A1\_P1\_Student

Riccardo Fusaroli

01/07/2018

# 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.

## 0. First an introduction on the data

# 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).

setwd("~/Documents/Cognitive Science/3rd semester/Experimental Methods 3/LanguageDevelopmentASD")

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>

#loading datasets  
demo\_train <- read.csv("demo\_train.csv")  
token\_train <- read.csv("token\_train.csv")  
lu\_train <- read.csv("LU\_train.csv")

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>).

library(pacman)  
p\_load(gdata, pdflatex, TeX)

## Warning: package 'pdflatex' is not available (for R version 3.4.1)

## Bioconductor version 3.6 (BiocInstaller 1.28.0), ?biocLite for help

## A new version of Bioconductor is available after installing the most  
## recent version of R; see http://bioconductor.org/install

## Warning in p\_install(package, character.only = TRUE, ...):

## Warning in library(package, lib.loc = lib.loc, character.only = TRUE,  
## logical.return = TRUE, : there is no package called 'pdflatex'

## Warning: package 'TeX' is not available (for R version 3.4.1)

## Bioconductor version 3.6 (BiocInstaller 1.28.0), ?biocLite for help  
## A new version of Bioconductor is available after installing the most  
## recent version of R; see http://bioconductor.org/install

## Warning in p\_install(package, character.only = TRUE, ...):

## Warning in library(package, lib.loc = lib.loc, character.only = TRUE,  
## logical.return = TRUE, : there is no package called 'TeX'

## Warning in p\_load(gdata, pdflatex, TeX): Failed to install/load:  
## pdflatex, TeX

#renaming the ID variable in all datasets to "ID"  
demo\_train <- rename.vars(demo\_train, "Child.ID", "ID")

##   
## Changing in demo\_train   
## From: Child.ID  
## To: ID

lu\_train <- rename.vars(lu\_train, "SUBJ", "ID")

##   
## Changing in lu\_train   
## From: SUBJ  
## To: ID

token\_train <- rename.vars(token\_train, "SUBJ", "ID")

##   
## Changing in token\_train   
## From: SUBJ  
## To: ID

#renaming the visit variable in all datasets to "Visit"  
lu\_train <- rename.vars(lu\_train, "VISIT", "Visit")

##   
## Changing in lu\_train   
## From: VISIT  
## To: Visit

token\_train <- rename.vars(token\_train, "VISIT", "Visit")

##   
## Changing in token\_train   
## From: VISIT  
## To: Visit

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.

p\_load(stringr)  
  
#homogenizing the visit variable; stringr extracts the numbers, these are assigned  
lu\_train$Visit <- str\_extract(lu\_train$Visit, "[1-6]")  
token\_train$Visit <- str\_extract(token\_train$Visit, "[1-6]")

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.

#removing periods in naming  
demo\_train$ID <- str\_replace\_all(demo\_train$ID, "[:punct:]", "")  
lu\_train$ID <- str\_replace\_all(lu\_train$ID, "[:punct:]", "")  
token\_train$ID <- str\_replace\_all(token\_train$ID, "[:punct:]", "")

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)

p\_load(dplyr, tidyverse)  
  
sub\_demo <- select(demo\_train, c(ID, Visit, Ethnicity, Diagnosis, Gender, Age, ADOS, MullenRaw, ExpressiveLangRaw))  
sub\_lu <- select(lu\_train, c(ID, Visit, MOT\_MLU, MOT\_LUstd, CHI\_MLU, CHI\_LUstd))  
sub\_token <- select(token\_train, c(ID, Visit, types\_MOT, types\_CHI, tokens\_MOT, tokens\_CHI))

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.

p\_load(plyr)  
  
#combining all three datasets  
all <- join(sub\_lu, sub\_token)

## Joining by: ID, Visit

all\_full <- join(sub\_demo, all)

## Joining by: ID, 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).

p\_load(sjmisc, tidyverse)  
  
#Making a subset with the relevant variables  
rep\_val <- subset(all\_full, Visit == 1, c(ID, ADOS, MullenRaw, ExpressiveLangRaw))  
  
