Supervised & Unsupervised Learning and Data Visualization

- MATH 4990: Applied and Introductory Data Science -

Final Project Report

Iurii Shamkin (T00036016)

Thompson Rivers University

Department of Mathematics and Statistics

Contents

Chapter 1

Introduction

This report is based on a final project for the Introductory Data Science course. Main topics covered are data visualization, unsupervised learning (Multidimensional scaling, t-Distributed Stochastic Neighbor Embedding, PAM clustering), supervised learning (Multiple Linear and Log-linear Regressions, Decision (regression) trees and Random Forests) and related metrics. This project was planned to not only explore separate topics of supervised and unsupervised learning but also to compare them whereas possible, to determine the best approach to use considering presented data. Because of this, the report is structured around datasets rather than specific methods.

Working with two personal datasets, I did data preparation, various data visualizations including regression trees, comparison of performing dimension reduction, clustering, and comparison of predictive models.

Although my code isn't referenced from the text, it's all located in chronological order in an Appendix with the same headings as topics from the report, so one should be able to navigate it easily.

Chapter 2

Personal data sets - Garmin activity tracker and Moment mobile app

2.1 Intro

Initially, I intended to work on continuous heart rate along with sleep stages (light/deep/movements) from my activity tracking device - Garmin Vivosmart 3. I was hoping to find correlations between both sets of metrics and build a predictive model. However, after numerous attempts to extract data from their custom-format .FIT files, I wasn't able to succeed. Despite stating that it's supported, even popular open-source data analysis tool for cyclists and triathletes - GoldenCheetah - failed to extract any meaningful data from mine .FIT files. I guess that it can be because my device is relatively new and file structure should've changed to accommodate new metrics like stress level.

As a result, I had to create my data set by hand. Thus it was feasible to use only average values for the whole day. Having data set with so few average samples, it wasn't enough data to make any conclusions. I decided to add data from Moment application I was using since mid-January. This is an iOS app that automatically tracks how much you use your phone during the day, as well as records time and duration of each phone pickup.

Combining both data sets, I intended to find a correlation between phone use and my vitality metrics. The end goal was to determine in a which way should I use my phone to increase the quality of my health.

2.2 Data Preparation

Moment app data set Moment data set is a JSON file with an array of objects with only three property: minuteCount, pickupCount, and date. minuteCount represents the total amount of phone usage during a day in minutes; pickupCount holds a total number of phone pickups (unlocks) during a day; a date is just a day when recording has been made. It was unusual to work with JSON in R, but it was only two lines to turn it into dataframe of lists with properties as columns.

First look at dataset revealed that there are no missing values. However, there are sentinel values presented. Summary of the dataset shows that we have as little phone usage as 3 min and no pickups during the day at all which does look suspicious, see figure 4.1. We would have to investigate this further. There are two identical outliers (see figure 4.2) which do look like sentinel values. After dropping these two rows, our dataset is ready for analysis. Distribution of total minutes per day and total pickups per day seems normal, figures 4.3 and 4.4 respectively.

Garmin wearable data set As I've already described in section 2.1, I wasn't able to process original data automatically. Thus I had to manually create CSV file by copying values from Garmin Connect portal by hand. It includes following columns: totalSteps (total count of steps I've done in a day), restingHeartRate (my average heart rate while being physically inactive during a day), stressLevel (proprietary Garmin algorithm for determing your stress level based on heart rate and movements, basically if you not moving and your pulse are rising high, you are under stress), deepSleep (duration of being in a deep sleep stage in minutes), lightSleep (duration of being in a light sleep phase in minutes), awakeSleep(duration of being awake during the night in minutes), and observationDate (date of measurements). I've picked the same interval as I get from Minute dataset - 01/22/18 - 04/20/18.

Based on the summary of Garmin data (figure 4.5), I can say that data looks reasonable, except an amount of NAs - 26% of data set's rows are incomplete. This means we cannot just leave out incomplete samples as this is a big portion of our data set. There are many ways to deal with missing values.

I'm planning to use a mean replacement for totalSteps, restingHeartRate and stressLevel (two missing values per column) and regression replacement for deepSleep, lightSleep, awakeSleep (23 missing values for each column). Figure 4.6 shows a result of fitting a linear model to find missing deepSleep values. Considering high standard errors for residuals and intercept values, as well as low R-squared value I've concluded that simple linear model wouldn't work here. I've also tried to fit a log-linear model (figure 4.7). Obviously, residual errors went down; however, R-squared metric still suggests that this model can explain only 18 percent of the variance in the dataset. Figure 4.8 shows a result of fitting generalized linear model instead. We have huge deviance, both amongst residuals and null, suggesting that glm wouldn't work either.

After giving it some thought, quite possibly that sleep measurements don't correlate with total steps, resting heart rate or stress level at all and it's impossible to predict missing values accurately. As a result, I decided to go for diversity of dataset (6 parameters with 66 samples) rather than quantity (3 parameters with 89 samples).

Next, I'm looking at the distribution of all columns. Total steps column (figure 4.9) has normal symmetrical distribution, whereas resting heart rate (figure 4.10) seems to have multiple very low sentinel values. All other measurements - stress level (figure 4.11), amount of deep sleep (figure 4.12), light sleep (figure 4.13) and being awake during sleep (figure 4.14) - have positively skewed normal distributions, meaning their mean is greater than the median.

I've also plotted a parallel-coordinate plot of Garmin dataset (figure 4.15) to envision variable correlations better. As one would expect, the stress level is positively correlated with resting heart rate as well as resting heart rate on average positively correlates with the number of steps per day. On the other hand, an interesting relation occurs with the total number of steps being negatively correlated with the amount of deep sleep in a previous night, and amount of deep sleep is negatively correlated with the amount of light sleep.

Having recordings for different sleep stages is sophisticated, but they are independent, whereas the total amount of sleep per day I would be able to control. Thus I've introduced a new column totalSleep - aggregation of all three sleep metrics. Figure 4.16 shows a parallel-coordinate plot considering only aggregated sleep. From it is seen a more obvious relation between the total amount of sleep and total steps - negative correlation (which is odd).

Combining both datasets After combining two data sets, I've tried again to predict missing values for sleep measurements using regression. Unfortunately, I wasn't able to fit any of them (see figure 4.17). I concluded that missing columns are not depended on present ones, which is a sign of a good data set.

2.3 Regression Tree Analysis

I began my analysis with regression trees. The main reason for that isn't prediction model but a visual representation of data relation within the set. I'm a very visual person and 'seeing' data is much better than reading a summary.

Figure 4.18 shows a regression tree for predicting average stress level for a day. It contradicts my assumption that phone usage should affect our vital parameters since neither minuteCount nor pickupCount been included into the tree. Another interesting observation - looks like it's better to have a longer light stage and shorter deep stage during your sleep.

On figure 4.19 is represented a regression tree for predicting an amount of phone usage in minutes per day. The most important feature in this case - the total amount of sleep. This regression tree suggests that the more sleep I had, the longer I've used a phone during that day.

On figure 4.20 is represented a regression tree for predicting a number of phone pickups per day. The most important feature for determining them is a total number of steps per day - more steps involves taking the phone out more often. Another observation, which is suggesting that model is over-fitted (5 splits) - higher resting heart rate corresponds to fewer phone pickups whereas higher stress level corresponds to a higher number of pickups. However, we've discovered that heart rate and stress level are positively correlated. Prove of this can be obtained by checking a complexity parameter plot on this tree (figure 4.21), where the tree with 4 splits is optimal.

2.4 Multidimensional Scaling & t-SNE Dimension Reduction

I wanted to step away from supervised learning and look for possible groups within data set. Before using any clustering algorithm, I wanted to try to look for clusters myself. As I mentioned previously, I'm a visual person, and the best way to find any groupings is to visualize the data. Since we have about 10 dimensions in the combined dataset, it's infeasible to visualize it as is. For this purpose, we would need some dimension reduction approach.

At first, I've applied classical multidimensional scaling with euclidean distances used in distance matrix. I've added color grading in chronological order and labeled points with their record date. Figures 4.22 and 4.23 shows 2-dimensional MDS of our data set with separate and combined sleep stages respectively. To my regret, these visualizations haven't brought any significant insights. MDS on the dataset with aggregated sleep groups does create more evenly spread plot, but that's about it. Dataset seems to be grouped neither chronologically nor somehow else.

Secondly, I've used t-Distributed Stochastic Neighbor Embedding technique (t-SNE) for exploring dimension reduction on my data. This popular and prize-winning [5] non-linear dimensionality reduction algorithm "has become widespread in the field of machine learning, since it has an almost magical ability to create compelling two-dimensional "maps" from data..." [6]. t-SNE is a variation of Stochastic Neighbor Embedding which "uses a Student-t distribution rather than a Gaussian to compute the similarity between two points in the low-dimensional space. t-SNE employs a heavy-tailed distribution in the low-dimensional space to alleviate both the crowding problem and the optimization problems of SNE"[4].

Figure 4.24 shows resulting t-SNE projection as a 2-dimensional scatter plot. Hyperparameters play a big role in the quality of t-SNE and allow to optimize process a lot; thus I was using this article as a guide [6]. The objective function in t-SNE is minimized using a gradient descent optimization that is initiated randomly. As a result, another run with identical parameters produces different (rotated) solution; however, the overall shape is preserved (see figure 4.25).

