

Linear Interpolation

Zarni Htet

March 15, 2018

Introduction

The goal of this code file is to generate an imputed data set of **BMI** and **Media Exposure** for young children from low income families. The existing raw data set has been linearly interpolated to fill in the missing values for **BMI** and **Media Exposure** at missing time points. A sample of the raw data set along with the existing issue and one proposed method for resolution is demonstrated in the objective section below.

Datasets Involved

Refer to the Belle Lab Power Point to add this portion

Administration

The project is supervised by Professor Marc Scott and Professor Daphna Harel. The data is from the Belle Lab at the Bellevue Hospital. Additional background on the project is in the *README* at the root directory of the Github repository associated with the project.

R Libraries

This block has all the *required* libraries for this code file.

```
#For the dta raw files
library(foreign)
#For importing different types of data set without specification
library(rio)
#For processing long form data
library(dplyr)
#GTools library for ordering numeric variables
library(gtools)
#For filling NA values
library(tidyr)
#Loading Rmarkdown library for rendering
library(rmarkdown)
#knitr library for rendering
library(knitr)
#for missing data
library(mi)
#Sourcing the code file
source("../zarni/01b_Function_LInterpolation.R")
```

I: Uploading Raw data

In this section, the raw data is processed to a .csv format. The data files will then be saved to a processing data folder to protect the integrity of the raw data.

```
#The BMI data extract
bmi <- read.dta("../data/raw/MASextract1.dta")
#The Media data extract
media <- read.dta("../data/raw/MASextract2.dta")
#Writing the BMI data to processing file
write.csv(bmi, "../data/processing/bmi.csv")
#Writing the media data to processing file
write.csv(media, "../data/processing/media.csv")
```

This code chunk is reloading and doing minor cleaning to the working version of the data to be used throughout the rest of the code file.

Note to self Insert a custom function to remove the V1 from the data set.

Objective

The overarching goal of the project is to assess whether infant media exposure is associated with weight/bmi trajectories during their infant to early childhood periods. In order to examine that association, media exposure data and weight/bmi data need to be synchronized (exactly how the response will be modeled as a function of the predictors is to be determined, but a first step is examining aligned data). As can be seen below in our short data snippet, media exposure and weight/bmi are collected at **asynchronous** time points. Our goal is to impute these missing data at missing time points via **linear interpolation**. For more on **linear interpolation**, please see the *Appendix* section.

	##	ID_	AgeMos	zBMI	lnmediatimespent	sqrtmediatimespent
## 7	1	7.786448	NA	5.463832	15.32971	
## 8	1	12.813142	1.011972	NA	NA	
## 9	1	15.244353	NA	4.948760	11.83216	

II: Data Exploration

This section examines the two datasets. The focus here is to explore the distribution of BMI and Media Exposure as well as cases of missing data and distribution of time points for each data set.

The BMI data set overview

	##	V1	ID_	AgeMos	zBMI
## 1	1	1	0.0000000	-3.5407891	
## 2	2	1	0.1314168	-3.1878707	
## 3	3	1	0.5585216	-0.2831618	
## 4	4	1	1.5441478	-1.2716171	
## 5	5	1	4.3039017	-1.1837007	
## 6	6	1	6.3737168	-2.5585830	
## 7	7	1	12.8131418	1.0119723	
## 8	8	2	0.0000000	1.2820979	
## 9	9	2	1.5441478	2.6198270	
## 10	10	2	2.5297742	1.0224092	
## 11	11	2	6.7351131	1.0519278	
## 12	12	2	10.8418894	1.9389179	
## 13	13	2	11.7289524	0.9665915	
## 14	14	2	26.4147835	1.7445436	
## 15	15	2	38.6365509	2.4713204	

