Preparing Sleep and Steps Data for Analysis

Brandon Sepulvado

11/14/2018

# Getting the data in order

This file serves as a brief report on the sleep and steps trends for each dyad in the study. Data preparation will not be covered in this report; if you would like to examine the code, it may be found in the data\_import\_clean.R, create\_dyad\_trends.R, and explore\_data.R files as well as in my private netheath Gihub repo (which does not however contain any of the data).

# Exploring the data

I first load two data objects: dyad\_sleep\_diffs and dyad\_steps\_diffs. The code chunk below shows that both objects are lists with 390,000 elements, which in this ase are dyads (*not edges*).

# verify object classes  
class(dyad\_sleep\_diffs)

## [1] "list"

class(dyad\_steps\_diffs)

## [1] "list"

# number of elements in each  
length(dyad\_sleep\_diffs)

## [1] 390000

length(dyad\_steps\_diffs)

## [1] 390000

As I show below, each list element is a tibble/data frame with four variables (i.e., datadate, diff, dyad\_id, node1, and node2). The diff is the difference between the activity of interest, and the dyad\_id is simply the two node ids concatenated.

head(dyad\_sleep\_diffs[[1]], 10)

## # A tibble: 10 x 5  
## datadate diff dyad\_id node1 node2   
## <chr> <dbl> <chr> <chr> <chr>   
## 1 2015-07-28 177 3M5ZZL3LZJFK 3M5ZZL 3LZJFK  
## 2 2015-07-29 12 3M5ZZL3LZJFK 3M5ZZL 3LZJFK  
## 3 2015-07-30 49 3M5ZZL3LZJFK 3M5ZZL 3LZJFK  
## 4 2015-07-31 92 3M5ZZL3LZJFK 3M5ZZL 3LZJFK  
## 5 2015-08-01 62 3M5ZZL3LZJFK 3M5ZZL 3LZJFK  
## 6 2015-08-02 67 3M5ZZL3LZJFK 3M5ZZL 3LZJFK  
## 7 2015-08-03 14 3M5ZZL3LZJFK 3M5ZZL 3LZJFK  
## 8 2015-08-04 23 3M5ZZL3LZJFK 3M5ZZL 3LZJFK  
## 9 2015-08-05 22 3M5ZZL3LZJFK 3M5ZZL 3LZJFK  
## 10 2015-08-06 13 3M5ZZL3LZJFK 3M5ZZL 3LZJFK

# Missing data

Missing data can occur within each list element in two manners. First, the two individuals for a given dyad might not have been in the study on the same dates. Second, at least one of the two individuals might have no data for a given date. A list element will have no rows if the first possibility occurs, but, if the two individuals overlapped yet at least one has missing dat, the diff value for a given date will be NA.

The next bit of code shows how many participants overlapped.

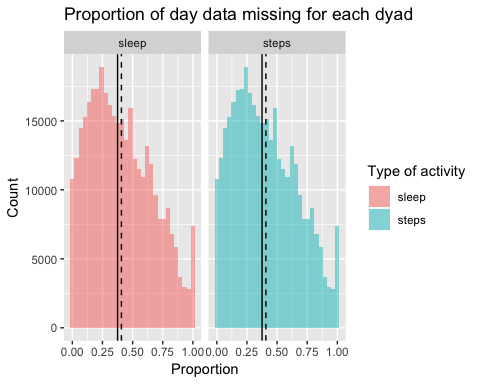
# count how many were in study at same time  
same\_time\_steps <- sum(sapply(dyad\_steps\_diffs,   
 function(x) nrow(x) > 0))  
# should be the same  
same\_time\_sleep <- sum(sapply(dyad\_sleep\_diffs,   
 function(x) nrow(x) > 0))  
same\_time\_sleep

## [1] 354116

same\_time\_steps

## [1] 354116

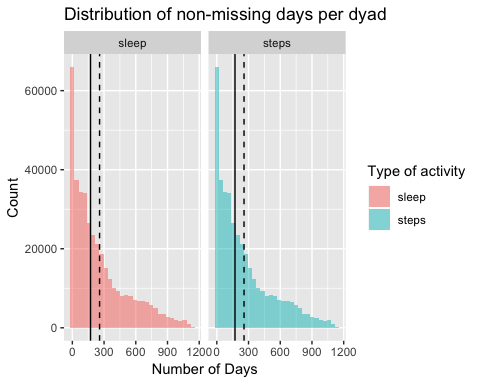
There are 354,116 dyads that participated in NetHealth at the same time. Now, for participants whose tenure overlapped, let’s look at how many have missing data and what proportion of their tenure that consistutes.



