**Introduction to Corpus Linguistics**

by Dr. Natalia Levshina

**Sessions 1 and 2. First steps with R. Extracting types and tokens from the Universal Dependencies Corpora.**

**I. First steps with R**

What is R?

* statistical computing environment (from *t*-test to generalized linear models, and more…)

- core distribution “base”

- add-on packages

* programming language
* tools for creation of publication-quality plots

Where to get R?

* Distribution and packages: CRAN (Comprehensive R Archive Network) <http://cran.r-project.org/>
* Information: <http://www.r-project.org/>

RStudio:

* Highly recommended (easy to manage projects, packages, data, graphs, etc.)!
* Available from <http://www.rstudio.com/products/RStudio/>

Basic operations in R:

**2 + 2**

[1] 4

**a <- 3**

**a**

[1] 3

**a + 5**

[1] 8

Numeric and character vectors in R:

* numeric vectors

**vnum <- 1:5** # a vector of integers from 1 to 5

**vnum**

[1] 1 2 3 4 5

**is(vnum)**

[1] "integer" "numeric" "vector"

[…]

**Frequencies <- c(1080, 201, 33, 5)** #some word frequencies

**Frequencies**

**[1] 1080 201 33 5**

**length(Frequencies)**

[1] 4

**Frequencies[4]**

[1] 5

* character vectors

**words <- c("you", "computer", "today")**

**words**

[1] "you" "computer" "today"

**is(words)**

[1] "character" "vector" "data.frameRowLabels" "SuperClassMethod" "index"

[6] "atomicVector"

**length(words)**

[1] 3

**words[1]**

[1] "you"

**Exercise**

Ask your colleagues and create a character vector with their names. Next, create a numeric vector with the number of languages they speak.

**II. The Universal Dependencies Corpora: an introduction**

In order to view the files, you will need some text editor. I recommend installing Notepad++. <https://notepad-plus-plus.org/>.

What are the UD corpora?

* More than 100 treebanks in over 70 languages
* <http://universaldependencies.org> > Home
* To download the files, use the link on the page (> 200M).

The structure of a UD file:

# newdoc id = weblog-blogspot.com\_nominations\_20041117172713\_ENG\_20041117\_172713

# sent\_id = weblog-blogspot.com\_nominations\_20041117172713\_ENG\_20041117\_172713-0001

# text = From the AP comes this story :

1 From from ADP IN \_ 3 case 3:case \_

2 the the DET DT Definite=Def|PronType=Art 3 det 3:det \_

3 AP AP PROPN NNP Number=Sing 4 obl 4:obl:from \_

4 comes come VERB VBZ Mood=Ind|Number=Sing|Person=3|Tense=Pres|VerbForm=Fin 0 root 0:root \_

5 this this DET DT Number=Sing|PronType=Dem 6 det 6:det \_

6 story story NOUN NN Number=Sing 4 nsubj 4:nsubj \_

7 : : PUNCT : \_ 4 punct 4:punct \_

Position 1: ID of the word

Position 2: Wordform (mind the capital letters!)

Position 3: Lemma (dictionary form)

Position 4: POS tag

* [ADJ](http://universaldependencies.org/u/pos/ADJ.html): adjective
* [ADP](http://universaldependencies.org/u/pos/ADP.html): adposition
* [ADV](http://universaldependencies.org/u/pos/ADV.html): adverb
* [AUX](http://universaldependencies.org/u/pos/AUX_.html): auxiliary
* [CCONJ](http://universaldependencies.org/u/pos/CCONJ.html): coordinating conjunction
* [DET](http://universaldependencies.org/u/pos/DET.html): determiner
* [INTJ](http://universaldependencies.org/u/pos/INTJ.html): interjection
* [NOUN](http://universaldependencies.org/u/pos/NOUN.html): noun
* [NUM](http://universaldependencies.org/u/pos/NUM.html): numeral
* [PART](http://universaldependencies.org/u/pos/PART.html): particle
* [PRON](http://universaldependencies.org/u/pos/PRON.html): pronoun
* [PROPN](http://universaldependencies.org/u/pos/PROPN.html): proper noun
* [PUNCT](http://universaldependencies.org/u/pos/PUNCT.html): punctuation
* [SCONJ](http://universaldependencies.org/u/pos/SCONJ.html): subordinating conjunction
* [SYM](http://universaldependencies.org/u/pos/SYM.html): symbol
* [VERB](http://universaldependencies.org/u/pos/VERB.html): verb
* [X](http://universaldependencies.org/u/pos/X.html): other

Position 5: Alternative tags (language-specific)

Position 6: Morphological features (number, mood, tense, etc.)