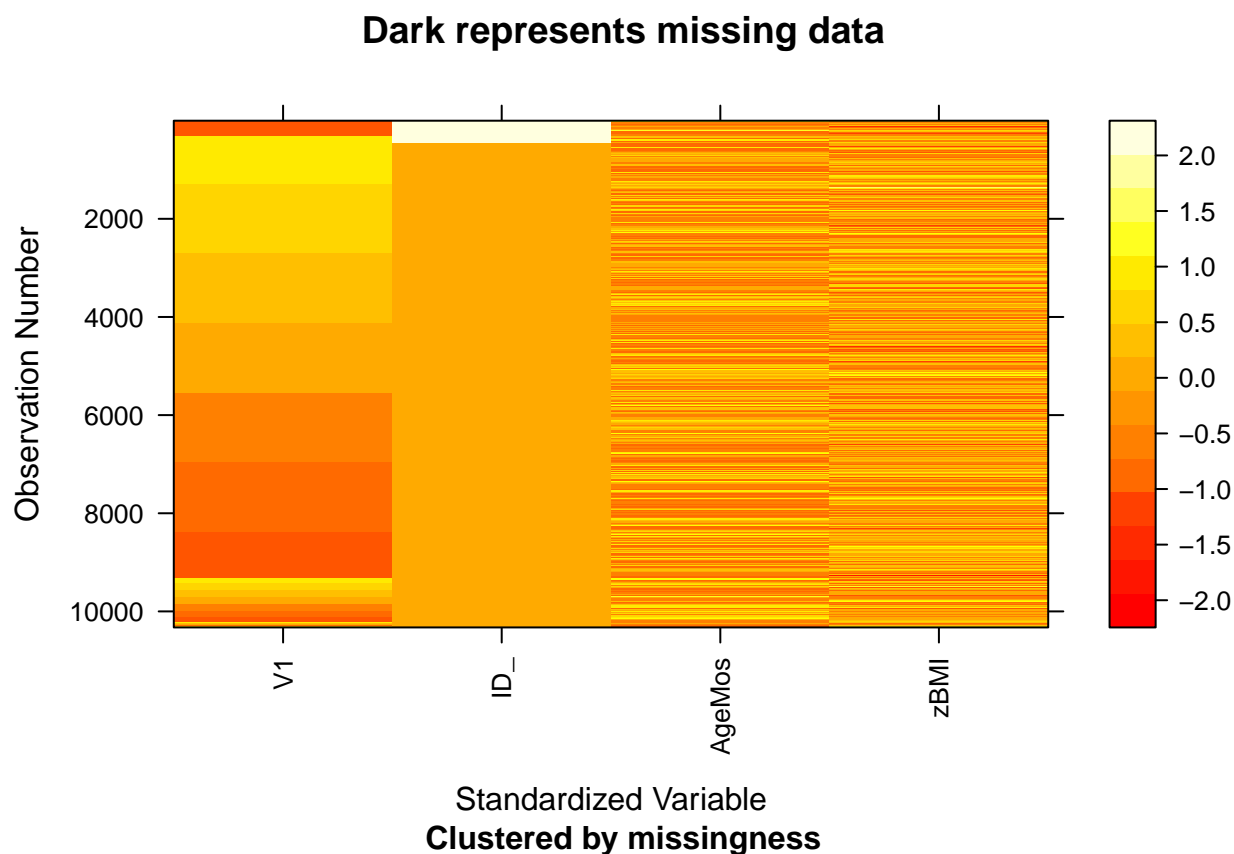
Each subject has different time points. For subject 1, months may be 0, 0.13 , 0.55 while subject 2 has months in 0, 1.5, 2.5 etc.

Missing data exploration

As can be seen from the visuals below, the BMI dataset by itself before joining across subjectID and time with the Media Exposure dataset has no missing values.

```
mdf_bmi = missing_data.frame(p_bmi)
```

```
image(mdf_bmi)
```