This visualization of dimension reduction drastically differs from what we've seen with MDS. t-SNE created a more ordered plot, where it's now possible to assume possible clusters. One can say that there are 2, 3, 4, 5 or even 8 clusters, depending on perception. However, we wouldn't know for sure without running some clustering algorithm on the data.

2.5 PAM Clustering & its Visualizations

For clustering, I've chosen K-medoids or PAM (Partitioning Around Medoids) algorithm, which is related to k-means clustering. "The term medoid refers to an object within a cluster for which average dissimilarity between it and all the other the members of the cluster is minimal. It corresponds to the most centrally located point in the cluster" [2]. In PAM each cluster is represented by one medoid in the cluster, rather than mean value of points belongings to cluster (k-means). This means k-medoids algorithm is less sensitive to anomalous points or outliers.

The same as k-means algorithm, k-medoids require specifying number of clusters to partition to. A variety of metrics exist to determine the number of clusters, and today I've chosen silhouette width - "... an internal validation metric which is an aggregated measure of how similar observation is to its own cluster compared its closest neighboring cluster" [1]. Figure 4.26 shows a plot of silhouette widths when clustering data set from 2 to 20 groups. The highest width is observed when the dataset is partitioned into 8 clusters.

After running PAM and generating 8 clusters, I wanted to visualize them. First, I've used t-SNE projection with 8 clusters being differently colored (see figure 4.27). Our assumption about how data might be clustered is confirmed, having clusters group one after another.

Next, I've returned to MDS plot; however, now I've colored clusters as well (see figure 4.28). Obviously, t-SNE did a better job on data visualization, having spread and torn clusters with MDS. Figure 4.29 shows the same MDS plot with date labels added to it. It's easy to observe that calendar date has nothing to do with clustering neither.

2.6 Linear Regression, Log-linear Regression and Random Forest - Evaluation of Predictive Models

Back to supervised learning, I'm planning to evaluate different predictive models: linear regression, log-linear model, and random forest. At first, we need evaluation metric for comparing different methods [3]. I've chosen the ones we've cover in class: RMSE (root mean squared error) and MAE (mean absolute error).

After that, I'm dividing our dataset into a train (70%) and test (30%) partitions. I'm training all 3 models on train subset, predict stress level against test partition and calculate both RMSE and MAE for each model. I repeat the process for 100 times to get average error rates.

Figure 4.30 shows calculated error metrics table. Random forest algorithm ended up being the most suitable predictive model no matter which metric considering (RMSE = 5.28 & MAE = 3.87), which is expected. Next is log-linear model (RMSE = 5.83 & MAE = 4.14). And the worst model to predict stress level turned to be a simple linear regression (RMSE = 6.25 & MAE = 4.66).

Chapter 3

Conclusion

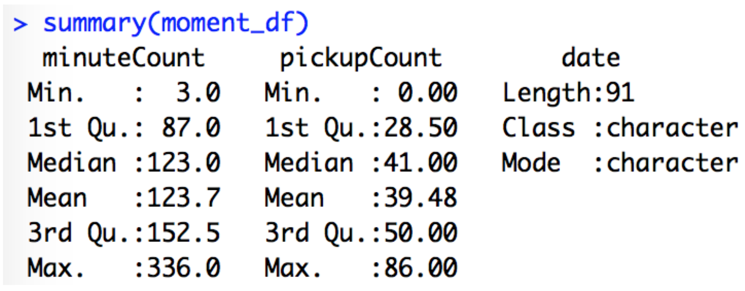
After thoroughly analyzing personal datasets, I did not receive results I was hoping for. My primary objective was to determine how phone usage is affecting my vitality parameters recorded by the activity tracker. However, none of my tests showed a strong correlation between the total amount of phone usage or the total number of phone pickups with my stress level, resting heart rate, the number of steps, and sleep stages. This can be due to a couple of reasons: short observation period, inaccurate data measuring from the activity tracker or nature of this features is indeed independent.

Nevertheless, I did derive some interesting observations. A total amount of sleep is negatively correlated with a number of steps meaning the more I'm sleeping the less I'm going to walk that day. Also, duration of light sleep stage is negatively correlated with deep sleep stage's length, and it's better to have longer light stages and shorter deep sleep stages. And the last one, number of phone pickups is positively correlated with a number of steps meaning that I'm using my phone the most when I'm either being physically active or traveling (which does make sense).

Regards methods I've used, t-SNE has proven to summarize dataset better comparing with MDS, and ensemble approach of Random Forest has shown being better suited for regression than linear and log-linear regressions themselves at least on my personal datasets.

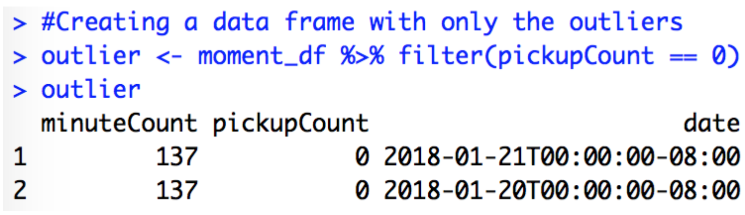
Chapter 4

Figures & Tables



```
> summary(moment_df)
  minuteCount     pickupCount          date          
 Min.   :  3.0   Min.   : 0.00   Length:91         
 1st Qu.: 87.0   1st Qu.:28.50   Class :character  
 Median :123.0   Median :41.00   Mode  :character  
 Mean   :123.7   Mean   :39.48                     
 3rd Qu.:152.5   3rd Qu.:50.00                     
 Max.   :336.0   Max.   :86.00                     
```

Figure 4.1: Summary of Moment data after transformations



```
> #Creating a data frame with only the outliers
> outlier <- moment_df %>% filter(pickupCount == 0)
> outlier
  minuteCount pickupCount                      date
1         137           0 2018-01-21T00:00:00-08:00
2         137           0 2018-01-20T00:00:00-08:00
```

Figure 4.2: Outliers within Moment data

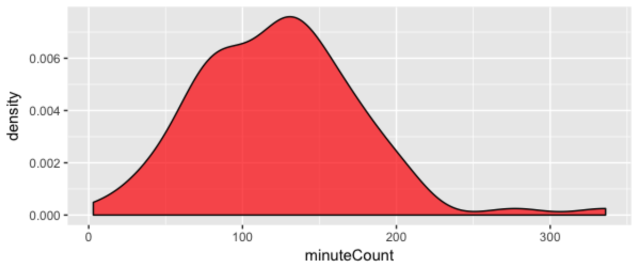


Figure 4.3: Distribution of total minutes per day

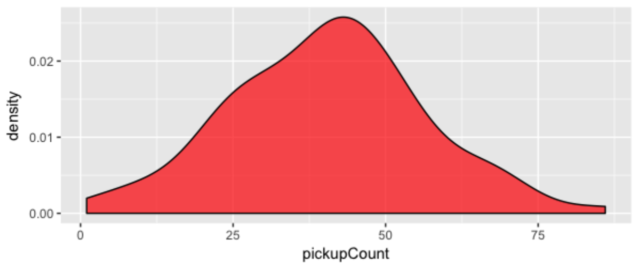
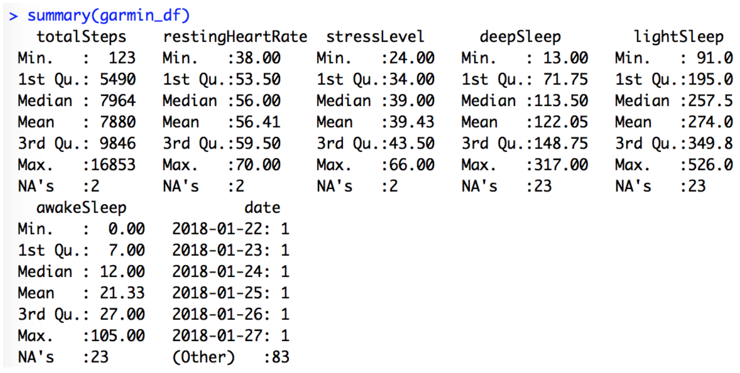
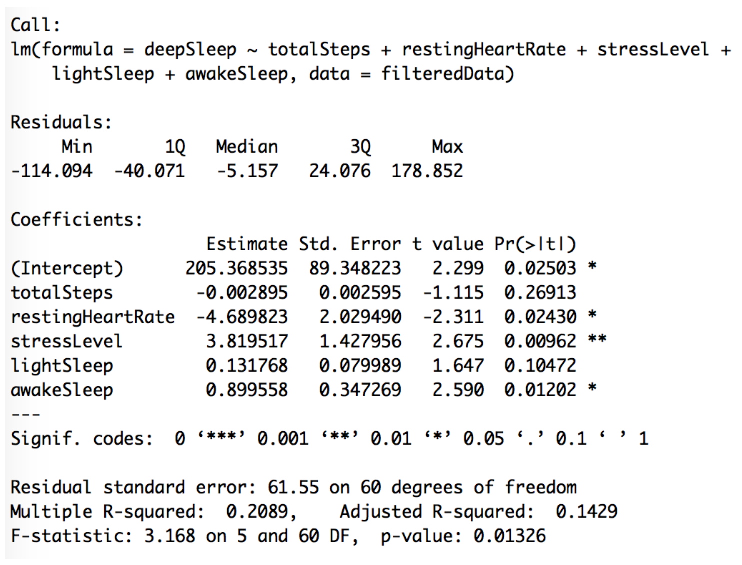


