Aesthetic Literature Project Scripts

Yuchao Wang

6/7/2020

Preprocessing of Qualtrics

- Using a `.csv` file, `forWordSeparation_cleaned`, to create a *word* by *participant* matrix with behavioral ratings (*emotional arousal* or *literariness*) from Qualtrics survey outputs.
- Specifically, `forWordSeparation_cleaned.csv` is read in to create `template`, which is used to search for the rating per word per participant.

```
library(tidyr)
#Clean up csv files containing all rated words in survey----
template <- read.csv(file = "Qualtrics/forWordSeparation_cleaned.csv",
                     header = F, sep = ",", stringsAsFactors = FALSE)
template <- t(template)
colnames(template) <- template[1, ]
template <- template[-1, ]
template <- as.data.frame(template) #data type converted to matrix after transpose, due to rectangular
template <- gather(template, "qnLabel", "word", 1:17)
```

```
## Warning: attributes are not identical across measure variables;
## they will be dropped
```

```
template <- template[!(template$word == ""), ]
template <- template[ , c("word","qnLabel")]
template$qnLabel <- trimws(template$qnLabel)
template$qnLabel <- as.factor(template$qnLabel)
template$qnLabel <- factor(template$qnLabel, levels = c("3.1","5.1","7.1","9.1",
                                                        "11.1","13.1","15.1","17.1","19.1","24.1","26.1
                                                        "28.1","30.1","32.1","34.1","36.1","38.1")) #qu
qnLabel <- levels(template$qnLabel)


#Match emo results----
emoResult <- read.csv(file="Qualtrics/Verhalen_Emotional+Arousal_10_1_19.csv",
                      header = TRUE, sep = ",", stringsAsFactors = FALSE)
emoResult <- emoResult[-c(1,2), ] #get rid of title row
emoResult <- emoResult[-c(1:9), ] #get rid of preview Test results
emoResult <- emoResult[-c(15:17,28), ] #get rid of incomplete results


totalParticipant <- nrow(emoResult)
totalWord <- nrow(template)
emoResultClean <- template
```

```
for (indexResponse in 1:totalParticipant) {
  for (indexWord in 1:totalWord) {
    searchWord <- template[indexWord,1]
    searchQn <- template[indexWord,2]
    resultCol <- indexResponse+2
    emoResultClean[indexWord,resultCol] <- NA #initialize a new column
      for (possibleRating in 1:7) {
        if (!is.na(emoResultClean[indexWord,resultCol])) {
          break
        }
        currentColName <- paste ("Q", searchQn, "_", possibleRating, sep = "", collapse = NULL)
        if (grepl(searchWord, emoResult[indexResponse,currentColName], fixed = TRUE)) {
            emoResultClean[indexWord,resultCol] <- substr(currentColName,
                                                          nchar(currentColName), nchar(currentColName))
        }
        else {
          if ((possibleRating == 7) & (is.na(emoResultClean[indexWord,resultCol]))) {
            emoResultClean[indexWord,resultCol] <- NA
          }
        }
      }
  }
}


#Match lit results----
litResult <- read.csv(file="Qualtrics/Verhalen_Literariness_10_1_19.csv",
                      header = TRUE, sep = ",", stringsAsFactors = FALSE)
litResult <- litResult[-c(1,2), ] #get rid of title row
litResult <- litResult[-c(1:10), ] #get rid of preview Test results


totalParticipant <- nrow(litResult)
totalWord <- nrow(template)
litResultClean <- template


for (indexResponse in 1:totalParticipant) {
  for (indexWord in 1:totalWord) {
    searchWord <- template[indexWord,1]
    searchQn <- template[indexWord,2]
    resultCol <- indexResponse+2
    litResultClean[indexWord,resultCol] <- NA #initialize a new column
      for (possibleRating in 1:2) {
        if (!is.na(litResultClean[indexWord,resultCol])) {
          break
        }
        currentColName <- paste ("Q", searchQn, "_", possibleRating, sep = "", collapse = NULL)
        if (grepl(searchWord, litResult[indexResponse,currentColName], fixed = TRUE)) {
            litResultClean[indexWord,resultCol] <- substr(currentColName,
                                                          nchar(currentColName), nchar(currentColName))
        }
        else {
```

```
          if ((possibleRating == 2) & (is.na(litResultClean[indexWord,resultCol]))) {
            litResultClean[indexWord,resultCol] <- NA
          }
        }
      }
  }
}
```

