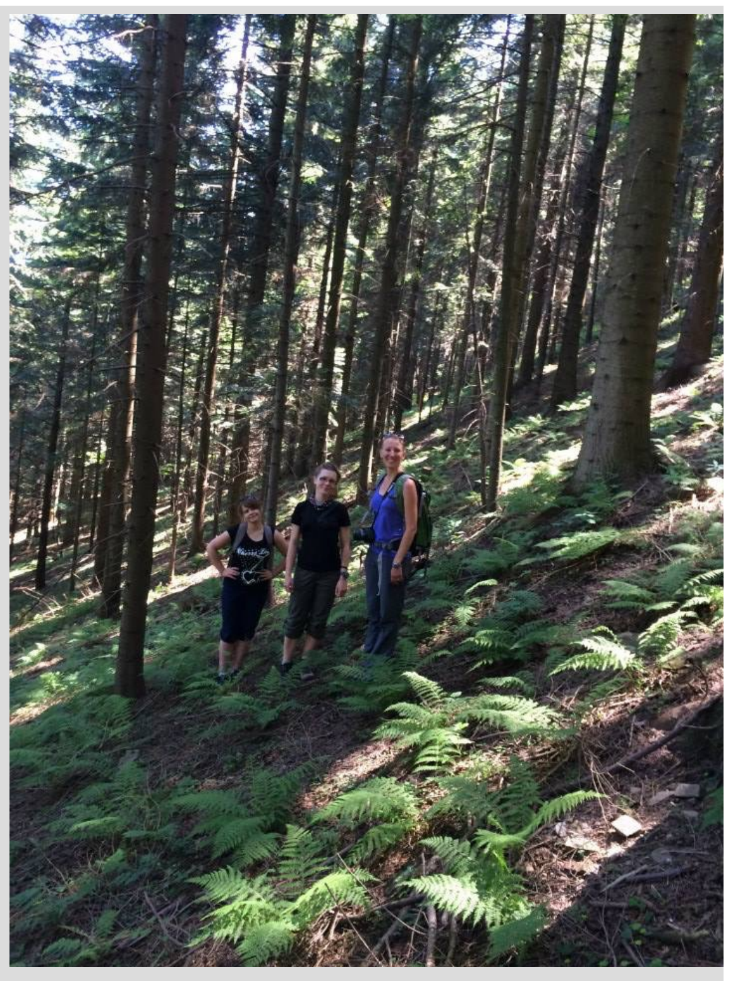
Physical Activity Analysis: A project in many languages

Katharine Lee

Department of Anthropology, University of Illinois at Urbana-Champaign



Project Goals:

- Use commercially available physical activity monitors (FitBit One) to collect detailed data about physical activity in adult women in rural Poland
- Analyze effects of physical activity in relation to markers of bone turnover and levels of reproductive hormones
- Leverage continuous data for time-use analysis methods and daily summary data for overall activity level
- Avoid spending research funds on *very* expensive activity monitors and software

Outcomes:

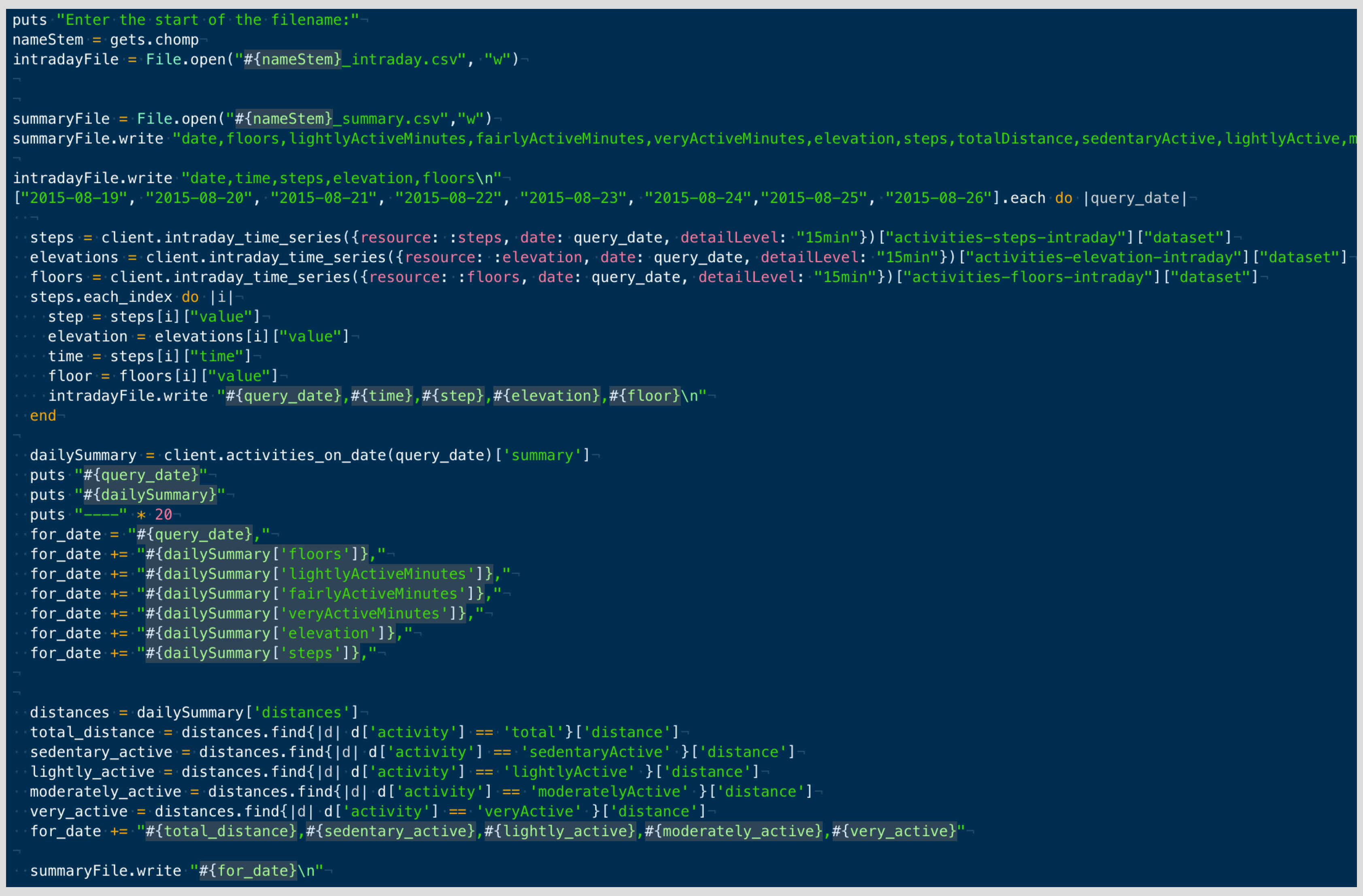
- Code is written to allow for quick downloading of physical activity data and fast, consistent parsing of files, which will be useful for the additional data collection I will be performing in Summer 2016 and Summer 2017.
- Results from this analysis have been presented at 2016 Association of Physical Anthropologists annual meeting in Atlanta, GA (April 2016)
- Results from this analysis are accepted for presentation at 2016 International Society for Evolutionary Medicine and Public Health annual meeting in Durham, NC (June 2016)
- I will be applying to present work generated in this class at the Feminist Biology Symposium at the University of Wisconsin (October 2016)

Acknowledgements:

- This work was part of a Focal Point grant funded by the Graduate College at the University of Illinois at Urbana-Champaign
- FitBit One trackers purchased with funding from Beckman Institute Cognitive Science/Artificial Intelligence grant and UIUC Dept of Anthropology Summer Graduate Research Assistance Award
- Data collection was possible with M. Rogers and K. Lee NSF GRFP awards, American Philosophical Society's Lewis and Clark Fund for Exploration, and NSF Clancy #1317140
- Additional thanks are due to research participants, Polish field assistants, Aaron Lee, and the instructors of this class.