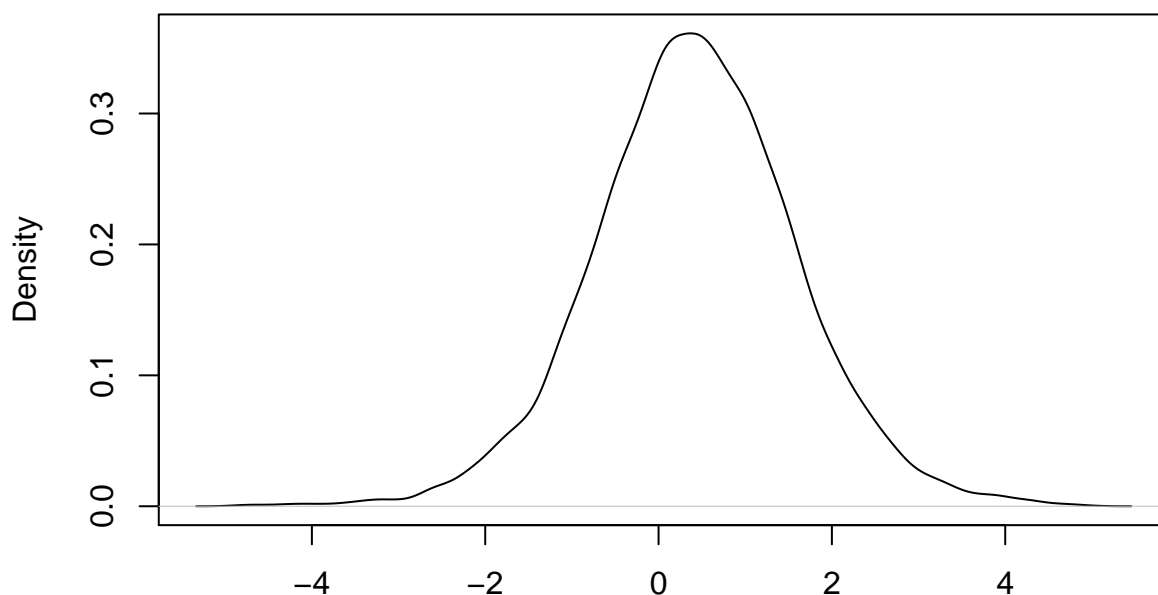


Distribution of BMI values

The BMI values are more or less normalized as can be seen below.

```
plot(density(p_bmi$zBMI), main = "Distribution of BMI Values")
```

Distribution of BMI Values

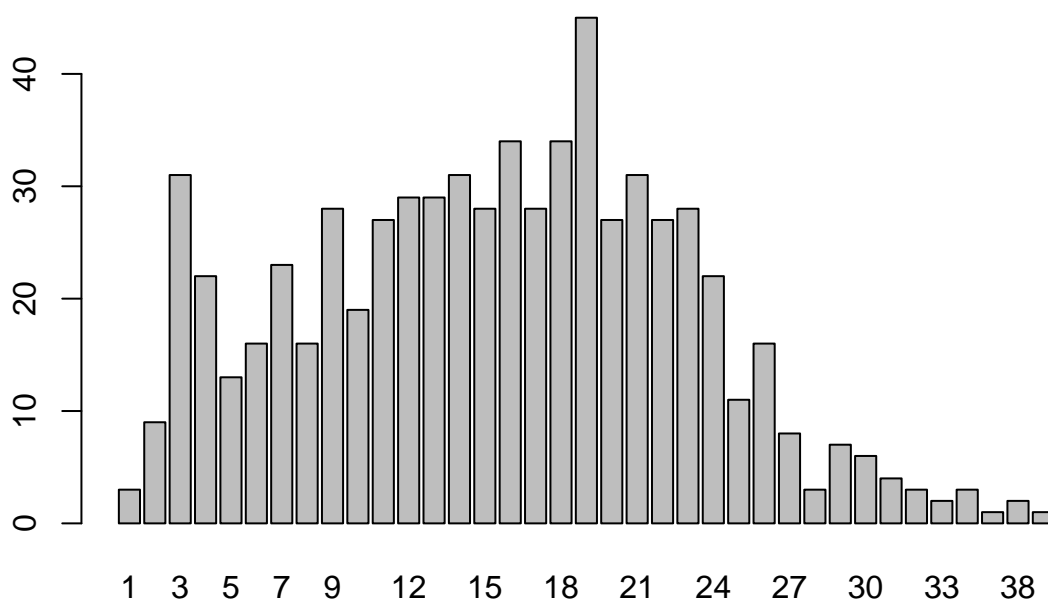


N = 10326 Bandwidth = 0.1572

Distribution of Number of Time Intervals by Subject Count for BMI

The number of time points counted over the number of subject ID within each count of time points can be seen below. The majority of the subjects appear to have between 11 and 23 time points.

Number of Time Points Distribution by Subject Count



The count of time points per subject ID is also calculated. As shown below, at least one subject ID has only 1 timepoint and 39 timepoints respectively. This insight is useful in that the built-in linear interpolation cannot impute data with only one timepoint. Therefore, those cases will have to be handled separately.

```
bmi_timed <- p_bmi %>%
  group_by(ID_) %>%
  summarize(n = n())
print(max(bmi_timed$n))
```

```
## [1] 39
```

```
print(min(bmi_timed$n))
```

```
## [1] 1
```

