VMMT1 REPORT AND SOLUTIONS

This project consists of programming quests we have the school course ‘Matematické modelování textu 1’ (EN: Mathematical modelling of text 1). The completeness of this project means to be success of this course (VMMT1). Later, I added this project to my Github repository for illustration purposes.

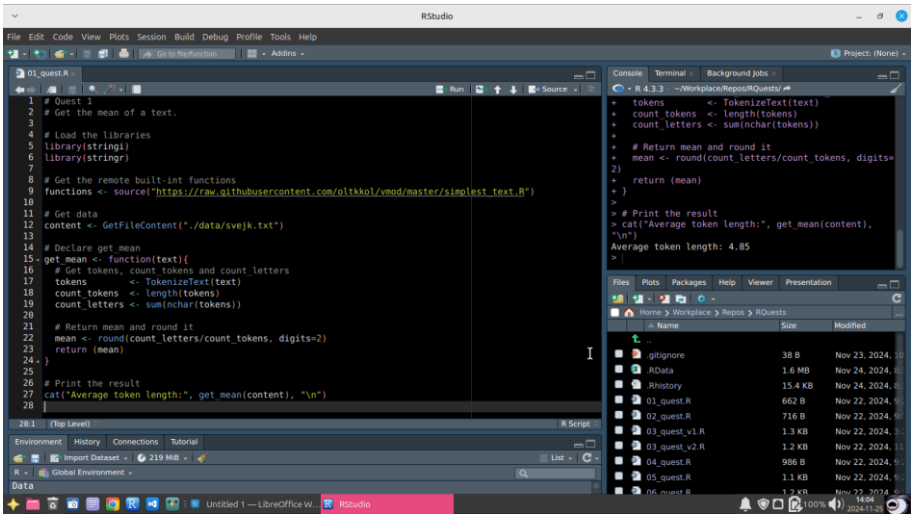
Each chapter for a quest is built with:

- Screenshot
- Raw cade as plaint text
- Output as in the comment form

Contact and more information about me:

Github: https://github.com/kivanc57/RQuests

01_quest.R

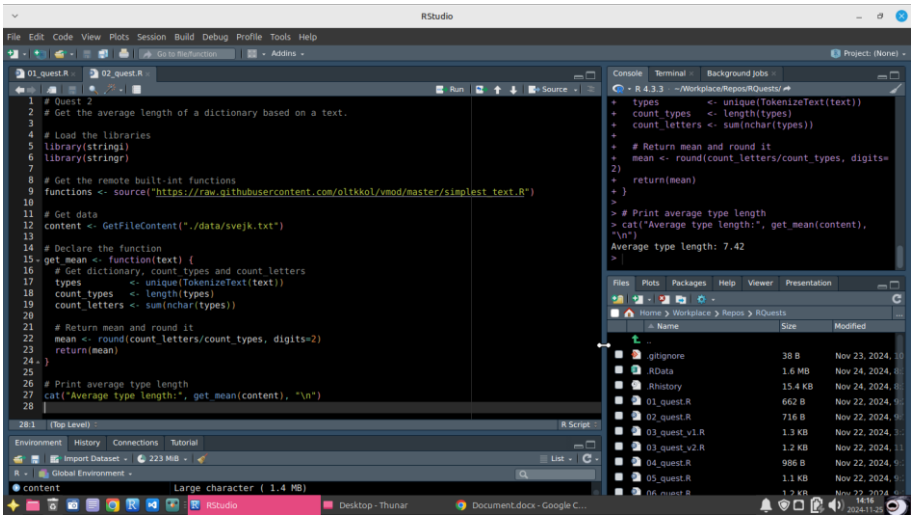



```
# Quest 1
# Get the mean of a text.

# Load the libraries
library(stringi)
library(stringr)

# Get the remote built-int functions
functions <- source("https://raw.githubusercontent.com/oltkkol/vmod/master/simplest_text.R")

# Get data
content <- GetFileContent("./data/svejk.txt")

# Declare get_mean
get_mean <- function(text){
 # Get tokens, count_tokens and count_letters
 tokens       <- TokenizeText(text)
 count_tokens  <- length(tokens)
 count_letters <- sum(nchar(tokens))

 # Return mean and round it
```

```
 mean <- round(count_letters/count_tokens, digits=2)
 return (mean)
}

# Print the result
cat("Average token length:", get_mean(content), "\n")
```

Commented [GK1]: Average token length: 4.85

02_quest.R

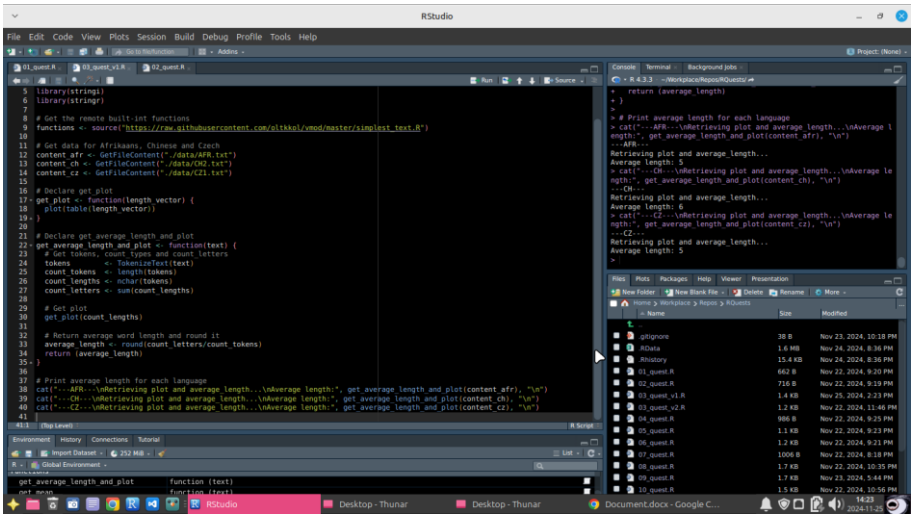



```
# Quest 2
# Get the average length of a dictionary based on a text.

# Load the libraries
library(stringi)
library(stringr)

# Get the remote built-int functions
functions <- source("https://raw.githubusercontent.com/oltkkol/vmod/master/simplest_text.R")

# Get data
content <- GetFileContent("./data/svejk.txt")

# Declare the function
get_mean <- function(text) {
```

```
# Get dictionary, count_types and count_letters
types        <- unique(TokenizeText(text))
count_types  <- length(types)
count_letters <- sum(nchar(types))

# Return mean and round it
mean <- round(count_letters/count_types, digits=2)
return(mean)
}

# Print average type length
cat("Average type length:", get_mean(content), "\n")
```