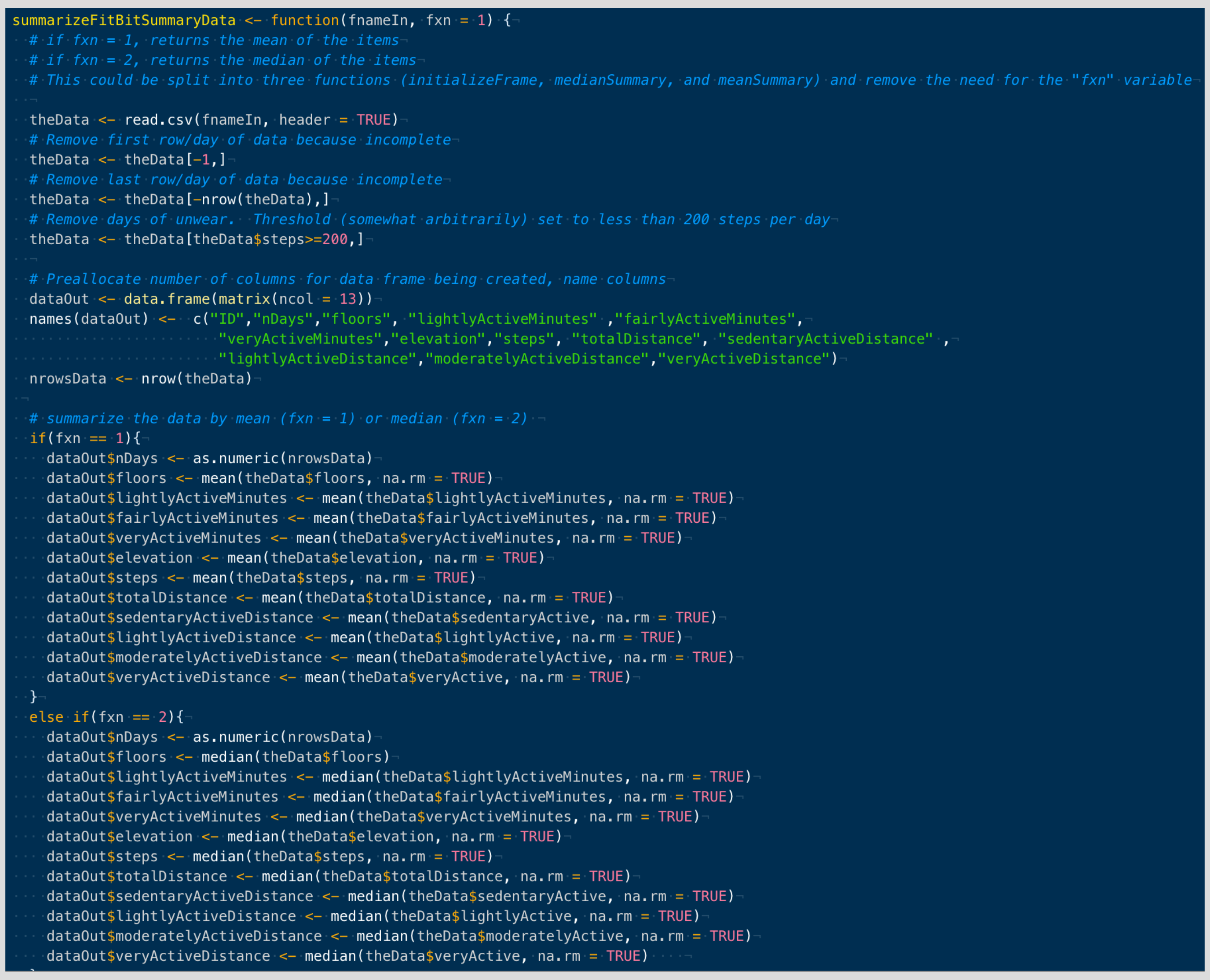
Challenges & Solutions:

- FitBit does not provide continuous data from web interface
  - Request developer API from company with special permissions
  - Download physical activity data from FitBit servers using Ruby



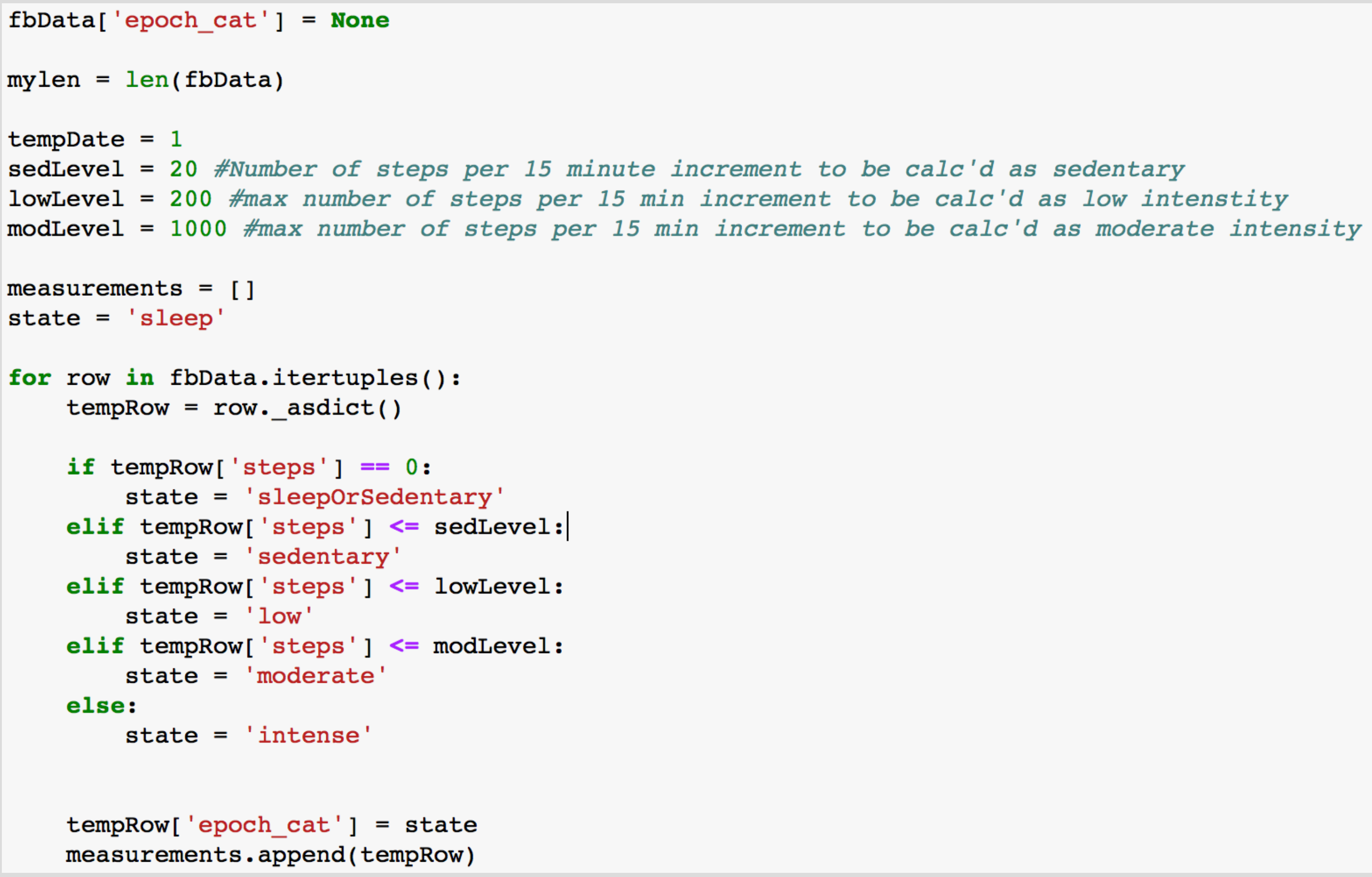
```ruby
puts "Enter the start of the filename:"
nameStem = gets.chomp
intradayFile = File.open("#{nameStem}_intraday.csv", "w")


summaryFile = File.open("#{nameStem}_summary.csv","w")
summaryFile.write "date,floors,lightlyActiveMinutes,fairlyActiveMinutes,veryActiveMinutes,elevation,steps,totalDistance,sedentaryActive,lightlyActive,m

intradayFile.write "date,time,steps,elevation,floors\n"
["2015-08-19", "2015-08-20", "2015-08-21", "2015-08-22", "2015-08-23", "2015-08-24","2015-08-25", "2015-08-26"].each do |query_date|

  steps = client.intraday_time_series({resource: :steps, date: query_date, detailLevel: "15min"})["activities-steps-intraday"]["dataset"]
  elevations = client.intraday_time_series({resource: :elevation, date: query_date, detailLevel: "15min"})["activities-elevation-intraday"]["dataset"]
  floors = client.intraday_time_series({resource: :floors, date: query_date, detailLevel: "15min"})["activities-floors-intraday"]["dataset"]
  steps.each_index do |i|
    step = steps[i]["value"]
