SCR\_PAM\_R.r

timotheepremat

2021-02-05

#Timothée Premat | 22/01/2020  
#Script to print plots based on PAM analysis results  
  
#Load packages  
library(scales)  
library(tidyverse)

## ── Attaching packages ─────────────────────────────────────── tidyverse 1.3.0 ──

## ✓ ggplot2 3.3.3 ✓ purrr 0.3.4  
## ✓ tibble 3.0.6 ✓ dplyr 1.0.3  
## ✓ tidyr 1.1.2 ✓ stringr 1.4.0  
## ✓ readr 1.4.0 ✓ forcats 0.5.1

## ── Conflicts ────────────────────────────────────────── tidyverse\_conflicts() ──  
## x readr::col\_factor() masks scales::col\_factor()  
## x purrr::discard() masks scales::discard()  
## x dplyr::filter() masks stats::filter()  
## x dplyr::lag() masks stats::lag()

library(readxl)  
library(ggpubr)  
library(ggrepel)  
library(Hmisc)

## Loading required package: lattice

## Loading required package: survival

## Loading required package: Formula

##   
## Attaching package: 'Hmisc'

## The following objects are masked from 'package:dplyr':  
##   
## src, summarize

## The following objects are masked from 'package:base':  
##   
## format.pval, units

#Set directory  
setwd("~/Documents/GitHub/PhD/Metrical Analysis/BFM/roland")  
  
#Import datasets  
##Import global data  
df\_loc\_line <- read.table(  
 "line\_by\_line\_meter.txt",  
 header=T)  
df\_epC <- df\_loc\_line %>%  
 mutate(meter=replace(meter, ces\_4=="4épC" & meter==11, 10)) %>%  
 as.data.frame()  
 #This replace meter=11 by meter=10 when ces\_4=4épC, in order to display lines with m:11 and 4épC as having only 10 metrified syllable. They are merged with the 'natural' m:10 lines.  
df\_epC <- df\_epC %>%  
 mutate(meter=replace(meter, ces\_6=="6épC" & meter==11, 10)) %>%  
 as.data.frame()  
 #This replace meter=11 by meter=10 when ces\_6=6épC, in order to display lines with m:11 and 6épC as having only 10 metrified syllable. They are merged with the 'natural' m:10 lines.  
 #We don't do that for line m:>11, because location of metrical break is not reliable for them.  
#rawscore <- df\_epC[, c("Meter","Score1", "Score2")]  
#rawscore <- melt(rawscore, id = c("Type")) %>%  
# group\_by(variable) %>% summarize(value=mean(value), count = n())  
 #Allow to add labels of count  
 #From now, df\_epC is ready to be dataframe for global analysis and graph production.  
 #Seems to be (a) non-needed, and (b) source of error. Keep it there for archive for some time, then delete it!  
  
##Import global data for schwa/no schwa  
 PAM\_raw\_xlsx <- read\_excel("all.xlsx")

## Warning in read\_fun(path = enc2native(normalizePath(path)), sheet\_i = sheet, :  
## Expecting logical in V2727 / R2727C22: got 're'

## Warning in read\_fun(path = enc2native(normalizePath(path)), sheet\_i = sheet, :  
## Expecting logical in AA2989 / R2989C27: got 've'

## Warning in read\_fun(path = enc2native(normalizePath(path)), sheet\_i = sheet, :  
## Expecting logical in AA2993 / R2993C27: got 'ge'

## Warning in read\_fun(path = enc2native(normalizePath(path)), sheet\_i = sheet, :  
## Expecting logical in AA3695 / R3695C27: got 'i'

## Warning in read\_fun(path = enc2native(normalizePath(path)), sheet\_i = sheet, :  
## Expecting logical in AC3695 / R3695C29: got 'gne'

## Warning in read\_fun(path = enc2native(normalizePath(path)), sheet\_i = sheet, :  
## Expecting logical in AA4147 / R4147C27: got 'gιes'

## Warning in read\_fun(path = enc2native(normalizePath(path)), sheet\_i = sheet, :  
## Expecting logical in AC4147 / R4147C29: got 'er'

## Warning in read\_fun(path = enc2native(normalizePath(path)), sheet\_i = sheet, :  
## Expecting logical in AA4581 / R4581C27: got 'os'

