# Stabilization Analyses for Characters: StringBase

Chris Bentz

25/02/2021

### Load libraries

If the libraries are not installed yet, you need to install them using, for example, the command: install.packages("ggplot2"). For the Hrate package this is different, since it comes from github. The devtools library needs to be installed, and then the install_github() function is used.

```
library(stringr)
library(ggplot2)
library(plyr)
library(entropy)
library(ggExtra)
library(gsubfn)
```

```
## Loading required package: proto
```

```
# library(devtools)
# install_github("dimalik/Hrate")
library(Hrate)
```

### List files

Create list with all the files in the directory "corpus".

```
# give file paths to be processed
file.list <- list.files(path = "~/Data/100LC_Dumps/output_test",
                        recursive = T, full.names = T)
# /home/chris/Github/StringBase/corpus/original
#print(file.list)
length(file.list)
```

```
## [1] 43
```

## Stabilization analysis per file

```
# set counter
counter = 0
# set the maximal number of units (n), and the stepsize for stabilization analysis
# (i.e. in steps of how many units are values calculated?)
n = 100
stepsize = 10
# initialize dataframe to append results to
```

```
stabilization.df <- data.frame(filename = character(0), subcorpus = character(0),
                               code = character(0), huni.chars = numeric (0),
                               hrate.chars = numeric(0), ttr = numeric(0),
                               units = numeric(0))
# start time
start_time <- Sys.time()
for (file in file.list)
{
  # basic processing
  # loading textfile
  textfile <- scan(file, what = "char", quote = "",
                   comment.char = "", encoding = "UTF-8", sep = "\n" , skip = 16, nmax = 20)
  # nmax is the maximum number of lines to be read
  # remove tabs and parentheses
  textfile <- gsubfn(".", list("\t" = "", "(" = "", ")" = "", "]" = "",
                               "[" = "", "}" = "",  "{" = ""), textfile)
  # first select the lines with the original text, i.e. marked by <line_x>,
  # this is only relevant for the 100LC corpus
  textfile <- na.omit(str_match(textfile, "<line_.*"))
  # remove annotations marked by '<>'
  textfile <- gsub("<.*>","",textfile)
  # print(head(textfile))
  # get filename
  filename <- basename(file)
  # print(filename) # for visual inspection
  # get subcorpus category
  subcorpus <- sub("_.*", "", filename)
  # print(subcorpus) # for visual inspection
  # get the three letter identification code + the running number
  code <- substring(filename, 1, nchar(filename)-4)
  # print(code) # for visual inspection
  # split the textfile into individual utf-8 characters. The output of strsplit()
  # is a list, so it needs to be "unlisted"" to get a vector. Note that white spaces    # are counted a
  chars <- unlist(strsplit(textfile, ""))
  chars <- chars[1:n] # use only maximally n units
  chars <- chars[!is.na(chars)] # remove NAs for vectors which are already shorter
  # than n
  chars <- chars[chars != " "] # remove white spaces from character vector
  # define the number of units (i.e. characters) used for analyses (note that k is      # always either
  k = length(chars)
  for (i in 1:(k/stepsize))
  {
    # unigram entropy estimation
    # calculate unigram entropy for characters
    chars.df <- as.data.frame(table(chars[1:(i*stepsize)]))
    # print(chars.df)
    huni.chars <- entropy(chars.df$Freq, method = "ML", unit = "log2")
    # entropy rate estimation
    # note: the values chosen for max.length and every.word will crucially
    # impact processing time. max.length = NULL means all units in the file are
    # considered.
    hrate.chars <- get.estimate(text = chars[1:(i*stepsize)], every.word = 1,
                                max.length = NULL)
```

```
    # calculate type-token ratio (ttr)
    ttr.chars <- nrow(chars.df)/sum(chars.df$Freq)
    # append results to dataframe
    local.df <- data.frame(filename, subcorpus, code, huni.chars, hrate.chars,
                           ttr.chars, units = i*stepsize)
    stabilization.df <- rbind(stabilization.df, local.df)
  }
  # counter
  counter <- counter + 1
  # print(counter)
}
end_time <- Sys.time()
end_time - start_time
```

