Signe Kløve Kjær

01/07/2018

Link to code:

<https://github.com/YpipY/EXPMETH3/blob/master/Assignment%201%20and%202%20(autism)/Signe_A1_P1_LangASD_code.Rmd>

# Assignment 1, Part 1: Language development in Autism Spectrum Disorder (ASD) - Brushing up your code skills

In this first part of the assignment we will brush up your programming skills, and make you familiar with the data sets you will be analysing for the next parts of the assignment.

In this first part of the assignment you will: 1) Create a Github (or gitlab) account and link it to your RStudio 2) Use small nifty lines of code to transform several data sets into just one. The final data set will contain only the variables that are needed for the analysis in the next parts of the assignment 3) Become familiar with the tidyverse package (especially the sub-packages stringr and dplyr), which you will find handy for later assignments.

# Language development in Autism Spectrum Disorder (ASD)

Background: Autism Spectrum Disorder is often related to language impairment. However, this phenomenon has not been empirically traced in detail: i) relying on actual naturalistic language production, ii) over extended periods of time. We therefore videotaped circa 30 kids with ASD and circa 30 comparison kids (matched by linguistic performance at visit 1) for ca. 30 minutes of naturalistic interactions with a parent. We repeated the data collection 6 times per kid, with 4 months between each visit. We transcribed the data and counted: i) the amount of words that each kid uses in each video. Same for the parent. ii) the amount of unique words that each kid uses in each video. Same for the parent. iii) the amount of morphemes per utterance (Mean Length of Utterance) displayed by each child in each video. Same for the parent.

## 1. Let's get started on GitHub

Follow the link to a Github tutorial: <https://support.rstudio.com/hc/en-us/articles/200532077-Version-Control-with-Git-and-SVN>

In the assignments you will be asked to upload your code on Github and the GitHub repositories will be part of the portfolio, therefore all students must make an account and link it to their RStudio (you'll thank us later for this!).

N.B. Create a GitHub repository for the Language Development in ASD set of assignments and link it to a project on your RStudio.

You may also use Gitlab instead of Github (Malte will explain in class)

## 2. Now let's take dirty dirty data sets and make them into a tidy one

If you're not in a project in Rstudio, make sure to set your working directory here. If you created an rstudio project, then your working directory (the directory with your data and code for these assignments) is the project directory. You can always see which working directory is used with getwd(). Note that this may be different in the console than in the Rmd cells (don't ask me why).

Load the three data sets, after downloading them from dropbox and saving them in your working directory: \* Demographic data for the participants: <https://www.dropbox.com/s/w15pou9wstgc8fe/demo_train.csv?dl=0> \* Length of utterance data: <https://www.dropbox.com/s/usyauqm37a76of6/LU_train.csv?dl=0> \* Word data: <https://www.dropbox.com/s/8ng1civpl2aux58/token_train.csv?dl=0>

#load data and turn strings into factors   
demo<-read.csv("demo\_train.csv", stringsAsFactors = F)  
lu<-read.csv("LU\_train.csv", stringsAsFactors = F)  
token<-read.csv("token\_train.csv", stringsAsFactors = F)  
  
#load library  
library(plyr) #probably bad idea to install this, fucks up with group\_by()   
library(dplyr)

##   
## Attaching package: 'dplyr'

## The following objects are masked from 'package:plyr':  
##   
## arrange, count, desc, failwith, id, mutate, rename, summarise,  
## summarize

## The following objects are masked from 'package:stats':  
##   
## filter, lag

## The following objects are masked from 'package:base':  
##   
## intersect, setdiff, setequal, union

library(stringr)  
library(tidyverse)

## Loading tidyverse: ggplot2  
## Loading tidyverse: tibble  
## Loading tidyverse: tidyr  
## Loading tidyverse: readr  
## Loading tidyverse: purrr

## Conflicts with tidy packages ----------------------------------------------

## arrange(): dplyr, plyr  
## compact(): purrr, plyr  
## count(): dplyr, plyr  
## failwith(): dplyr, plyr  
## filter(): dplyr, stats  
## id(): dplyr, plyr  
## lag(): dplyr, stats  
## mutate(): dplyr, plyr  
## rename(): dplyr, plyr  
## summarise(): dplyr, plyr  
## summarize(): dplyr, plyr

Explore the 3 datasets (e.g. visualize them, summarize them, etc.). You will see that the data is messy, since the psychologists collected the demographic data, a linguist analyzed the length of utterance in May 2014 and the same linguist analyzed the words several months later. In particular: - the same variables might have different names (e.g. identifier of the child) - the same variables might report the values in different ways (e.g. visit) Welcome to real world of messy data :-)

Before being able to combine the data sets we need to make sure the relevant variables have the same names and the same kind of values.