## Warning in read\_fun(path = enc2native(normalizePath(path)), sheet\_i = sheet, :  
## Expecting logical in T5113 / R5113C20: got 'e'

## Warning in read\_fun(path = enc2native(normalizePath(path)), sheet\_i = sheet, :  
## Expecting logical in T5445 / R5445C20: got 'e'

## Warning in read\_fun(path = enc2native(normalizePath(path)), sheet\_i = sheet, :  
## Expecting logical in AA7319 / R7319C27: got 'de'

## Warning in read\_fun(path = enc2native(normalizePath(path)), sheet\_i = sheet, :  
## Expecting logical in AC7319 / R7319C29: got 'les'

## Warning in read\_fun(path = enc2native(normalizePath(path)), sheet\_i = sheet, :  
## Expecting logical in AA7407 / R7407C27: got 'le'

#First, let's apply some transformations to clean up and properly reorganize data!  
 ##Moove columns  
 PAM\_raw\_xlsx <- PAM\_raw\_xlsx %>% relocate(meter, ces\_3, ces\_4, ces\_5, ces\_6, ces\_7)  
   
 ##Delete uneven lines (which contains syllables and not tags)  
 toDelete <- seq(1, nrow(PAM\_raw\_xlsx), 2)  
 PAM\_tag <- PAM\_raw\_xlsx[ toDelete ,]  
  
 ##Delete empty columns (xlsx export from PAM creates extra columns to be sure to be large enough for big too-long lines) (would also delete unused -1 (elided schwa) columns, but it shouldn't have any consequences and should be very rare, if existing)  
 PAM\_tag <- PAM\_tag[,colSums(is.na(PAM\_tag))<nrow(PAM\_tag)]  
  
 ##Compute the 'Yes/No' criteria for presence/absence of final schwas in the line  
 #Computation of feminine rimes (schwa in feminine rime should not count in counting below as it cannot made the line ill-formed)  
 #Define a function  
 lastValue <- function(x) tail(x[!is.na(x)], 1)  
 #Apply it to the all df  
 PAM\_tag$rime <- apply(PAM\_tag, 1, lastValue)  
   
 #Computation of the number of schwas in the line  
 #For non elided final schwa  
 PAM\_tag$schwa\_C\_sum\_temp <- (rowSums(PAM\_tag == 0, na.rm = TRUE))  
 #For elided final schwa  
 PAM\_tag$schwa\_V\_sum <- (rowSums(PAM\_tag == -1, na.rm = TRUE))  
 #Re-ajust the number of elided final schwa to exclude (a) line-final schwa (should not count here as it cannot made the line ill-formed) and (b) the criteria rime=='1' criteria (meaning the rime is feminine)  
 PAM\_tag <- PAM\_tag %>%  
 mutate(schwa\_C\_sum = case\_when(  
 rime == 0 ~ (schwa\_C\_sum\_temp - 2),  
 rime > 0 ~ schwa\_C\_sum\_temp  
 ))  
  
 #Computation of the boolean variables on final schwa  
 #TO DO?  
 PAM\_tag <- PAM\_tag %>%  
 mutate(schwa\_C = case\_when(  
 schwa\_C\_sum > 0 ~ "Yes",  
 schwa\_C\_sum == 0 ~ "No"  
 ))  
 PAM\_tag <- PAM\_tag %>%  
 mutate(schwa\_V = case\_when(  
 schwa\_V\_sum > 0 ~ "Yes",  
 schwa\_V\_sum == 0 ~ "No"  
 ))  
  
 #Calculate the percentage of final schwa per line  
 PAM\_tag <- PAM\_tag %>%  
 mutate(percent\_schwa\_C = schwa\_C\_sum / meter)  
 PAM\_tag <- PAM\_tag %>%  
 mutate(percent\_schwa\_V = schwa\_V\_sum / meter)  
  
 #Reduce meter==11 to meter==10 when 4épC or 6épC detected  
 PAM\_tag\_epC <- PAM\_tag %>%  
 mutate(meter=replace(meter, ces\_4=="4épC" & meter==11, 10)) %>%  
 as.data.frame()  
 PAM\_tag\_epC <- PAM\_tag\_epC %>%  
 mutate(meter=replace(meter, ces\_6=="6épC" & meter==11, 10)) %>%  
 as.data.frame()  
  