Stats - Intraclass Correlation Coefficient

- To check how well participants agree with each other on ratings.

```
library(lme4)
```

```
## Loading required package: Matrix
```

```
##
## Attaching package: 'Matrix'
```

```
## The following objects are masked from 'package:tidyr':
##
##     expand, pack, unpack
```

```
library(psych)
emoResultCleanICC <- matrix(data = NA, nrow = 0, ncol = 27)
for (index in 1:2354) {
  emoResultCleanICC <- rbind(emoResultCleanICC, emoResultClean[index, 3:29])
}
for (j in 1:27) {
    emoResultCleanICC[,j] <- as.numeric(emoResultCleanICC[,j])
}
ICC(emoResultCleanICC, missing = FALSE, alpha = 0.05)
```

```
## Call: ICC(x = emoResultCleanICC, missing = FALSE, alpha = 0.05)
##
## Intraclass correlation coefficients
##                          type  ICC  F  df1   df2 p lower bound upper bound
## Single_raters_absolute   ICC1 0.18  7 2353 61204 0        0.17        0.19
## Single_random_raters     ICC2 0.19 12 2353 61178 0        0.16        0.23
## Single_fixed_raters      ICC3 0.28 12 2353 61178 0        0.27        0.29
## Average_raters_absolute ICC1k 0.86  7 2353 61204 0        0.85        0.86
## Average_random_raters   ICC2k 0.86 12 2353 61178 0        0.83        0.89
## Average_fixed_raters    ICC3k 0.91 12 2353 61178 0        0.91        0.92
##
##  Number of subjects = 2354     Number of Judges =  27
```

```
litResultCleanICC <- matrix(data = NA, nrow = 0, ncol = 27)
for (index in 1:2354) {
  litResultCleanICC <- rbind(litResultCleanICC, litResultClean[index, 3:29])
}
for (j in 1:27) {
    litResultCleanICC[,j] <- as.numeric(litResultCleanICC[,j])
}
ICC(litResultCleanICC, missing = FALSE, alpha = 0.05)
```

```
## Call: ICC(x = litResultCleanICC, missing = FALSE, alpha = 0.05)
##
```

```
## Intraclass correlation coefficients
##                            type  ICC    F  df1   df2 p lower bound
## Single_raters_absolute    ICC1 0.24  9.3 2353 61204 0        0.22
## Single_random_raters      ICC2 0.24 10.6 2353 61178 0        0.22
## Single_fixed_raters       ICC3 0.26 10.6 2353 61178 0        0.25
## Average_raters_absolute ICC1k 0.89  9.3 2353 61204 0        0.89
## Average_random_raters   ICC2k 0.89 10.6 2353 61178 0        0.88
## Average_fixed_raters    ICC3k 0.91 10.6 2353 61178 0        0.90
##                           upper bound
## Single_raters_absolute           0.25
## Single_random_raters             0.25
## Single_fixed_raters              0.28
## Average_raters_absolute          0.90
## Average_random_raters            0.90
## Average_fixed_raters             0.91
##
##  Number of subjects = 2354     Number of Judges =  27
```