So:

2a. Find a way to transform variable names. Tip: Look into the package dplyr (part of tidyverse) Tip: Or google "how to rename variables in R". Tip: Or look through the chapter on data transformation in R for data science (<http://r4ds.had.co.nz>).

#rename variable names   
colnames(demo)[colnames(demo) == "Child.ID"] <- "SUBJ"  
colnames(demo)[colnames(demo) == "Visit"] <- "VISIT"  
  
#could also use rename instead

2b. Find a way to homogeneize the way "visit" is reported. If you look into the original data sets, you will see that in the LU data and the Token data, Visits are called "visit 1"" instead of just "1"" (which is the case in the demographic data set). Tip: There is a package called stringr, which will be very handy for manipulating (text) strings also in furture assignments. We will return to this package later, but for now use the str\_extract () to extract only the number from the variable Visit in each data set.

#change content of VISIT column  
#stupid way, but possible, use revalue   
token$VISIT <- revalue(token$VISIT, c("visit1." = "1", "visit2." ="2","visit3." = "3", "visit4." = "4", "visit5." = "5", "visit6." = 6 ))  
  
#much smarter way, use str\_extract   
token$VISIT<-str\_extract(token$VISIT, "[1-9]")  
lu$VISIT<-str\_extract(lu$VISIT, "[1-9]")  
  
#done in another way  
token$VISIT<-str\_extract(token$VISIT, "[:digit:]")  
  
  
#could also use gsub (a regular expression)  
#str\_extract (a regular expression), str\_extract tells R to look for a pattern, like a digit and extract it  
#regular expressions use the same langugae (but it's a little difficult)

2c. We also need to make a small adjustment to the content of the Child.ID coloumn in the demographic data. Within this column, names that are not abbreviations do not end with "." (i.e. Adam), which is the case in the other two data sets (i.e. Adam.). If The content of the two variables isn't identical the data sets will not be merged sufficiently. We wish to remove the "." at the end of names in the LU data and the tokens data.

Tip: stringr is helpful again. Look up str\_replace\_all Tip: You can either have one line of code for each child name that is to be changed (easier, more typing) or specify the pattern that you want to match (more complicated: look up "regular expressions", but less typing)

Tip: You will have to do identical work for both data sets, so to save time on the copy/paste use the cmd+f/ctrl+f function. Add the data frame name (e.g. LU\_data) in the first box, and the data frame name (e.g. Tokens\_data) you wish to change it to in the other box, and press replace. Or create a function that takes the data set and does the transformation. Then call the function on both data sets.

#make subject variable into character  
lu$SUBJ = as.character(lu$SUBJ)  
token$SUBJ = as.character(token$SUBJ)  
  
#loop that goes through all rows in subject and check if there is only one ".", if that's the case, replace it with nothing  
for (row in 1:nrow(token)) {  
 if(str\_count(token[row, "SUBJ"], "[.]") == 1){  
 token[row, "SUBJ"] <- str\_replace(token[row, "SUBJ"], "[.]", "")  
 }  
}  
  
#loop that goes through all rows in subject and check if there is only one ".", if that's the case, replace it with nothing  
for (row in 1:nrow(lu)) {  
 if(str\_count(lu[row, "SUBJ"], "[.]") == 1){  
 lu[row, "SUBJ"] <- str\_replace(lu[row, "SUBJ"], "[.]", "")  
 }  
}  
  
  
#could also just remove all the dots, use [:punct:] and str\_replace  
  
#could also replace last character by using "\\.$" and str\_replace

2d. Now that the nitty gritty details of the different data sets are fixed, we want to make a subset of each data set only containig the variables that we wish to use in the final data set. For this we use the tidyverse package dplyr, which contains the function select().