#Renaming variables in new data set  
rep\_val <- rename.vars(rep\_val, "ADOS", "ADOS1")

##   
## Changing in rep\_val   
## From: ADOS   
## To: ADOS1

rep\_val <- rename.vars(rep\_val, "MullenRaw", "MullenRaw1")

##   
## Changing in rep\_val   
## From: MullenRaw   
## To: MullenRaw1

rep\_val <- rename.vars(rep\_val, "ExpressiveLangRaw", "ExpressiveLangRaw1")

##   
## Changing in rep\_val   
## From: ExpressiveLangRaw   
## To: ExpressiveLangRaw1

#Merging the variables back into the data frame  
full\_visit <- join(rep\_val, all\_full)

## Joining by: ID

#Removing the old, empty variables  
full\_visit\_clean <- remove\_var(full\_visit, c(ADOS, MullenRaw, ExpressiveLangRaw))  
  
#Using select to make a clean dataset with variables in the right order (removal above can be omitted)  
clean <- select(full\_visit\_clean, c(ID, Diagnosis, Gender, Age, Ethnicity, Visit, ADOS1, MullenRaw1, ExpressiveLangRaw1, CHI\_MLU, CHI\_LUstd, MOT\_MLU, MOT\_LUstd, types\_CHI, types\_MOT, tokens\_CHI, tokens\_MOT))

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".

clean$ID <- as.factor(clean$ID)  
clean$ID <- as.numeric(clean$ID)  
  
#Turning the 1s and 2s into M and F in the gender variable  
clean$Gender <- ifelse(clean$Gender == "1", "F", "M")  
  
#Turning the As and Bs intro ASD and TD in the diagnosis variable  
clean$Diagnosis <- ifelse(clean$Diagnosis == "A", "ASD", "TD")

Write the data set into a csv file.

#Writing csv  
write.csv(clean, "LanguageASDv2.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 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

#Kids with MLU > 2.7  
clean %>% filter(CHI\_MLU > 2.7)

## # A tibble: 97 x 17  
## ID Diagnosis Gender Age Ethnicity Visit ADOS1 MullenRaw1  
## <dbl> <chr> <chr> <dbl> <fctr> <int> <int> <int>  
## 1 1 TD F 35.90 White 5 0 28  
## 2 1 TD F 40.13 White 6 0 28  
## 3 3 TD M 35.53 White 4 1 29  
## 4 3 TD M 39.47 White 5 1 29  
## 5 3 TD M 45.07 White 6 1 29  
## 6 9 TD M 31.07 White 4 0 24  
## 7 9 TD M 35.00 White 5 0 24  
## 8 12 TD F 40.27 White 6 0 21  
## 9 2 ASD F 33.17 White 2 13 34  
## 10 2 ASD F 37.07 White 3 13 34  
## # ... with 87 more rows, and 9 more variables: ExpressiveLangRaw1 <int>,  
## # CHI\_MLU <dbl>, CHI\_LUstd <dbl>, MOT\_MLU <dbl>, MOT\_LUstd <dbl>,  
## # types\_CHI <int>, types\_MOT <int>, tokens\_CHI <int>, tokens\_MOT <int>

#Kids with MLU < 1.5 first visit  
clean %>% filter(CHI\_MLU < 1.5, Visit == "1")

## # A tibble: 47 x 17  
## ID Diagnosis Gender Age Ethnicity Visit ADOS1 MullenRaw1  
## <dbl> <chr> <chr> <dbl> <fctr> <int> <int> <int>  
## 1 1 TD F 19.80 White 1 0 28  
## 2 9 TD M 18.30 White 1 0 24  
## 3 12 TD F 19.23 White 1 0 21  
## 4 5 ASD F 34.03 White 1 9 34  
## 5 6 ASD M 26.17 Bangladeshi 1 17 20  
## 6 7 ASD M 41.00 White 1 18 24  
## 7 10 TD F 19.27 White 1 3 27  
## 8 13 TD F 20.07 White 1 0 30  
## 9 15 TD F 18.97 White 1 0 23  
## 10 16 TD F 19.27 White 1 0 24  
## # ... with 37 more rows, and 9 more variables: ExpressiveLangRaw1 <int>,  
## # CHI\_MLU <dbl>, CHI\_LUstd <dbl>, MOT\_MLU <dbl>, MOT\_LUstd <dbl>,  
## # types\_CHI <int>, types\_MOT <int>, tokens\_CHI <int>, tokens\_MOT <int>