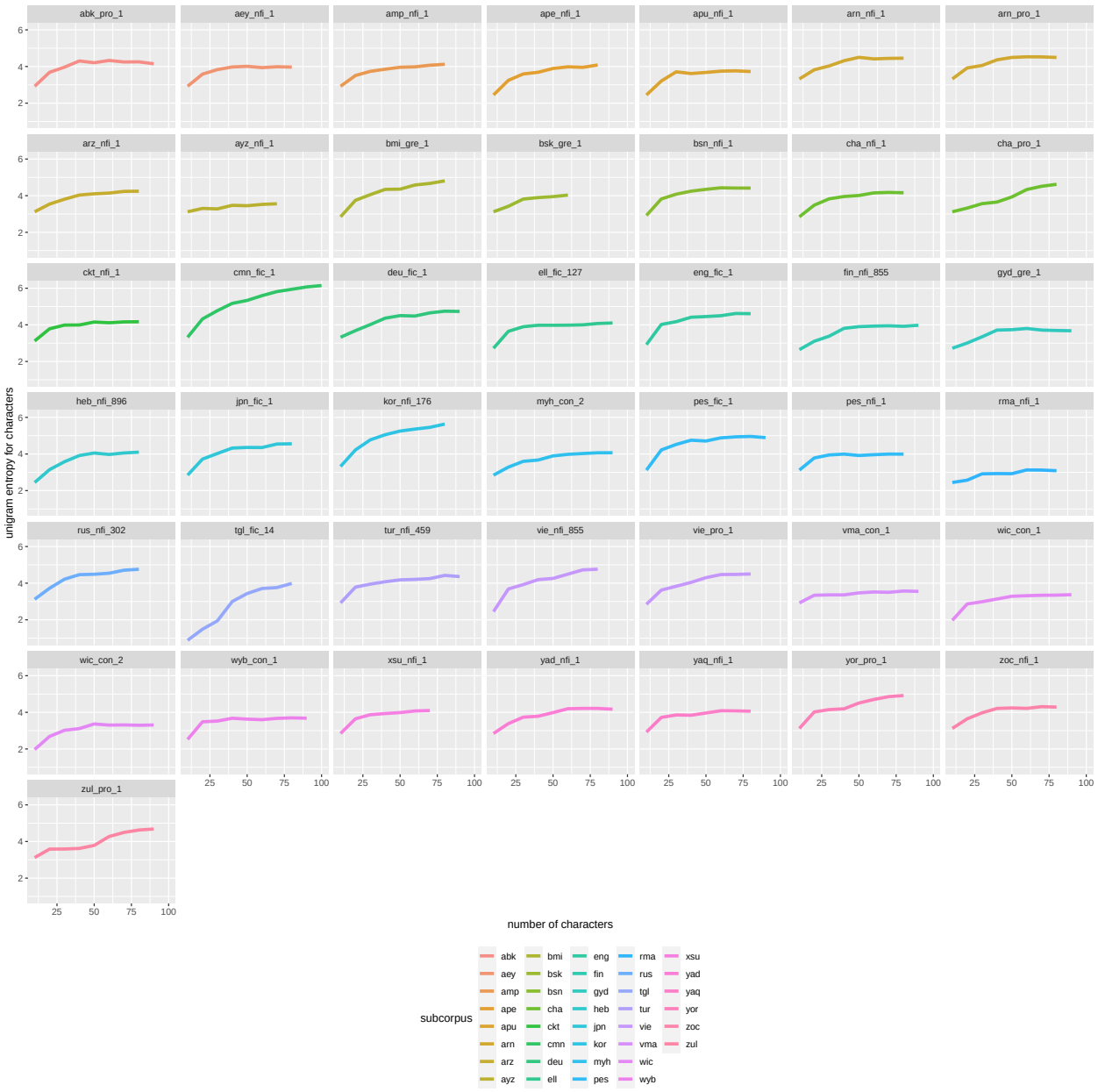
```
## Time difference of 4.106965 secs
```

```
# stabilization.df
```

# Stabilization plots

## Unigram entropy characters

```
huni.chars.plot <- ggplot(stabilization.df, aes(x = units, y = huni.chars,
                                                colour = subcorpus)) +
  geom_line(alpha = 0.8, size  = 1.5) +
  theme(legend.position = "bottom") +
  labs(x = "number of characters", y = "unigram entropy for characters") +
  facet_wrap(~code)
huni.chars.plot
```