In these plots, the solid line represents the median proportion, and the dashed line represents the mean proportion. I now will plot the distribution of the raw number of days for which each dyad has data.

# get number of non\_missing days for dyads  
nonmissing\_steps\_days <- sapply(dyad\_steps\_diffs,   
 function(x) sum(!is.na(x['diff'])))  
nonmissing\_sleep\_days <- sapply(dyad\_sleep\_diffs,   
 function(x) sum(!is.na(x['diff'])))  
  
# make tibble to be able to facet histogram  
steps\_nonmiss\_t <- as\_tibble(nonmissing\_steps\_days) %>%   
 rename(n\_days = value) %>%   
 mutate(activity = "steps")  
sleep\_nonmiss\_t <- as\_tibble(nonmissing\_sleep\_days) %>%   
 rename(n\_days = value) %>%   
 mutate(activity = "sleep")  
nonmissing\_combined <- bind\_rows(steps\_nonmiss\_t,  
 sleep\_nonmiss\_t)  
  
# get mean for missing proportion types for histograms below  
mean\_steps <- mean(steps\_nonmiss\_t$n\_days, na.rm = TRUE)  
mean\_sleep <- mean(sleep\_nonmiss\_t$n\_days, na.rm = TRUE)  
  
# get median for missing proportion types   
# get mean for missing proportion types for histograms below  
median\_steps <- median(steps\_nonmiss\_t$n\_days, na.rm = TRUE)  
median\_sleep <- median(sleep\_nonmiss\_t$n\_days, na.rm = TRUE)  
  
# plot histogram faceted  
nonmissing\_combined %>%   
 ggplot(aes(x = n\_days, fill = as.factor(activity))) +  
 geom\_histogram(alpha = 0.5) +  
 facet\_wrap(~as.factor(activity)) +  
 labs(title = "Distribution of non-missing days per dyad",  
 x = "Number of Days",  
 y = "Count") +  
 scale\_fill\_discrete(name = "Type of activity") +  
 geom\_vline(data=filter(nonmissing\_combined, as.factor(activity)=="steps"),  
 aes(xintercept=mean\_steps), linetype = "dashed", colour="black") +  
 geom\_vline(data=filter(nonmissing\_combined, as.factor(activity)=="sleep"),  
 aes(xintercept=mean\_sleep), linetype = "dashed", colour="black") +  
 geom\_vline(data=filter(nonmissing\_combined, as.factor(activity)=="steps"),  
 aes(xintercept=median\_steps), linetype = "solid", colour="black") +  
 geom\_vline(data=filter(nonmissing\_combined, as.factor(activity)=="sleep"),  
 aes(xintercept=median\_sleep), linetype = "solid", colour="black")

## `stat\_bin()` using `bins = 30`. Pick better value with `binwidth`.

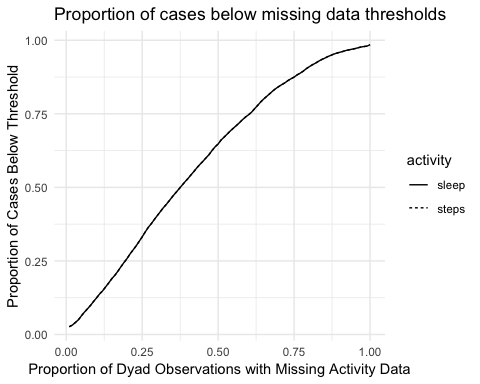


But, how *much* missing data for each dyad is there?