#Filtering all rows containing NAs, hereby assuming that a visit/trial was not completed if there is no data  
#We also assume that if CHI\_MLU does not exist, then neither do the other variables (which seems to be the case)  
clean %>% filter(is.na(CHI\_MLU))

## # A tibble: 20 x 17  
## ID Diagnosis Gender Age Ethnicity Visit ADOS1 MullenRaw1  
## <dbl> <chr> <chr> <dbl> <fctr> <int> <int> <int>  
## 1 66 TD F 18.07 White 1 15 NA  
## 2 9 TD M NA White 6 0 24  
## 3 2 ASD F 45.23 White 5 13 34  
## 4 7 ASD M 60.33 White 6 18 24  
## 5 8 TD F 28.17 White 3 5 32  
## 6 11 ASD M 31.77 White 1 0 28  
## 7 11 ASD M 37.13 White American 2 0 28  
## 8 18 ASD F 46.67 White 4 14 25  
## 9 23 ASD F 35.47 Latino 1 19 26  
## 10 23 ASD F 39.87 Hispanic 2 19 26  
## 11 23 ASD F 43.57 Latino/Hispanic 3 19 26  
## 12 28 ASD F 34.43 White 2 11 32  
## 13 40 TD M 40.23 White 6 1 29  
## 14 42 ASD F 42.27 Lebanese 5 13 27  
## 15 46 TD F 39.93 White 6 3 30  
## 16 48 TD F 23.13 White 1 0 27  
## 17 48 TD F NA White 2 0 27  
## 18 50 TD F NA White 1 4 17  
## 19 50 TD F NA White 2 4 17  
## 20 59 TD F 36.83 White 5 0 26  
## # ... with 9 more variables: ExpressiveLangRaw1 <int>, CHI\_MLU <dbl>,  
## # CHI\_LUstd <dbl>, MOT\_MLU <dbl>, MOT\_LUstd <dbl>, types\_CHI <int>,  
## # types\_MOT <int>, tokens\_CHI <int>, tokens\_MOT <int>

#Summarising solution counting the amount of NA rows pr. child  
clean$ID <- as.character(clean$ID)  
  
detach(package:plyr)  
library(dplyr)  
  
clean %>%  
 group\_by(ID) %>%  
 filter(is.na(CHI\_MLU)) %>%  
 summarise(n())

## # A tibble: 15 x 2  
## ID `n()`  
## <chr> <int>  
## 1 11 2  
## 2 18 1  
## 3 2 1  
## 4 23 3  
## 5 28 1  
## 6 40 1  
## 7 42 1  
## 8 46 1  
## 9 48 2  
## 10 50 2  
## 11 59 1  
## 12 66 1  
## 13 7 1  
## 14 8 1  
## 15 9 1

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.

#Kids producing most words on the last visit  
clean %>% filter(Visit == "6") %>% arrange(desc(tokens\_CHI))

## # A tibble: 60 x 17  
## ID Diagnosis Gender Age Ethnicity Visit ADOS1 MullenRaw1  
## <chr> <chr> <chr> <dbl> <fctr> <int> <int> <int>  
## 1 59 TD F 41.93 White 6 0 26  
## 2 28 ASD F 51.00 White 6 11 32  
## 3 19 TD F 44.07 White 6 0 29  
## 4 27 TD F 42.47 White 6 0 29  
## 5 45 ASD F 57.37 White 6 14 42  
## 6 39 ASD F 58.77 White 6 7 33  
## 7 14 TD F 39.40 White 6 0 25  
## 8 24 TD M 39.23 White 6 0 21  
## 9 15 TD F 39.43 White 6 0 23  
## 10 4 ASD F 51.37 White/Latino 6 8 31  
## # ... with 50 more rows, and 9 more variables: ExpressiveLangRaw1 <int>,  
## # CHI\_MLU <dbl>, CHI\_LUstd <dbl>, MOT\_MLU <dbl>, MOT\_LUstd <dbl>,  
## # types\_CHI <int>, types\_MOT <int>, tokens\_CHI <int>, tokens\_MOT <int>

