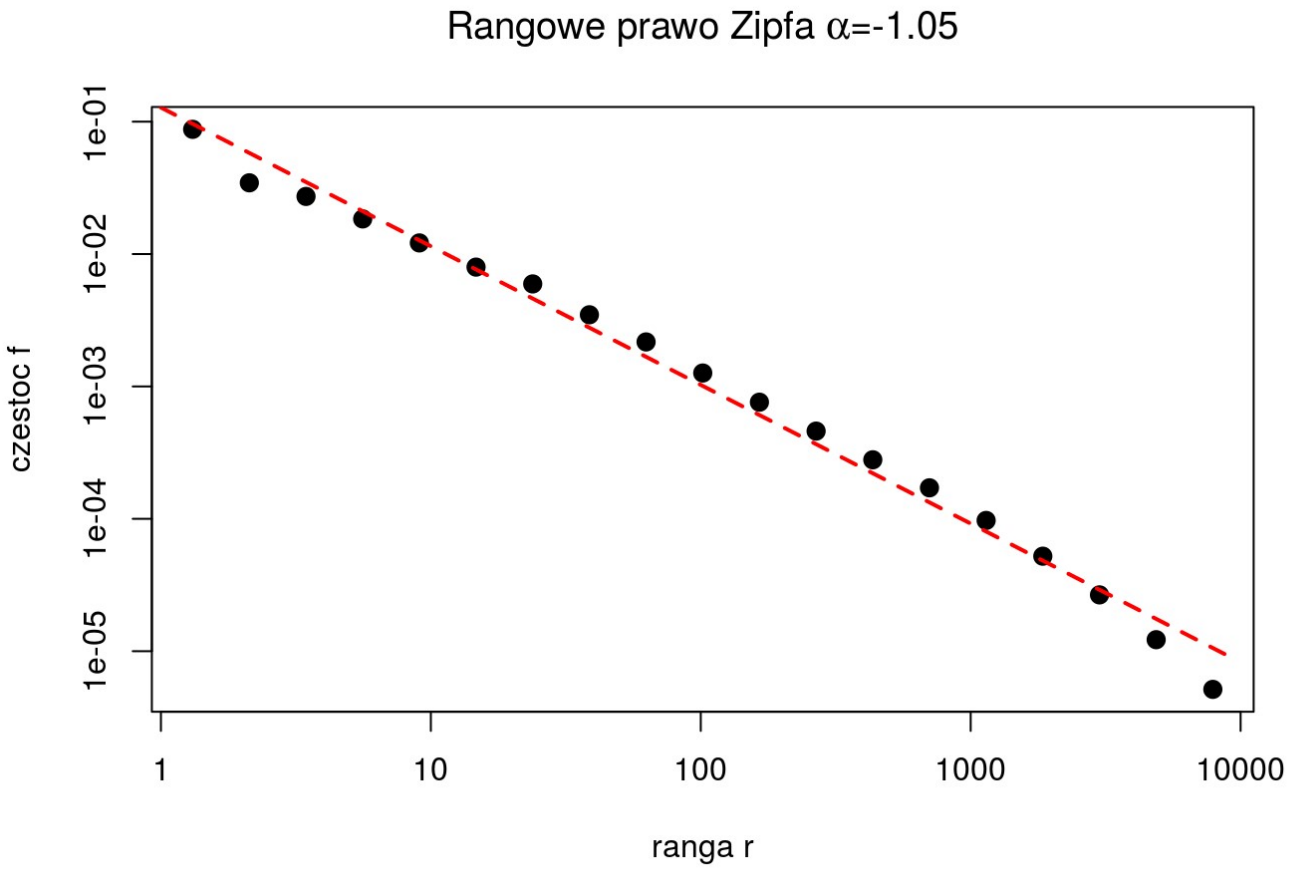
```
## $centers
##  [1]    1.310864    2.125865    3.447576    5.591031    9.067132
##  [6]   14.704424   23.846579   38.672669   62.716558  101.709210
## [11]  164.944693  267.495460  433.804930  703.513688 1140.907984
## [16] 1850.242648 3000.590674 4866.142503 7891.560508
##
## $breaks
##  [1]    1.000000    1.621728    2.630002    4.265149    6.916912
##  [6]   11.217352   18.191496   29.501661   47.843676   77.589439
## [11]  125.828981  204.060406  330.930514  536.679346  870.348029
## [16] 1411.467939 2289.017357 3712.163991 6020.121015 9763.000000
##
## $stats
##                          1          2            3           4           5
## N              1.00000000 1.00000000 2.0000000000 2.000000000 5.000000000
## mean           0.08750901 0.03452393 0.0272413793 0.018419969 0.012136902
## Std.Dev.               NA         NA 0.0008006356 0.001812348 0.002211185
## min            0.08750901 0.03452393 0.0266752445 0.017138446 0.009747813
## Q1             0.08750901 0.03452393 0.0269583119 0.017779207 0.010823469
## median         0.08750901 0.03452393 0.0272413793 0.018419969 0.011183736
## Q3             0.08750901 0.03452393 0.0275244467 0.019060731 0.014029851
## max            0.08750901 0.03452393 0.0278075142 0.019701493 0.014899640
## missing values 0.00000000 0.00000000 0.0000000000 0.000000000 0.000000000
##                          6            7            8            9
## N              7.0000000000 1.100000e+01 1.800000e+01 3.000000e+01
## mean           0.0079685317 5.947691e-03 3.477726e-03 2.168983e-03
## Std.Dev.       0.0006912135 6.470557e-04 4.660845e-04 3.595664e-04
## min            0.0073340196 4.956253e-03 2.943901e-03 1.744725e-03
## Q1             0.0074292331 5.476068e-03 3.089295e-03 1.909418e-03
## median         0.0078126608 6.098816e-03 3.386516e-03 2.022645e-03
## Q3             0.0084148224 6.466804e-03 3.690170e-03 2.472980e-03
## max            0.0089449305 6.762738e-03 4.343798e-03 2.882141e-03
## missing values 0.0000000000 0.000000e+00 0.000000e+00 0.000000e+00
##                         10           11           12           13
## N              4.800000e+01 7.900000e+01 1.260000e+02 2.060000e+02
## mean           1.262867e-03 7.601451e-04 4.597293e-04 2.791200e-04
## Std.Dev.       2.227493e-04 1.094365e-04 6.789197e-05 4.058047e-05
## min            9.572826e-04 6.021616e-04 3.602676e-04 2.213073e-04
## Q1             1.062790e-03 6.562017e-04 4.014411e-04 