The variables we need are: Child.ID, Visit, Ethnicity, Diagnosis, Gender, Age, ADOS, MullenRaw, ExpressiveLangRaw, MOT\_MLU, MOT\_LUstd, CHI\_MLU, CHI\_LUstd, types\_MOT, types\_CHI, tokens\_MOT, tokens\_CHI.

* ADOS indicates the severity of the autistic symptoms (the higher the score, the worse the symptoms)
* MLU stands for mean length of utterance
* types stands for unique words (e.g. even if "doggie" is used 100 times it only counts for 1)
* tokens stands for overall amount of words (if "doggie" is used 100 times it counts for 100)
* MullenRaw indicates non verbal IQ
* ExpressiveLangRaw indicates verbal IQ

It would be smart to rename the last two into something you can remember (i.e. nonVerbalIQ, verbalIQ)

#rename last two columns   
colnames(demo)[colnames(demo) == "MullenRaw"] <- "nonVerbalIQ"  
colnames(demo)[colnames(demo) == "ExpressiveLangRaw"] <- "verbalIQ"  
  
#select only the variables we need  
selected\_demo <- dplyr::select(demo,SUBJ, VISIT, Ethnicity, Diagnosis, Gender, Age, ADOS, nonVerbalIQ, verbalIQ)  
selected\_token <- dplyr::select(token,c(SUBJ, VISIT, types\_MOT, types\_CHI, tokens\_MOT, tokens\_CHI))  
selected\_lu <- dplyr::select(lu,c(SUBJ, VISIT, MOT\_MLU, MOT\_LUstd, CHI\_MLU, CHI\_LUstd))  
  
  
#important to think about what you throw out and what you keep

2e. Finally we are ready to merge all the data sets into just one. Tip: Look up "joins" in R for data science, or google "How to merge datasets in R" Tip: Joining / merging only works for two data frames at the time.

#put the three dataframes together  
#inner join, left join, right join, full join   
#have to decide what you want to do with missing datapoints, should you throw away data?   
#turn visit into character  
selected\_demo$VISIT <- as.character(selected\_demo$VISIT)  
  
#use full join to merge the first two dataframes  
merge\_demo\_lu <- full\_join(selected\_demo, selected\_lu, by = c("SUBJ", "VISIT"))  
  
#use full join to merge wiht the third dataframe  
merged\_data <-full\_join(merge\_demo\_lu, selected\_token, by = c("SUBJ", "VISIT"))

Are we done? If you look at the data set now, you'll se a lot of NA's in the variables ADOS, nonVerbalIQ (MullenRaw) and verbalIQ (ExpressiveLangRaw). These measures were not taken at all visits. Additionally, we only want these measures for the first visit (Riccardo will explain why in class). So let's make sure that we select only these variables as collected during the first visit for each child and repeat these values throughout all other visits.

* A way to get around this in R, is to make a subset of the data containing only the values we are interested in, i.e. the values at visit 1. We only want to keep the relevant variables in this data set, i.e the ones which values are to be repeated. Either the the subset() function or select() and filter() can be used here. Solve this issue with both functions to familiarize yourself with these useful functions. N.B. save the subset of the dataset as a new dataset, do not overwrite the old one.
* In order to merge these new variables to the final data set, they'll need new names. E.g change the ADOS variable to ADOS1.
* Once you've changed the names, the subset can be merged to the final data set, and the score at the 1st visit in each variable will be spread to all 6.
* Lastly, there are too many unneccesary variables in the data set by now. Use the select() to choose only the variables you want in the data (e.g. remove the old ADOS, verbal and nonVerbal IQ variables, so you will not get confused later on) and define the order of the variables. Hint: You want the child identifier, followed by diagnosis, followed by demographic, cognitive and clinical features, followed by indexes of linguistic performance (utterances lenghts, types and tokens of words).

