01_Linear_Interpolation

Zarni Htet (zh938@nyu.edu)

Linear Interpolation

This Markdown is filling the missing data for BMI and media exposure at asynchronous time points using linear interpolation. The project is supervised by Professor Marc Scott and Professor Daphna Harel. The data is from the Belle Lab at the Bellevue Hospital. More details of the project scope in the repository under lit folder.

R Libraries

This code block has all the needed R libraries

```r
#For the dta raw files
library(foreign)
```

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

```r
#For importing different types of data set without specification
library(rio)
```

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

```r
#For processing long form data
library(dplyr)
```

```
##
## Attaching package: 'dplyr'
```

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

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

Uploading Raw data

In this code chunk, we are uploading raw .dta data and converting to it a csv. This will then be saved to a processing data folder to protect the integrity of the raw data.

```r
#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")
```

Loading the Data back from Processing Folder

This code chunk is loading the working version of the data extra to be used throughout the document.

```
#processing bmi data
p_bmi <- import("../../data/processing/bmi.csv")
p_media <- import("../../data/processing/media.csv")
```

Data Exploration

This code chunks examine the two data sets. In particular, the focus here is on the key variables and the time intervals they are recorded. At the end of each code block for each data set, there is a short summary of what the data consists of.

The BMI data set overview

```
head(p_bmi)
```

```
##   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
```

```
tail(p_bmi)
```

```
##          V1   ID_    AgeMos       zBMI
## 10321 10321 91008 0.9199179  1.4600797
## 10322 10322 91008 1.2813141  1.8749880
## 10323 10323 91118 0.0000000 -0.9925033
## 10324 10324 91118 0.5913758  1.3112072
## 10325 10325 91118 1.5112936  1.9073267
## 10326 10326 91118 3.9096510  2.6738663
```

```
dim(p_bmi) #10326, 4
```

```
## [1] 10326     4
```

```
#check the number of unique subjects
length(unique(p_bmi$ID_)) #667
```

```
## [1] 667
```

```
length(unique(p_bmi$AgeMos)) #1951
```

```
## [1] 1951
```

Each subject has different time points. For subject 1, months may be 0, 0.5, 1.0 while subject 2 has months in 0, 0.7, 1.2 etc.

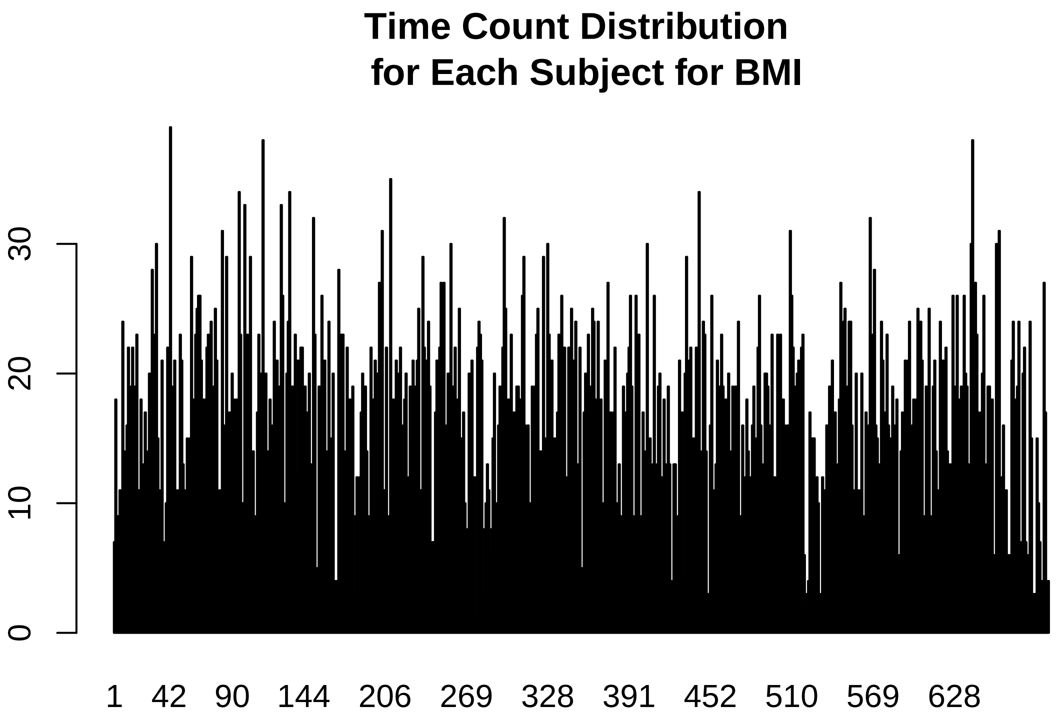
This is to explore the number of time intervals each subject has.

```
#This uses dplyr to group by each subject and count their instances. This effectively counts the number
bmi_timed <- p_bmi %>%
  group_by(ID_) %>%
  summarize(n = n())
print(bmi_timed)
```

```
## # A tibble: 667 × 2
##       ID_     n
##     <int> <int>
## 1       1     7
## 2       2    18
## 3       3     9
## 4       4     9
## 5       5    11
## 6       6     5
## 7       7    24
## 8       8    14
## 9       9    10
## 10     10    16
## # ... with 657 more rows
```