## Safe figure to file

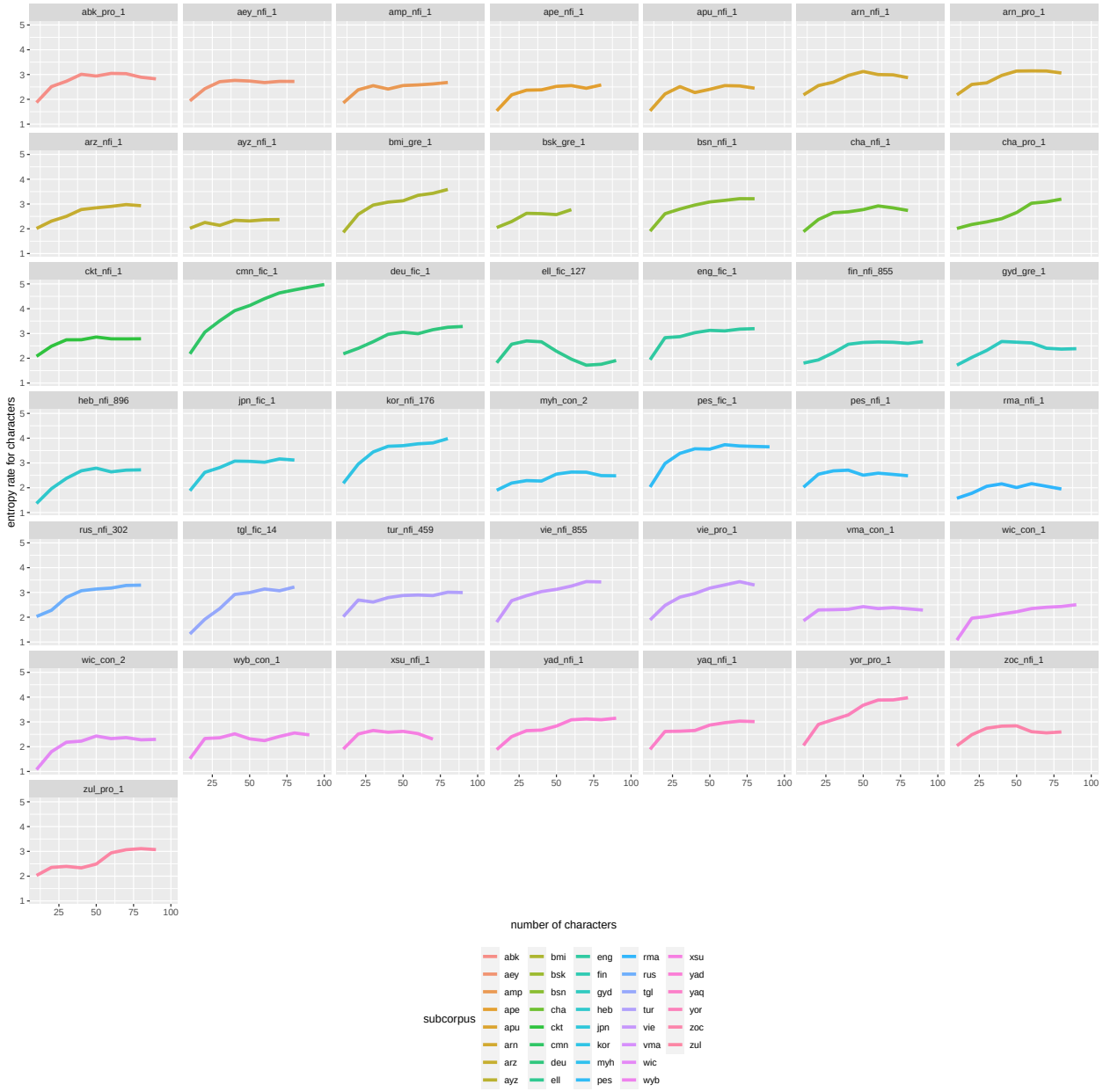
```
ggsave("Figures/stabilization_huni_chars_100LC.pdf", huni.chars.plot, dpi = 300,
       scale = 1, device = cairo_pdf)
```

```
## Saving 15 x 15 in image
```

## Entropy rate characters

```
hrate.chars.plot <- ggplot(stabilization.df, aes(x = units, y = hrate.chars,
                                                 colour = subcorpus)) +
```

```
  geom_line(alpha = 0.8, size  = 1.5) +
  theme(legend.position = "bottom") +
  labs(x = "number of characters", y = "entropy rate for characters") +
  facet_wrap(~code)
hrate.chars.plot
```