Media exposure data set overview

##	V1	ID_	AgeMos	lnmediatimespent	sqrtmediatimespent
## 1	2	1	7.786448	5.463832	15.329710
## 2	1	1	15.244353	4.948760	11.832160
## 3	5	2	6.735113	4.330733	8.660254
## 4	3	2	24.147844	4.795791	10.954452
## 5	4	2	42.940453	3.433987	5.477226
## 6	6	2	60.714581	4.795791	10.954452
## 7	7	3	9.494866	5.888878	18.973665
## 8	8	4	5.848049	3.433987	5.477226
## 9	9	4	31.605749	4.110874	7.745967
## 10	14	5	6.965092	4.510859	9.486833
## 11	10	5	17.314169	5.198497	13.416408
## 12	11	5	24.147844	5.602119	16.431677
## 13	13	5	42.381931	5.484797	15.491934
## 14	12	5	62.357288	4.510859	9.486833
## 15	17	6	6.866530	5.017280	12.247449

Like with the BMI data set before each Media data set subject has different time points. For subject 1, months are 7.786, 15.24 while subject 2 has months such as 6.73, 24.14 etc.

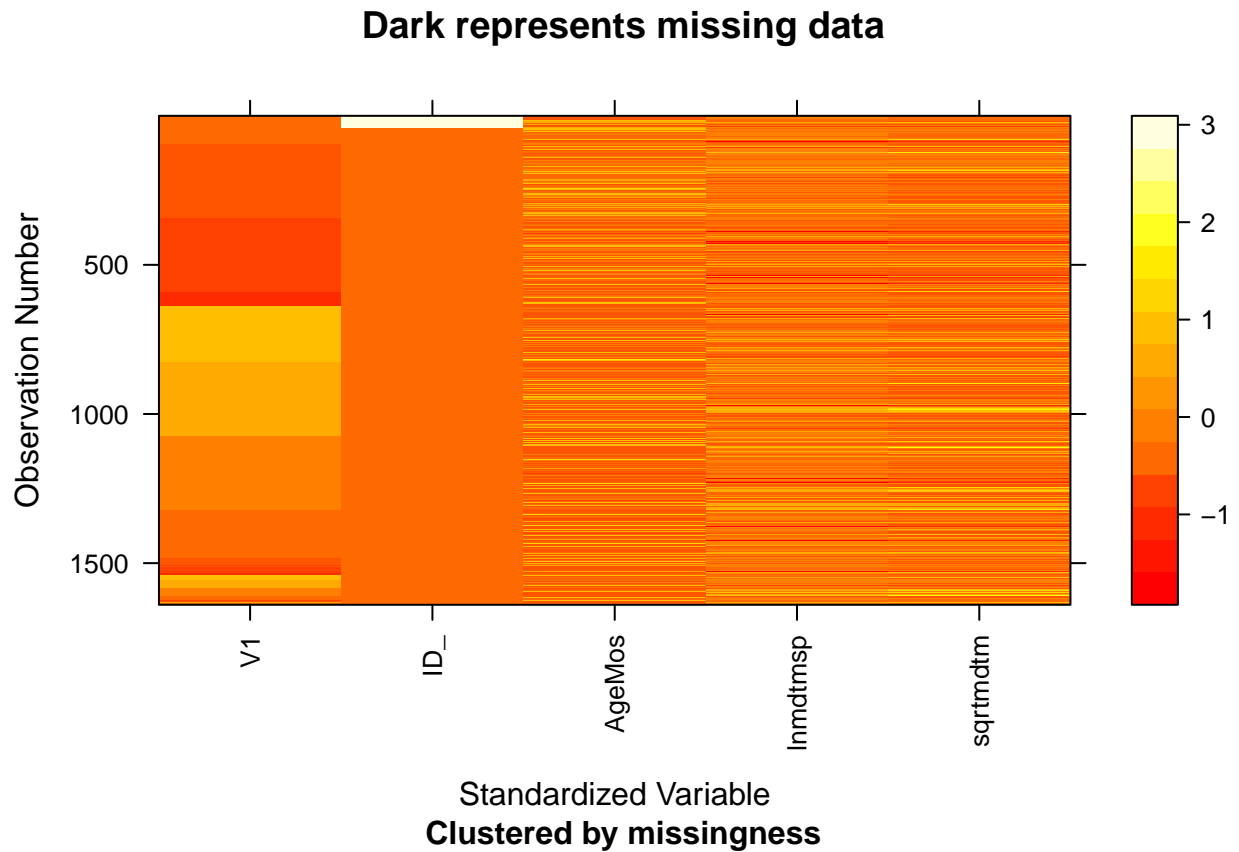
Missing data exploration

As can be seen from the visuals below, the Media dataset by itself before joining across subjectID and time with the BMI dataset has no missing values.

```
mdf_media = missing_data.frame(p_media)
```

```
## Warning in .local(.Object, ...): lnmediatimespent and sqrtmediatimespent have the same rank ordering
## Please verify whether they are in fact distinct variables.
```

```
image(mdf_media)
```