Stats - Emo ~ Lit by story

- By story
- To check if *emotional arousal* and *literariness* are distinct and can serve as regressors in separate fMRI GLMs.
- Specifically, checked both the correlation using 1. the original word-by-word mean values and 2. the rollmean with two preceding and two succeeding words

```
#Simple linear regression ANOVA of Emo on Lit ratings, with and without boxcar smoothing
library(zoo)
```

```
##
## Attaching package: 'zoo'
```

```
## The following objects are masked from 'package:base':
##
##     as.Date, as.Date.numeric
```

```
library(psych)

#EMO
temoResultClean <- t(emoResultClean)
temoResultClean <- temoResultClean[-c(1:2),]

emoTrend <- describe(temoResultClean)
```

```
## Converted non-numeric matrix input to numeric.  Are you sure you wanted to do this. Please check your
```

```
emoDHTrend <- emoTrend[c(1:1237),]
emoDMTrend <- emoTrend[c(1238:2354),]
emoDMTrend$vars <- seq(length=nrow(emoDMTrend))


#LIT
tlitResultClean <- t(litResultClean)
tlitResultClean <- tlitResultClean[-c(1:2),]
litTrend <- describe(tlitResultClean)
```

```
## Converted non-numeric matrix input to numeric.  Are you sure you wanted to do this. Please check your
```

```r
litDHTrend <- litTrend[c(1:1237),]
litDMTrend <- litTrend[c(1238:2354),]
litDMTrend$vars <- seq(length=nrow(litDMTrend))

#Cor.test
emoDHTrend <- cbind(emoDHTrend, rollmean(emoDHTrend$mean, k = 5, fill = NA, align = "center"))
names(emoDHTrend)[14] <- "rollmean"
litDHTrend <- cbind(litDHTrend, rollmean(litDHTrend$mean, k = 5, fill = NA, align = "center"))
names(litDHTrend)[14] <- "rollmean"
cor.test(emoDHTrend$mean, litDHTrend$mean, method = "pearson")
```

```
##
##  Pearson's product-moment correlation
##
## data:  emoDHTrend$mean and litDHTrend$mean
## t = 11.077, df = 1234, p-value < 2.2e-16
## alternative hypothesis: true correlation is not equal to 0
## 95 percent confidence interval:
##  0.2491395 0.3506013
## sample estimates:
##      cor
## 0.300721
```

```r
cor.test(emoDHTrend$rollmean, litDHTrend$rollmean, method = "pearson")
```

```
##
##  Pearson's product-moment correlation
##
## data:  emoDHTrend$rollmean and litDHTrend$rollmean
## t = 10.532, df = 1226, p-value < 2.2e-16
## alternative hypothesis: true correlation is not equal to 0
## 95 percent confidence interval:
##  0.2359002 0.3385254
## sample estimates:
##       cor
## 0.2880396
```

```r
emoDMTrend <- cbind(emoDMTrend, rollmean(emoDMTrend$mean, k = 5, fill = NA, align = "center"))
names(emoDMTrend)[14] <- "rollmean"
litDMTrend <- cbind(litDMTrend, rollmean(litDMTrend$mean, k = 5, fill = NA, align = "center"))
names(litDMTrend)[14] <- "rollmean"
cor.test(emoDMTrend$mean, litDMTrend$mean, method = "pearson")
```

```
##
##  Pearson's product-moment correlation
##
## data:  emoDMTrend$mean and litDMTrend$mean
## t = -0.78249, df = 1115, p-value = 0.4341
## alternative hypothesis: true correlation is not equal to 0
## 95 percent confidence interval:
##  -0.08196979  0.03527649
## sample estimates:
##         cor
## -0.02342721
```

```r
cor.test(emoDMTrend$rollmean, litDMTrend$rollmean, method = "pearson")
```

```
##
##  Pearson's product-moment correlation
##
## data:  emoDMTrend$rollmean and litDMTrend$rollmean
## t = -1.6976, df = 1111, p-value = 0.08987
## alternative hypothesis: true correlation is not equal to 0
## 95 percent confidence interval:
##  -0.109297957  0.007920187
## sample estimates:
##         cor
## -0.05086406
```