2.418940e-04
## median         1.188883e-03 7.617087e-04 4.554812e-04 2.727741e-04
## Q3             1.457797e-03 8.466289e-04 5.146680e-04 3.178075e-04
## max            1.672671e-03 9.521359e-04 6.021616e-04 3.602676e-04
## missing values 0.000000e+00 0.000000e+00 0.000000e+00 0.000000e+00
##                         14           15           16           17
## N              3.340000e+02 5.410000e+02 8.780000e+02 1.423000e+03
## mean           1.714276e-04 9.718759e-05 5.218195e-05 2.662675e-05
## Std.Dev.       2.588171e-05 1.638287e-05 1.027434e-05 5.540803e-06
## min            1.286670e-04 7.205353e-05 3.602676e-05 2.058672e-05
## Q1             1.492537e-04 8.234689e-05 4.117344e-05 2.058672e-05
## median         1.698405e-04 9.778693e-05 5.146680e-05 2.573340e-05
```

```
## Q3             1.904272e-04 1.132270e-04 6.176016e-05 3.088008e-05
## max            2.161606e-04 1.286670e-04 7.205353e-05 3.602676e-05
## missing values 0.000000e+00 0.000000e+00 0.000000e+00 0.000000e+00
##                          18          19
## N              2.308000e+03 3.74300e+03
## mean           1.221333e-05 5.14668e-06
## Std.Dev.       2.689094e-06 0.00000e+00
## min            5.146680e-06 5.14668e-06
## Q1             1.029336e-05 5.14668e-06
## median         1.029336e-05 5.14668e-06
## Q3             1.544004e-05 5.14668e-06
## max            2.058672e-05 5.14668e-06
## missing values 0.000000e+00 0.00000e+00
```

```
zipf1.lm <- lm(log(freq.sb$stats[2,]) ~ log(freq.sb$centers))
summary(zipf1.lm)
```

```
##
## Call:
## lm(formula = log(freq.sb$stats[2, ]) ~ log(freq.sb$centers))
##
## Residuals:
##      Min       1Q   Median       3Q      Max
## -0.71927 -0.11508  0.07752  0.22600  0.25906
##
## Coefficients:
##                       Estimate Std. Error t value Pr(>|t|)
## (Intercept)           -2.05670    0.13783  -14.92 3.36e-11 ***
## log(freq.sb$centers) -1.04766    0.02587  -40.49  < 2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 0.2987 on 17 degrees of freedom
## Multiple R-squared:  0.9897, Adjusted R-squared:  0.9891
## F-statistic:  1640 on 1 and 17 DF,  p-value: < 2.2e-16
```

```
A <- zipf1.lm$coefficients[1]
alfa <- zipf1.lm$coefficients[2]

plot(freq.sb$centers, freq.sb$stats[2,], log="xy", xlab = "ranga r",
     ylab = "czestoc f", pch = 19, cex = 1.2,
     main = substitute(paste("Rangowe prawo Zipfa ", alpha,"=",a),list(a = round(alfa,
2))))

lines(ranks.scale, exp(A) * ranks.scale ** alfa, col = "red", lty = 2, lwd = 2)
```