Commented [GK2]: Average type length: 7.42

03_quest_v1.R

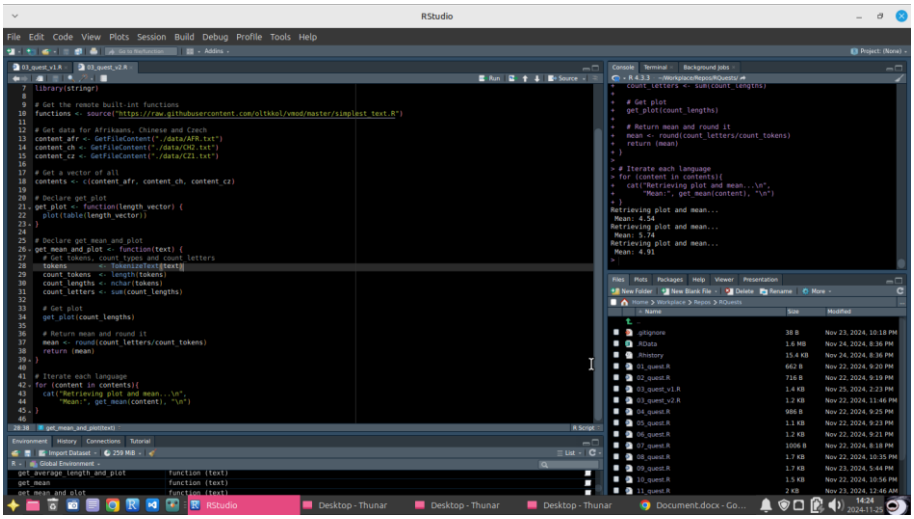



```
# Quest 3
# Get the average length of three different languages and make histograms.

# Load the libraries
library(stringi)
library(stringr)

# Get the remote built-int functions
functions <- source("https://raw.githubusercontent.com/oltkkol/vmod/master/simplest_text.R")
```

```
# Get data for Afrikaans, Chinese and Czech
content_afr <- GetFileContent("./data/AFR.txt")
content_ch <- GetFileContent("./data/CH2.txt")
content_cz <- GetFileContent("./data/CZ1.txt")

# Declare get_plot
get_plot <- function(length_vector) {
 plot(table(length_vector))
}

# Declare get_average_length_and_plot
get_average_length_and_plot <- function(text) {
 # Get tokens, count_types and count_letters
 tokens       <- TokenizeText(text)
 count_tokens  <- length(tokens)
 count_lengths <- nchar(tokens)
 count_letters <- sum(count_lengths)

 # Get plot
 get_plot(count_lengths)

 # Return average word length and round it
 average_length <- round(count_letters/count_tokens)
 return (average_length)
}

# Print average length for each language
cat("---AFR---\nRetrieving plot and average_length...\nAverage length:", get_average_length_and_plot(content_afr), "\n")
cat("---CH---\nRetrieving plot and average_length...\nAverage length:", get_average_length_and_plot(content_ch), "\n")
cat("---CZ---\nRetrieving plot and average_length...\nAverage length:", get_average_length_and_plot(content_cz), "\n")
```

Commented [GK3]: ---AFR---
Retrieving plot and average_length...
Average length: 5
> cat("---CH---\nRetrieving plot and average_length...\nAverage length:", get_average_length_and_plot(content_ch), "\n")
---CH---
Retrieving plot and average_length...
Average length: 6
> cat("---CZ---\nRetrieving plot and average_length...\nAverage length:", get_average_length_and_plot(content_cz), "\n")
---CZ---
Retrieving plot and average_length...
Average length: 5

03_quest_v2.R



Quest 3

Get the mean of three different languages and make histograms.

PS: This version is the iterative version with for

Load the libraries

library(stringi)

library(stringr)

Get the remote built-int functions

functions <- source("https://raw.githubusercontent.com/oltkkol/vmod/master/simplest_text.R")

Get data for Afrikaans, Chinese and Czech

content_afr <- GetFileContent("./data/AFR.txt")

content_ch <- GetFileContent("./data/CH2.txt")

content_cz <- GetFileContent("./data/CZ1.txt")

Get a vector of all

contents <- c(content_afr, content_ch, content_cz)

Declare get_plot

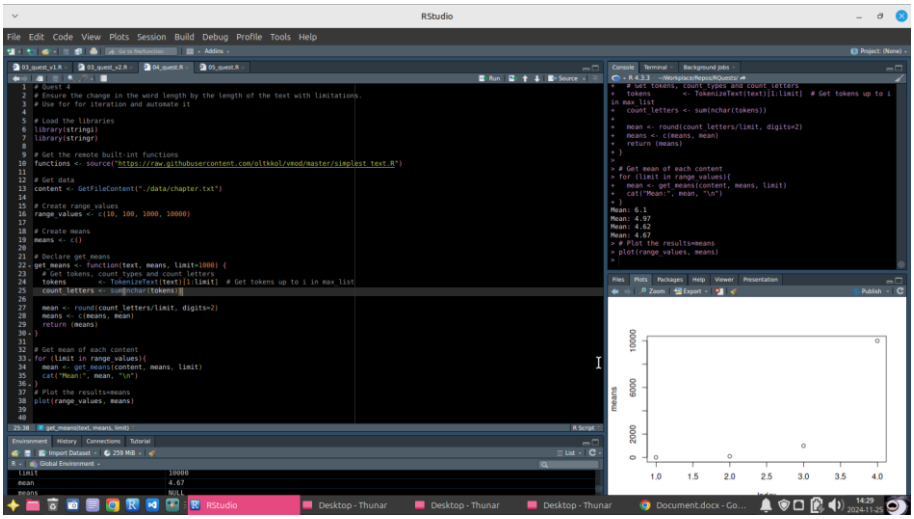
get_plot <- function(length_vector) {

 plot(table(length_vector))

```
}

# Declare get_mean_and_plot
get_mean_and_plot <- function(text) {
 # Get tokens, count_types and count_letters
 tokens        <- TokenizeText(text)
 count_tokens  <- length(tokens)
 count_lengths <- nchar(tokens)
 count_letters <- sum(count_lengths)

 # Get plot
 get_plot(count_lengths)

 # Return mean and round it
 mean <- round(count_letters/count_tokens)
 return (mean)
}

# Iterate each language
for (content in contents){
 cat("Retrieving plot and mean...\n",
     "Mean:", get_mean(content), "\n")
}
```

Commented [GK4]:
Retrieving plot and mean...
Mean: 4.54
Retrieving plot and mean...
Mean: 5.74
Retrieving plot and mean...
Mean: 4.91