<http://universaldependencies.org/u/feat/index.html>

Position 7: ID of the Head

Position 8: Syntactic dependency with regard to the Head

See the full list here:

<http://universaldependencies.org/u/dep/all.html>

Positions 9-10: language specific, different combinations of dependencies

**Exercise**

Using the UD annotation style, annotate the following sentence:

*The girl is reading an interesting book*.

**III. Exploring the UD corpora with R**

1. How to open a UD file in R

We’ll use the file en\_ewt-ud-dev.conllu. Download it from github.com/levshina/Jena\_CorpLing > Data\_UD. When you click on the file in the list, there’s a Download button. Save or copy it somewhere in a local directory (e.g. My Documents).

R code for reading the file interactively. R will open a dialog window where you can choose the corpus file:

**ud <- scan(file = file.choose(), what = "character", sep = "\n", comment.char = "#", encoding = "UTF-8")**

**head(ud)**

[1] "1\tFrom\tfrom\tADP\tIN\t\_\t3\tcase\t3:case\t\_"

[2] "2\tthe\tthe\tDET\tDT\tDefinite=Def|PronType=Art\t3\tdet\t3:det\t\_"

...

**length(ud)**

[1] 25150

2. How to get a vector of all wordforms

First, we need to split the lines into the UD slots by tab characters. The result is a list, a special object type in R:

**ud\_list <- strsplit(ud, "\t")**

**head(ud\_list)**

[[1]]

[1] "1" "From" "from" "ADP" "IN" "\_" "3" "case" "3:case" "\_"

…

Every UD line is now an element of a list. See the double squared brackets [[1]]. Inside each list element, there is a character vector. The elements of the vector are the UD slots: ID, wordform, lemma, etc.

In order to get all corpus tokens, we can select the second element from each list element (UD line):

**tokens <- sapply(ud\_list, function (x) x[2])**

**head(tokens)**

[1] "From" "the" "AP" "comes" "this" "story"

**length(tokens)**

[1] 25150

This is our token frequency (for the time being, we ignore the punctuation marks).

If you need to extract lemmata, POS or syntactic dependencies, you should replace 2 with the corresponding index. See how it is done for lemmata below.

**lemmata <- sapply(ud\_list, function (x) x[3])**

**head(lemmata)**

[1] "from" "the" "AP" "come" "this" "story"

Question: How can you see that these are the lemmata?

3. How to get a vector with unique wordforms

We ignore the punctuation marks and treat the upper-case letters separately for the moment.

**types <- unique(tokens)**

**length(types)**

[1] 5495

4. How to compute the type-token ratio

**Type-token ratio (TTR)** is the ratio of types (the number of different words in a corpus) to tokens (the total number of words). Ratios are computed by **division**:

Very simple:

**TTR <- length(types)/length(tokens)**

**TTR**

[1] 0.2184891

In other words, the relation between types and tokens is roughly 1:5. This means, every individual wordform (type) occurs about five times.

**Exercise**

Download a UD file with another language from GitHub. Open it in R. Compute the token and type frequencies. Compute the type-token ratio and compare it with English.

5. How to visualize the results with the help of bar plots

Create a numeric vector with TTR scores:

TTR\_sample <- c(0.25, 0.17, 0.41, 0.32, 0.22, 0.37, 0.48, 0.41, 0.37)

Create a character vector with language names:

languages <- c("af", "ca", "cs", "de", "en", "eu", "fi", "hu", "id")

Use the language names as names for the TTR vector elements:

names(TTR\_sample) <- languages

TTR\_sample

af ca cs de en eu fi hu id

0.25 0.17 0.41 0.32 0.22 0.37 0.48 0.41 0.37

Sort the TTR vector:

sort(TTR\_sample) #in ascending order

ca en af de eu id cs hu fi

0.17 0.22 0.25 0.32 0.37 0.37 0.41 0.41 0.48

sort(TTR\_sample, decreasing = TRUE) #in descending order

fi cs hu eu id de af en ca

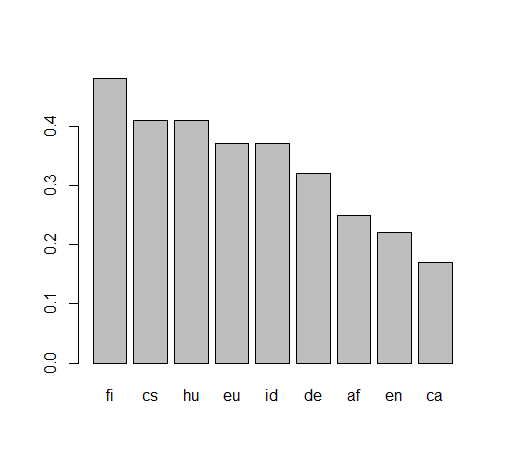
0.48 0.41 0.41 0.37 0.37 0.32 0.25 0.22 0.17