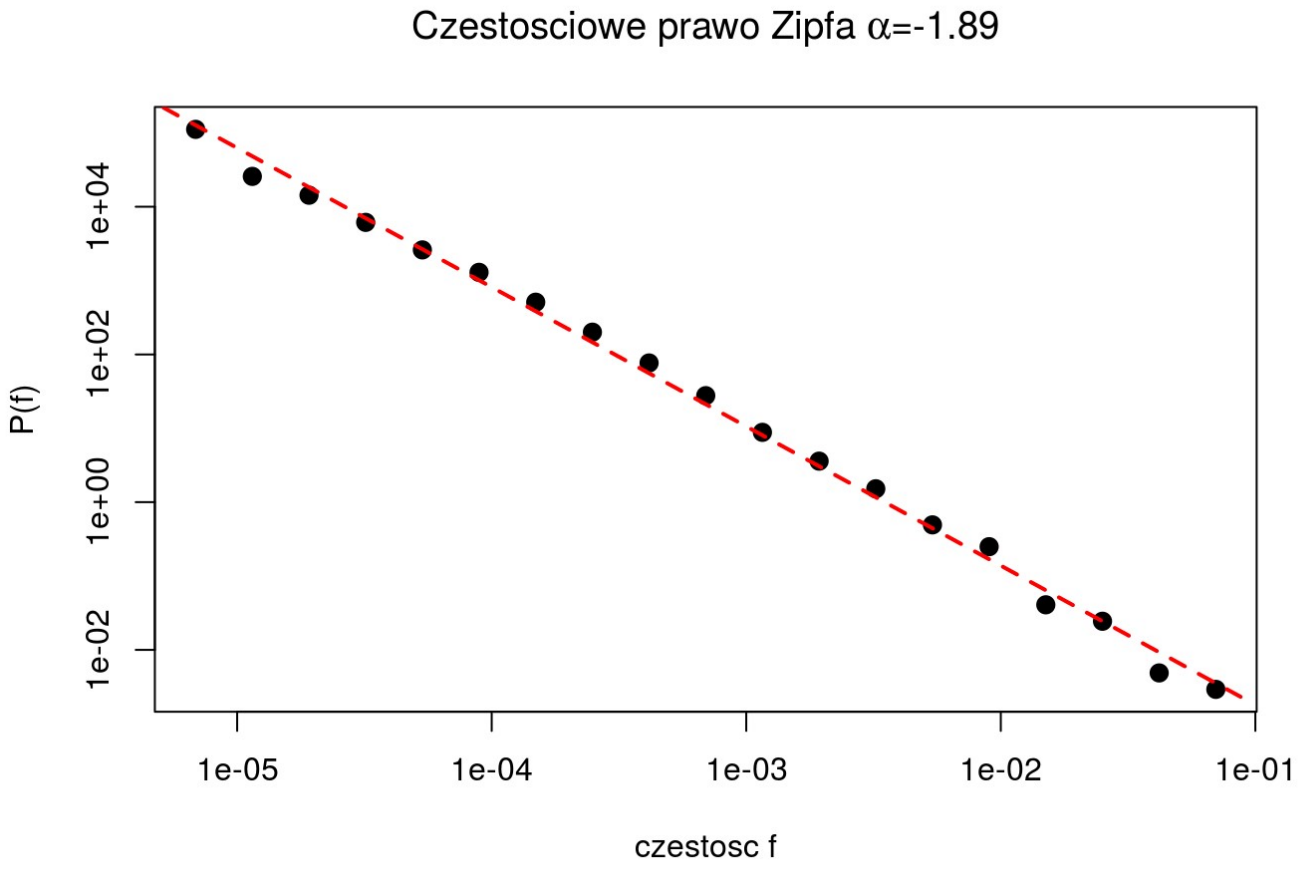


Przprowadzimy teraz podobną procedurę w przypadku drugiej postaci prawa Zipfa - związanej z rozkładem częstości. Tu różnica jest taka, że zamiast wykorzystywać funkcję **stats.bin()**, potrzebną w przypadku relacji y(x), uzyjemy po prostu histogramu, bo zależność jest jednowymiarowa.

```
# PRZYKŁAD 4.8

freqs.scale <- with(data, seq(min(freq), max(freq), length.out = 20))
freqs.scale <- log.scale(freqs.scale, 20)

h <- hist(data$freq, breaks = freqs.scale, plot = FALSE)

freq.lm <- lm(log(h$density) ~ log(h$mids))
summary(freq.lm)
```

```
##
## Call:
## lm(formula = log(h$density) ~ log(h$mids))
##
## Residuals:
##      Min       1Q   Median       3Q      Max
## -0.64458 -0.15989  0.09071  0.26627  0.39186
##
## Coefficients:
##             Estimate Std. Error t value Pr(>|t|)
## (Intercept) -10.6604     0.2090  -51.01   <2e-16 ***
## log(h$mids)  -1.8852     0.0268  -70.33   <2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 0.3281 on 17 degrees of freedom
## Multiple R-squared:  0.9966, Adjusted R-squared:  0.9964
## F-statistic:  4947 on 1 and 17 DF,  p-value: < 2.2e-16
```

```
A <- freq.lm$coefficients[1]
alfa <- freq.lm$coefficients[2]

plot(h$mids, h$density, log="xy", xlab = "czestosc f", ylab = "P(f)",
     pch = 19, cex = 1.2,
     main = substitute(paste("Czestosciowe prawo Zipfa ", alpha,"=",a),list(a = round(
alfa, 2))))

lines(freqs.scale, exp(A) * freqs.scale ** alfa, col = "red", lty = 2, lwd = 2)
```



Prawo Heapa

Drugim prawem, które wyjątkowo często jest sprawdzane we wszelkiego rodzaju korpusach i tekstach jest **prawo Heapa**, które wiąże liczbę unikalnych słów w tekście V z calkowitą liczbą słów N relacją:

$V \sim N^{\beta}$

przy czym zwykle $0 < \beta < 1$.