04_quest.R

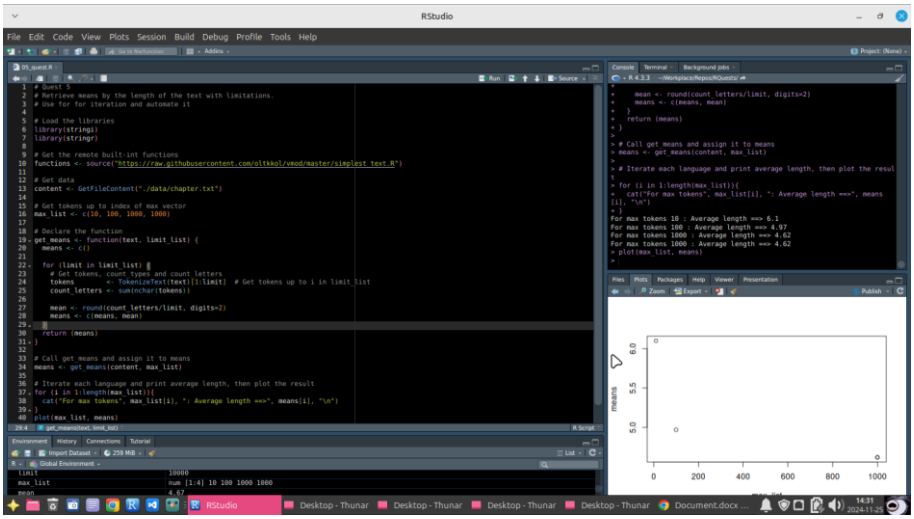



```
# Quest 4

# Ensure the change in the word length by the length of the text with limitations.

# Use for for iteration and automate it



# Load the libraries

library(stringi)

library(stringr)



# Get the remote built-int functions

functions <- source("https://raw.githubusercontent.com/oltkkol/vmod/master/simplest_text.R")



# Get data

content <- GetFileContent("./data/chapter.txt")



# Create range_values

range_values <- c(10, 100, 1000, 10000)



# Create means

means <- c()



# Declare get_means

get_means <- function(text, means, limit=1000) {
```

```
 # Get tokens, count_types and count_letters
 tokens      <- TokenizeText(text)[1:limit]  # Get tokens up to i in max_list
 count_letters <- sum(nchar(tokens))


 mean <- round(count_letters/limit, digits=2)
 means <- c(means, mean)
 return (means)
}

# Get mean of each content
for (limit in range_values){
 mean <- get_means(content, means, limit)
 cat("Mean:", mean, "\n")
}
# Plot the results=means
plot(range_values, means)
```

Commented [GK5]: Mean: 6.1
Mean: 4.97
Mean: 4.62
Mean: 4.67

05_quest.R

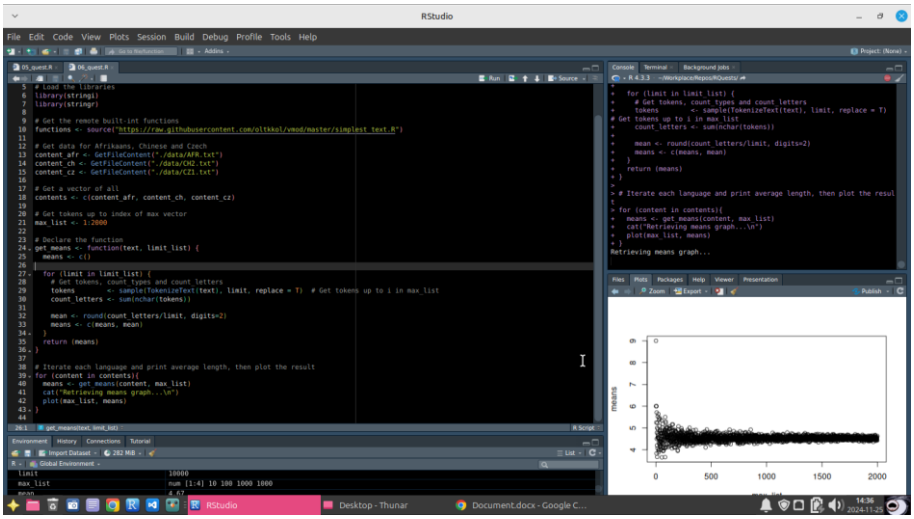



```
# Quest 5
# Retrieve means by the length of the text with limitations.
# Use for for iteration and automate it

# Load the libraries
```

```
library(stringi)
library(stringr)

# Get the remote built-int functions
functions <- source("https://raw.githubusercontent.com/oltkkol/vmod/master/simplest_text.R")

# Get data
content <- GetFileContent("../data/chapter.txt")

# Get tokens up to index of max vector
max_list <- c(10, 100, 1000, 1000)

# Declare the function
get_means <- function(text, limit_list) {
 means <- c()

 for (limit in limit_list) {
  # Get tokens, count_types and count_letters
  tokens       <- TokenizeText(text)[1:limit]  # Get tokens up to i in limit_list
  count_letters <- sum(nchar(tokens))

  mean <- round(count_letters/limit, digits=2)
  means <- c(means, mean)
 }
 return (means)
}

# Call get_means and assign it to means
means <- get_means(content, max_list)

# Iterate each language and print average length, then plot the result
for (i in 1:length(max_list)){
 cat("For max tokens", max_list[i], ": Average length ==>", means[i], "\n")
}
plot(max_list, means)
```

Commented [GK6]: For max tokens 10 : Average length ==> 6.1
For max tokens 100 : Average length ==> 4.97
For max tokens 1000 : Average length ==> 4.62
For max tokens 1000 : Average length ==> 4.62

06_quest.R



Quest 6

Get the mean of three different languages and make histograms.

Get the tokens randomly for each.

Load the libraries

library(stringi)

library(stringr)

Get the remote built-int functions

functions <- source("https://raw.githubusercontent.com/oltkkol/vmod/master/simplest_text.R")

Get data for Afrikaans, Chinese and Czech

content_afr <- GetFileContent("./data/AFR.txt")

content_ch <- GetFileContent("./data/CH2.txt")

content_cz <- GetFileContent("./data/CZ1.txt")

Get a vector of all

contents <- c(content_afr, content_ch, content_cz)

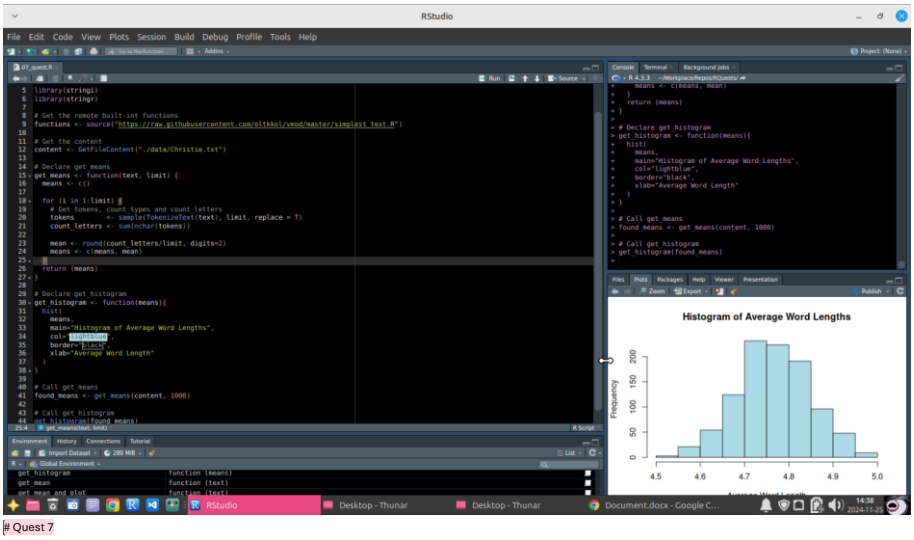
Get tokens up to index of max vector

max_list <- 1:2000

```
# Declare the function
get_means <- function(text, limit_list) {
 means <- c()

 for (limit in limit_list) {
  # Get tokens, count_types and count_letters
  tokens      <- sample(TokenizeText(text), limit, replace = T)  # Get tokens up to i in max_list
  count_letters <- sum(nchar(tokens))

  mean <- round(count_letters/limit, digits=2)
  means <- c(means, mean)
 }
 return (means)
}

# Iterate each language and print average length, then plot the result
for (content in contents){
 means <- get_means(content, max_list)
 cat("Retrieving means graph....\n")
 plot(max_list, means)
}
```