Figure 4.4: Distribution of total pickups per day



```
> summary(garmin_df)
   totalSteps      restingHeartRate  stressLevel       deepSleep         lightSleep
 Min.   :  123    Min.   :38.00    Min.   :24.00    Min.   : 13.00    Min.   : 91.0
 1st Qu.: 5490    1st Qu.:53.50    1st Qu.:34.00    1st Qu.: 71.75    1st Qu.:195.0
 Median : 7964    Median :56.00    Median :39.00    Median :113.50    Median :257.5
 Mean   : 7880    Mean   :56.41    Mean   :39.43    Mean   :122.05    Mean   :274.0
 3rd Qu.: 9846    3rd Qu.:59.50    3rd Qu.:43.50    3rd Qu.:148.75    3rd Qu.:349.8
 Max.   :16853    Max.   :70.00    Max.   :66.00    Max.   :317.00    Max.   :526.0
 NA's   :2        NA's   :2        NA's   :2        NA's   :23        NA's   :23
   awakeSleep           date
 Min.   :  0.00   2018-01-22: 1
 1st Qu.:  7.00   2018-01-23: 1
 Median : 12.00   2018-01-24: 1
 Mean   : 21.33   2018-01-25: 1
 3rd Qu.: 27.00   2018-01-26: 1
 Max.   :105.00   2018-01-27: 1
 NA's   :23       (Other)   :83
```

Figure 4.5: Summary of Garmin data



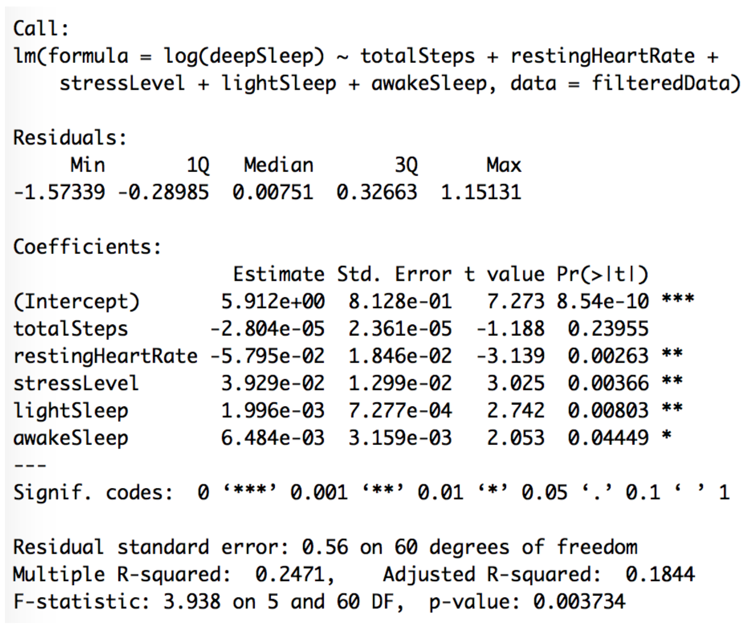
```
Call:
lm(formula = deepSleep ~ totalSteps + restingHeartRate + stressLevel +
    lightSleep + awakeSleep, data = filteredData)

Residuals:
     Min       1Q   Median       3Q      Max
-114.094  -40.071   -5.157   24.076  178.852

Coefficients:
                   Estimate Std. Error t value Pr(>|t|)
(Intercept)      205.368535  89.348223   2.299  0.02503 *
totalSteps        -0.002895   0.002595  -1.115  0.26913
restingHeartRate  -4.689823   2.029490  -2.311  0.02430 *
stressLevel        3.819517   1.427956   2.675  0.00962 **
lightSleep         0.131768   0.079989   1.647  0.10472
awakeSleep         0.899558   0.347269   2.590  0.01202 *
---
Signif. codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1

Residual standard error: 61.55 on 60 degrees of freedom
Multiple R-squared:  0.2089,	Adjusted R-squared:  0.1429
F-statistic: 3.168 on 5 and 60 DF,  p-value: 0.01326
```

Figure 4.6: Predicitng missing values of deepSleep with lm on Garmin data set



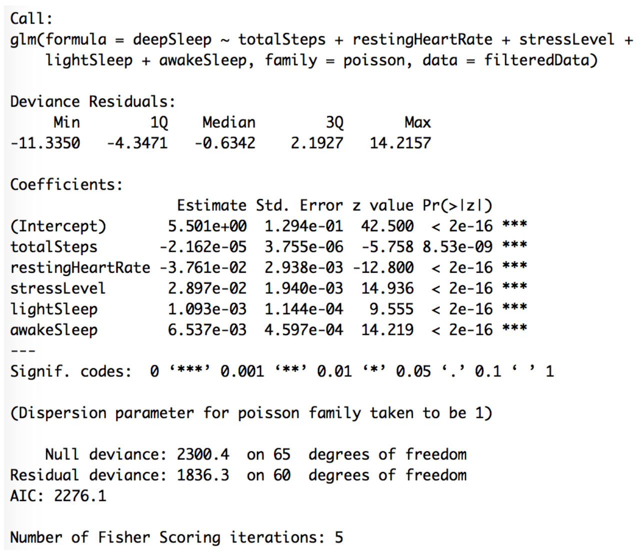
```
Call:
lm(formula = log(deepSleep) ~ totalSteps + restingHeartRate +
    stressLevel + lightSleep + awakeSleep, data = filteredData)

Residuals:
     Min       1Q   Median       3Q      Max
-1.57339 -0.28985  0.00751  0.32663  1.15131

Coefficients:
                   Estimate Std. Error t value Pr(>|t|)
(Intercept)       5.912e+00  8.128e-01   7.273 8.54e-10 ***
totalSteps       -2.804e-05  2.361e-05  -1.188  0.23955
restingHeartRate -5.795e-02  1.846e-02  -3.139  0.00263 **
stressLevel       3.929e-02  1.299e-02   3.025  0.00366 **
lightSleep        1.996e-03  7.277e-04   2.742  0.00803 **
awakeSleep        6.484e-03  3.159e-03   2.053  0.04449 *
---
Signif. codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1

Residual standard error: 0.56 on 60 degrees of freedom
Multiple R-squared:  0.2471,	Adjusted R-squared:  0.1844
F-statistic: 3.938 on 5 and 60 DF,  p-value: 0.003734
```

Figure 4.7: Fitting log linear model to predict missing deepSleep values



```
Call:
glm(formula = deepSleep ~ totalSteps + restingHeartRate + stressLevel +
    lightSleep + awakeSleep, family = poisson, data = filteredData)

Deviance Residuals:
     Min        1Q    Median        3Q       Max
-11.3350   -4.3471   -0.6342    2.1927   14.2157

Coefficients:
                   Estimate Std. Error z value Pr(>|z|)
(Intercept)       5.501e+00  1.294e-01  42.500  < 2e-16 ***
totalSteps       -2.162e-05  3.755e-06  -5.758 8.53e-09 ***
restingHeartRate -3.761e-02  2.938e-03 -12.800  < 2e-16 ***
stressLevel       2.897e-02  1.940e-03  14.936  < 2e-16 ***
lightSleep        1.093e-03  1.144e-04   9.555  < 2e-16 ***
awakeSleep        6.537e-03  4.597e-04  14.219  < 2e-16 ***
---
Signif. codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1

(Dispersion parameter for poisson family taken to be 1)

    Null deviance: 2300.4  on 65  degrees of freedom
Residual deviance: 1836.3  on 60  degrees of freedom
AIC: 2276.1

Number of Fisher Scoring iterations: 5
```