 #Reorder data for readability  
 PAM\_tag\_epC <- PAM\_tag\_epC %>% relocate(meter, rime, schwa\_C, schwa\_C\_sum, schwa\_V, schwa\_V\_sum, percent\_schwa\_C, percent\_schwa\_V)  
 PAM\_tag\_epC$schwa\_C\_sum\_temp <- NULL  
  
#Prepare data for regression line (does not work with count numbers (in fake % shape).  
#PAM\_tag\_epC <- PAM\_tag\_epC %>%  
 # mutate(meter\_cont = as.numeric(as.character(meter)))  
  
#Import data for m:9  
#df\_m9 <- read.table("m9.txt", header=T)  
  
#PLOT IT!  
##LOC = dotplot that shows location of each type of line  
LOC <- ggplot(  
 df\_epC,  
 aes(  
 x=num\_l,  
 y=meter)) +  
  
geom\_jitter(  
 width=0,  
 height=0.05) +  
  
 ###Good looking parameters  
 labs(  
 x="Line number",  
 y="Number of metrified syllables",  
 title = "Line lenght\* per line",  
 caption = "\*metrified syllables only") +  
 #Very bad method to set the breaks, but still works. To be changed in the future.  
 scale\_y\_continuous(breaks=c(1,2,3,4,5,6,7,8,9,10,11,12,13,14,15,16,17,18,19,20)) +  
 theme\_bw()  
  
##DISTRIB\_lines = barplot of each type of m:n[x]  
DISTRIB\_lines <-ggplot(df\_epC, aes(meter)) + geom\_bar() +  
 geom\_text(stat='count', aes(label=..count..), vjust=-0.5) +  
 ###Good looking parameters  
 labs(  
 x="Number of metrified syllables",  
 y="Number of lines (squared log.)",  
 title = "Distribution of line lenght\*",  
 caption = "\*metrified syllables only") +  
 theme\_classic() +  
 coord\_trans(y='sqrt') +  
 scale\_x\_continuous(breaks=c(7,8,9,10,11,12,13,14,15)) +  
 scale\_y\_continuous(breaks=c(5,10,50,150,1000,4000))  
  
##Apply transformations (needed for ploting) to the df  
PAM\_tag\_epC <- PAM\_tag\_epC %>%  
 mutate(schwa\_C\_sum\_disc = as\_factor(schwa\_C\_sum))  
PAM\_tag\_epC <- PAM\_tag\_epC %>%  
 mutate(schwa\_V\_sum\_disc = as\_factor(schwa\_V\_sum))  
PAM\_tag\_epC <- PAM\_tag\_epC %>%  
 mutate(meter\_cont = as\_factor(meter))  
  
cor.test(x = PAM\_tag\_epC$meter, y = PAM\_tag\_epC$schwa\_C\_sum, method="kendall")

##   
## Kendall's rank correlation tau  
##   
## data: PAM\_tag\_epC$meter and PAM\_tag\_epC$schwa\_C\_sum  
## z = 4.8959, p-value = 9.786e-07  
## alternative hypothesis: true tau is not equal to 0  
## sample estimates:  
## tau   
## 0.072317

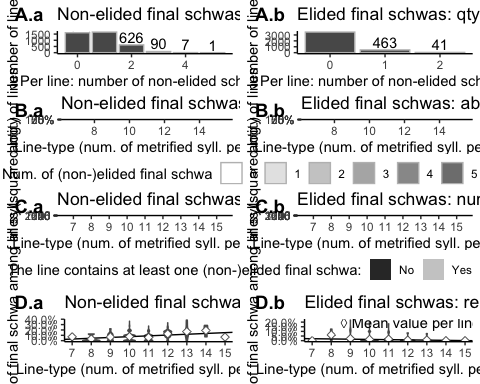
##NUM\_SCHWA = barplot with quantity of schwa per line and metrical status  
# A.a  
NUM\_SCHWA\_C <- ggplot(PAM\_tag\_epC, aes(schwa\_C\_sum, fill = meter)) + geom\_bar(position="stack", color="grey") +  
 geom\_text(stat='count', aes(label=..count..), vjust = -0.25) +  
 labs(  
 x="Per line: number of non-elided schwas",  
 y="Number of lines",  
 title = "Non-elided final schwas: qty. per line") +  
 theme\_classic()  
  