    elevation = elevations[i]["value"]
    time = steps[i]["time"]
    floor = floors[i]["value"]
    intradayFile.write "#{query_date},#{time},#{step},#{elevation},#{floor}\n"
  end

  dailySummary = client.activities_on_date(query_date)['summary']
  puts "#{query_date}"
  puts "#{dailySummary}"
  puts "----" * 20
  for_date = "#{query_date},"
  for_date += "#{dailySummary['floors']},"
  for_date += "#{dailySummary['lightlyActiveMinutes']},"
  for_date += "#{dailySummary['fairlyActiveMinutes']},"
  for_date += "#{dailySummary['veryActiveMinutes']},"
  for_date += "#{dailySummary['elevation']},"
  for_date += "#{dailySummary['steps']},"


  distances = dailySummary['distances']
  total_distance = distances.find{|d| d['activity'] == 'total'}['distance']
  sedentary_active = distances.find{|d| d['activity'] == 'sedentaryActive' }['distance']
  lightly_active = distances.find{|d| d['activity'] == 'lightlyActive' }['distance']
  moderately_active = distances.find{|d| d['activity'] == 'moderatelyActive' }['distance']
  very_active = distances.find{|d| d['activity'] == 'veryActive' }['distance']
  for_date += "#{total_distance},#{sedentary_active},#{lightly_active},#{moderately_active},#{very_active}"

  summaryFile.write "#{for_date}\n"
```