Figure 4.8: Fitting glm to predict missing deepSleep values

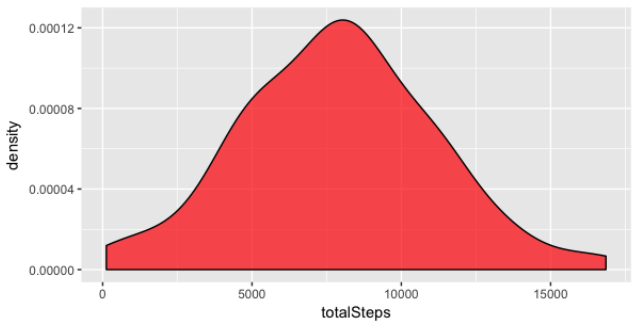


Figure 4.9: Distribution of total steps per day

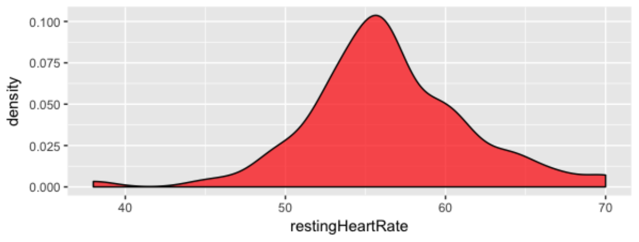


Figure 4.10: Distribution of resting heart rate values

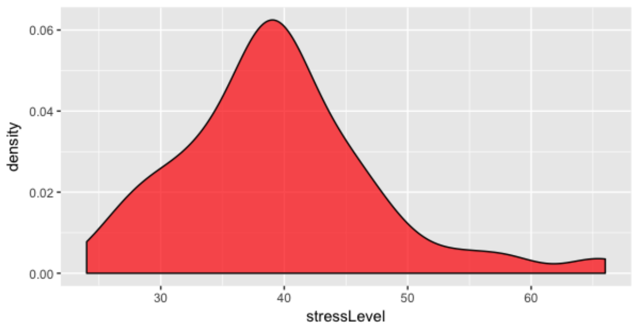


Figure 4.11: Distribution of average stress level per day

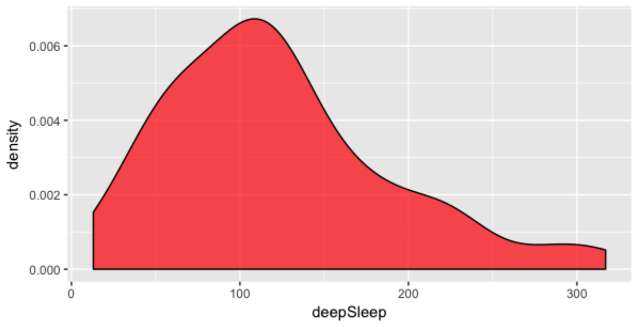


Figure 4.12: Distribution of deep sleep cycle duration during a night

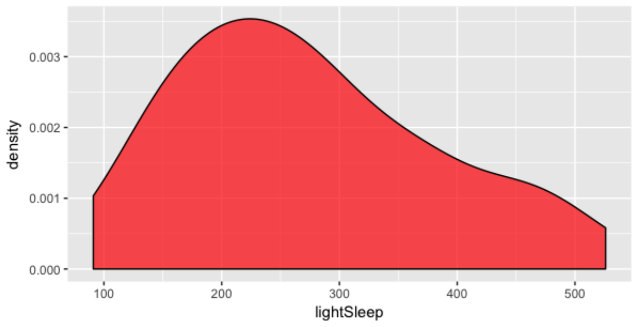


Figure 4.13: Distribution of light sleep cycle duration during a night

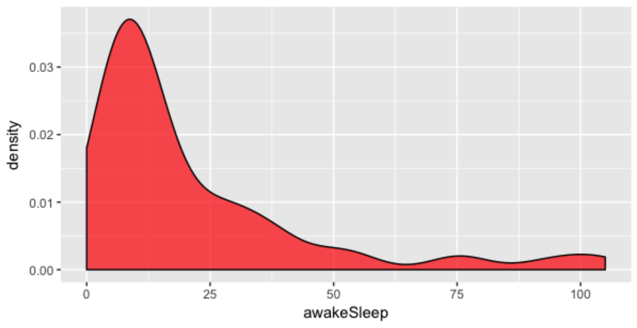


Figure 4.14: Distribution of being awake during a night

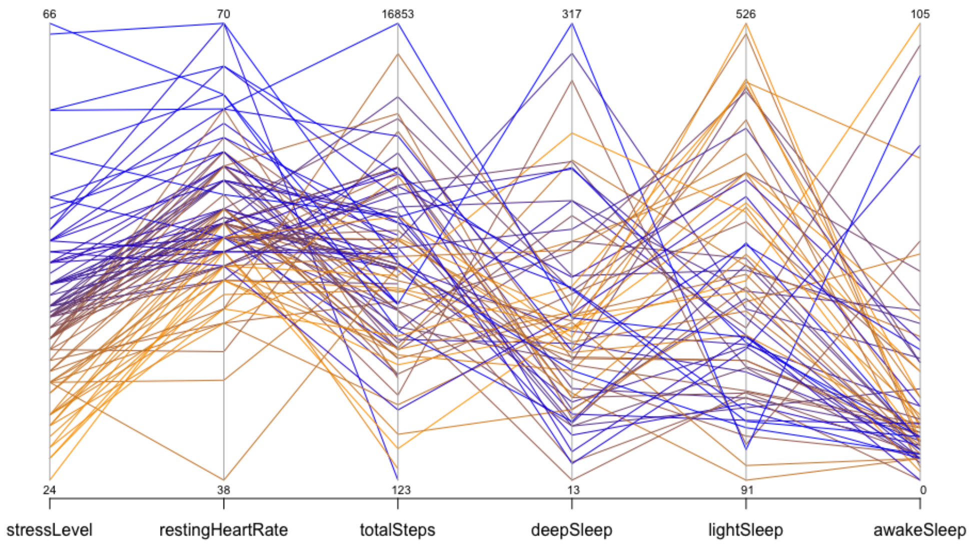


Figure 4.15: Parallel-coordinate plot of Garmin data

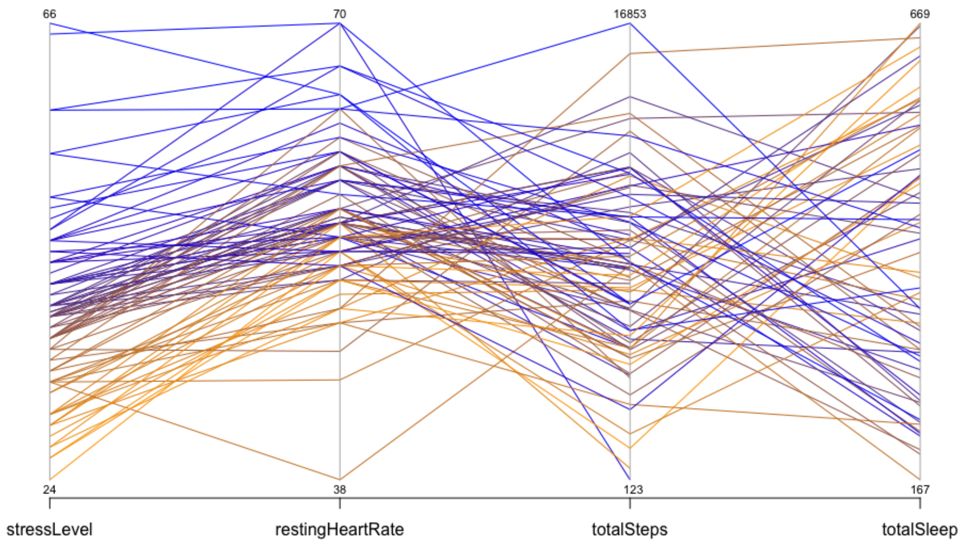
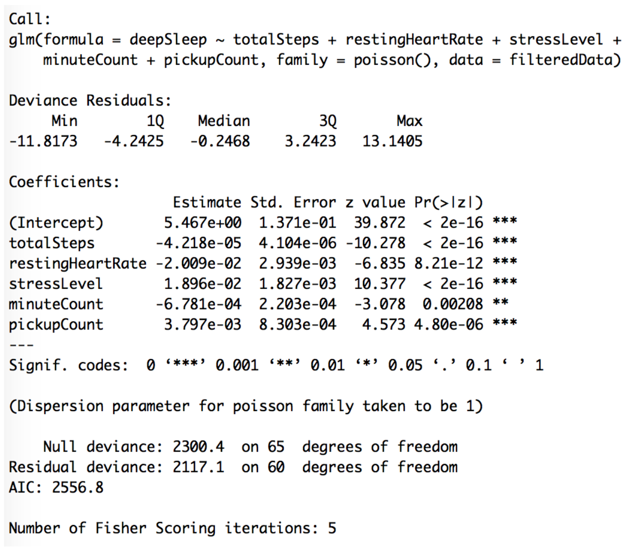


Figure 4.16: Parallel-coordinate plot of Garmin data w/ aggregated sleep



```
Call:
glm(formula = deepSleep ~ totalSteps + restingHeartRate + stressLevel +
    minuteCount + pickupCount, family = poisson(), data = filteredData)

Deviance Residuals:
     Min        1Q    Median        3Q       Max
-11.8173   -4.2425   -0.2468    3.2423   13.1405

Coefficients:
                   Estimate Std. Error z value Pr(>|z|)
(Intercept)       5.467e+00  1.371e-01  39.872  < 2e-16 ***
totalSteps       -4.218e-05  4.104e-06 -10.278  < 2e-16 ***
restingHeartRate -2.009e-02  2.939e-03  -6.835 8.21e-12 ***
stressLevel       1.896e-02  1.827e-03  10.377  < 2e-16 ***
minuteCount      -6.781e-04  2.203e-04  -3.078  0.00208 **
pickupCount       3.797e-03  8.303e-04   4.573 4.80e-06 ***
---
Signif. codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1

(Dispersion parameter for poisson family taken to be 1)

    Null deviance: 2300.4  on 65  degrees of freedom
Residual deviance: 2117.1  on 60  degrees of freedom
AIC: 2556.8

Number of Fisher Scoring iterations: 5
```

Figure 4.17: Predicitng missing values of deepSleep with glm on combined data set