We will do a quick barplot to see the distribution of time points for each subject has

```
#Using the table function and barplot to draw the distribution of time.
barplot(table(bmi$ID_), main = "Time Count Distribution \n for Each Subject for BMI")
```



Check the Minimum/Maximum time intervals. This is to see if we have to explore edge cases later down the road for Last Value Carried forward at the end for each subject

```
min(bmi_timed$n) #1
```

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

```
max(bmi_timed$n) #39
```

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

At least 1 subject has only 1 time interval for BMI.

Media exposure data set overview

```r
head(p_media)
```

```
##   V1 ID_    AgeMos lnmediatimespent sqrtmediatimespent
## 1  1   1 15.244353         4.948760          11.832160
## 2  2   1  7.786448         5.463832          15.329710
## 3  3   2 24.147844         4.795791          10.954452
## 4  4   2 42.940453         3.433987           5.477226
## 5  5   2  6.735113         4.330733           8.660254
## 6  6   2 60.714581         4.795791          10.954452
```

```r
tail(p_media)
```

```
##        V1   ID_   AgeMos lnmediatimespent sqrtmediatimespent
## 1634 1634 90372 16.09856         5.602119          16.431677
## 1635 1635 90406 50.43121         5.198497          13.416408
## 1636 1636 90425 23.65503         3.433987           5.477226
## 1637 1637 90448 36.66530         4.510859           9.486833
## 1638 1638 90448 27.92608         4.110874           7.745967
## 1639 1639 90448 59.79466         5.198497          13.416408
```

```r
dim(p_media) #1639, 5
```

```
## [1] 1639    5
```

```r
#check the number of unique subjects
length(unique(p_media$ID_)) #542
```

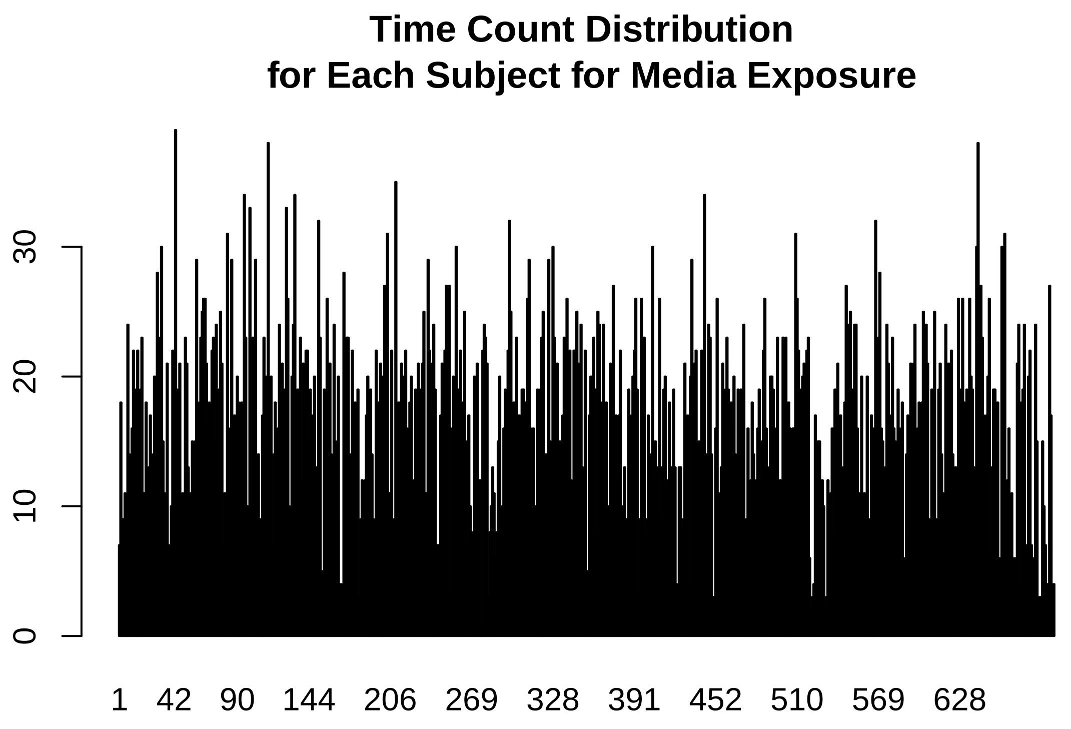
```
## [1] 542
```

```r
length(unique(p_media$AgeMos)) #745
```

```
## [1] 745
```