# A.b  
NUM\_SCHWA\_V <- ggplot(PAM\_tag\_epC, aes(schwa\_V\_sum, fill = meter)) + geom\_bar(position="stack", color="grey") +  
 geom\_text(stat='count', aes(label=..count..), vjust = -0.25) +  
 labs(  
 x="Per line: number of non-elided schwas",  
 y="Number of lines",  
 title = "Elided final schwas: qty. per line") +  
 theme\_classic()  
  
##QUANTITY\_SCHWA = filled barplot with quantity of schwa per line and metrical status  
## B.a  
QUANTITY\_SCHWA\_C <- ggplot(PAM\_tag\_epC, aes(meter, fill = forcats::fct\_rev(schwa\_C\_sum\_disc))) + geom\_bar(position="fill", color="grey") +  
 geom\_text(stat='count', aes(label=..count..), position = position\_fill(vjust = .5), color="black") +  
 labs(  
 x="Line-type (num. of metrified syll. per line)",  
 y="Quantity of lines",  
 fill="Num. of (non-)elided final schwa",  
 title = "Non-elided final schwas: absolute qty. per line-type") +  
 theme\_classic() +  
 theme(legend.position = "bottom") +  
 scale\_fill\_manual(values = c("grey50", "grey60", "grey70", "grey80", "grey90", "white")) +  
 scale\_y\_continuous(labels=scales::percent) +  
 guides(fill = guide\_legend(nrow = 1, reverse=T))  
  
## B.c  
QUANTITY\_SCHWA\_V <- ggplot(PAM\_tag\_epC, aes(meter, fill = forcats::fct\_rev(schwa\_V\_sum\_disc))) + geom\_bar(position="fill", color="grey") +  
 geom\_text(stat='count', aes(label=..count..), position = position\_fill(vjust = .5), color="black") +  
 labs(  
 x="Line-type (num. of metrified syll. per line)",  
 y="Quantity of lines",  
 fill="Num. of (non-)elided final schwa",  
 title = "Elided final schwas: absolute qty. per line-type") +  
 theme\_classic() +  
 theme(legend.position = "bottom") +  
scale\_fill\_manual(values = c("grey80", "grey90", "white")) +  
 scale\_y\_continuous(labels=scales::percent) +  
 guides(fill = guide\_legend(nrow = 1, reverse=T))  
  
##YESNO = barplot with quantity of lines that have/have not at least one final schwa  
# C.a  
YESNO\_C <- ggplot(PAM\_tag\_epC, aes(x=meter, fill=schwa\_C)) + geom\_bar(position=position\_dodge()) +  
 labs(  
 x="Line-type (num. of metrified syll. per line)",  
 y="Number of lines (squared log.)",  
 title = "Non-elided final schwas: number of lines",  
 fill = "The line contains at least one (non-)elided final schwa:") +  
 theme\_classic() + theme(legend.position = "bottom") +  
 scale\_x\_continuous(breaks=c(7,8,9,10,11,12,13,14,15)) +  
 scale\_y\_continuous(breaks=c(5,10,25,50,125, 250, 500, 1000, 2000, 3000, 4000, 5000)) +  
 scale\_fill\_grey() +  
 coord\_trans(y='sqrt')  
  
# C.b  
YESNO\_V <- ggplot(PAM\_tag\_epC, aes(x=meter, fill=schwa\_V)) + geom\_bar(position=position\_dodge()) +  
 labs(  
 x="Line-type (num. of metrified syll. per line)",  
 y="Number of lines (squared log.)",  
 title = "Elided final schwas: number of lines",  
 fill = "The line contains at least one (non-)elided final schwa:") +  
 theme\_classic() + theme(legend.position = "bottom") +  
 scale\_x\_continuous(breaks=c(7,8,9,10,11,12,13,14,15)) +  
 scale\_y\_continuous(breaks=c(5,10,25,50,125, 250, 500, 1000, 2000, 3000, 4000, 5000)) +  
 scale\_fill\_grey() +  
 coord\_trans(y='sqrt')  
  