Zbiór do testowania prawa Heapa uda nam sie uzyskac w dośc prosty sposób -- użyjemy wygodnej funkcji **duplicated()**, która zwraca fałsz, gdy dany token pojawił się już we wcześniejszych elementach wektora:

```
# PRZYKŁAD 4.9

verne_heaps <- v %>%
  unnest_tokens(word, text) %>%
  group_by(title) %>%
  mutate(M = row_number(), V = cumsum(!duplicated(word)))

verne_heaps
```

```
## # A tibble: 350,155 x 4
## # Groups:   title [3]
##    title                                          word      M     V
##    <chr>                                          <chr> <int> <int>
##  1 From the Earth to the Moon; and, Round the Moon from      1     1
##  2 From the Earth to the Moon; and, Round the Moon the       2     2
##  3 From the Earth to the Moon; and, Round the Moon earth     3     3
##  4 From the Earth to the Moon; and, Round the Moon to        4     4
##  5 From the Earth to the Moon; and, Round the Moon the       5     4
##  6 From the Earth to the Moon; and, Round the Moon moon      6     5
##  7 From the Earth to the Moon; and, Round the Moon by        7     6
##  8 From the Earth to the Moon; and, Round the Moon jules     8     7
##  9 From the Earth to the Moon; and, Round the Moon verne     9     8
## 10 From the Earth to the Moon; and, Round the Moon table    10     9
## # ... with 350,145 more rows
```

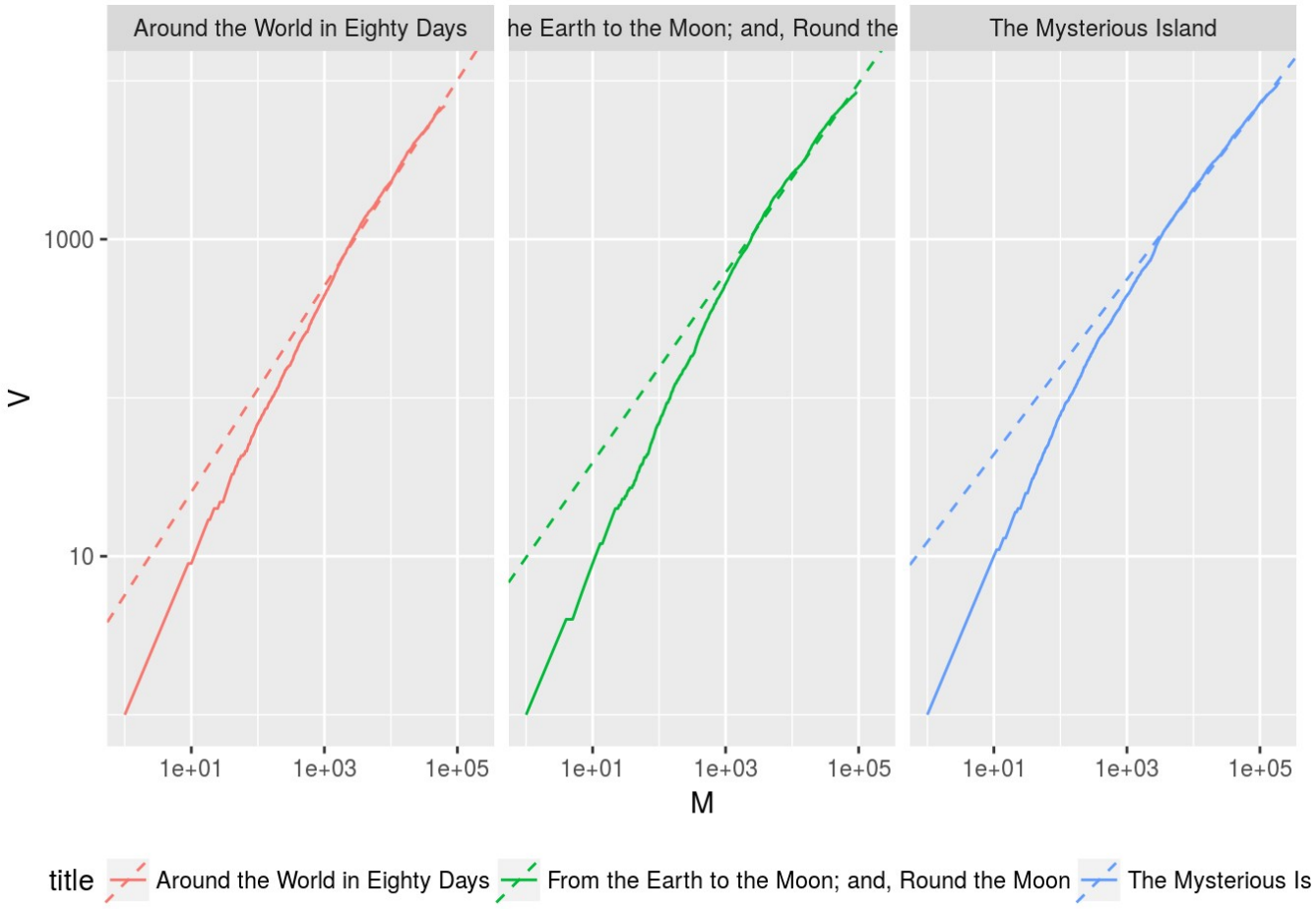
Następnie użyjemy funkcji **summarise()** do wyznaczenia współczynników regresji dla poszczególnych książek:

```
# PRZYKŁAD 4.10

verne_sum <- verne_heaps %>%
  summarise(a = lm(log10(V) ~ log10(M))$coefficients[1], b = lm(log10(V) ~ log10(M))$c
oefficients[2])

verne_sum
```

```
## # A tibble: 3 x 3
##   title                                             a     b
##   <chr>                                         <dbl> <dbl>
## 1 Around the World in Eighty Days               0.755 0.650
## 2 From the Earth to the Moon; and, Round the Moon 0.992 0.599
## 3 The Mysterious Island                         1.09  0.552
```