Commented [GK7]: (returns only plots)

07_quest.R

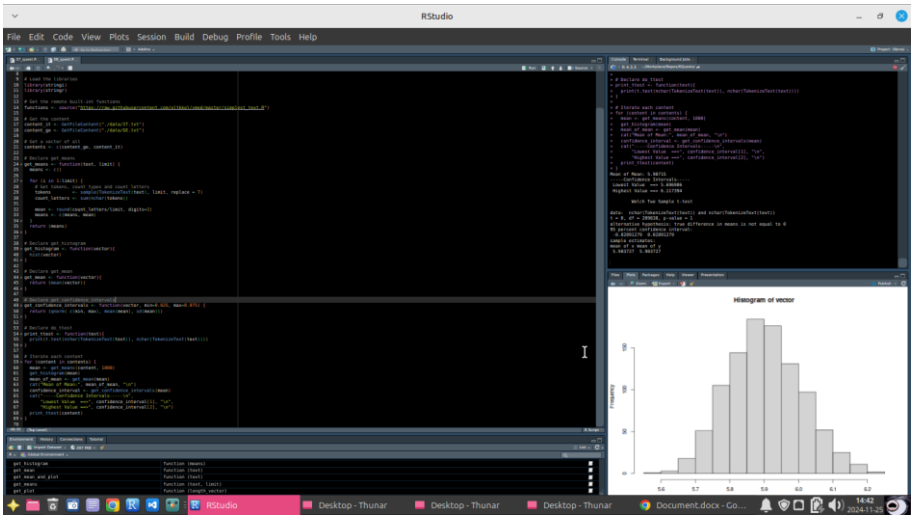



```
# Quest 7
# Make an estimation graph of the average word length for any excerpt.
```

```
# Load the libraries
library(stringi)
library(stringr)

# Get the remote built-int functions
functions <- source("https://raw.githubusercontent.com/oltkkol/vmod/master/simplest_text.R")

# Get the content
content <- GetFileContent("./data/Christie.txt")

# Declare get_means
get_means <- function(text, limit) {
 means <- c()

 for (i in 1:limit) {
  # Get tokens, count_types and count_letters
  tokens      <- sample(TokenizeText(text), limit, replace = T)
  count_letters <- sum(nchar(tokens))

  mean <- round(count_letters/limit, digits=2)
  means <- c(means, mean)
 }
 return (means)
}

# Declare get_histogram
get_histogram <- function(means){
 hist(
  means,
  main="Histogram of Average Word Lengths",
  col="lightblue",
  border="black",
  xlab="Average Word Length"
 )
}

# Call get_means
found_means <- get_means(content, 1000)

# Call get_histogram
get_histogram(found_means)
```

Commented [GK8]: (only histogram is plotted)

08_quest.R

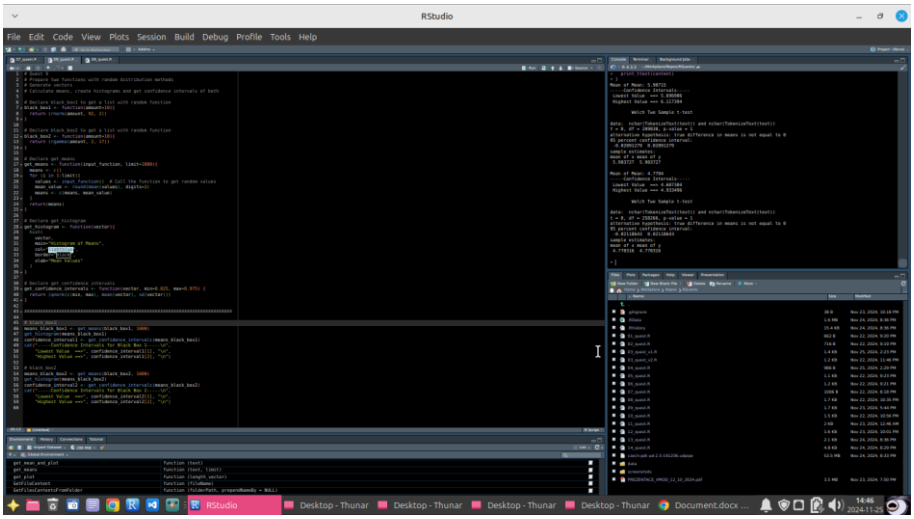



```
# Quest 8

# Read text, tokenize, get length of words

# Create histogram

# Get confidential interval

# Get mean of mean

# Test the results with t.test

# Compare Italian and German


# Load the libraries

library(stringi)

library(stringr)


# Get the remote built-int functions

functions <- source("https://raw.githubusercontent.com/oltkkol/vmod/master/simplest_text.R")


# Get the content

content_it <- GetFileContent("./data/IT.txt")

content_ge <- GetFileContent("./data/GE.txt")


# Get a vector of all

contents <- c(content_ge, content_it)


# Declare get_means
```

```
get_means <- function(text, limit) {
 means <- c()

 for (i in 1:limit) {
  # Get tokens, count_types and count_letters
  tokens       <- sample(TokenizeText(text), limit, replace = T)
  count_letters <- sum(nchar(tokens))

  mean <- round(count_letters/limit, digits=2)
  means <- c(means, mean)
 }
 return (means)
}

# Declare get_histogram
get_histogram <- function(vector){
 hist(vector)
}

# Declare get_mean
get_mean <- function(vector){
 return (mean(vector))
}

# Declare get_confidence_intervals
get_confidence_intervals <- function(vector, min=0.025, max=0.975) {
 return (qnorm( c(min, max), mean(mean), sd(mean)))
}

# Declare do_ttest
print_ttest <- function(text){
 print(t.test(nchar(TokenizeText(text)), nchar(TokenizeText(text))))
}

# Iterate each content
for (content in contents) {
 mean <- get_means(content, 1000)
 get_histogram(mean)
 mean_of_mean <- get_mean(mean)
 cat("Mean of Mean:", mean_of_mean, "\n")
 confidence_interval <- get_confidence_intervals(mean)
 cat("-----Confidence Intervals-----\n",
```

```
  "Lowest Value ==>", confidence_interval[1], "\n",
  "Highest Value ==>", confidence_interval[2], "\n")
print_ttest(content)
}
```

09_quest.R




```
# Quest 9
# Prepare two functions with random distribution methods
# Generate vectors
# Calculate means, create histograms and get confidence intervals of both

# Declare black_box1 to get a list with random function
black_box1 <- function(amount=10){
 return (rnorm(amount, 92, 2))
}

# Declare black_box2 to get a list with random function
black_box2 <- function(amount=10){
 return (rgamma(amount, 2, 17))
}

# Declare get_means
get_means <- function(input_function, limit=1000){
 means <- c()
 for (i in 1:limit){
  values <- input_function()  # Call the function to get random values
```