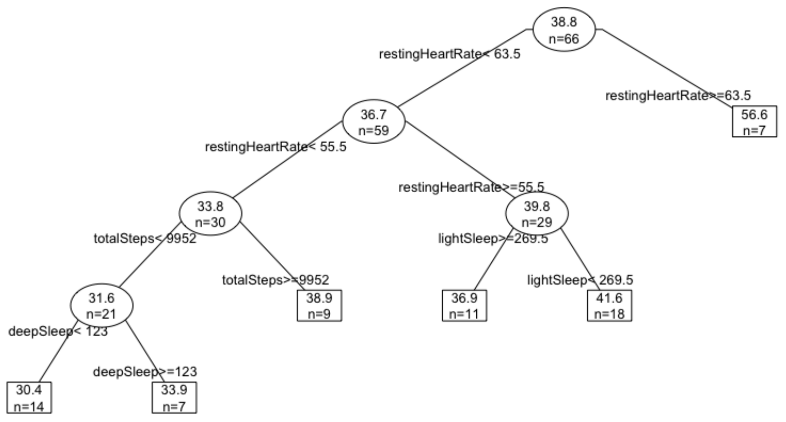


Figure 4.18: Regression tree on predicting stress level, combined data

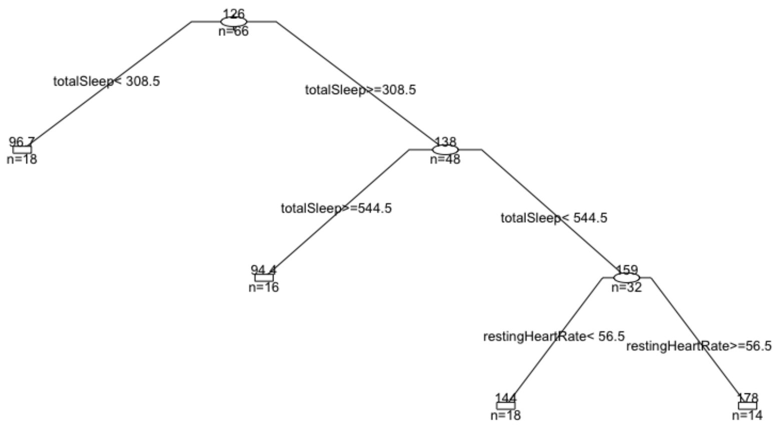


Figure 4.19: Regression tree on predicting amount of phone usage (minutes)

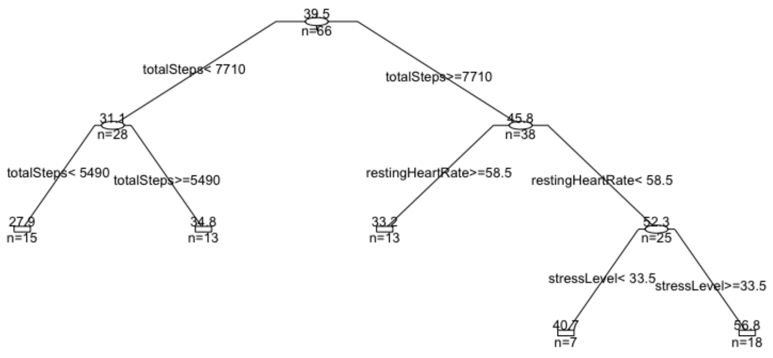


Figure 4.20: Regression tree on predicting number of phone pickups

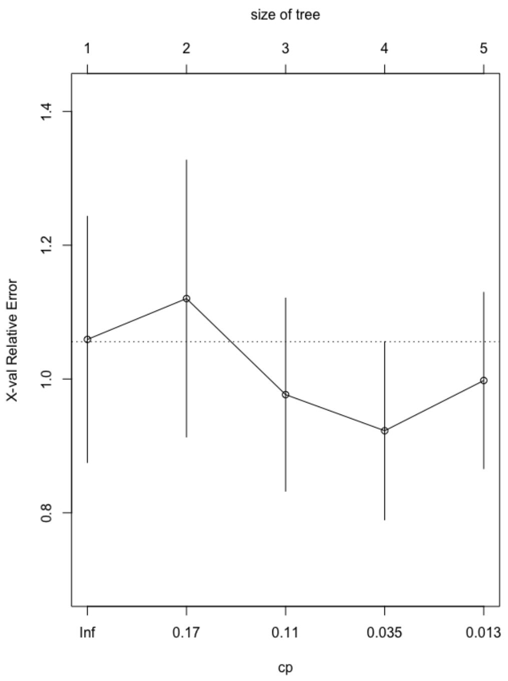


Figure 4.21: Plot of complexity parameter table for regression tree on phone pickups