# number of cases at diff missing values thresholds  
get\_missing\_cases <- function(data, focal\_var, focal\_value, missing\_var, missing\_prop){  
 focal\_var <- enquo(focal\_var)  
 missing\_var <- enquo(missing\_var)  
   
 below\_prop <- data %>%   
 filter(!! focal\_var == focal\_value) %>%   
 filter(!! missing\_var < missing\_prop) %>%   
 nrow()  
   
 return(below\_prop)  
}  
  
# get these number for various thresholds  
thresholds <- tibble::tibble(threshold = seq(from = .01,   
 to = 1.0,   
 by = .01))  
  
# apply to steps  
num\_cases\_steps <- sapply(thresholds$threshold, function(x){  
 get\_missing\_cases(missing\_prop\_combined, activity, "steps", proportion, x)  
})  
  
# apply to sleep  
num\_cases\_sleep <- sapply(thresholds$threshold, function(x){  
 get\_missing\_cases(missing\_prop\_combined, activity, "sleep", proportion, x)  
})  
  
# join together  
num\_cases\_steps <- bind\_cols(thresholds, as\_tibble(num\_cases\_steps)) %>%   
 rename(n\_cases = value) %>%   
 mutate(activity = "steps")  
num\_cases\_sleep <- bind\_cols(thresholds, as\_tibble(num\_cases\_sleep)) %>%   
 rename(n\_cases = value) %>%   
 mutate(activity = "sleep")  
  
# merge together  
num\_cases <- bind\_rows(num\_cases\_steps, num\_cases\_sleep)  
  
# plot   
num\_cases %>%   
 ggplot(aes(x = threshold, y = n\_cases), color = activity) +  
 geom\_line(aes(linetype = activity)) +   
 ggtitle("Number of Cases Below Missing Data Thresholds") +  
 xlab("Missing Data Threshold") +  
 ylab("Number of Cases Below Threshold") +  
 theme\_minimal()



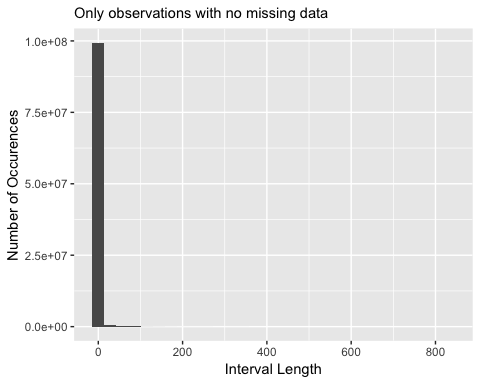
# get proportion  
num\_cases %>%   
 mutate(proportion = n\_cases / 354116) %>% # n cases in study at same time  
 ggplot(aes(x = threshold, y = proportion), color = activity) +  
 geom\_line(aes(linetype = activity)) +  
 ggtitle("Proportion of cases below missing data thresholds ") +  
 xlab("Proportion of Dyad Observations with Missing Activity Data") +  
 ylab("Proportion of Cases Below Threshold") +  
 theme\_minimal()



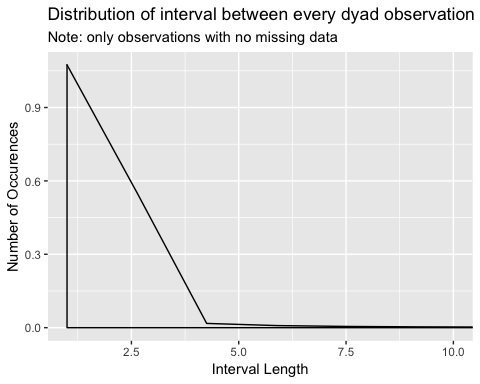
To be able to reliably impute or interpolate data, it is necessary to examine the length of the gaps between observed day observations for each dyad. For instance, it is easy to interpolate a single day between two observed days, but it is much less obvious what to do if an entire week of observations are missing for a given dyad. The following code does two things. First, I write a function to generate a vector of intervals between each non-missing observation on the specific activity variable; note that this vevtor will have a length of n - 1, where n is the number of observed data points, and that the interval of time between two points is returned in days. Second, it applies the function to the dyad\_sleep\_diffs and dyad\_steps\_diffs lists.