Commented [GK9]: Mean of Mean: 5.90715
-----Confidence Intervals-----
Lowest Value ==> 5.696906
Highest Value ==> 6.117394

Welch Two Sample t-test

data: nchar(TokenizeText(text)) and nchar(TokenizeText(text))
t = 0, df = 209838, p-value = 1
alternative hypothesis: true difference in means is not equal to 0
95 percent confidence interval:
-0.02891279 0.02891279
sample estimates:
mean of x mean of y
5.903727 5.903727

Mean of Mean: 4.7704
-----Confidence Intervals-----
Lowest Value ==> 4.607304
Highest Value ==> 4.933496

Welch Two Sample t-test

data: nchar(TokenizeText(text)) and nchar(TokenizeText(text))
t = 0, df = 258266, p-value = 1
alternative hypothesis: true difference in means is not equal to 0
95 percent confidence interval:
-0.02118643 0.02118643
sample estimates:
mean of x mean of y
4.770316 4.770316

>

```
  mean_value <- round(mean(values), digits=2)
  means <- c(means, mean_value)
 }
 return(means)
}

# Declare get_histogram
get_histogram <- function(vector){
 hist(
  vector,
  main="Histogram of Means",
  col="lightblue",
  border="black",
  xlab="Mean Values"
 )
}

# Declare get_confidence_intervals
get_confidence_intervals <- function(vector, min=0.025, max=0.975){
 return (qnorm(c(min, max), mean(vector), sd(vector)))
}

################################################################################

# black_box1
means_black_box1 <- get_means(black_box1, 1000)
get_histogram(means_black_box1)
confidence_interval1 <- get_confidence_intervals(means_black_box1)
cat("-----Confidence Intervals for Black Box 1-----\n",
  "Lowest Value  ==>", confidence_interval1[1], "\n",
  "Highest Value ==>", confidence_interval1[2], "\n")

# black_box2
means_black_box2 <- get_means(black_box2, 1000)
get_histogram(means_black_box2)
confidence_interval2 <- get_confidence_intervals(means_black_box2)
cat("-----Confidence Intervals for Black Box 2-----\n",
  "Lowest Value  ==>", confidence_interval2[1], "\n",
  "Highest Value ==>", confidence_interval2[2], "\n")
```

Commented [GK10]: Mean of Mean: 5.90715
-----Confidence Intervals-----
Lowest Value ==> 5.696906
Highest Value ==> 6.117394

Welch Two Sample t-test

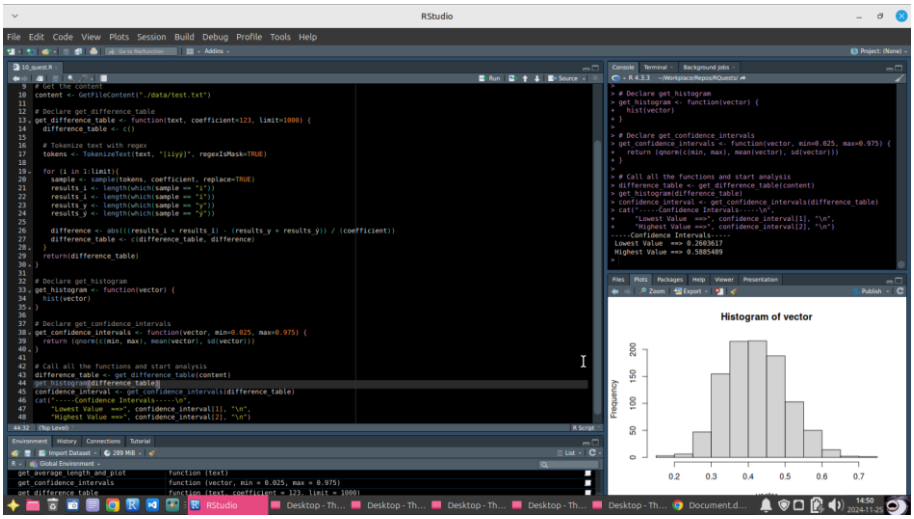
data: nchar(TokenizeText(text)) and nchar(TokenizeText(text))
t = 0, df = 209838, p-value = 1
alternative hypothesis: true difference in means is not equal to 0
95 percent confidence interval:
-0.02891279 0.02891279
sample estimates:
mean of x mean of y
5.903727 5.903727

Mean of Mean: 4.7704
-----Confidence Intervals-----
Lowest Value ==> 4.607304
Highest Value ==> 4.933496

Welch Two Sample t-test

data: nchar(TokenizeText(text)) and nchar(TokenizeText(text))
t = 0, df = 258266, p-value = 1
alternative hypothesis: true difference in means is not equal to 0
95 percent confidence interval:
-0.02118643 0.02118643
sample estimates:
mean of x mean of y
4.770316 4.770316

10_quest.R

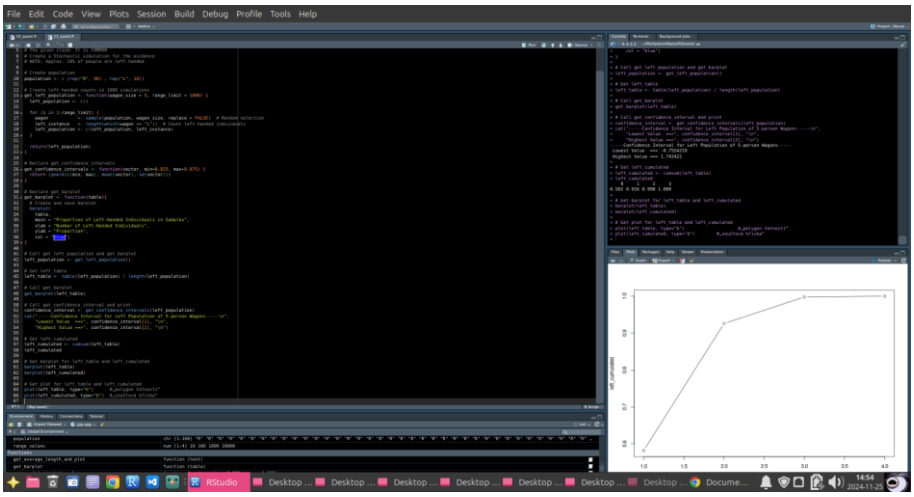



```
# Quest 10

# Find if "i" or "y" is more frequent in a Czech text

# Get histogram

# Get confidence interval



# Get the remote built-int functions

functions <- source("https://raw.githubusercontent.com/oltkkol/vmod/master/simplest_text.R")



# Get the content

content <- GetFileContent("./data/test.txt")



# Declare get_difference_table

get_difference_table <- function(text, coefficient=123, limit=1000) {

 difference_table <- c()



 # Tokenize text with regex

 tokens <- TokenizeText(text, "[iíyý]", regexIsMask=TRUE)



 for (i in 1:limit){

  sample <- sample(tokens, coefficient, replace=TRUE)

  results_i <- length(which(sample == "i"))

  results_í <- length(which(sample == "í"))
```

```
  results_y <- length(which(sample == "y"))
  results_ý <- length(which(sample == "ý"))


  difference <- abs(((results_i + results_í) - (results_y + results_ý)) / (coefficient))
  difference_table <- c(difference_table, difference)
 }
 return(difference_table)
}


# Declare get_histogram
get_histogram <- function(vector) {
 hist(vector)
}


# Declare get_confidence_intervals
get_confidence_intervals <- function(vector, min=0.025, max=0.975) {
 return (qnorm(c(min, max), mean(vector), sd(vector)))
}


# Call all the functions and start analysis
difference_table <- get_difference_table(content)
get_histogram(difference_table)
confidence_interval <- get_confidence_intervals(difference_table)
cat("-----Confidence Intervals-----\n",
  "Lowest Value  ==>", confidence_interval[1], "\n",
  "Highest Value ==>", confidence_interval[2], "\n")
```

