A1_P1_Student

Céline Carlsen

29/8/2017

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 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, 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-S

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 (including a working directory where you will save all your data and code for these assignments)

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

Set the working directory (the directory with your data and code for these assignments):

```
setwd("~/Desktop/3. Semester/Experimental methods 3/Portfolio 1/Portfolio 1")
```

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

```
Demo = read.csv("demo_train(1).csv")
LU = read.csv("LU_train.csv")
Token = read.csv("token_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 data.table, or google "how to rename variables in R"

```
library(data.table)
```

```
## Warning: package 'data.table' was built under R version 3.3.2
```

```
library(plyr)

Demo = plyr::rename(Demo, c("Child.ID" = "ID"))
LU = plyr::rename(LU, c("SUBJ" = "ID"))
Token = plyr::rename(Token, c("SUBJ" = "ID"))

Demo = plyr::rename(Demo, c("Visit" = "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" in stead 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 you also in future 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. Tip: type ?str_extract() after loading the library, for examples of how to use it.

```
library(stringr)
```

```
## Warning: package 'stringr' was built under R version 3.3.2
```

```
LU$VISIT = str_extract(LU$VISIT, "\\d")
Token$VISIT = str_extract(Token$VISIT, "\\d")
```

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. To do these a subfunction of apply(), called sapply() can be used.

Tip: Take a look into the gsub() function. Tip: A possible solution has one line of code for each child name that is to be changed. Another combines mutate() and recode()

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.

```
LU$ID=gsub('[[:punct:] ]+','',LU$ID)
Token$ID=gsub('[[:punct:] ]+','',Token$ID)
Demo$ID=gsub('[[:punct:] ]+','',Demo$ID)
```

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, which contain 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 worse)
- MullenRaw indicates non verbal IQ
- ExpressiveLangRaw indicates verbal IQ
- 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)

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

```
library(tidyverse)
```

```
## Warning: package 'tidyverse' was built under R version 3.3.2
```

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

```
## Warning: package 'ggplot2' was built under R version 3.3.2
```

```
## Warning: package 'readr' was built under R version 3.3.2
```

```
## Warning: package 'purrr' was built under R version 3.3.2
```

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

```
## arrange():   dplyr, plyr
## between():   dplyr, data.table
## compact():   purrr, plyr
## count():     dplyr, plyr
## failwith():  dplyr, plyr
## filter():    dplyr, stats
## first():     dplyr, data.table
## id():        dplyr, plyr
## lag():       dplyr, stats
## last():      dplyr, data.table
## mutate():    dplyr, plyr
## rename():    dplyr, plyr
## summarise(): dplyr, plyr
## summarize(): dplyr, plyr
## transpose(): purrr, data.table
```

```
DataDemo=select(Demo, ID, VISIT, Ethnicity, Diagnosis, Gender, Age, ADOS, MullenRaw, ExpressiveLangRaw)
DataLU=select(LU, ID, VISIT, MOT_MLU, MOT_LUstd, CHI_MLU, CHI_LUstd)
DataToken=select(Token, ID, VISIT, types_MOT, types_CHI, tokens_MOT, tokens_CHI)

DataDemo = plyr::rename(DataDemo, c("ExpressiveLangRaw" = "verbalIQ"))
DataDemo = plyr::rename(DataDemo, c("MullenRaw" = "nonVerbalIQ"))
```

2e. Finally we are ready to merge all the data sets into just one. Google "How to merge datasets in R" Tip: Use the merge() function for this. Tip: Merge only works for two data frames at the time. Tip: Check the number of observations in the datasets before and after merging. What is going on?

```
#Merge data
DATA1=merge(DataDemo, DataLU)
DATA=merge(DATA1, DataToken)

# Reason for less datapoints is due to a larger number og demographic data than in the other datasets.
```

Are we done yet?

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.

Tip: one solution requires you to select only the rows corresponding to visit 1 in a new dataset, to rename the columns of the relevant variables and to merge it back to the old dataset. Tip: subset() and select() might be useful. Tip: the final dataset should have as many rows as the the old one.

```
#Only data from visit 1
Visit1Data=subset(DATA, DATA$VISIT=="1")

#Select relevant columns
ID_visit=select(Visit1Data, ID, ADOS, nonVerbalIQ, verbalIQ)

#Remove old data from relevant columns
NewData1=DATA[-7:-9]

#Insert the new data from the relevant columns
NewData=merge(NewData1, ID_visit, by = "ID")
```