Match word ratings with onset

- Match Qualtrics ratings per word (for 1st person perspective stories) with the onset time of that word in both 1st and 3rd person perspective story recordings.
- Praat timing files adjusted to eliminate mismatches due to formatting or transcription differences.
- Add a column `onset`.

```r
#READ IN ALL 4 ADJUSTED PRAAT WORD ONSETS AND DURATIONS----
colLabel <- c("Word", "Onset", "Duration")
DHO <- read.delim("Praat/De Hond_O_adjusted.csv", header = FALSE, sep = ",", stringsAsFactors = FALSE)
DHO <- DHO[-c(1:3), ] #get rid of title
colnames(DHO) <- colLabel

DMO <- read.delim("Praat/De Muur_O_adjusted.csv", header = FALSE, sep = ",", stringsAsFactors = FALSE)
DMO <- DMO[-c(1,2), ] #get rid of title
colnames(DMO) <- colLabel

DH2V <- read.delim("Praat/De Hond_2V_adjusted.csv", header = FALSE, sep = ",", stringsAsFactors = FALSE)
DH2V <- DH2V[-c(1:3), ] #get rid of title
colnames(DH2V) <- colLabel

DM2V <- read.delim("Praat/De Muur_2V_adjusted.csv", header = FALSE, sep = ",", stringsAsFactors = FALSE)
DM2V <- DM2V[-c(1,2), ] #get rid of title
colnames(DM2V) <- colLabel

#MATCH, FOR ORIGINAL(FIRST PERSON PERSPEC), THE TIMING WITH RATED WORDS----

#SETUP
colNumEmo <- ncol(emoResultClean)
colNumLit <- ncol(litResultClean)

emoOGwTiming <- emoResultClean
emo2VwTiming <- emoResultClean
litOGwTiming <- litResultClean
lit2VwTiming <- litResultClean


addTimingCols <- function(totalWord, praatClean, resultClean, resultwTiming, colNum) {
  if (totalWord == 1237) {
     for (index in 1:totalWord) {
```

```
        if (grepl(praatClean[index,1], resultClean[index,1], fixed = TRUE)) {
          resultwTiming[index, c(colNum+1, colNum+2)] <- praatClean[index, c(2,3)]
        }
        else {
          resultwTiming[index, c(colNum+1, colNum+2)] <- c("mismatch", "mismatch")
        }
     }
  }
  else {
    for (index in 1:totalWord) {
      if (grepl(praatClean[index,1], resultClean[index + 1237,1], fixed = TRUE)) {
        resultwTiming[index + 1237, c(colNum+1, colNum+2)] <- praatClean[index, c(2,3)]
      }
      else {
        resultwTiming[index + 1237, c(colNum+1, colNum+2)] <- c("mismatch", "mismatch")
      }
    }
  }
  return(resultwTiming)
}

emoOGwTiming <- addTimingCols(1237, DHO, emoResultClean, emoOGwTiming, colNumEmo)
emoOGwTiming <- addTimingCols(1117, DMO, emoResultClean, emoOGwTiming, colNumEmo)
litOGwTiming <- addTimingCols(1237, DHO, litResultClean, litOGwTiming, colNumLit)
litOGwTiming <- addTimingCols(1117, DMO, litResultClean, litOGwTiming, colNumLit)
emo2VwTiming <- addTimingCols(1237, DH2V, emoResultClean, emo2VwTiming, colNumEmo)
emo2VwTiming <- addTimingCols(1117, DM2V, emoResultClean, emo2VwTiming, colNumEmo)
lit2VwTiming <- addTimingCols(1237, DH2V, litResultClean, lit2VwTiming, colNumLit)
lit2VwTiming <- addTimingCols(1117, DM2V, litResultClean, lit2VwTiming, colNumLit)
```