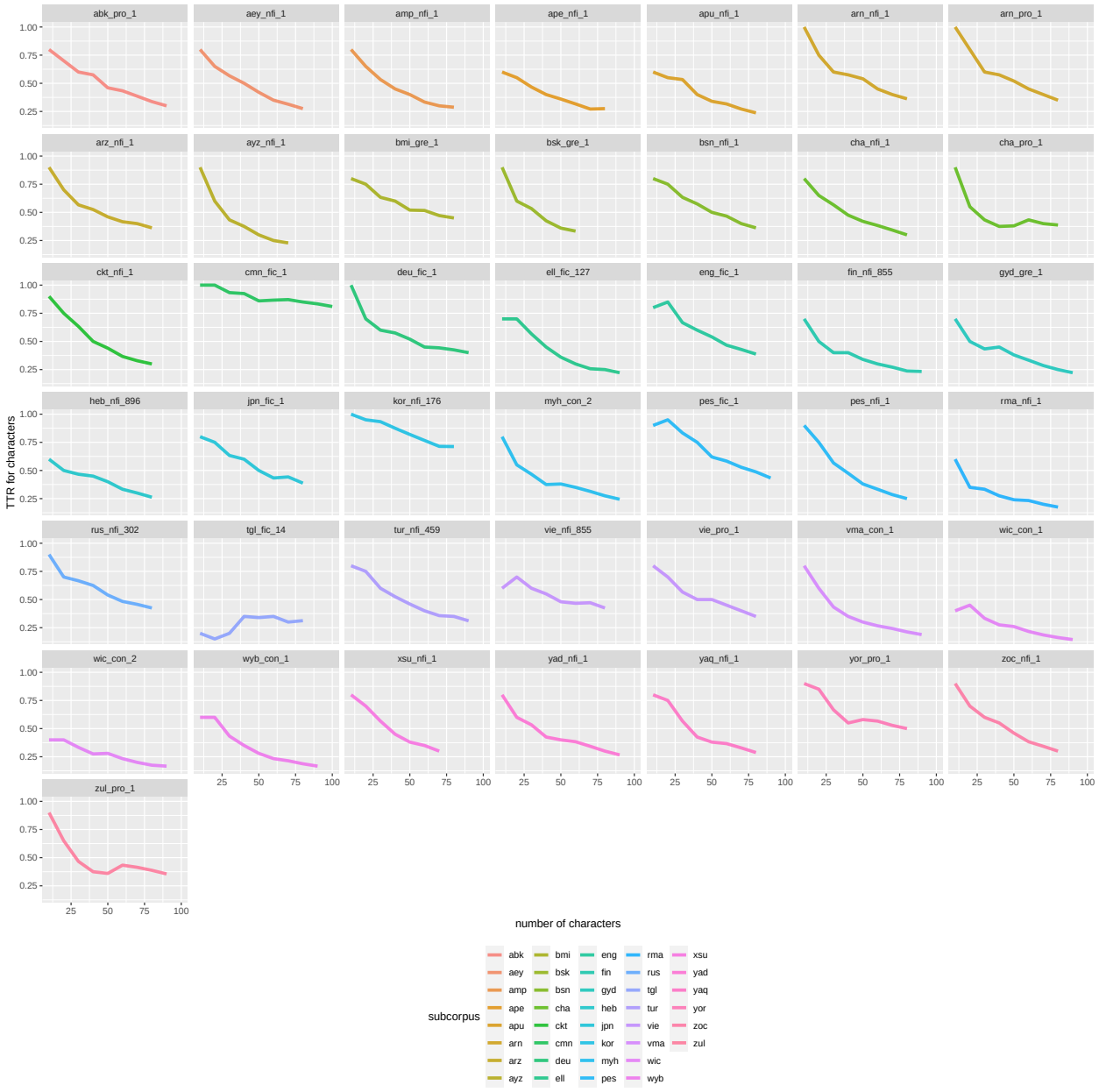

## Safe figure to file

```
ggsave("Figures/stabilization_hrate_chars_100LC.pdf", hrate.chars.plot, dpi = 300,
       scale = 1, device = cairo_pdf)
```

```
## Saving 15 x 15 in image
```

## TTR characters

```
ttr.chars.plot <- ggplot(stabilization.df, aes(x = units, y = ttr.chars,
                                               colour = subcorpus)) +
  geom_line(alpha = 0.8, size  = 1.5) +
  theme(legend.position = "bottom") +
  labs(x = "number of characters", y = "TTR for characters") +
  facet_wrap(~code)
ttr.chars.plot
```



### Safe figure to file

```
ggsave("Figures/stabilization_ttr_chars_100LC.pdf", ttr.chars.plot, dpi = 300,
       scale = 1, device = cairo_pdf)
```

```
## Saving 15 x 15 in image
```