##PERCENT\_SCHWA = filled barplot with percentage of schwa per line and metrical status  
# D.a  
coefs\_C <- coef(lm(percent\_schwa\_C ~ meter, data = PAM\_tag\_epC))  
PERCENT\_SCHWA\_C <- ggplot(PAM\_tag\_epC, aes(x=meter, y=percent\_schwa\_C, group=meter\_cont)) + geom\_violin(fill="grey35",color="grey35") +  
 labs(  
 x="Line-type (num. of metrified syll. per line)",  
 y="Percentage of final schwa among all syll.",  
 title = "Non-elided final schwas: relative qty. per line-type") +  
 theme\_classic() +  
 theme(legend.position = "bottom") +  
 scale\_x\_continuous(breaks=c(1,2,3,4,5,6,7,8,9,10,11,12,13,14,15,16,17,18,19,20)) +  
 scale\_y\_continuous(labels=scales::percent) +  
 geom\_abline(intercept = coefs\_C[1], slope = coefs\_C[2]) +  
 stat\_summary(fun.y=mean, geom="point", shape=23, size=2, color="grey35", fill="white")

## Warning: `fun.y` is deprecated. Use `fun` instead.

# D.e  
coefs\_V <- coef(lm(percent\_schwa\_V ~ meter, data = PAM\_tag\_epC))  
PERCENT\_SCHWA\_V <- ggplot(PAM\_tag\_epC, aes(x=meter, y=percent\_schwa\_V, group=meter\_cont)) + geom\_violin(fill="grey35",color="grey35") +  
 labs(  
 x="Line-type (num. of metrified syll. per line)",  
 y="Percentage of final schwa among all syll.",  
 title = "Elided final schwas: relative qty. per line-type") +  
 theme\_classic() +  
 theme(legend.position = "bottom") +  
 scale\_x\_continuous(breaks=c(1,2,3,4,5,6,7,8,9,10,11,12,13,14,15,16,17,18,19,20)) +  
 scale\_y\_continuous(labels=scales::percent) +  
 geom\_abline(intercept = coefs\_V[1], slope = coefs\_V[2]) +  
 stat\_summary(fun.y=mean, geom="point", shape=23, size=2, color="grey35", fill="white") +  
 annotate("text", x = 13, y = 0.2, label = "\U25CA Mean value per line-type")

## Warning: `fun.y` is deprecated. Use `fun` instead.

#The wanted symbol is:U25C7, but it does not work... No time for this!  
  
##ARRANGE: compiles plot on only one figure  
###QUANTITY SCHWA for \_C and \_V  
QUANTITY\_SCHWA <- ggarrange(QUANTITY\_SCHWA\_C, QUANTITY\_SCHWA\_V,  
 labels = "AUTO",  
 ncol=2, nrow=1,  
 common.legend=TRUE,  
 legend="bottom")  
  
NUM\_SCHWA <- ggarrange(NUM\_SCHWA\_C, NUM\_SCHWA\_V,  
 labels = "AUTO",  
 ncol=2, nrow=1)  
  
YESNO <- ggarrange(YESNO\_C, YESNO\_V,  
 labels = "AUTO",  
 ncol=2, nrow=1)  
  
PERCENT <- ggarrange(PERCENT\_SCHWA\_C,PERCENT\_SCHWA\_V,  
 labels = "AUTO",  
 ncol=2,nrow=1)  
  
ALL\_SCHWA <- ggarrange(  
 ggarrange(NUM\_SCHWA\_C, NUM\_SCHWA\_V,  
 labels = c("A.a","A.b"),  
 ncol=2,  
 nrow=1),  
 ggarrange(QUANTITY\_SCHWA\_C, QUANTITY\_SCHWA\_V,  
 labels = c("B.a","B.b"),  
 ncol=2,  
 nrow=1,  
 common.legend=TRUE,  
 legend="bottom"),  
 ggarrange(YESNO\_C, YESNO\_V,  
 labels = c("C.a","C.b"),  
 ncol=2,  
 nrow=1,  
 common.legend=TRUE,  
 legend="bottom"),  
 ggarrange(PERCENT\_SCHWA\_C,PERCENT\_SCHWA\_V,  
 labels = c("D.a","D.b"),  
 ncol=2,  
 nrow=1),  
 ncol=1)  
  