Like the BMI from before, each subject has different count of time as well as time intervals where the data is collected.

```r
#Using the table function and barplot to draw the distribution of time.
barplot(table(bmi$ID_), main = "Time Count Distribution \n for Each Subject for Media Exposure")
```



Next Steps?

Should all the X variables have similar time? Are we only matching it subject to subject but not acros subjects with the media exposure?

Merging the two data sets

Check the number of ID matches between the two files

The two ids to merge across the data sets are not of equal length. We will use the smaller one as base and see how much more are missing.

```
#Writing in the Media Exposure Smaller one first to fill in
matched_index <- match(p_media$ID_, p_bmi$ID_)
#Checking the number of non matches using is.na
non_matches <- sum(is.na(matched_index))
print(non_matches) #16
```

```
## [1] 16
```

```
#Checking total matches by substracting from maximum unique ID of smaller set to the missing ones
total_matches <- length(unique(p_media$ID_)) - non_matches
print(total_matches) #526
```

```
## [1] 526
```

```
#This is not perfect total number. The best way is to match it to match the IDs by innerjoin and captur
```

Join the two tables to fill in missing Xs and missing Ys for Each Subject

```
#Believe this will be use of InnerJoin of ID_s? Confirm!
#Goal is to have each column of X and Y filled with each of the missing counterparts.
```

Base Linear Interpolation Function

The linear interpolation equation to be used in the base function is below. The y_0 and y_1 would be either BMI or Media exposure variable. The x_0 and x_1 would be the time variable.

The y variable is the missing value we are looking for at time x. For BMI variable, the x corresponds to time from Media exposure that is missing between the x_0 and the x_1 intervals. The converse can be said of the Media Exposure variable to BMI as well.

Source: Linear Interpolation, Wikipedia

$$y = y_0 + (x - x_0)\frac{y_1 - y_0}{x_1 - x_0}$$

Use ApproxFun: https://stat.ethz.ch/R-manual/R-devel/library/stats/html/approxfun.html

OR

Base Function below.

```
#Should the base function be under a source file?
#Objective of Function is to take in y1,y0,x1,x0 with x value to spit out y.

#Question: if there are more than 1 x value between x1 and x0, we still use the same interval x1 and x0
```

Filling using Base Function above Function

```
#Parameters, X and Y values of each subject with missing NAs for the Y values

#Within function
## skip the first time point
## from the second time point and onwards
### if a missing NA is encountered for Y, go to the non-missing X and Y pair and the next one before.
### Calculate the time points in between.

#This could be much easily done if I use indexes of missing and non-missing.
##Have an index vector with the two X and Ys.
## Missing indexes can be two types
### It could be an index of count 1 and multiple
### For either case, pick the x0 and y0 and fill it up
```

Applying for Last Value Carried Forward Function

```
#Apply Last Value Carried Forward/Backward for the values
```

Applying to all the subjects function

Execution of the Functions

```
#All BMI subjects
```

```
#All Media Exposure subjects
```