#Kids producing least words on the first visit  
clean %>% filter(Visit == "1") %>% arrange(tokens\_CHI)

## # A tibble: 66 x 17  
## ID Diagnosis Gender Age Ethnicity Visit ADOS1 MullenRaw1  
## <chr> <chr> <chr> <dbl> <fctr> <int> <int> <int>  
## 1 61 ASD M 41.07 White 1 15 28  
## 2 34 ASD F 34.27 White 1 21 21  
## 3 38 TD F 19.20 White 1 3 19  
## 4 32 ASD F 36.53 White 1 12 31  
## 5 26 ASD F 37.47 White 1 20 13  
## 6 6 ASD M 26.17 Bangladeshi 1 17 20  
## 7 35 ASD M 25.33 African American 1 14 25  
## 8 18 ASD F 34.80 White 1 14 25  
## 9 55 TD M 20.87 Asian 1 1 22  
## 10 52 ASD F 31.63 White 1 17 28  
## # ... with 56 more rows, and 9 more variables: ExpressiveLangRaw1 <int>,  
## # CHI\_MLU <dbl>, CHI\_LUstd <dbl>, MOT\_MLU <dbl>, MOT\_LUstd <dbl>,  
## # types\_CHI <int>, types\_MOT <int>, tokens\_CHI <int>, tokens\_MOT <int>

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?

#Filtering only ASD children and selecting these for a new subset including the two other variables  
clean %>% filter(Diagnosis == "ASD") %>% select(Diagnosis, CHI\_MLU, tokens\_CHI)

## # A tibble: 176 x 3  
## Diagnosis CHI\_MLU tokens\_CHI  
## <chr> <dbl> <int>  
## 1 ASD 2.276860 461  
## 2 ASD 3.453039 562  
## 3 ASD 3.119318 983  
## 4 ASD 4.302326 674  
## 5 ASD NA NA  
## 6 ASD 3.413502 698  
## 7 ASD 2.097222 269  
## 8 ASD 2.563025 555  
## 9 ASD 3.518072 490  
## 10 ASD 3.257143 479  
## # ... with 166 more rows

#Attempting with a variable included multiple times  
clean %>% filter(Diagnosis == "ASD") %>% select(Diagnosis, Diagnosis, CHI\_MLU, tokens\_CHI)

## # A tibble: 176 x 3  
## Diagnosis CHI\_MLU tokens\_CHI  
## <chr> <dbl> <int>  
## 1 ASD 2.276860 461  
## 2 ASD 3.453039 562  
## 3 ASD 3.119318 983  
## 4 ASD 4.302326 674  
## 5 ASD NA NA  
## 6 ASD 3.413502 698  
## 7 ASD 2.097222 269  
## 8 ASD 2.563025 555  
## 9 ASD 3.518072 490  
## 10 ASD 3.257143 479  
## # ... with 166 more rows

#Nothing happens. It's still only included once.

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 a column in the data set containing the mean amount of words produced by each trial (CHILD????) 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?

#Adding a column with mean words spoken across visits for the individual child  
clean$ID <- as.character(clean$ID)  
clean\_means <- clean %>% mutate(mean\_overall = mean(tokens\_CHI, na.rm = T))  
  
#Adding a column with mean words spoken across visits for the individual child  
clean\_means <- clean\_means %>% group\_by(ID) %>% mutate(mean\_tokens = mean(tokens\_CHI, na.rm = T))  
  
#3: Some of the other variables might be relevant to the interpretation of language production - for example, if the MLU of the parent is also low, this might help determine some of the predisposistions of the individual child. All the demographic data is also relevant in the same sense. In general one would lose a lot of valuable information by only referring to the means, especially when it comes to interpreting and understanding observed differences.