# function to get interval vector  
get\_date\_differences <- function(data, format = "day"){  
 #date\_var <- enquo(date\_var)  
   
 temp <- data %>%   
 filter(!is.na(diff)) %>%   
 select(datadate)  
   
 temp <- ymd(temp$datadate)  
   
 diffs <- temp %>%   
 int\_diff() %>%   
 as.duration() %>%  
 as.numeric()  
   
 if(format == "day") {  
 return(diffs / 86400)  
 } else if (format == "seconds") {  
 return(diffs)  
 } else {  
 stop("You have chosen an unsupported date format!")  
 }  
}  
  
###  
# this bit of code only needs to be run once  
# apply function to sleep data  
#date\_diffs\_sleep <- lapply(dyad\_sleep\_diffs, get\_date\_differences)  
#date\_diffs\_sleep <- parallel::mclapply(dyad\_sleep\_diffs,   
 #get\_date\_differences,  
 #mc.cores = getOption("mc.cores", 4L))  
  
# only run once  
#saveRDS(date\_diffs\_sleep, "date\_diffs\_sleep.rds")  
  
# apply function to steps data  
#date\_diffs\_steps <- lapply(dyad\_steps\_diffs, get\_date\_differences)  
  
# only run once  
#saveRDS(date\_diffs\_steps, "date\_diffs\_steps.rds")  
  
# collapse all \*sleep\* list elements into a single object  
###  
  
# load data objects created in commented out section  
date\_diffs\_sleep <- readRDS("date\_diffs\_sleep.rds")  
date\_diffs\_steps <- readRDS("date\_diffs\_steps.rds")  
  
# first make tibble  
date\_diffs\_sleep <- lapply(date\_diffs\_sleep, function(x){  
 x %>%   
 as\_tibble() %>%   
 rename(interval = value)  
})  
  
# next, merge all elements into a single object  
date\_diffs\_sleep\_t <- bind\_rows(date\_diffs\_sleep)  
  
# collapse all \*step\* list elements into a single object  
  
# first make tibble  
date\_diffs\_steps <- lapply(date\_diffs\_steps, function(x){  
 x %>%   
 as\_tibble() %>%   
 rename(interval = value)  
})  
  
# next, merge all elements into a single object  
date\_diffs\_steps\_t <- bind\_rows(date\_diffs\_steps)  
   
# histogram of sleep intervals  
date\_diffs\_sleep\_t %>%   
 ggplot(aes(x = interval)) +  
 geom\_histogram(show.legend = FALSE) +  
 labs(main = "Distribution of interval between every dyad observation",  
 subtitle= "Only observations with no missing data",  
 x = "Interval Length",  
 y = "Number of Occurences")

## `stat\_bin()` using `bins = 30`. Pick better value with `binwidth`.



# how many total intervals? use to get proportion   
# dim(date\_diffs\_sleep\_t)  
 # 100076296, 1  
  
# look at density, no longer visualizing 1 (but keeping in proportions)  
date\_diffs\_sleep\_t %>%   
 ggplot(aes(x = interval)) +  
 geom\_density(show.legend = FALSE) +  
 #coord\_cartesian(xlim=c(2, max(date\_diffs\_sleep\_t$interval))) +  
 #coord\_cartesian(xlim=c(2, 200)) +  
 coord\_cartesian(xlim=c(1, 10)) +  
 labs(title = "Distribution of interval between every dyad observation",  
 subtitle= "Note: only observations with no missing data",  
 x = "Interval Length",  
 y = "Number of Occurences")



# get proportion of cases with intervals from 2-4  
prop\_intervals <- sapply(seq(from = 2, to = 5, by = 1),  
 function(x){  
 date\_diffs\_sleep\_t %>%   
 filter(interval < x) %>%   
 summarise(n\_filtered = n(),  
 prop\_filtered = n\_filtered / 100076296,  
 days\_interval = x-1)  
 })  
  
# print   
prop\_intervals

## [,1] [,2] [,3] [,4]   
## n\_filtered 94863664 96921304 97765562 98225608   
## prop\_filtered 0.9479134 0.9684741 0.9769103 0.9815072  
## days\_interval 1 2 3 4

The visualizations indicate that the overwhelming majority of observations do not have large intervals (in terms of days) between their observations. The prop\_intervals object displays the number of cases kept, proportion of such cases, and the thresholds for each set of numbers. If we restrict the data to observations on consecutive days, then we will lose almost 6% of observations; if we increase the cutoff to four days with no data, we will lose almost 2% of cases. *These numbers of course do not indicate the number of dyads we will lose if we exclude them based upon whether or not they contain such periods of missing observations. Ideally, we will run this again after making a choice about the proportion of missing data threshold to verify that no systematic relationship between number of observations for a dyad and the length of time between observations occurs.*