Add mean and SE for behavioral ratings

- Add rating mean and SE as additional columns to matrix, and clean mismatched rows
- Save resulted matrix per story (`DM` and `DH`) per perspective (`O` and `2V`) per rating (`emo` and `lit`) as `.csv` files.

```
stderr <- function(x) {
  sd(x, na.rm = TRUE)/sqrt(length(x[!is.na(x)]))
}

addMNSECol <- function(resultwTiming, colNum) {
  for (colIndex in 3: colNum) {
    resultwTiming[, colIndex] <- as.numeric(resultwTiming[, colIndex])
  }
  resultwTiming <- transform(resultwTiming, MN = rowMeans(resultwTiming[ ,3:colNum], na.rm = TRUE))
  rowNum <- nrow(resultwTiming)
  for (rowIndex in 1:rowNum) {
    resultwTiming[rowIndex, colNum+4] <- stderr(resultwTiming[rowIndex, 3:colNum])
  }
  names(resultwTiming)[colNum+4] <- 'SE'
  return(resultwTiming)
}

emoOGwTiming <- addMNSECol(emoOGwTiming, colNumEmo)
```

```
emo2VwTiming <- addMNSECol(emo2VwTiming, colNumEmo)
litOGwTiming <- addMNSECol(litOGwTiming, colNumLit)
lit2VwTiming <- addMNSECol(lit2VwTiming, colNumLit)



#WRITE OUTPUT FILES
emoDHOFinal <- matrix(data = NA, nrow = 0, ncol = colNumEmo+4)
emoDMOFinal <- matrix(data = NA, nrow = 0, ncol = colNumEmo+4)
emoDH2VFinal <- matrix(data = NA, nrow = 0, ncol = colNumEmo+4)
emoDM2VFinal <- matrix(data = NA, nrow = 0, ncol = colNumEmo+4)
litDHOFinal <- matrix(data = NA, nrow = 0, ncol = colNumLit+4)
litDMOFinal <- matrix(data = NA, nrow = 0, ncol = colNumLit+4)
litDH2VFinal <- matrix(data = NA, nrow = 0, ncol = colNumLit+4)
litDM2VFinal <- matrix(data = NA, nrow = 0, ncol = colNumLit+4)

#This function cleans mismatched entries and write CSV output files.
cleanOutput <- function(totalWord, colNum, rawData, outputFile, outputName) {
  if (totalWord == 1237) {
    for (index in 1:totalWord) {
      if ((rawData[index, colNum+2] == "mismatch") | (rawData[index, colNum+2] == "0")) {
        next
      }
      else {
        outputFile <- rbind(outputFile, rawData[index, ])
      }
    }
  }
  else {
    for (index in 1:totalWord) {
      if ((rawData[index+1237, colNum+2] == "mismatch") | (rawData[index+1237, colNum+2] == "0")) {
        next
      }
      else {
        outputFile <- rbind(outputFile, rawData[index+1237, ])
      }
    }
  }
  write.csv(outputFile, file = paste("ProcessedByWord/", outputName, ".csv", sep=""))
  return(outputFile)
}


emoDHOFinal <- cleanOutput(1237, colNumEmo, emoOGwTiming, emoDHOFinal, "emoDHOFinal")
emoDMOFinal <- cleanOutput(1117, colNumEmo, emoOGwTiming, emoDMOFinal, "emoDMOFinal")
emoDH2VFinal <- cleanOutput(1237, colNumEmo, emo2VwTiming, emoDH2VFinal, "emoDH2VFinal")
emoDM2VFinal <- cleanOutput(1117, colNumEmo, emo2VwTiming, emoDM2VFinal, "emoDM2VFinal")
litDHOFinal <- cleanOutput(1237, colNumLit, litOGwTiming, litDHOFinal, "litDHOFinal")
litDMOFinal <- cleanOutput(1117, colNumLit, litOGwTiming, litDMOFinal, "litDMOFinal")
litDH2VFinal <- cleanOutput(1237, colNumLit, lit2VwTiming, litDH2VFinal, "litDH2VFinal")
litDM2VFinal <- cleanOutput(1117, colNumLit, lit2VwTiming, litDM2VFinal, "litDM2VFinal")
```