Now, we are almost ready to actually start working with 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: as.numeric() might be a useful function, but not alone.
- Note that visit is (probably) not defined as numeric. Turn it into a numeric variable
- 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: Google "how to rename levels in R".

```
#Visit is already numeric

#Making ID anonymous
NewData$ID=as.factor(NewData$ID)
NewData$ID=as.numeric(NewData$ID)

#Change gender variable
NewData$Gender=as.factor(NewData$Gender)
NewData$Gender=revalue(NewData$Gender, c("1"="M", "2"="F"))

#Change diagnosis variable
NewData$Diagnosis=revalue(NewData$Diagnosis, c("A"="ASD", "B"="TD"))
```

Save the data set using into a csv file. Hint: look into write.csv()

```
write.csv(NewData, "clean_data.csv", sep=",")
```

```
## Warning in write.csv(NewData, "clean_data.csv", sep = ","): attempt to set
## 'sep' ignored
```

3) 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

1) 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

```
#Load clean data
CleanData=read_csv("clean_data.csv")
```

```
## Warning: Missing column names filled in: 'X1' [1]
```

```
## Parsed with column specification:
## cols(
##   X1 = col_integer(),
##   ID = col_integer(),
##   VISIT = col_integer(),
##   Ethnicity = col_character(),
##   Diagnosis = col_character(),
##   Gender = col_character(),
##   Age = col_double(),
##   MOT_MLU = col_double(),
##   MOT_LUstd = col_double(),
##   CHI_MLU = col_double(),
##   CHI_LUstd = col_double(),
##   types_MOT = col_integer(),
##   types_CHI = col_integer(),
##   tokens_MOT = col_integer(),
##   tokens_CHI = col_integer(),
##   ADOS = col_integer(),
##   nonVerbalIQ = col_integer(),
##   verbalIQ = col_integer()
## )
```

```
CleanData=CleanData[-1]

#Mean length of utterance of more than 2.7 morphemes
filter(CleanData, CHI_MLU > 2.7)
```

```
## # A tibble: 97 × 17
##       ID VISIT    Ethnicity Diagnosis Gender   Age  MOT_MLU MOT_LUstd
##    <int> <int>        <chr>     <chr>  <chr> <dbl>    <dbl>     <dbl>
## 1      1     5        White        TD      M 35.90 5.209615  2.814165
## 2      1     6        White        TD      M 40.13 4.664013  2.765261
## 3      2     2        White       ASD      M 33.17 4.964664  2.499220
## 4      2     3        White       ASD      M 37.07 4.147059  2.803222
## 5      2     4        White       ASD      M 41.07 5.309804  2.842621
## 6      2     6        White       ASD      M 49.70 4.588477  2.783585
## 7      3     4        White        TD      F 35.53 5.301053  2.912026
## 8      3     5        White        TD      F 39.47 4.566038  2.792687
## 9      3     6        White        TD      F 45.07 5.229885  3.014147
## 10     4     3 White/Latino       ASD      M 38.90 3.818681  2.420551
## # ... with 87 more rows, and 9 more variables: CHI_MLU <dbl>,
## #   CHI_LUstd <dbl>, types_MOT <int>, types_CHI <int>, tokens_MOT <int>,
## #   tokens_CHI <int>, ADOS <int>, nonVerbalIQ <int>, verbalIQ <int>
```

```
#Mean length of utterance of less than 1.5 morphemes at first visit
filter(CleanData, VISIT == 1, CHI_MLU < 1.5)
```

```
## # A tibble: 47 × 17
##       ID VISIT   Ethnicity Diagnosis Gender   Age  MOT_MLU MOT_LUstd
##    <int> <int>       <chr>     <chr>  <chr> <dbl>    <dbl>     <dbl>
## 1      1     1       White        TD      M 19.80 3.621993  2.164553
## 2      5     1       White       ASD      M 34.03 3.986357  2.500713
## 3      6     1 Bangladeshi       ASD      F 26.17 2.618729  1.935874
```

```
## 4      7     1       White       ASD      F 41.00 2.244755  1.510878
## 5      9     1       White        TD      F 18.30 3.544419  2.272387
## 6     10     1       White        TD      M 19.27 4.204846  2.384767
## 7     11     1       White        TD      M 19.23 3.380463  2.214518
## 8     12     1       White        TD      M 20.07 4.195335  2.280551
## 9     14     1       White        TD      M 18.97 3.420315  2.273399
## 10    15     1       White        TD      M 19.27 3.967078  2.302921
## # ... with 37 more rows, and 9 more variables: CHI_MLU <dbl>,
## #   CHI_LUstd <dbl>, types_MOT <int>, types_CHI <int>, tokens_MOT <int>,
## #   tokens_CHI <int>, ADOS <int>, nonVerbalIQ <int>, verbalIQ <int>
```

```
#All kids who have not completed all trials

missed = CleanData %>% group_by(ID) %>% tally()
filter(missed, n < 6)
```

```
## # A tibble: 13 × 2
##       ID     n
##    <int> <int>
## 1      2     5
## 2      7     5
## 3      8     5
## 4      9     5
## 5     17     5
## 6     26     5
## 7     38     5
## 8     40     5
## 9     44     5
## 10    45     5
## 11    48     5
## 12    55     5
## 13    56     4
```