Make a new sorted version of the TTR vector:

TTR\_sort <- sort(TTR\_sample, decreasing = TRUE)

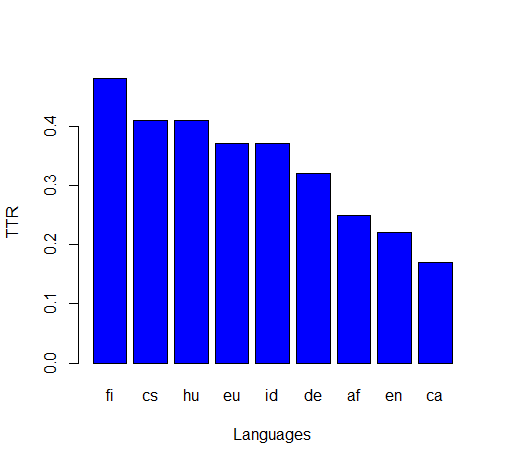
Make a bar plot:

barplot(TTR\_sort)



Make the barplot more explicit:

barplot(TTR\_sort, col = "blue", xlab = "Languages", ylab = "TTR")



Explore the colours:

colors()

[1] "white" "aliceblue" "antiquewhite"

[4] "antiquewhite1" "antiquewhite2" "antiquewhite3"

…

Use our own colour!

6. Some fine-tuning

a) How to eliminate punctuation marks

ud\_list[1:10]

[[1]]

[1] "1" "From" "from" "ADP" "IN" "\_" "3" "case" "3:case" "\_"

…

[[7]]

[1] "7" ":" ":" "PUNCT" ":" "\_" "4" "punct" "4:punct"

…

Make a vector with TRUE and FALSE values. If the fourth element in a vector is “PUNCT”, then the value TRUE. If not, the value is FALSE.

ud\_list\_punct <- sapply(ud\_list, function (x) x[4] == "PUNCT")

ud\_list\_punct[1:10]

[1] FALSE FALSE FALSE FALSE FALSE FALSE TRUE FALSE FALSE FALSE

Exclude these elements from the list of tokens:

tokens <- sapply(ud\_list, function (x) x[2])

tokens[1:10]

[1] "From" "the" "AP" "comes" "this" "story" ":"

[8] "President" "Bush" "on"

tokens\_nopunct <- tokens[!ud\_list\_punct]

tokens\_nopunct[1:10]

[1] "From" "the" "AP" "comes" "this" "story" "President"

[8] "Bush" "on" "Tuesday"

length(tokens)

[1] 25150

length(tokens\_nopunct)

[1] 22067

The last number is our new, corrected token frequency.

b) How to normalize the forms, using only low-case letters

tokens\_lower <- tolower(tokens\_nopunct)

tokens\_lower[1:10]

[1] "from" "the" "ap" "comes" "this" "story" "president"

[8] "bush" "on" "tuesday"

c) How to draw a random subsample of equal size (e.g. 1000 tokens)

First, how to draw a sample of 5 numbers from a sequence from 1 to 100:

sample(100, 5) #by default, without replacement!

[1] 93 7 19 67 99

You’ll get different results each time you take the sample:

sample(100, 5)

[1] 52 73 14 64 68

If you want to get reproducible results, you should always use the same random seed **immediately** before running the function sample():

set.seed(42)

sample(100, 5)

[1] 92 93 29 81 62

sample(100, 5)

[1] 52 73 14 64 68

set.seed(42)

sample(100, 5)

[1] 92 93 29 81 62

tokens\_sample <- tokens[sample(length(tokens\_lower), 5000)]

length(tokens\_sample)

[1] 5000

types\_sample <- unique(tokens\_sample)

length(types\_sample)

[1] 1876

1876/5000

[1] 0.3752

This is the new, normalized TTR for the English corpus with the fixed number of tokens (5,000).

**Exercise**

Redo the analyses with one of the languages, using the corrected procedure. After everyone has obtained the updated scores, aggregate them and make a new bar plot. Have the results changed?