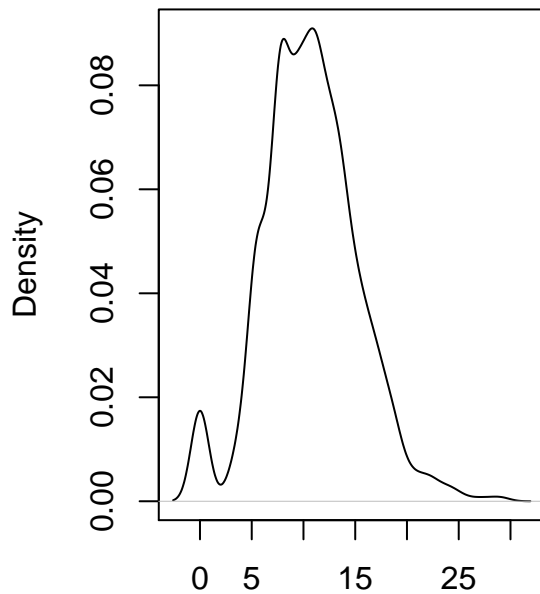


Distribution of the Media Variable

The Media exposure data set has two measures of media time spent for infants. One is log transformed and the other is square root transformed. A plot of both transformations is compared below.

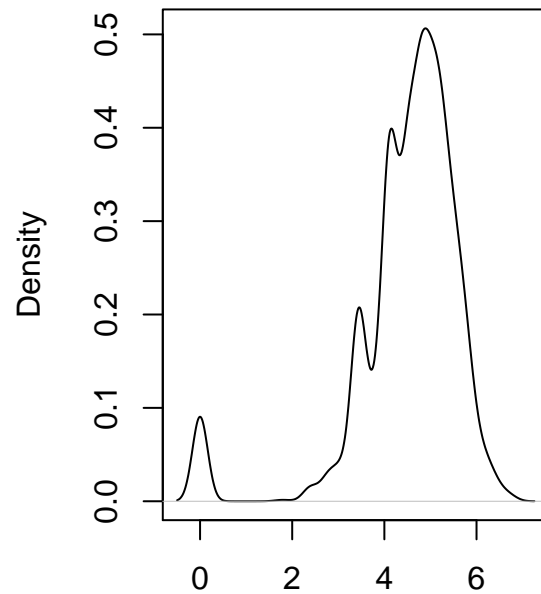
```
par(mfrow = c(1,2))
plot(density(p_media$sqrtmediatimespent), main = "Sqrt Media Time Spent")
plot(density(p_media$lnmediatimespent), main = "Log Media Time Spent")
```