ADOS, nonVerbalIQ (MullenRaw) and verbalIQ (ExpressiveLangRaw)

#use subset and filter function  
subset\_data <- subset(merged\_data, VISIT == "1", select = c("SUBJ", "VISIT", "ADOS", "nonVerbalIQ","verbalIQ"))  
  
#filter\_data <- filter(merged\_data, VISIT == "1")  
  
#select\_data <- select(filter\_data, SUBJ, VISIT, ADOS, nonVerbalIQ,verbalIQ )  
  
  
#change column names  
colnames(subset\_data)[colnames(subset\_data) == "ADOS"] <- "ADOS1"  
colnames(subset\_data)[colnames(subset\_data) == "nonVerbalIQ"] <- "nonVerbalIQ1"  
colnames(subset\_data)[colnames(subset\_data) == "verbalIQ"] <- "verbalIQ1"  
  
#merge full data and subset data, to have value from visit one at all visits for ados1, nonverbal1 and verbal IQ1  
merged\_data\_0 <- full\_join(merged\_data, subset\_data, by = "SUBJ")  
  
  
#make dataframe with only the new columns  
merged\_data\_1 <- subset(merged\_data\_0, select =-c(ADOS, nonVerbalIQ, verbalIQ))  
  
  
  
#make dataframe with variables in the right order  
merged\_data\_2 <- dplyr::select(merged\_data\_1, c(SUBJ, VISIT.x, Diagnosis, Ethnicity, Gender, Age, ADOS1, nonVerbalIQ1, verbalIQ1,types\_MOT, types\_CHI, tokens\_MOT, tokens\_CHI,MOT\_MLU, MOT\_LUstd, CHI\_MLU, CHI\_LUstd))  
  
#change variable name for visit.x to visit  
colnames(merged\_data\_2)[colnames(merged\_data\_2) == "VISIT.x"] <- "VISIT"

Now, we are almost ready to start analysing the data. However, here are some additional finishing touches:

* in some experiments your participants must be anonymous. Therefore we wish to turn the CHILD.ID into numbers.

Tip: You will probably need to turn it into a factor first, then a number Tip: google "R how to convert character to integer" or look up the as.??? functions

* In order to make it easier to work with this nice, clean dataset in the future, it is practical to make sure the variables have sensible values. E.g. right now gender is marked 1 and 2, but in two weeks you will not be able to remember, which gender were connected to which number, so change the values from 1 and 2 to F and M in the gender variable. For the same reason, you should also change the values of Diagnosis from A and B to ASD (autism spectrum disorder) and TD (typically developing). Tip: Try taking a look at ifelse(), or google "how to rename levels in R".

#turn child ID into factors  
merged\_data\_2$SUBJ <- as.factor(merged\_data\_2$SUBJ)  
  
#turn child ID into numbers   
merged\_data\_2$SUBJ <- as.integer(merged\_data\_2$SUBJ)  
  
  
#change numbers in gender to a factor  
merged\_data\_2$SUBJ <- as.factor(merged\_data\_2$SUBJ)  
  
#change numbers in gender to F and M  
merged\_data\_2$Gender <-ifelse(merged\_data\_2$Gender == "1", "M", "F")  
  
#change A and B in diagnosis to ASD and TD  
merged\_data\_2$Diagnosis <-ifelse(merged\_data\_2$Diagnosis == "A", "ASD", "TD")

Write the data set into a csv file.

#write csv file  
write.csv(merged\_data\_2, file = "autism\_data.csv")

1. Now that we have a nice clean data set to use for the analysis next week, we shall play a bit around with it. The following exercises are not relevant for the analysis, but are here so you can get familiar with the functions within the tidyverse package.

Here's the link to a very helpful book, which explains each function: <http://r4ds.had.co.nz/index.html>

USING FILTER List all kids (CHI) who: 1. have a mean length of utterance (across all visits) of more than 2.7 morphemes. 2. have a mean length of utterance of less than 1.5 morphemes at the first visit 3. have not completed all trials. Tip: Use pipes %>% to solve this

#create new dataset with kids who have a mean length of utterances of more than 2.7 morphenes across all visits  
utterance\_all\_visits <- merged\_data\_2 %>%  
 group\_by(SUBJ) %>% #group by subject  
 filter(mean(CHI\_MLU) > 2.7) #filter by means bigger than 2.7  
  
  
#create dataset with kids with MLU less than 1.5 morphemes at the first visit  
  
#use filter to make dataframe with only the first visits  
utterance\_first\_visit <- filter(merged\_data\_2, VISIT == 1)  
  