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.

```
#Sorting kids to find the one that produced most words on the 6th visit
Visit_6=subset(CleanData, CleanData$VISIT==6)
arrange(Visit_6, c(Visit_6$tokens_CHI))
```

```
## # A tibble: 56 × 17
##       ID VISIT        Ethnicity Diagnosis Gender   Age  MOT_MLU MOT_LUstd
##    <int> <int>            <chr>     <chr>  <chr> <dbl>    <dbl>     <dbl>
## 1     21     6 African American       ASD      M 48.97 3.943636  2.819817
## 2     50     6            White       ASD      M 62.40 3.886842  2.583416
## 3     56     6            White       ASD      M    NA 3.474725  2.095238
## 4     17     6            White       ASD      M 54.43 3.650235  2.613115
## 5     47     6            White        TD      F 40.37 4.676101  2.442830
## 6     57     6            White       ASD      F 61.70 3.241422  2.096168
## 7     33     6 African American       ASD      F 46.17 3.965392  2.431706
```

```
## 8     32     6            White       ASD      M 54.63 4.003257  2.537020
## 9     39     6      White/Asian       ASD      M 53.63 4.100707  2.162669
## 10    46     6            White       ASD      M 57.43 3.460548  2.287079
## # ... with 46 more rows, and 9 more variables: CHI_MLU <dbl>,
## #   CHI_LUstd <dbl>, types_MOT <int>, types_CHI <int>, tokens_MOT <int>,
## #   tokens_CHI <int>, ADOS <int>, nonVerbalIQ <int>, verbalIQ <int>
```

```
#ID 55 produced most words

#Sorting kids to find the one that produced least words on the 1st visit
Visit_1=subset(CleanData, CleanData$VISIT==1)
arrange(Visit_1, c(Visit_1$tokens_CHI))
```

```
## # A tibble: 61 × 17
##       ID VISIT        Ethnicity Diagnosis Gender   Age  MOT_MLU MOT_LUstd
##    <int> <int>            <chr>     <chr>  <chr> <dbl>    <dbl>     <dbl>
## 1     57     1            White       ASD      F 41.07 3.833770  2.417727
## 2     32     1            White       ASD      M 34.27 3.686347  2.650865
## 3     36     1            White        TD      M 19.20 3.921109  2.376179
## 4     30     1            White       ASD      M 36.53 3.607088  2.340376
## 5     24     1            White       ASD      M 37.47 2.917355  2.073392
## 6      6     1      Bangladeshi       ASD      F 26.17 2.618729  1.935874
## 7     33     1 African American       ASD      F 25.33 2.287293  1.928359
## 8     17     1            White       ASD      M 34.80 3.182390  2.269630
## 9     51     1            Asian        TD      F 20.87 3.435743  2.257715
## 10    48     1            White       ASD      M 31.63 3.765528  2.329794
## # ... with 51 more rows, and 9 more variables: CHI_MLU <dbl>,
## #   CHI_LUstd <dbl>, types_MOT <int>, types_CHI <int>, tokens_MOT <int>,
## #   tokens_CHI <int>, ADOS <int>, nonVerbalIQ <int>, verbalIQ <int>
```

```
#ID 57 produced least words
```

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?

```
#Making a subset only including children with ASD
ASD_Data1=subset(CleanData, CleanData$Diagnosis=="ASD")
ASD_Data=select(ASD_Data1, ID, VISIT, CHI_MLU, tokens_CHI)

#Including a variable name multiple times
ASD_Data2=select(ASD_Data1, ID, ID, VISIT, CHI_MLU, tokens_CHI)
#Nothing happens. The variable only appears 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 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?

```r
#Adding a column to the data set that represents mean number of words spoken during all visits
CleanData = CleanData %>% group_by(VISIT) %>% mutate(MeanWordVisit = sum(tokens_CHI)/length(tokens_CHI))

#Adding a column in the data set containing the mean amount of words produced by each trial across all

CleanData = CleanData %>% group_by(ID) %>% mutate(MeanWordChild = sum(tokens_CHI)/length(tokens_CHI))

#3
#If you only use the average value, you miss out on the personal development that each child undergoes.
```