Sqrt Media Time Spent



N = 1639 Bandwidth = 0.8666

Log Media Time Spent



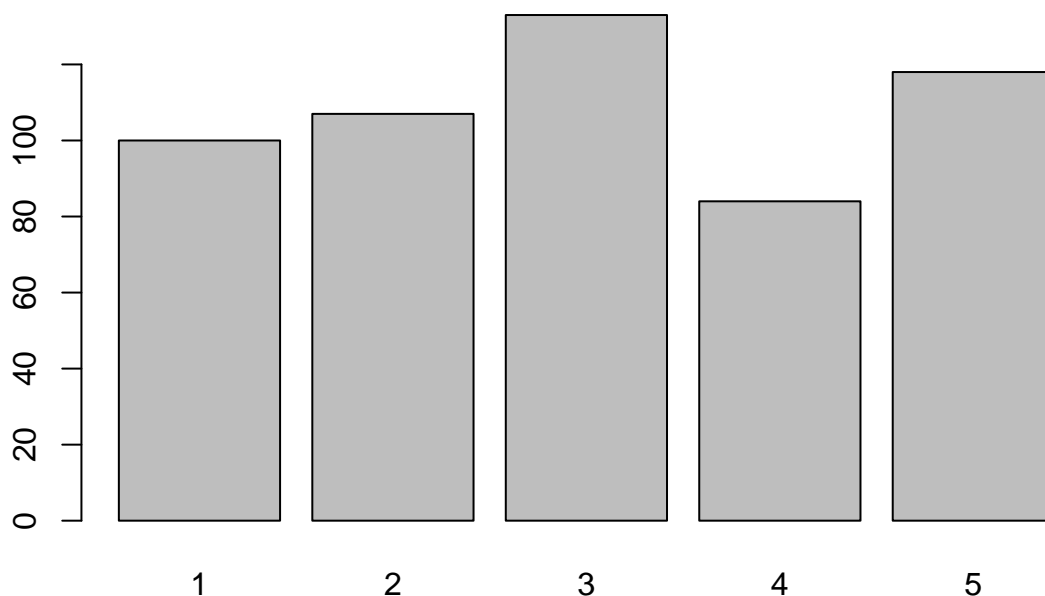
N = 1639 Bandwidth = 0.1662

The square root transformation looks more **normal** than log transformation. Subsequently, it is utilized in the rest of the code file.

Distribution of Number of Time Intervals by Subject Count for Media Exposure

```
#Using the table function and barplot to draw the distribution of time.  
media_tt = table(table(p_media$ID_))  
barplot(media_tt, main = "Number of Time Points Distribution by Subject Count")
```

Number of Time Points Distribution by Subject Count



The number of time points counted over the number of subject ID within each count of time points can be seen below. Unlike BMI data set, all the Media data set subjects have time points between **1** and **5** inclusive. It indicates that more time points are missing in Media data set compared to the BMI data set.

III: Interpolation Scenarios

This section lays out the different interpolation scenarios as the two data sets are joined to capture missing values in each data set. The goal as stated before is to impute these values using **linear interpolation**.

Scenario I:

In this case, there are missing BMI and Media Exposure values at varying time points. The built-in linear interpolation function (as described in the *Appendix*) can be applied to each value column of BMI and Media Exposure. The function will work smoothly for cases like these as there is at least **two** values for the respective missing columns. If there are less than **two** non-missing values in BMI or Media Exposure column, the built-in function will fail. Such cases are handled with a custom function as demonstrated later.

##	ID_	AgeMos	zBMI	lnmediatimespent	sqrtmediatimespent
## 6	1	6.373717	-2.558583	NA	NA
## 7	1	7.786448	NA	5.463832	15.32971
## 8	1	12.813142	1.011972	NA	NA
## 9	1	15.244353	NA	4.948760	11.83216

Scenario II:

In this case, one column value (zBMI) is completely filled across all the time points. The Media Exposure value on the other hand has only **one** value. In that case, we cannot utilize the built-in linear interpolation function. A custom function has been created to execute last value carried forward and backward on these cases. This type of scenario can happen similarly to zBMI where it only has **one** value while the Media Exposure column is filled. Additionally, it could be that one of the columns does not necessarily have to be filled completely. Rather that it has at least **two** values or more. Those cases would fall under this scenario as well.

##	ID_	AgeMos	zBMI	lnmediatimespent	sqrtmediatimespent
## 31	3	0.000000	-0.2047709	NA	NA
## 32	3	0.164271	-0.8374662	NA	NA
## 33	3	0.624230	1.2653127	NA	NA
## 34	3	1.577002	1.0108550	NA	NA
## 35	3	2.825462	1.4965336	NA	NA
## 36	3	3.745380	0.9784334	NA	NA
## 37	3	6.275154	0.2728231	NA	NA
## 38	3	9.494866	0.6977530	5.888878	18.97367
## 39	3	13.634497	1.8940200	NA	NA

Scenario III:

In this case, the subject ID only exists in one data set and does not exist in the other. The confirmation to that case is that all the Media Exposure values are NAs as shown below. As shown before, the Media data has no missing values by itself. Thus, this is a case of subject ID mismatch. There is no point in constructing a study where the subject altogether does not exist in one data set or the other. These cases will be taken care of through only matching common ids between the two data sets.

##	ID_	AgeMos	zBMI	lnmediatimespent	sqrtmediatimespent
## 622	39	0.000000	0.6400806	NA	NA
## 623	39	0.624230	0.2247941	NA	NA
## 624	39	1.314168	0.9656801	NA	NA
## 625	39	1.774127	1.4292920	NA	NA
## 626	39	4.106776	2.3997307	NA	NA


```
## 627 39 11.203285 2.7849188 NA NA
```

Scenario IV:

Scenario IV is where the subject ID from both BMI and Media data sets match. Additionally, they only have 1 time point each and that time point is exactly equal as well. In that case, there is **no** other time point to linearly interpolate. As a result such data rows must be temporarily removed before applying our linear interpolation functions. Fortunately, as can be seen below, there is only one subject ID of one time instance in both data sets that where the subject ID matches. However, the exact time instance does not match. Therefore, this scenario is avoided.

```
##      ID_   AgeMos      zBMI lnmediatimespent sqrtmediatimespent
## 10551 626   0.00000 -1.546457             NA             NA
## 10552 626  36.79671       NA       5.888878       18.97367
```

IV: Creating Linearly Interpolated Data Set

In this section, the end goal is to create a linearly interpolated data set while resolving scenario I to scenario III of interpolation from the above section. The section is subdivided into the following themes.