Adding MN and Onset based on event boundaries

- Event boundaries were created by Franziska Hartung on the original, 1st person perspective version (`0`) of both stories.
- The same event boundaries were created on the 3rd person perspective versions (`2V`) using a separate script.
- **Do not** run this chunk again because all `_eventMarked.csv`, which initially only contains a column of event markers in addition to the *per word* `Final` files, have been processed to add MN and SE *per event*.

```
emoDH0marked <- read.csv(file = "emoDH0Final_eventMarked.csv", header = T, sep = ",", stringsAsFactors =
emoDH2Vmarked <- read.csv(file = "emoDH2VFinal_eventMarked.csv", header = T, sep = ",", stringsAsFactors
emoDM0marked <- read.csv(file = "emoDM0Final_eventMarked.csv", header = T, sep = ",", stringsAsFactors =
emoDM2Vmarked <- read.csv(file = "emoDM2VFinal_eventMarked.csv", header = T, sep = ",", stringsAsFactors
litDH0marked <- read.csv(file = "litDH0Final_eventMarked.csv", header = T, sep = ",", stringsAsFactors =
litDH2Vmarked <- read.csv(file = "litDH2VFinal_eventMarked.csv", header = T, sep = ",", stringsAsFactors
litDM0marked <- read.csv(file = "litDM0Final_eventMarked.csv", header = T, sep = ",", stringsAsFactors =
litDM2Vmarked <- read.csv(file = "litDM2VFinal_eventMarked.csv", header = T, sep = ",", stringsAsFactors

addEventParam <- function(dataEventMarked) {
#initialize condition for search
  begin_row <- 1
  end_row <- 1
  total_row <- nrow(dataEventMarked)
  #search loop
  while (end_row <= total_row) {
    if  (is.na(dataEventMarked$marker_event[end_row])) {
      end_row <- end_row+1
    }
    else {
      dataEventMarked$onset_event[end_row] <- dataEventMarked$Onset[begin_row]
      dataEventMarked$duration_event[end_row] <- sum(dataEventMarked$Duration[begin_row:end_row])
      dataEventMarked$MN_event[end_row] <- mean(dataEventMarked$MN[begin_row:end_row], na.rm = TRUE)
      begin_row <- end_row+1
      end_row <- end_row+1
    }
  }
  return(dataEventMarked)
}

emoDH0marked <- addEventParam(emoDH0marked)
emoDH2Vmarked <- addEventParam(emoDH2Vmarked)
emoDM0marked <- addEventParam(emoDM0marked)
emoDM2Vmarked <- addEventParam(emoDM2Vmarked)
litDH0marked <- addEventParam(litDH0marked)
litDH2Vmarked <- addEventParam(litDH2Vmarked)
litDM0marked <- addEventParam(litDM0marked)
litDM2Vmarked <- addEventParam(litDM2Vmarked)

write.csv(emoDH0marked, file = "emoDH0Final_eventMarked.csv", na = "")
write.csv(emoDH2Vmarked, file = "emoDH2VFinal_eventMarked.csv", na = "")
write.csv(emoDM0marked, file = "emoDM0Final_eventMarked.csv", na = "")
write.csv(emoDM2Vmarked, file = "emoDM2VFinal_eventMarked.csv", na = "")
write.csv(litDH0marked, file = "litDH0Final_eventMarked.csv", na = "")
write.csv(litDH2Vmarked, file = "litDH2VFinal_eventMarked.csv", na = "")
```

```
write.csv(litDMOmarked, file = "litDMOFinal_eventMarked.csv", na = "")
write.csv(litDM2Vmarked, file = "litDM2VFinal_eventMarked.csv", na = "")
```

Final Visualization

- To be used in manuscript
- `AllOriginal_eventMarked.csv` created by manually concatenating `emoDHOFinal_eventMarked`, `emoDMOFinal_eventMarked`, `litDHOFinal_eventMarked`, `litDMOFinal_eventMarked`.
- Overall plots of ratings against onset, per story, per type of rating (`emo` and `lit`).
- For 1. per word ratings 2. per semantic event ratings.
- SE for literariness is remapped to 0 because its a binary decision and SE does not make sense.

```
library(ggplot2)
```

```
##
## Attaching package: 'ggplot2'
```

```
## The following objects are masked from 'package:psych':
##
##     %+%, alpha
```