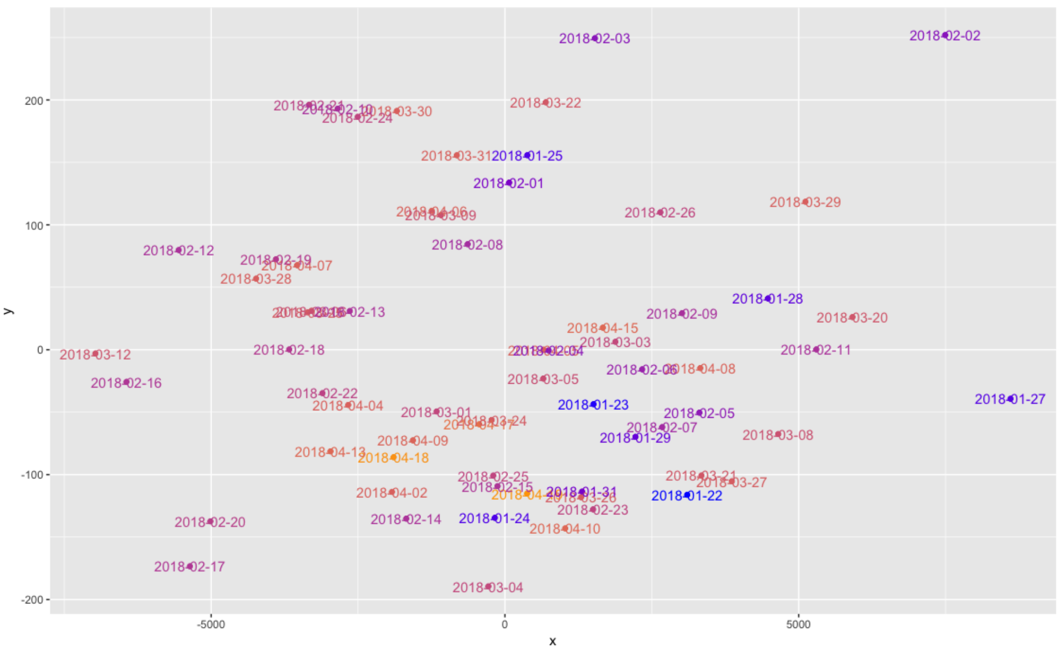


Figure 4.22: Scatter plot of MDS on combined data with separate sleep stages

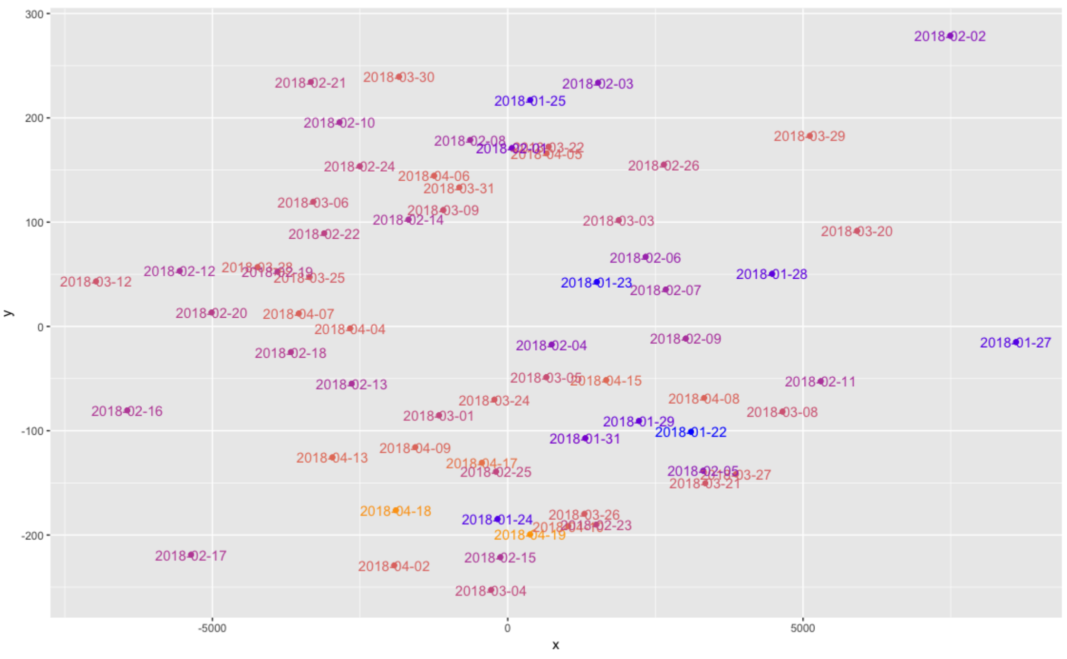


Figure 4.23: Scatter plot of MDS on combined data with aggregated sleep stages

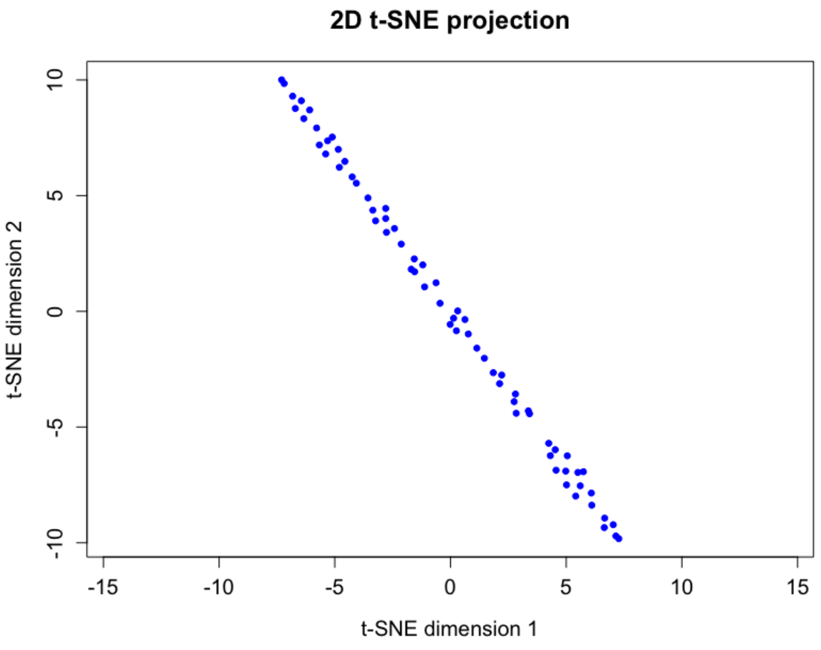


Figure 4.24: Scatter plot of t-SNE on combined data

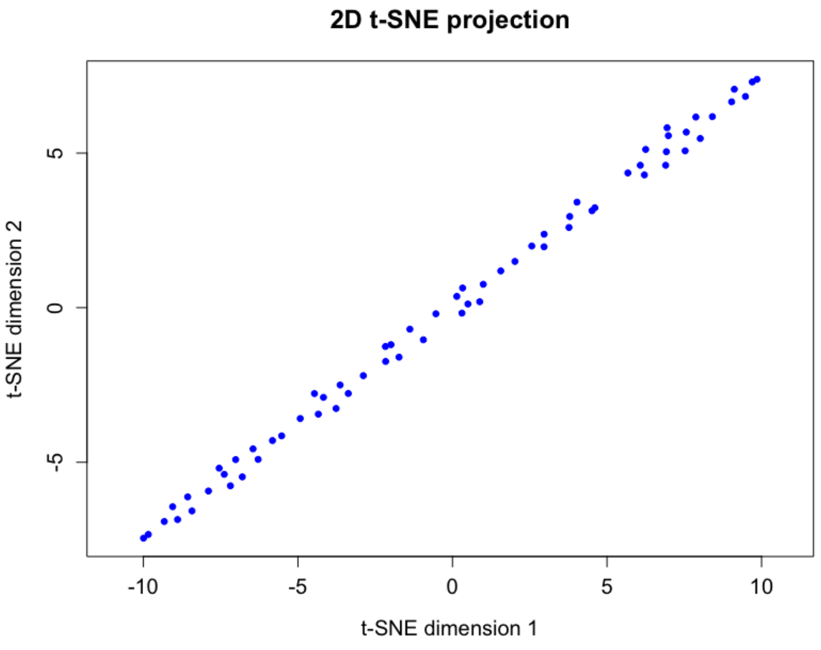


Figure 4.25: Scatter plot of t-SNE on combined data, re-run

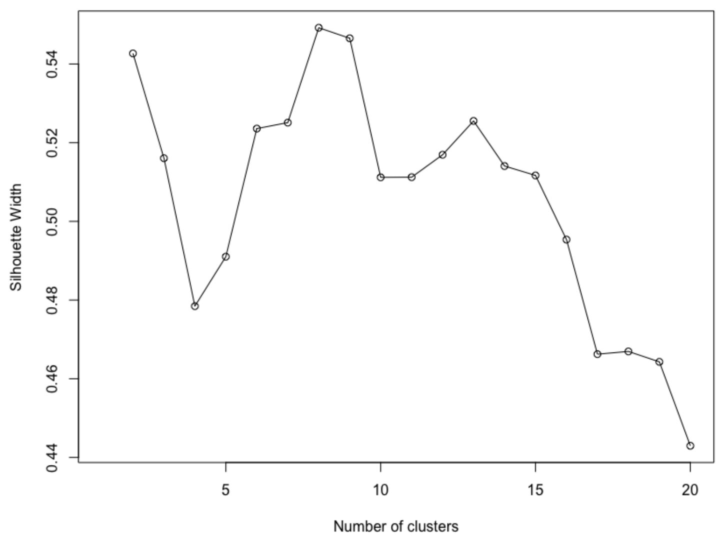


Figure 4.26: Plot of silhouette width (higher is better)