Teraz w łatwy sposób można wykreślić zarówno obserowane prawo Heapa jak i otrzymane wartości dopasowania na jednym wykresie:

```
# PRZYKŁAD 4.10

ggplot(verne_heaps, aes(M, V, color = title)) +
  geom_line() +
  geom_abline(data = verne_sum, aes(intercept = a, slope = b, color = title), linetype
= "dashed") +
  theme(legend.position = "bottom") +
  scale_x_log10() + scale_y_log10() +
  facet_grid(~title)
```



Nie da się jednak ukryć, że w zależności od zakresu osi N wartości dopasowania mogą się mocno róznić pomiędzy sobą. Warto przyjrzeć się temu w sposób bardziej systematyczny. W tym celu stworzymy wrapper na funkcję **lm()**, który będzie przyjmował jako parametr górną granicę N, dla której będzie wyznaczane dopasowanie, a zwracał współczynniki regresji.

```
# PRZYKŁAD 4.11

make.lm <- function(data, threshold) {
  data %>%
    filter(M > 10 & M < threshold) %>%
    summarise(th = threshold, A = lm(log(V) ~ log(M))$coefficients[1], beta = lm(log(V
) ~ log(M))$coefficients[2])
}

threshold <- seq(100, 20000, 100)

lm.res <- do.call(rbind, lapply(threshold, make.lm, data = verne_heaps))
lm.res
```

```
## # A tibble: 600 x 4
##    title                                              th       A  beta
##    <chr>                                           <dbl>   <dbl> <dbl>
##  1 Around the World in Eighty Days                   100 0.333   0.841
##  2 From the Earth to the Moon; and, Round the Moon   100 0.264   0.850
##  3 The Mysterious Island                             100 0.0662  0.928
##  4 Around the World in Eighty Days                   200 0.368   0.832
##  5 From the Earth to the Moon; and, Round the Moon   200 0.00704 0.922
##  6 The Mysterious Island                             200 0.153   0.905
##  7 Around the World in Eighty Days                   300 0.383   0.829
##  8 From the Earth to the Moon; and, Round the Moon   300 0.0769  0.905
##  9 The Mysterious Island                             300 0.209   0.891
## 10 Around the World in Eighty Days                   400 0.415   0.821
## # ... with 590 more rows
```

Funkcję wywołaliśmy dla całego zestawu parametrów, a wyniki zostały sklejone do jednej ramki danych. Teraz po prostu możemy wykreślic wykładnik β w funkcji progu:

```
# PRZYKŁAD 4.12

ggplot(lm.res) +
  geom_line(aes(th, beta, color = title)) +
  theme(legend.position = "bottom") +
  facet_grid(~title)
```