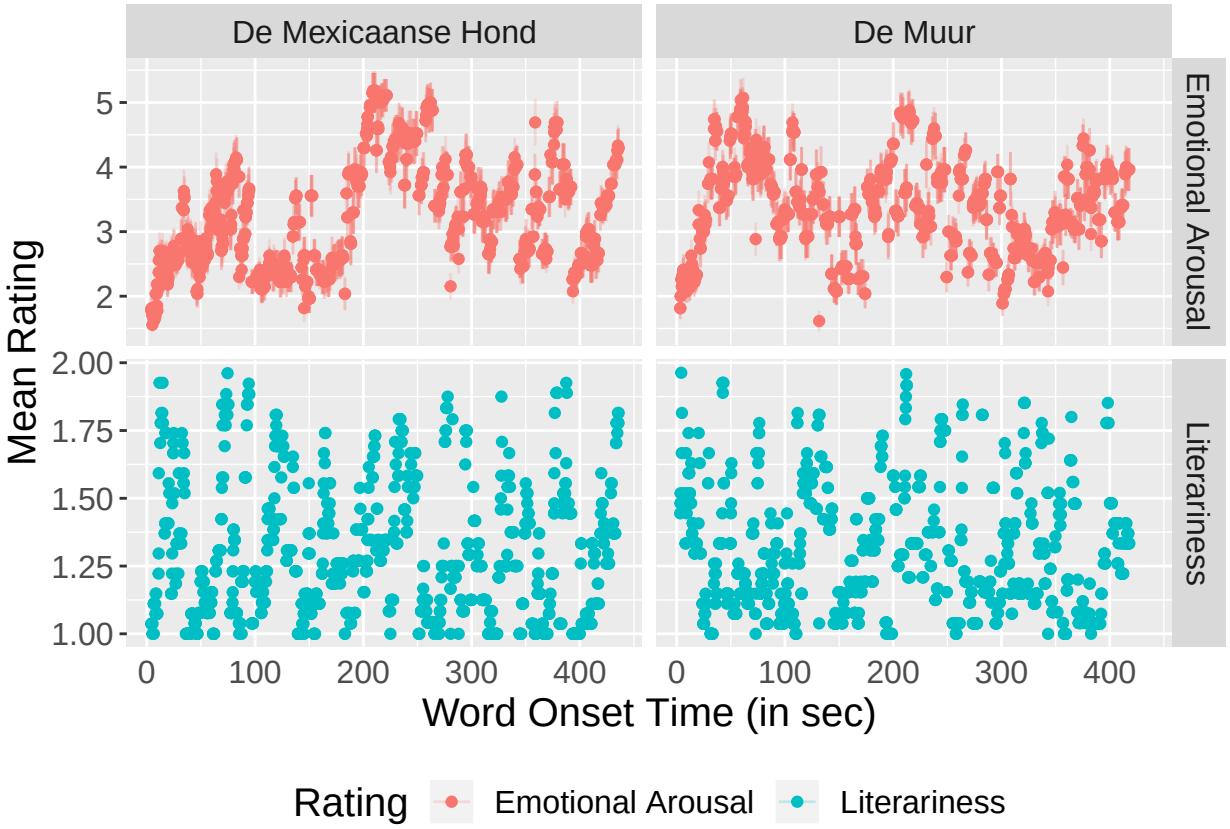
```
allRatingMarked <- read.csv(file = "ProcessedByEvent/OriginalCombined_eventMarked.csv", header = T, sep

#correct for interpretation of literariness SE by changing all lit SE to 0
for (i in 1:nrow(allRatingMarked)) {
  if (allRatingMarked$Rating[i] == "Literariness") {
    allRatingMarked$SE[i] = 0
  }
}

limits <- aes(ymax = allRatingMarked$MN + allRatingMarked$SE,
              ymin = allRatingMarked$MN - allRatingMarked$SE)
word_plot <- ggplot(data = allRatingMarked, aes(x = Onset, y = MN, color = Rating)) +
    geom_point(size=1.5) +
    geom_errorbar(limits, width = 0.1, alpha = 0.2) +
    xlab("Word Onset Time (in sec)") +
    ylab("Mean Rating") +
    theme(text = element_text(size=15),  axis.ticks.x=element_blank(), legend.position="bottom") +
    facet_grid(Rating ~ Story, scales = "free_y")
event_plot <- ggplot(data = allRatingMarked, aes(x = Onset_event, y = MN_event, color = Rating)) +
    geom_step(direction = "hv", size=0.5) +
    xlab("Event Onset Time (in sec)") +
    ylab("Mean Rating") +
    theme(text = element_text(size=15),  axis.ticks.x=element_blank(), legend.position="bottom") +
    facet_grid(Rating ~ Story, scales = "free_y")
word_plot
```