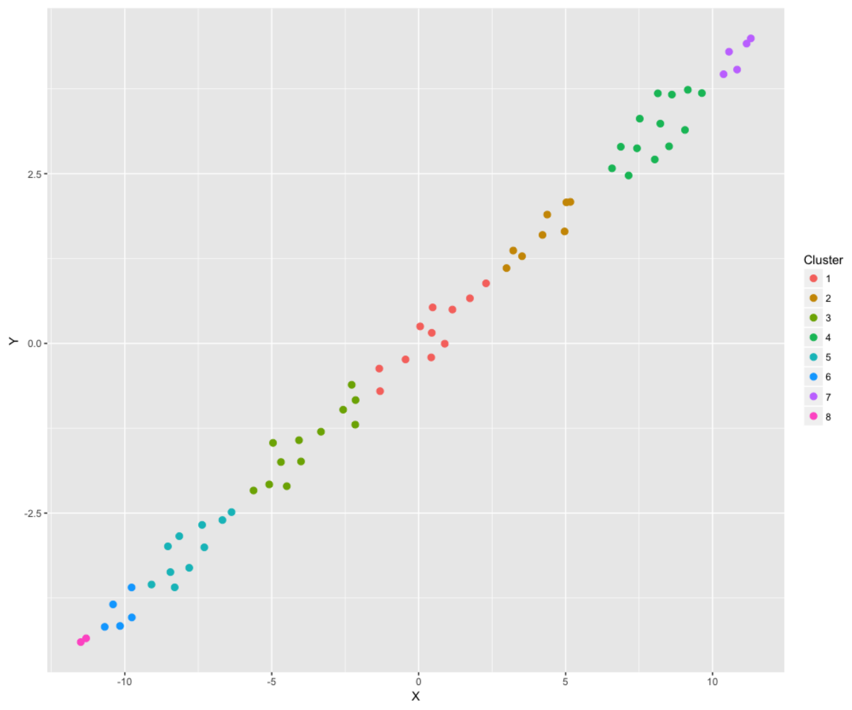


Figure 4.27: Scatter plot of t-SNE with PAM cluste

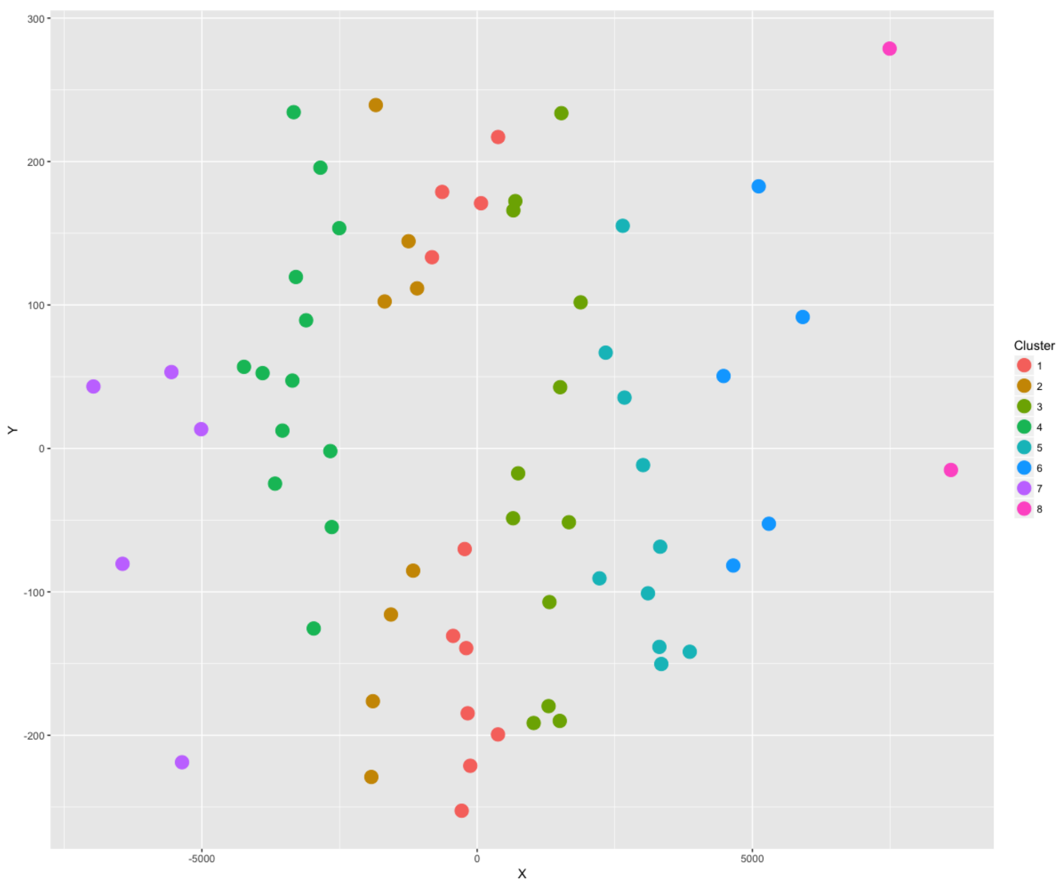


Figure 4.28: Scatter plot of MDS with PAM clusters

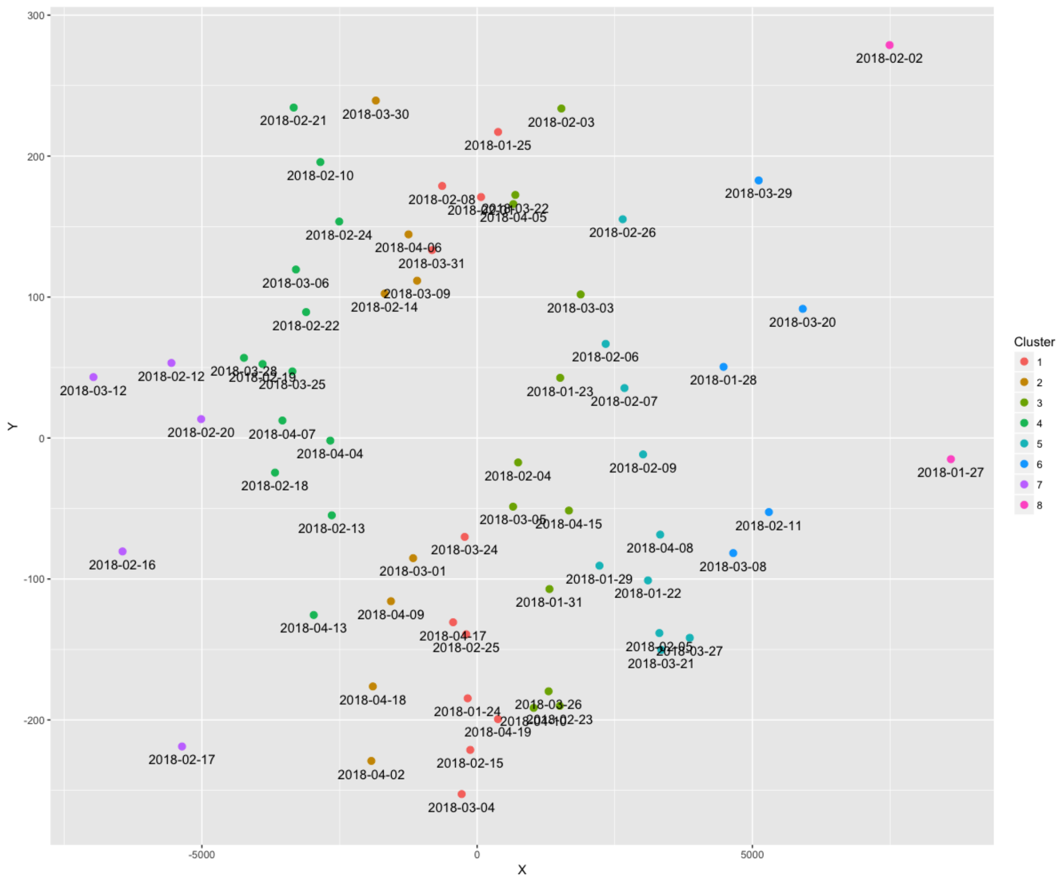


Figure 4.29: Scatter plot of MDS with PAM clusters and date labels

```
                 Method RMSE  MAE
1     Linear Regression 6.25 4.66
2 Log Linear Regression 5.83 4.14
3         Random Forest 5.28 3.87
```

Figure 4.30: Error metrics table

Bibliography

[1] *Clustering Mixed Data Types in R*. Wicked Good Data - r. 2016. URL: https://www.r-bloggers.com/clustering-mixed-data-types-in-r/.

[2] Alboukadel Kassambara. "K-Medoids Essentials". In: (2017). URL: http://www.sthda.com/english/articles/27-partitioning-clustering-essentials/88-k-medoids-essentials/.

[3] Pedro M. "Part 4a: Modelling - predicting the amount of rain". In: (2015). URL: http://theanalyticalminds.blogspot.ca/2015/04/part-4a-modelling-predicting-amount-of.html.

[4] Laurens van der Maaten and Geoffrey Hinton. "Visualizing Data Using t-SNE". In: *Journal of Machine Learning Research 9 (Nov):2579-2605* (2008). URL: http://jmlr.org/papers/volume9/vandermaaten08a/vandermaaten08a.pdf.

[5] Laurentius Johannes Paulus van der Maaten. "t-Distributed Stochastic Neighbor Embedding Wins Merck Viz Challenge". In: *Kaggle Blog* (2012). URL: http://blog.kaggle.com/2012/11/02/t-distributed-stochastic-neighbor-embedding-wins-merck-viz-challenge/.

[6] Martin Wattenberg, Fernanda Viégas, and Ian Johnson. "How to Use t-SNE Effectively". In: *Distill* (2016). DOI: 10.23915/distill.00002. URL: http://distill.pub/2016/misread-tsne.