#list the children with a MLU at less than 1.5 morphenes  
utterance\_first\_visit <- filter(utterance\_first\_visit, CHI\_MLU < 1.5)  
  
  
#list kids who have not completed all trials  
na\_kids <- merged\_data\_2 %>%  
 group\_by(SUBJ) %>% #group by subject  
 filter(is.na(CHI\_MLU)) #take all the subjects with na values

USING ARRANGE

1. Sort kids to find the kid who produced the most words on the 6th visit
2. Sort kids to find the kid who produced the least amount of words on the 1st visit.

#find kid who produced most words at 6th visit  
merged\_data\_2 %>%  
 filter(VISIT ==6) %>% #filter by 6th visit  
 arrange(desc(tokens\_CHI)) %>% #arrange the columns tokens\_CHI by descending order  
 slice(1) #take the first kid

## # A tibble: 1 x 17  
## SUBJ VISIT Diagnosis Ethnicity Gender Age ADOS1 nonVerbalIQ1  
## <fctr> <chr> <chr> <chr> <chr> <dbl> <int> <int>  
## 1 62 6 TD White M 41.93 0 26  
## # ... with 9 more variables: verbalIQ1 <int>, types\_MOT <int>,  
## # types\_CHI <int>, tokens\_MOT <int>, tokens\_CHI <int>, MOT\_MLU <dbl>,  
## # MOT\_LUstd <dbl>, CHI\_MLU <dbl>, CHI\_LUstd <dbl>

#find kid who produced the least words at 1st visit  
merged\_data\_2 %>%  
 filter(VISIT ==1) %>% #filter by 6th visit  
 arrange(tokens\_CHI) %>% #arrange the columns tokens\_CHI, ascending by default  
 slice(1) #take the first kid

## # A tibble: 1 x 17  
## SUBJ VISIT Diagnosis Ethnicity Gender Age ADOS1 nonVerbalIQ1  
## <fctr> <chr> <chr> <chr> <chr> <dbl> <int> <int>  
## 1 64 1 ASD White F 41.07 15 28  
## # ... with 9 more variables: verbalIQ1 <int>, types\_MOT <int>,  
## # types\_CHI <int>, tokens\_MOT <int>, tokens\_CHI <int>, MOT\_MLU <dbl>,  
## # MOT\_LUstd <dbl>, CHI\_MLU <dbl>, CHI\_LUstd <dbl>

USING SELECT

1. Make a subset of the data including only kids with ASD, mlu and word tokens
2. What happens if you include the name of a variable multiple times in a select() call?

#use subset and select to make dataframe with kids with ASD and the columns CHI\_MLU and tokens\_CHI  
ASD\_data <- subset(merged\_data\_2, Diagnosis == "ASD", select = c("SUBJ", "CHI\_MLU", "tokens\_CHI"))  
  
#check what happens if you include a variable multiple times in select()  
  
ASD\_data\_test <- subset(merged\_data\_2, Diagnosis == "ASD", select = c("SUBJ", "CHI\_MLU", "tokens\_CHI", "SUBJ"))  
  
#it write it two times, but with .1 in the end

USING MUTATE, SUMMARISE and PIPES 1. Add a column to the data set that represents the mean number of words spoken during all visits. 2. Use the summarise function and pipes to add an column in the data set containing the mean amount of words produced by each trial across all visits. HINT: group by Child.ID 3. The solution to task above enables us to assess the average amount of words produced by each child. Why don't we just use these average values to describe the language production of the children? What is the advantage of keeping all the data?

#create dataframe  
#group\_by does not work if plyr is installed  
detach(package:plyr)  
  
#create dataframe with all variables and mean\_token in the end  
merged\_data\_3 <- merged\_data\_2 %>%  
 group\_by(SUBJ) %>%  
 mutate(mean\_token\_CHI = mean(tokens\_CHI, na.rm = TRUE))  
  
#create dataframe with mean\_token and subject number  
merged\_data\_4 <- merged\_data\_2%>%  
 group\_by(SUBJ)%>%  
 summarise(mean\_token = mean(tokens\_CHI, na.rm = TRUE))