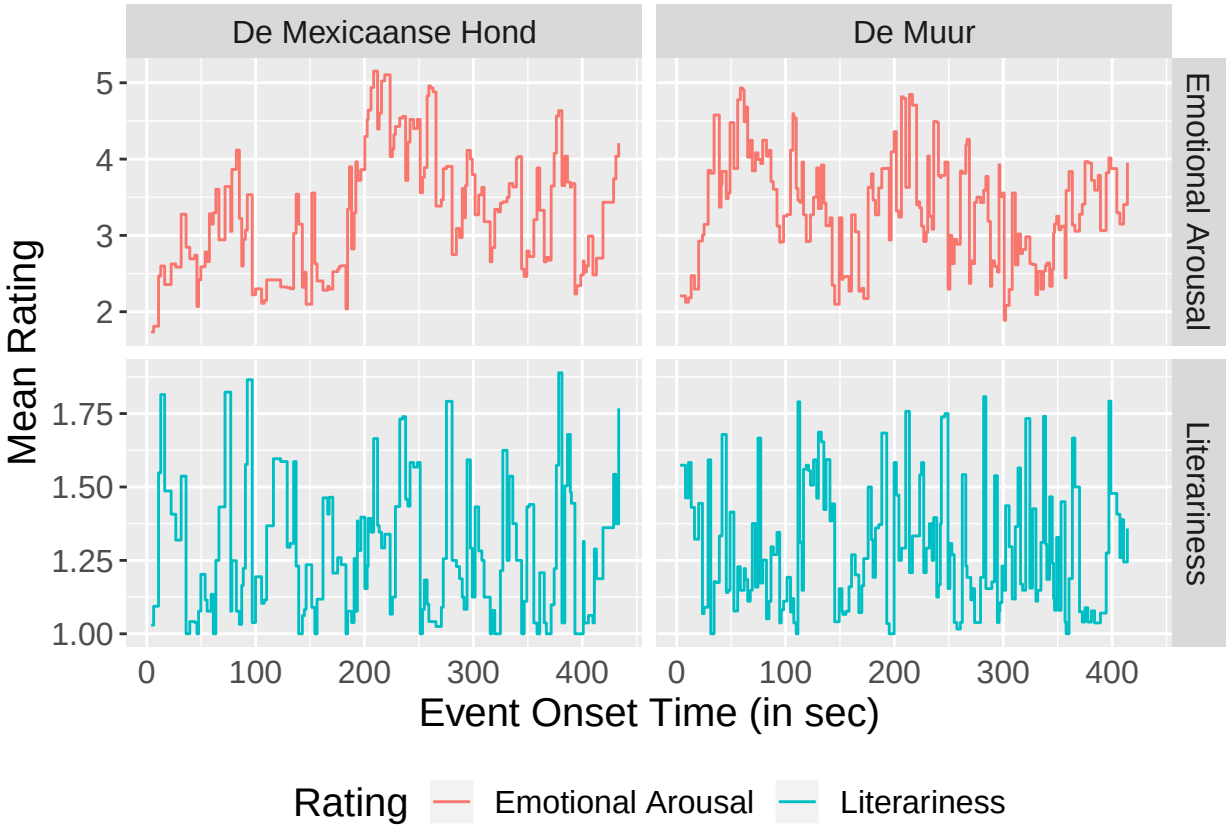
```
## Warning: Removed 1 rows containing missing values (geom_point).
```

```
## Warning: Removed 1 rows containing missing values (geom_errorbar).
```




```
event_plot
```

```
## Warning: Removed 8 rows containing missing values (geom_path).
```




```
ggsave("allWordRating.png", plot = word_plot, device = png(),
    scale = 1, width = 400, height = 240, units = c("mm"),
    dpi = 300, limitsize = TRUE)
```

```
## Warning: Removed 1 rows containing missing values (geom_point).
```

```
## Warning: Removed 1 rows containing missing values (geom_errorbar).
```

```
ggsave("allEventRating.png", plot = event_plot, device = png(),
    scale = 1, width = 400, height = 240, units = c("mm"),
    dpi = 300, limitsize = TRUE)
```

```
## Warning: Removed 8 rows containing missing values (geom_path).
```

Stats - Emo ~ Lit (stories concatenated)

- Using original, 1st perspec stories.

```
allEmoWordRating <- unlist(subset(allRatingMarked, Rating == "Emotional Arousal", select = "MN"))
allLitWordRating <- unlist(subset(allRatingMarked, Rating == "Literariness", select = "MN"))
allEmoEventRating <- unlist(subset(allRatingMarked, Rating == "Emotional Arousal", select = "MN_event"))
allLitEventRating <- unlist(subset(allRatingMarked, Rating == "Literariness", select = "MN_event"))
cor.test(allEmoWordRating, allLitWordRating, method = "pearson")
```

```
##
##  Pearson's product-moment correlation
##
## data:  allEmoWordRating and allLitWordRating
```

```
## t = 8.1896, df = 2341, p-value = 4.258e-16
## alternative hypothesis: true correlation is not equal to 0
## 95 percent confidence interval:
##  0.1272542 0.2059923
## sample estimates:
##       cor
## 0.1668893
```

```
cor.test(allEmoEventRating, allLitEventRating, method = "pearson")
```

```
##
##  Pearson's product-moment correlation
##
## data:  allEmoEventRating and allLitEventRating
## t = 2.8498, df = 323, p-value = 0.004655
## alternative hypothesis: true correlation is not equal to 0
## 95 percent confidence interval:
##  0.04864803 0.26095741
## sample estimates:
##       cor
## 0.1566114
```