- Subsetting both data sets via **common** subject IDs between the two data sets. This action takes care of scenario III.
- Creating a combined from the two subsetting data sets above for interpolation.
- Applying custom last value carried forward and backward function to handle scenario II.
- Applying the built-in interpolation function to execute imputation for scenario I.

These themes are carried out in a total of **7** steps as demonstrated below.

Step 1: Finding the Common ID

Finding the subject IDs that match across both data sets

```
common_ID <- intersect(p_bmi$ID_, p_media$ID_) # intersection works like in Set theory
print(length(common_ID)) #537 common subject IDs
```

Step 2: Subsetting BMI and Media Data set based on shared ID

```
#Matched BMI
m_bmi <- p_bmi[(p_bmi$ID_ %in% common_ID), ]
#Matched Media
m_media <- p_media[(p_media$ID_ %in% common_ID),]
```

Step 3: Combining both data sets for create missing time points in each data set

```
#Merging both data sets by ID and Months
c_data <- m_bmi %>% full_join(m_media, by =c("ID_" = "ID_", "AgeMos" = "AgeMos"))
#Extracting needed columns esp sqrtmediatimespent as it is more normal than the log transformation
c_data <- c_data[,c("ID_", "AgeMos", "sqrtmediatimespent", "zBMI")]
#Renaming the variables
colnames(c_data) <- c("ID", "Months", "Media", "zBMI")

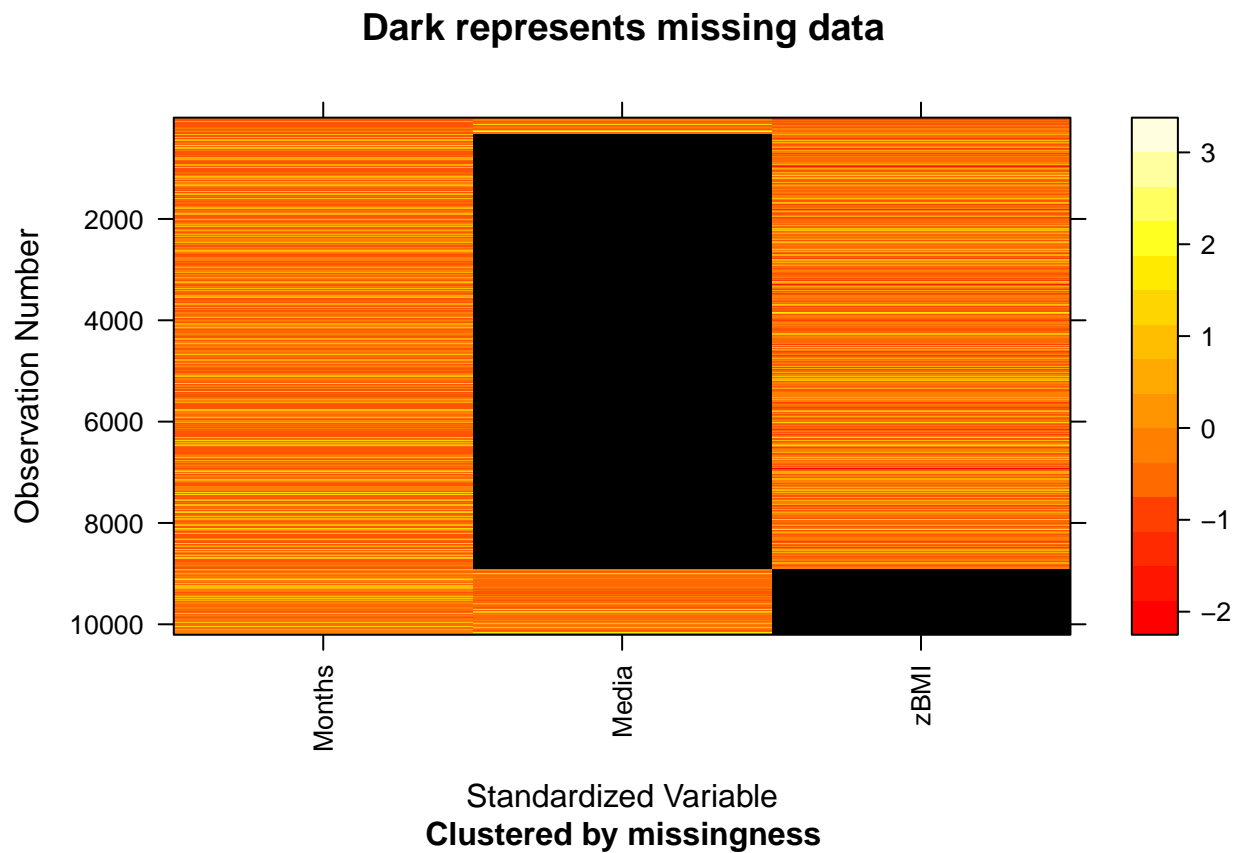
### Doing minor cleaning

#Converting Months into numeric for sorting later
c_data$Months <- as.numeric(as.character(c_data$Months))
#Converting the square root media value to square media for later linear interpolation
c_data$Media <- c_data$Media^2
#Rearrange the data by GroupID then within groups by Time
c_data_arr <- c_data %>% arrange(ID, Months)
```