Commented [GK11]: -----Confidence Intervals-----
Lowest Value ==> 0.2603617
Highest Value ==> 0.5885489

11_quest.R



```
# Quest 11 (Humanities Quest)

# There are 5 people settled in a wagon

# 3 on the table side, 2 on the window side

# Those 2 on the window side are left-handed

# The given claim: It is COMMON

# Create a Stochastic simulation for the evidence

# NOTE: Approx. 10% of people are left-handed



# Create population

population <- c (rep("R", 90) , rep("L", 10))



# Create left-handed counts in 1000 simulations

get_left_population <- function(wagon_size = 5, range_limit = 1000) {

 left_population <- c()



 for (i in 1:range_limit) {

  wagon         <- sample(population, wagon_size, replace = FALSE)  # Random selection

  left_instance  <- length(which(wagon == "L"))  # Count left-handed individuals

  left_population <- c(left_population, left_instance)

 }



 return(left_population)

}
```

```
# Declare get_confidence_intervals
get_confidence_intervals <- function(vector, min=0.025, max=0.975){
 return (qnorm(c(min, max), mean(vector), sd(vector)))
}

# Declare get_barplot
get_barplot <- function(table){
 # Create and save barplot
 barplot(
  table,
  main = "Proportion of Left-Handed Individuals in Samples",
  xlab = "Number of Left-Handed Individuals",
  ylab = "Proportion",
  col = "blue")
}

# Call get_left_population and get_barplot
left_population <- get_left_population()

# Get left_table
left_table <- table(left_population) / length(left_population)

# Call get_barplot
get_barplot(left_table)

# Call get_confidence_interval and print
confidence_interval <- get_confidence_intervals(left_population)
cat("-----Confidence Interval for Left Population of 5-person Wagons-----\n",
  "Lowest Value  ==>", confidence_interval[1], "\n",
  "Highest Value ==>", confidence_interval[2], "\n")

# Get left_cumulated
left_cumulated <- cumsum(left_table)
left_cumulated

# Get barplot for left_table and left_cumulated
barplot(left_table)
```

barplot(left_cumulated)

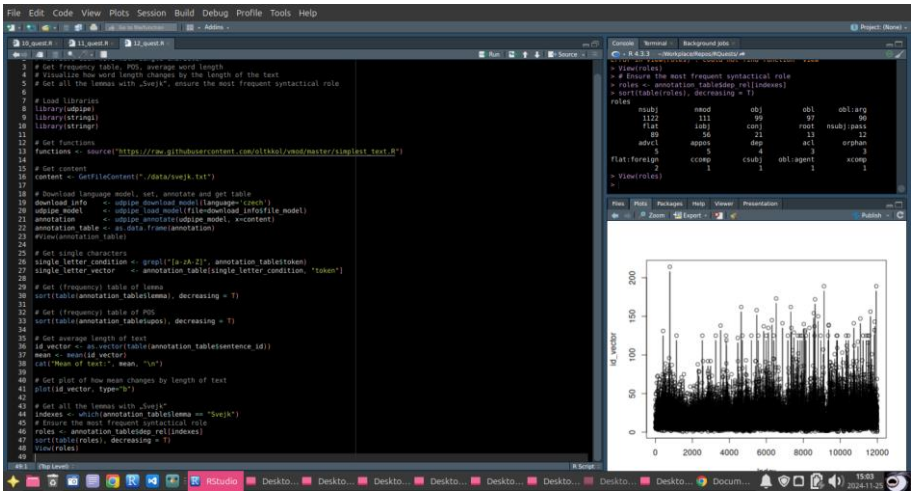
Get plot for left_table and left_cumulated

plot(left_table, type="b") #„polygon četností"

plot(left_cumulated, type="b") #„součtová křivka"

Commented [GK12]: -----Confidence Interval for Left Population of 5-person Wagons-----
Lowest Value ==> -0.7554219
Highest Value ==> 1.743422
> ....
> left_cumulated
0 1 2 3
0.582 0.926 0.998 1.000

12_quest.R



Quest 12 - Udpipe

Retrieve each word with single character

Get frequency table, POS, average word length

Visualize how word length changes by the length of the text

Get all the lemmas with „Švejk", ensure the most frequent syntactical role

Load libraries

library(udpipe)

library(stringi)

library(stringr)

Get functions

functions <- source("https://raw.githubusercontent.com/oltkkol/vmod/master/simplest_text.R")

Get content

content <- GetFileContent("./data/svejk.txt")

```
# Download language model, set, annotate and get table

download_info   <- udpipe_download_model(language='czech')

udpipe_model    <- udpipe_load_model(file=download_info$file_model)

annotation      <- udpipe_annotate(udpipe_model, x=content)

annotation_table <- as.data.frame(annotation)

#View(annotation_table)



# Get single characters

single_letter_condition <- grepl("[a-zA-Z]", annotation_table$token)

single_letter_vector   <- annotation_table[single_letter_condition, "token"]



# Get (frequency) table of lemma

sort(table(annotation_table$lemma), decreasing = T)



# Get (frequency) table of POS

sort(table(annotation_table$upos), decreasing = T)



# Get average length of text

id_vector <- as.vector(table(annotation_table$sentence_id))

mean <- mean(id_vector)

cat("Mean of text:", mean, "\n")



# Get plot of how mean changes by length of text

plot(id_vector, type="b")



# Get all the lemmas with „Švejk“

indexes <- which(annotation_table$lemma == "Švejk")

# Ensure the most frequent syntactical role

roles <- annotation_table$dep_rel[indexes]

sort(table(roles), decreasing = T)

View(roles)
```

Commented [GK13]: > # Get (frequency) table of lemma
> sort(table(annotation_table$lemma), decreasing = T)

```
     ,      .    být      "
 23601  11825   9283   8580
    se      a      -    ten
  7532   7051   6887   6077
    na     on     že      v
  3935   3681   3474   3210
    já      s     do    mít
  2388   2086   1760   1736
 Švejk      z      :   když
  1736   1674   1672   1631
```