#PRINT IT!  
##Uncomment the plot you want to print. Leave all uncommented to print all plots.  
#LOC  
# ggsave(LOC, filename = "loc\_meter.png", width=25,height=10, units="cm", scale=1, dpi="retina")  
  
#DISTRIB\_lines  
# ggsave(DISTRIB\_lines, filename = "distrib\_meter.png", width=25, height=20.13, units="cm", scale=1, dpi="retina")  
  
#NUM\_SCHWA  
# ggsave(NUM\_SCHWA, filename = "num\_schwa.png", width=25, height=12, units="cm", scale=1, dpi="retina")  
  
#QUANTITY\_SCHWA  
 #ggsave(QUANTITY\_SCHWA, filename = "qty\_schwa.png", width=25, height=12, units="cm", scale=1, dpi="retina")  
  
ALL\_SCHWA



ggsave(ALL\_SCHWA, filename = "all\_schwa.png", width=25, height=36, units="cm", scale=1, dpi="retina")

#ARCHIVE. Contains no print instruction, so can be let uncommented for now; to comment or delete to increase process speed.

SCATTER\_SCHWA\_V <- ggplot(PAM\_tag\_epC, aes(x=meter, y=schwa\_C\_sum)) + geom\_point() +# geom\_jitter(height=0.2) +  
labs(  
 x="Number of metrified syllables",  
 y="Number of non-elided final schwa(s)",  
 title = "Quantity of non-elided final schwas per line-type") +  
theme\_classic() +  
scale\_x\_continuous(breaks=c(7,8,9,10,11,12,13,14,15)) +  
geom\_smooth(method=NULL, se=FALSE)  
  
DENSITY\_SCHWA\_V <- ggplot(PAM\_tag\_epC, aes(x=meter, color=schwa\_C, fill=schwa\_C)) + geom\_density(adjust = 5, alpha = 0.1) +  
scale\_color\_grey() + scale\_fill\_grey() +  
coord\_trans(y='sqrt') +  
theme\_classic()  
  
LINE\_SCHWA\_V <- ggplot(PAM\_tag\_epC, aes(x=meter, y=schwa\_C\_sum\_disc, color=schwa\_C\_sum, fill=schwa\_C\_sum)) + geom\_line(alpha = 0.1) +  
scale\_color\_grey() + scale\_fill\_grey() +  
theme\_classic()  
  
PROB\_C <- ggplot(PAM\_tag\_epC, aes(x=schwa\_C\_sum, fill=meter\_cont, group=meter\_cont)) + geom\_bar(position="fill", color="grey") +  
 geom\_text(stat='count', aes(label=..count..), position = position\_fill(vjust = .5), check\_overlap=TRUE) +  
 labs(  
 x="Number of non-elided schwas per line",  
 y="Quantity of lines",  
 title = "Non-elided final schwas and line well-formedness",  
 fill = "Number of\nmetrified\nsyllables") +  
 theme\_classic() +  
 scale\_x\_continuous(breaks=c(0,1,2,3,4,5)) +  
 scale\_y\_continuous(labels=scales::percent) +  
 scale\_fill\_manual(values = c("grey10", "grey20", "grey30", "grey50", "grey70", "grey80", "grey90", "grey95", "white")) +  
 theme(legend.position = "none")  
  
PROB\_V <- ggplot(PAM\_tag\_epC, aes(x=schwa\_V\_sum, fill=meter\_cont, group=meter\_cont)) + geom\_bar(position="fill", color="grey") +  
 geom\_text(stat='count', aes(label=..count..), position = position\_fill(vjust = .5), check\_overlap=TRUE) +  
 labs(  
 x="Number of elided schwas per line",  
 y="Quantity of lines",  
 title = "Elided final schwas and line well-formedness",  
 fill = "Number of\nmetrified\nsyllables") +  
 theme\_classic() +  
 scale\_x\_continuous(breaks=c(0,1,2,3,4,5)) +  
 scale\_y\_continuous(labels=scales::percent) +  
 scale\_fill\_manual(values = c("grey10", "grey20", "grey30", "grey50", "grey70", "grey80", "grey90", "grey95", "white"))