Missing Values Creation Confirmation

As shown in the picture below, Media exposure missing values and BMI missing values at various time points are created. As to be expected from the data exploration before, there are more Media exposure missing values than that of BMI.

```
image(mdf_combined)
```



Step 4: Merging Duplicated Rows

This section checks if BMI and Media data has been recorded at the same time for the same subject ID. If those rows exist, the goal is to merge those same rows by the **average** values of both BMI and Media Exposure duplicated columns. A sample of what these duplicated rows looks like is show below.

```
#Group by gets the same ID and Month and summarise checks the count
dup_count <- c_data_arr %>% group_by(ID, Months) %>% summarise(n=n())
#Capture the duplicate cases
v_dup <- dup_count[dup_count$n >1,]
#Save an id of a duplicate case
id <- v_dup$ID[1]
#Spit out a duplicate case
print(c_data_arr[c_data_arr$ID == id,][3:6,])
```

```
##      ID  Months Media      zBMI
## 1213 70 1.215606   NA -0.2115094
## 1214 70 1.215606   NA -0.2115094
## 1215 70 1.839836   NA -0.0139306
## 1216 70 1.839836   NA -0.0139306
```

Custom Function applied to merge the duplicated rows

```
#Combined data that is arranged and merged.
c_data_arr_mer <- c_data_arr %>% group_by(ID,Months) %>% summarise_all(funs(my.rowmerge))

#Saving the arranged and merged data set
write.csv(c_data_arr_mer, ".././data/final/final_na_data.csv")
```

Step 5: Handling data Scenario II from Section III

As mentioned before, there are cases where there is only 1 single value of either BMI or Media Exposure data with the rest of the time points' values being missing. The built-in linear interpolation function cannot handle it. Therefore, last value carried forward/backward will be carried out with our own custom functions.

Separate the data of those single time instance values and not

```
#Getting all the rows with single value IDs
singleton_data <- c_data_arr_mer [(c_data_arr_mer$ID %in% all_singletons$ID_),]
#Getting all the rows with non single value IDs
non_singleton_data <- c_data_arr_mer[!(c_data_arr_mer$ID %in% all_singletons$ID_),]
```

Applying custom function for last value carried forward/backward

- Rewrite the custom function to only take in a single vector at a time
- lapply would work for the vector too. If not, check supply.

QUESTIONS HERE!

```
#Splitting the data frame by the subject ID
singleton_data_split <- split(singleton_data, singleton_data[,1])
#Pulling the BMI column out. I am hardcoding the column?
bmi <- lapply(singleton_data_split, function(x) x$zBMI)
#Plug it fill_NA (custom function) in the place of mean
bmi_fill_NA <- lapply(bmi, function(x) mean(x))
```

Old code

```
# Split by each groupID
# singleton_data_split <- split(singleton_data, singleton_data[,1])
# Apply NA fixes to each of the data splits
# singleton_NA_filled <- lapply(singleton_data_split, fill_NA)
# Collapse the Split Data into a single data frame
# singleton_NA_filled <- bind_rows(singleton_NA_filled)
# write.csv(singleton_NA_filled, ".././data/Intermediate/singleton_NA_filled.csv")
```