Get (frequency) table of POS
> sort(table(annotation_table$upos), decreasing = T)

```
PUNCT  NOUN  VERB   ADP  PRON   ADV   ADJ   DET
CCONJ SCONJ PROPN
55016 44247 32556 20871 17278 16970 16216 13703
9404  8572  8426
 AUX   NUM  PART  INTJ   SYM
 8296  2434  1358   171    32
```

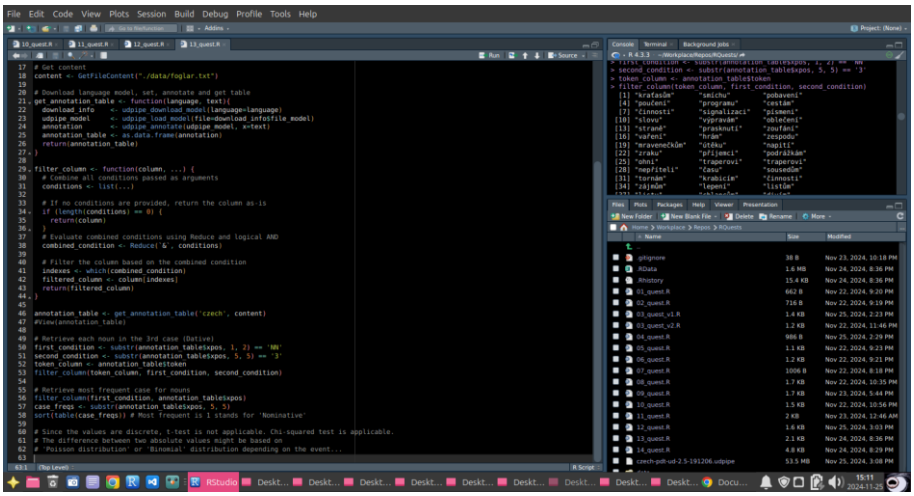
> # Get average length of text
> id_vector <-
as.vector(table(annotation_table$sentence_id))
> mean <- mean(id_vector)
> cat("Mean of text:", mean, "\n")
Mean of text: 21.37823

> roles <- annotation_table$dep_rel[indexes]
> sort(table(roles), decreasing = T)

```
roles
    nsubj      nmod       obj       obl    obl:arg
     1122       111        99        97        90
     flat      iobj      conj      root  nsubj:pass
       89        56        21        13        12
    advcl     appos       dep       acl     orphan
        5         5         4         3         3
flat:foreign     ccomp     csubj   obl:agent     xcomp
        2         1         1         1         1
```

> View(roles)

13_quest.R



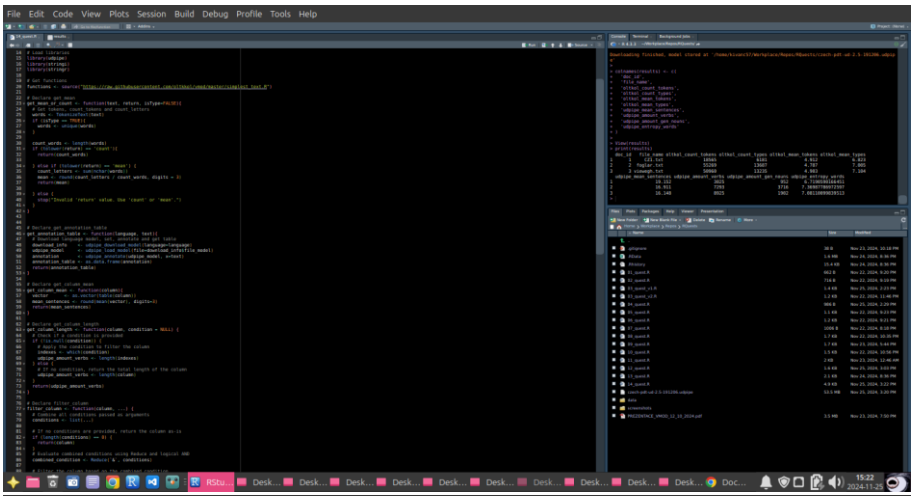
```
# Quest 13 - Udpipe

# Retrieve each noun in 3. case (Dative)

# Which nouns are the most common?

# How can you ensure statistically?

# Is it possible to apply t-test?

# (Which distribution leads to two absolute calculation?)

# Use substr(), 'xpos' column


# Load libraries

library(udpipe)

library(stringi)

library(stringr)


# Get functions

functions <- source("https://raw.githubusercontent.com/oltkkol/vmod/master/simplest_text.R")


# Get content

content <- GetFileContent("../data/foglar.txt")


# Download language model, set, annotate and get table

get_annotation_table <- function(language, text){

 download_info   <- udpipe_download_model(language=language)

 udpipe_model    <- udpipe_load_model(file=download_info$file_model)

 annotation      <- udpipe_annotate(udpipe_model, x=text)

 annotation_table <- as.data.frame(annotation)

 return(annotation_table)
```

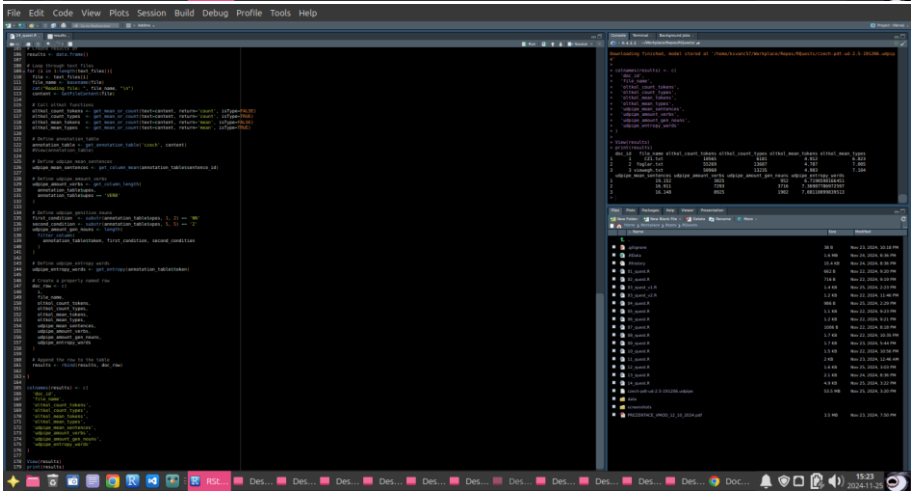
```
}

filter_column <- function(column, ...) {
 # Combine all conditions passed as arguments
 conditions <- list(...)

 # If no conditions are provided, return the column as-is
 if (length(conditions) == 0){
   return(column)
 }
 # Evaluate combined conditions using Reduce and logical AND
 combined_condition <- Reduce(`&`, conditions)

 # Filter the column based on the combined condition
 indexes <- which(combined_condition)
 filtered_column <- column[indexes]
 return(filtered_column)
}

annotation_table <- get_annotation_table('czech', content)
#View(annotation_table)

# Retrieve each noun in the 3rd case (Dative)
first_condition <- substr(annotation_table$xpos, 1, 2) == 'NN'
second_condition <- substr(annotation_table$xpos, 5, 5) == '3'
token_column <- annotation_table$token
filter_column(token_column, first_condition, second_condition)

# Retrieve most frequent case for nouns
filter_column(first_condition, annotation_table$xpos)
case_freqs <- substr(annotation_table$xpos, 5, 5)
sort(table(case_freqs)) # Most frequent is 1 stands for 'Nominative'

# Since the values are discrete, t-test is not applicable. Chi-squared test is applicable.
# The difference between two absolute values might be based on
# 'Poisson distribution' or 'Binomial' distribution depending on the event...
```