Appendix A

Code

Please see attached code on the next page.

```
#################################
# Final Project                 #
# MATH 4990                     #
# Iurii Shamkin (T00036016)     #
#################################

setwd('/Users/NN/Desktop/Winter18/MATH4990/final_project')


#######################################
#### Data preparation - Moment app ####
#######################################

library("rjson")
# loading JSON
moment_data = fromJSON(file="moment_data.json")
# binding into dataframe
moment_dataframe = as.data.frame(do.call("rbind", moment_data))

# we got 91 days of observations
dim(moment_dataframe)

# we have dataframe of lists
sapply(moment_dataframe, class)

# converting into actual dataframe
moment_df = data.frame(matrix(unlist(moment_dataframe),
nrow=nrow(moment_dataframe)),  stringsAsFactors=FALSE)
# copying column names
colnames(moment_df) = colnames(moment_dataframe)

#all columns are characters, have to convert to numeric
str(moment_df)
moment_df$minuteCount = as.numeric(as.character(moment_df$minuteCount))
moment_df$pickupCount = as.numeric(as.character(moment_df$pickupCount))

# we don't have missing values in our dataframe
complete_rows_ration = nrow(moment_df)/sum(complete.cases(moment_df))

# we have suspicious outliers
summary(moment_df)

library(dplyr)
#Creating a data frame with only the outliers
outlier <- moment_df %>% filter(pickupCount == 0)
outlier

#Removing the observations with sentinel value
moment_df <- moment_df %>% filter(pickupCount != 0)
```

```
library(ggplot2)
#Plotting numerical variables distributions
pl1 <- ggplot(moment_df, aes(minuteCount))
pl1 + geom_density(fill = "red", alpha = "0.7")

pl2 <- ggplot(moment_df, aes(pickupCount))
pl2 + geom_density(fill = "red", alpha = "0.7")




###################################
#### Data preparation - Garmin ####
###################################

garmin_df = read.csv(file="garmin_data.csv",sep=",", header = T)

dim(garmin_df)
sapply(garmin_df, class)
str(garmin_df)

# many NA's
summary(garmin_df)

# we have about 26% incomplete rows
missing_rows_percent = (1 -
sum(complete.cases(garmin_df))/nrow(garmin_df))*100


# using mean to replace the missing values in totalSteps variable
meanValue = mean(garmin_df$totalSteps[!is.na(garmin_df$totalSteps)])
garmin_df$totalSteps[is.na(garmin_df$totalSteps)] = meanValue

# using mean to replace the missing values in restingHeartRate variable
meanValue =
mean(garmin_df$restingHeartRate[!is.na(garmin_df$restingHeartRate)])
garmin_df$restingHeartRate[is.na(garmin_df$restingHeartRate)] =
meanValue

# using mean to replace the missing values in stressLevel variable
meanValue = mean(garmin_df$stressLevel[!is.na(garmin_df$stressLevel)])
garmin_df$stressLevel[is.na(garmin_df$stressLevel)] = meanValue


#Using linear regression to replace NA in deepSleep variable
filteredData = garmin_df %>% filter(!is.na(deepSleep))
deepSleep_model = lm(deepSleep ~ totalSteps + restingHeartRate +
stressLevel + lightSleep + awakeSleep, data = filteredData)
summary(deepSleep_model)

#Using log linear regression to replace NA in deepSleep variable
```

```r
filteredData = garmin_df %>% filter(!is.na(deepSleep))
deepSleep_model = lm(log(deepSleep) ~ totalSteps + restingHeartRate +
stressLevel + lightSleep + awakeSleep, data = filteredData)
summary(deepSleep_model)

#Using glm to replace NA in deepSleep variable
filteredData = garmin_df %>% filter(!is.na(deepSleep))
deepSleep_model = glm(deepSleep ~ totalSteps + restingHeartRate +
stressLevel + lightSleep + awakeSleep, data = filteredData, family =
poisson)
summary(deepSleep_model)
garmin_df$deepSleep[is.na(garmin_df$deepSleep)] =
predict(deepSleep_model, newdata =
garmin_df[is.na(garmin_df$deepSleep),])

summary(garmin_df)

#Plotting numerical variables distributions
plTotalSteps <- ggplot(garmin_df, aes(totalSteps))
plTotalSteps + geom_density(fill = "red", alpha = "0.7")

plRestingHeartRate <- ggplot(garmin_df, aes(restingHeartRate))
plRestingHeartRate + geom_density(fill = "red", alpha = "0.7")

plStressLevel <- ggplot(garmin_df, aes(stressLevel))
plStressLevel + geom_density(fill = "red", alpha = "0.7")

plDeepSleep <- ggplot(garmin_df, aes(deepSleep))
plDeepSleep + geom_density(fill = "red", alpha = "0.7")

plLightSleep <- ggplot(garmin_df, aes(lightSleep))
plLightSleep + geom_density(fill = "red", alpha = "0.7")

plAwakeSleep <- ggplot(garmin_df, aes(awakeSleep))
plAwakeSleep + geom_density(fill = "red", alpha = "0.7")



# parallel-coordinate plot
blueOrange = colorRampPalette(c("orange","blue"))
theColours = blueOrange(length(garmin_df[,3]))
coloursToPaint = theColours[rank(garmin_df[,3])]

library(MASS)

parcoord(
  garmin_df[, c(3,2,1,4,5,6)],
  col = coloursToPaint,
  var.label = T
)
```

```
# colorelss version
parcoord(
  garmin_df[, c(3,2,1,4,5,6)],
  var.label = T
)

# combining all sleep measurments together
garmin_df$totalSleep = garmin_df$deepSleep + garmin_df$lightSleep +
garmin_df$awakeSleep

parcoord(
  garmin_df[, c(3,2,1,8)],
  col = coloursToPaint,
  var.label = T
)




####################################################
#### Data preparation - Combining both datasets ####
####################################################

combined_df = cbind(garmin_df,moment_df)
combined_df = combined_df[, !(names(combined_df) %in% c("date"))]
combined_df = combined_df[, c(1,2,3,4,5,6,8,9,10,7)]

filteredData = combined_df %>% filter(!is.na(deepSleep))

#Using log linear regression to replace NA in deepSleep variable
deepSleep_model = lm(totalSleep ~ totalSteps + restingHeartRate +
stressLevel + minuteCount + pickupCount, data = filteredData)
summary(deepSleep_model)

#Using glm to replace NA in deepSleep variable
deepSleep_model = glm(deepSleep ~ totalSteps + restingHeartRate +
stressLevel + minuteCount + pickupCount, data = filteredData)
summary(deepSleep_model)




##################################
#### Regression Tree Analysis ####
##################################

filteredData = combined_df %>% filter(!is.na(deepSleep))
rt = rpart(formula= stressLevel~., data = filteredData[,
c(1,2,3,4,5,6,8,9)])
prettyTree(rt, fwidth = 0.2, fheight = 0.1, cex = 0.8)
```

```
rt = rpart(formula= minuteCount~., data = filteredData[, c(1,2,3,7,8)])
prettyTree(rt, fwidth = 0.2, fheight = 0.1, cex = 0.8)
plotcp(rt)

rt = rpart(formula= pickupCount~., data = filteredData[, c(1,2,3,7,9)])
prettyTree(rt, fwidth = 0.2, fheight = 0.1, cex = 0.8)
plotcp(rt)
printcp(rt)



#####################
#### MDS & t-SNE ####
#####################

# source: https://distill.pub/2016/misread-tsne/

library(cluster) # for gower similarity and pam
library(Rtsne) # for t-SNE plot

filteredData = combined_df %>% filter(!is.na(deepSleep))
col <- sort(rnorm(66))

# Classical MDS considering separate sleep stages
distances = dist(filteredData[, c(1,2,3,4,5,6,8,9)])
mds = cmdscale(distances)
colnames(mds) = c('x','y')
qplot(x, y, data=as.data.frame(mds), colour=col) +
scale_colour_gradient(low="orange", high="blue") + geom_point() +
geom_text(aes(label=filteredData$observationDate))

# Classical MDS considering only totalSleep
distances = dist(filteredData[, c(1,2,3,7,8,9)])
summary(distances)
mds = cmdscale(distances)
colnames(mds) = c('x','y')
qplot(x, y, data=as.data.frame(mds), colour=col) +
scale_colour_gradient(low="orange", high="blue") + geom_point() +
geom_text(aes(label=filteredData$observationDate))


# t-SNE dimesnion reduction
tsne_out = Rtsne(filteredData[, c(1,2,3,7,8,9)], perplexity = 15, theta
= 0.0, eta = 200, max_iter = 5000, verbose = TRUE)

plot(tsne_out$Y, asp = 1, pch = 20, col = "blue",
     cex = 1.15, cex.axis = 1.25, cex.lab = 1.25, cex.main = 1.5,
     xlab = "t-SNE dimension 1", ylab = "t-SNE dimension 2",
     main = "2D t-SNE projection")
```

```
##############################################
#### PAM Clustering & its Visualizations #####
##############################################

# Calculate silhouette width for many k using PAM
distances = dist(filteredData[, c(1,2,3,4,5,6,8,9)])
#distances = dist(filteredData[, c(1,2,3,7,8,9)])

sil_width <- c(NA)

for(i in 2:20){
  pam_fit <- pam(distances, diss = TRUE, k = i)
  sil_width[i] <- pam_fit$silinfo$avg.width
}

# Plot sihouette width (higher is better)
plot(1:20, sil_width,
     xlab = "Number of clusters",
     ylab = "Silhouette Width")
lines(1:20, sil_width)


# PAM clustering
pam_fit <- pam(distances, diss = TRUE, k = 8)

# t-SNE dimesnion reduction
tsne_obj <- Rtsne(distances, is_distance = TRUE, perplexity = 15, theta
= 0.0, eta = 200, max_iter = 5000, verbose = TRUE)

tsne_data <- tsne_obj$Y %>%
  data.frame() %>%
  setNames(c("X", "Y")) %>%
  mutate(Cluster = factor(pam_fit$clustering))

ggplot(aes(x = X, y = Y), data = tsne_data) +
  geom_point(aes(color = Cluster), size = 2.5)

# classical MDS
another_tsne_data <- mds %>%
  data.frame() %>%
  setNames(c("X", "Y")) %>%
  mutate(Cluster = factor(pam_fit$clustering))

ggplot(aes(x = X, y = Y), data=another_tsne_data) + geom_point(aes(color
= Cluster), size = 5)

ggplot(aes(x = X, y = Y), data=another_tsne_data) + geom_point(aes(color
= Cluster), size = 2.5) +
geom_text(aes(label=filteredData$observationDate), nudge_y = -9)
```

```
#######################################################
#### Linear Regression, Log-linear Regression and    ####
#### Random Forest - Evaluation of Predictive Models ####
#######################################################

#filteredData[, c(1,2,3,4,5,6,8,9)]
#filteredData[, c(1,2,3,7,8,9)]

data = filteredData[, c(1,2,3,4,5,6,8,9)]

set.seed(101)

for (i in 1:100) {

  # randomly pick 70% of the number of observations
  index <- sample(1:nrow(data),size = 0.7*nrow(data))
  # subset to include only the elements in the index
  train <- data[index,]
  # subset to include all but the elements in the index
  test <- data[-index,]

  # linear model
  lm.model <- lm(stressLevel ~ totalSteps + restingHeartRate + deepSleep
+ lightSleep + awakeSleep + pickupCount + minuteCount, data = train)
  lm.test <- predict(lm.model, test)
  RMSE.lm = sqrt(mean((lm.test - test$stressLevel)^2))
  MAE.lm = mean(abs(lm.test - test$stressLevel))

  # log linear model
  loglm.model = lm(log(stressLevel) ~ totalSteps + restingHeartRate +
deepSleep + lightSleep + awakeSleep + pickupCount + minuteCount, data =
train)
  loglm.test <- exp(predict(loglm.model, test))
  RMSE.loglm = sqrt(mean((loglm.test - test$stressLevel)^2))
  MAE.loglm = mean(abs(loglm.test - test$stressLevel))

  # random forest
  rf.model <- randomForest(stressLevel ~ totalSteps + restingHeartRate +
deepSleep + lightSleep + awakeSleep + pickupCount + minuteCount, data =
train, ntree = 10000)
  rf.test <- predict(rf.model, test)
  RMSE.rf = sqrt(mean((rf.test - test$stressLevel)^2))
  MAE.rf = mean(abs(rf.test - test$stressLevel))
}

# Create a data frame with the error metrics for each method
accuracy <- data.frame(Method = c("Linear Regression", "Log Linear
Regression", "Random Forest"), RMSE = c(RMSE.lm, RMSE.loglm, RMSE.rf),
MAE = c(MAE.lm, MAE.loglm, MAE.rf))

# Round the values and print the table
accuracy$RMSE <- round(accuracy$RMSE,2)
accuracy$MAE <- round(accuracy$MAE,2)

accuracy
```