- Summarized data includes days the device was delivered to & returned by study participant
  - Remove incomplete days of data collection, then
  - Average data from each individual across days
  - Analyzed in R because it would be tedious & error-prone in Excel



```r
summarizeFitBitSummaryData <- function(fnameIn, fxn = 1) {
  # if fxn = 1, returns the mean of the items
  # if fxn = 2, returns the median of the items
  # This could be split into three functions (initializeFrame, medianSummary, and meanSummary) and remove the need for the "fxn" variable

  theData <- read.csv(fnameIn, header = TRUE)
  # Remove first row/day of data because incomplete
  theData <- theData[-1,]
  # Remove last row/day of data because incomplete
  theData <- theData[-nrow(theData),]
  # Remove days of unwear.  Threshold (somewhat arbitrarily) set to less than 200 steps per day
  theData <- theData[theData$steps>=200,]

  # Preallocate number of columns for data frame being created, name columns
  dataOut <- data.frame(matrix(ncol = 13))
  names(dataOut) <-  c("ID","nDays","floors", "lightlyActiveMinutes" ,"fairlyActiveMinutes",
                       "veryActiveMinutes","elevation","steps", "totalDistance", "sedentaryActiveDistance" ,
                       "lightlyActiveDistance","moderatelyActiveDistance","veryActiveDistance")
  nrowsData <- nrow(theData)

  # summarize the data by mean (fxn = 1) or median (fxn = 2)
  if(fxn == 1){
    dataOut$nDays <- as.numeric(nrowsData)
    dataOut$floors <- mean(theData$floors, na.rm = TRUE)
    dataOut$lightlyActiveMinutes <- mean(theData$lightlyActiveMinutes, na.rm = TRUE)
    dataOut$fairlyActiveMinutes <- mean(theData$fairlyActiveMinutes, na.rm = TRUE)
    dataOut$veryActiveMinutes <- mean(theData$veryActiveMinutes, na.rm = TRUE)
    dataOut$elevation <- mean(theData$elevation, na.rm = TRUE)
    dataOut$steps <- mean(theData$steps, na.rm = TRUE)
    dataOut$totalDistance <- mean(theData$totalDistance, na.rm = TRUE)
    dataOut$sedentaryActiveDistance <- mean(theData$sedentaryActive, na.rm = TRUE)
    dataOut$lightlyActiveDistance <- mean(theData$lightlyActive, na.rm = TRUE)
    dataOut$moderatelyActiveDistance <- mean(theData$moderatelyActive, na.rm = TRUE)
    dataOut$veryActiveDistance <- mean(theData$veryActive, na.rm = TRUE)
  }
  else if(fxn == 2){
    dataOut$nDays <- as.numeric(nrowsData)
    dataOut$floors <- median(theData$floors)
    dataOut$lightlyActiveMinutes <- median(theData$lightlyActiveMinutes, na.rm = TRUE)
    dataOut$fairlyActiveMinutes <- median(theData$fairlyActiveMinutes, na.rm = TRUE)
    dataOut$veryActiveMinutes <- median(theData$veryActiveMinutes, na.rm = TRUE)
    dataOut$elevation <- median(theData$elevation, na.rm = TRUE)
    dataOut$steps <- median(theData$steps, na.rm = TRUE)
    dataOut$totalDistance <- median(theData$totalDistance, na.rm = TRUE)
    dataOut$sedentaryActiveDistance <- median(theData$sedentaryActive, na.rm = TRUE)
    dataOut$lightlyActiveDistance <- median(theData$lightlyActive, na.rm = TRUE)
    dataOut$moderatelyActiveDistance <- median(theData$moderatelyActive, na.rm = TRUE)
    dataOut$veryActiveDistance <- median(theData$veryActive, na.rm = TRUE)
```

- Continuous data must be categorized
  - Sleep time should not be included in further analyses.
  - Remove first & last day from analysis because of incomplete data
  - Classify time intensity of activity for each remaining time period
  - Used Python (Jupyter notebook) for flexibility with data structures



```python
fbData['epoch_cat'] = None

mylen = len(fbData)

tempDate = 1
sedLevel = 20 #Number of steps per 15 minute increment to be calc'd as sedentary
lowLevel = 200 #max number of steps per 15 min increment to be calc'd as low intenstity
modLevel = 1000 #max number of steps per 15 min increment to be calc'd as moderate intensity

measurements = []
state = 'sleep'

for row in fbData.itertuples():
    tempRow = row._asdict()

    if tempRow['steps'] == 0:
        state = 'sleepOrSedentary'
    elif tempRow['steps'] <= sedLevel:
        state = 'sedentary'
    elif tempRow['steps'] <= lowLevel:
        state = 'low'
    elif tempRow['steps'] <= modLevel:
        state = 'moderate'
    else:
        state = 'intense'


    tempRow['epoch_cat'] = state
    measurements.append(tempRow)
```