Commented [GK14]: > token_column <- annotation_table$token
> filter_column(token_column, first_condition, second_condition)
[1] "kraťasům" "smíchu" "pobavení"
[4] "poučení" "programu" "cestám"
[7] "činnosti" "signalizaci" "písmeni"
[10] "slovu" "výpravám" "oblečení"
[13] "straně" "prasknutí" "zoufání"
[16] "vaření" "hrám" "zespodu"
[19] "mravenečkům" "útěku" "napití"
[22] "zraku" "příjemci" "podrážkám"
[25] "ohni" "traperovi" "traperovi"

> sort(table(case_freqs)) # Most frequent is 1 stands for 'Nominative'
case_freqs
5 X 3 7 6 2 1 4 -
6 490 1727 2739 4762 7

14_quest.R





Quest 14 - Udpipe --FINAL--

#Create a script that...

#For a given directory, loads all text files and obtains:

#1-The number of tokens

#2-The number of types

#3-The average word length

#4-The average sentence length (using udpipe!)

#5-The total number of verbs (using udpipe!)

#6-The total number of nouns in the genitive case (using udpipe!)

#7-Calculates the entropy of words

```
#8-Writes all the finding to a custom table

# Load libraries
library(udpipe)
library(stringi)
library(stringr)

# Get functions
functions <- source("https://raw.githubusercontent.com/oltkkol/vmod/master/simplest_text.R")

# Declare get_mean
get_mean_or_count <- function(text, return, isType=FALSE){
 # Get tokens, count_tokens and count_letters
 words <- TokenizeText(text)
 if (isType == TRUE){
  words <- unique(words)
 }

 count_words <- length(words)
 if (tolower(return) == 'count'){
  return(count_words)

 } else if (tolower(return) == 'mean') {
  count_letters <- sum(nchar(words))
  mean <- round(count_letters / count_words, digits = 3)
  return(mean)

 } else {
  stop("Invalid 'return' value. Use 'count' or 'mean'.")
 }
}


# Declare_get_annotation_table
get_annotation_table <- function(language, text){
 # Download language model, set, annotate and get table
 download_info   <- udpipe_download_model(language=language)
 udpipe_model    <- udpipe_load_model(file=download_info$file_model)
 annotation      <- udpipe_annotate(udpipe_model, x=text)
 annotation_table <- as.data.frame(annotation)
 return(annotation_table)
}
```

```
# Declare get_column_mean
get_column_mean <- function(column){
 vector     <- as.vector(table(column))
 mean_sentences <- round(mean(vector), digits=3)
 return(mean_sentences)
}

# Declare get_column_length
get_column_length <- function(column, condition = NULL) {
 # Check if a condition is provided
 if (!is.null(condition)) {
  # Apply the condition to filter the column
  indexes <- which(condition)
  udpipe_amount_verbs <- length(indexes)
 } else {
  # If no condition, return the total length of the column
  udpipe_amount_verbs <- length(column)
 }
 return(udpipe_amount_verbs)
}

# Declare filter_column
filter_column <- function(column, ...) {
 # Combine all conditions passed as arguments
 conditions <- list(...)

 # If no conditions are provided, return the column as-is
 if (length(conditions) == 0) {
  return(column)
 }
 # Evaluate combined conditions using Reduce and logical AND
 combined_condition <- Reduce(`&`, conditions)

 # Filter the column based on the combined condition
 indexes <- which(combined_condition)
 filtered_column <- column[indexes]
 return(filtered_column)
}

# Declare get_entropy
get_entropy<- function(column){
```

```
 p <- table(column) / length(column)
 entropy <- -sum(p * log(p))
 return (entropy)
}

# main program
directory <- "./data/multiple_texts"
text_files <- list.files(directory, pattern = "\\.txt$", full.names = TRUE)

# Create results df
results <- data.frame()

# Loop through text files
for (i in 1:length(text_files)){
 file <- text_files[i]
 file_name <- basename(file)
 cat("Reading file: ", file_name, "\n")
 content <- GetFileContent(file)

 # Call oltkol functions
 oltkol_count_tokens <- get_mean_or_count(text=content, return='count', isType=FALSE)
 oltkol_count_types  <- get_mean_or_count(text=content, return='count', isType=TRUE)
 oltkol_mean_tokens  <- get_mean_or_count(text=content, return='mean', isType=FALSE)
 oltkol_mean_types  <- get_mean_or_count(text=content, return='mean', isType=TRUE)

 # Define annotation_table
 annotation_table <- get_annotation_table('czech', content)
 #View(annotation_table)

 # Define udpipe_mean_sentences
 udpipe_mean_sentences <- get_column_mean(annotation_table$sentence_id)

 # Define udpipe_amount_verbs
 udpipe_amount_verbs <- get_column_length(
  annotation_table$upos,
  annotation_table$upos == 'VERB'
 )

 # Define udpipe_genitive_nouns
 first_condition  <- substr(annotation_table$xpos, 1, 2) == 'NN'
 second_condition <- substr(annotation_table$xpos, 5, 5) == '2'
 udpipe_amount_gen_nouns <- length(
```

```
  filter_column(
    annotation_table$token, first_condition, second_condition
  )
 )

 # Define udpipe_entropy words
 udpipe_entropy_words <- get_entropy(annotation_table$token)

 # Create a properly named row
 doc_row <- c(
  i,
  file_name,
  oltkol_count_tokens,
  oltkol_count_types,
  oltkol_mean_tokens,
  oltkol_mean_types,
  udpipe_mean_sentences,
  udpipe_amount_verbs,
  udpipe_amount_gen_nouns,
  udpipe_entropy_words
 )

 # Append the row to the table
 results <- rbind(results, doc_row)

}

colnames(results) <- c(
 'doc_id',
 'file_name',
 'oltkol_count_tokens',
 'oltkol_count_types',
 'oltkol_mean_tokens',
 'oltkol_mean_types',
 'udpipe_mean_sentences',
 'udpipe_amount_verbs',
 'udpipe_amount_gen_nouns',
 'udpipe_entropy_words'
)

View(results)
print(results)
```

Commented [GK15]:

```
> View(results)
> print(results)
  doc_id  file_name oltkol_count_tokens
oltkol_count_types oltkol_mean_tokens
oltkol_mean_types
1     1    CZ1.txt           18565           6181          4.912
6.823
2     2  foglar.txt           55269          13687          4.787
7.005
3     3 viewegh.txt           50960          13235          4.903
7.104
  udpipe_mean_sentences udpipe_amount_verbs
udpipe_amount_gen_nouns udpipe_entropy_words
1                19.152                3025                 952
6.7190590166451
2                16.911                7293                3716
7.36987786972597
3                16.148                8925                1902
7.08110099839513
>
```