## Periodization of Missing Observations

In this section, I want to explore if there are temporal trends in the missing data. I first will collapse the dyad\_steps\_diffs and dyad\_sleep\_diffs lists into two data frames.

dyad\_sleep\_diffs\_t <- bind\_rows(dyad\_sleep\_diffs)  
dyad\_steps\_diffs\_t <- bind\_rows(dyad\_sleep\_diffs)

Next, I will count the number of missing observations for each date and plot the resutls.

dyad\_sleep\_diffs\_t %>%   
 filter(is.na(diff)) %>%   
 count(datadate) %>%   
 mutate(datadate = ymd(datadate)) %>%   
 ggplot(aes(x = datadate, y = n)) +  
 geom\_line(show.legend = FALSE) +  
 #scale\_x\_date(limits = c(min(dyad\_sleep\_diffs\_t$datadate),  
 # max(dyad\_sleep\_diffs\_t$datadate))) +  
 scale\_x\_date(date\_labels = "%b %Y",  
 date\_breaks = "6 months",  
 date\_minor\_breaks = "1 month",  
 limits = c(as.Date(min(dyad\_sleep\_diffs\_t$datadate)),  
 as.Date(max(dyad\_sleep\_diffs\_t$datadate)))) +  
 labs(title = "Number of Missing Observations Throughout Study",  
 subtitle = "Activity: Sleep",  
 x = "Date",  
 y = "Number of Missing Observations") +  
 theme\_minimal()

 This line graph plots the number of missing dyadic observations for each day of the study (through September 2018). To reiterate, missing data in this context means that (1) two individuals overlapped in the dates of their study participation but (2) at least one of them has no recorded activity data for a given activity indicator. The x-axis is the date and has vertical lines for every month; the y-axis is simply the raw count of missing observations for a given study day.

There seem to be three distinct periods. In the first, which consists of the inital academic year, the number of missing observation increases fairly linearly with occasional spikes. In the second, which consists of around the end of August 2016 to the end of August 2017, there is a markedly different trend. One observes a rough linear increase—again with spikes—until the beginning of the Spring 2017 semester, when there is a desrease in the number of missing observations. One also observes during Summer 2017 a drastic increase in the number of missing observations. Finally, in the remaining period, there is a dramatic decrease in missing observations at the beginning of Fall 2017 and Spring 2018. There is a spike in missing data during the winter holidays between the two semesters. After the end of Spring 2017, one sees a decrease in missing observations.

Before interpreting these trends, I will create the same visualization for the steps activity variable. The results should be virtually identical.

dyad\_steps\_diffs\_t %>%   
 filter(is.na(diff)) %>%   
 count(datadate) %>%   
 mutate(datadate = ymd(datadate)) %>%   
 ggplot(aes(x = datadate, y = n)) +  
 geom\_line(show.legend = FALSE) +  
 scale\_x\_date(date\_labels = "%b %Y",  
 date\_breaks = "6 months",  
 date\_minor\_breaks = "1 month",  
 limits = c(as.Date(min(dyad\_sleep\_diffs\_t$datadate)),  
 as.Date(max(dyad\_sleep\_diffs\_t$datadate)))) +  
 labs(title = "Number of Missing Observations Throughout Study",  
 subtitle = "Activity: Steps",  
 x = "Date",  
 y = "Number of Missing Observations") +  
 theme\_minimal()

 As expected, the trends resemble each other.

Why are these trends in missing data observed? The steady increase during the first academic year suggests a growing inattention to wearing one’s fitbit. The second period seems to be characterized by a cohort who maintain a somewhat steady level of fitbit adherence. One observes in the final academic year much less variation in the number of missing observations.

What does this mean for analyses? The first factor to check will be the number of participants within the study at any given time. Comparing this trend to the number of study participants might reveal the underlying causes of variation in the raw number of missing observations and stability versus volatility in the trend. These results might impact the ability to impute reliably missing data because the length of time between two observations might systematically relate to the date and because—given these seeming periodizations—one might expect different types of participants at distinct periods. These latent groups might exhibit different missing data patterns tied to their more or less (empirically ascertainable) behaviors that would enable us to specify potentially useful seasonality relationships in missing data imputation.