Connecting Time Use to Jobs

STAT288 Final Project

Gian Cercena gcercena@uvm.edu

Nils Dahlin ndahlin@uvm.edu

ABSTRACT

This paper examines how individuals spend their time and how that may be influenced by their job status, job flexibility, and income. The data used was obtained from the 2011 American Time Use Survey, which asks how people in the US spend the time on a daily basis. A linear regression model, PCA, clustering, logistic regression, and random forests were used.

1. INTRODUCTION

Each individual American spends time in different ways. Hobbies, jobs, and families can all affect someone's schedule. In the pursuit to gain insights on what the average American does, and how that connects to each of their income, job flexibility, and job status, we have created models that use data from the American Time Use Survey (ATUS). By examining the data given by the ATUS, we hope that we can find patterns or trends that shed light on how Americans spend their time, and how that may be affected by job status, job flexibility, and income.

2. Data

The data used was obtained from ICPSR, the Inter-university Consortium for Political and Social Research. ICPSR. Specifically, the data is from the 2011 American Time Use Survey. The ATUS is a nationally representative survey conducted by the Bureau of Labor Statistics that measures how people in the United States spend their time on a daily basis. It provides a wealth of information on a variety of topics including employment, leisure activities, and household activities. Each model required a modified dataset to suit the needs of that specific model, so additional cleaning and preprocessing will be described towards the beginning of each model's section.

3. Models

3.1 Linear Regression

The question that our linear regression model was trying to answer was if weekly income could be predicted based on the flexibility of the job you work in. Income was continuous, and had a right skewed distribution as shown in *fig. 1*. The minimum value was -0.01, the maximum value was 2884.61, the median was 700.00 (blue line), the mean was 856.77 (red line), and the standard deviation was 658.16 (green dashed lines). All but one of the predictor variables were categorical and relating to the flexibility of a job for time off due to things such as illnesses, changing work location, changing work hours, and whether or not a job has paid leave. Since this was a phone survey the options for them included "(-3) Refused", "(-2) Don't Know", "(-1) Blank", "(1) Yes", and "(2) No". "(-1) Blank" was recorded in some cases when either the individual refused to answer (as refused isn't used in some questions), or the question wasn't asked for another reason. One other variable was about hours of leave, which ranged from "(-2) Don't Know", and "(-1) Blank", to 88.
Running a linear model across all variables gave an adjusted R-squared of 0.2134, an F-statistic of 12.77 with 123 and 5214 degrees of freedom, a p-value of less than 0.0001, and a root mean squared error of 580.25. Following this we performed a stepwise variable selection in both directions. This dropped our model from 38 variables to 17. The smaller model gave an adjusted R-squared of 0.2133, and F-statistic of 31.78 with 47 and 5290 degrees of freedom, a p-value of less than 0.0001, and a root mean squared error of 577.17.
A small improvement is seen with the simpler model, which is excellent in terms of interpretability, as well as generalization. In both the saturated and simpler model many of the same predictors stood out as important, but were accentuated in the simpler model. The most important predictor was whether or not an individual could take time off from work without pay for the birth or

adoption of a child (LUUNBRTH) with t-values of -3.92 (Don't know), -3.89 (No), -3.79 (Yes), and 2.86 (Blank), all of which had a p-value less than 0.005. This is interesting since it seems that irregardless of whether or not an individual can take off from work, according to our data, it negatively impacts weekly income, with the negative coefficients generally being around -2250. Having the blank option be positive where the others are negative is also intriguing. A naive assumption is that the blanks are mostly males, who tend to have higher incomes on average, as they might not be asked that question as often on the survey, though further analysis would be needed to see if this assumption is correct.

Another stand out variable was whether or not an individual could vary the location they work. A "Yes" answer had a t-value of 3.426, and a p-value of 0.0006. If someone has the freedom to move their work location, the linear model's coefficient gives them an estimated increase of weekly salary of around 404.05. If they answered "No" to having that ability, the coefficient was 105.10, but since the standard error was 117.95, there wasn't any significance to that variable.

3.2 PCA

A Principal Component Analysis was performed on the data used in the linear regression. After turning all of the data into numeric and standardizing it, we plotted *fig.2*, which shows the cumulative variance for each component. If this were to be used in the future for analyzing data at reduced dimensions, the count of principal components that could be optimal to use would be around 7, shown by the dashed plotted line.

3.3 Cluster

When reduced to two dimensions the PCA seems to form 3 groups. Using K-means with 3 clusters on the data reveals *fig.3*. The numbering for each cluster is placed at the center at the respective cluster. Since such distinct clusters formed, a further dive into the differences of them could be insightful. The average salaries in *fig.4* are 560.03, 1079.28, and 1031.01, for cluster 1, 2, and 3 respectively. The large difference between cluster 1 to cluster 2 and 3 indicate that cluster 1 has lower earners on average, but since clusters 2 and 3 have a very similar weekly earning, their difference may be between the flexibility of their jobs. To look into this further 4 variables have been selected. Three were determined to be the most important features when performing the PCA test, and the other helps support the following findings.

The most important feature according to the PCA is whether or not the individual has ever taken unpaid leave off for a list of 7 reasons (LUUNEVR). These reasons included illness, child/eldercare, vacation, errands, or birth/adoption of a child. *fig. 5* shows the percentage answered for each option in each cluster. What stands out is the large amount of blank answers for cluster 2. Those in cluster 1 and 3 seem to have answered the question much more reliably than cluster 2. Since this question is asking whether or not the person has ever taken unpaid leave for various reasons, likely if earlier in the survey the interviewer heard that the interviewee can not take unpaid leave, this question isn't asked, and therefore left blank.

The second is whether or not the individual's employer offers separate paid leave for holidays (LUPTOHOL). Shown in *fig.6*, cluster 1, the lower-earning cluster, has this answered almost always as blank. Clusters 2 and 3 are very similar with this variable, closely matching each other's values.

The third is whether or not the individual receives paid leave at all from their job (LUPAID). *fig.7* shows a very extreme difference between the clusters. 99.7% in cluster 2 and 100% of individuals in cluster 3 have some sort of paid time. Meanwhile, 94.3% of responses from cluster 1 gave no as an answer. This stark difference lines up with the average weekly salaries earlier. It is much less common to get paid leave with lower paying jobs than with higher paid ones. *fig.8* confirms these findings as it shows roughly 90% of cluster 1 having a blank answer for paid leave for eldercare (LUPDEC). Since they answered no to having any paid leave at all, this question wouldn't be answered, and instead is left blank. Clusters 2 and 3 again have similar values.

Cluster 1 seems to be a lower-paid cluster that does not have access to any paid or unpaid leave, cluster 2 is an above average-paid cluster that has access to paid leave, but does not have access to unpaid leave, and cluster 3 is also an above average-paid cluster but has access to both paid and unpaid leave.

3.4 Logistic Regression

Using Logistic Regression our goal was to classify whether an individual was a part or full time employee at their job (full-time=0, part-time=1). We used a collection of numeric and categorical features to train the model on and due to the use of Lasso Regression later on (3.5) and the structure of the variables, One Hot Encoding was necessary to convert the categorical features. The categorical features included whether the individual had certain types of disabilities(difficulty seeing, hearing, walking, dressing, with memory, and running errands), their sex, type of household and the type of ownership, their education level, and a few more. The numerical features included their age, residents in the household, and then all of the specific activities and how much time they spent weekly on each. After examining the data it was apparent that there was a large imbalance of data for part and full-time time workers, full-time workers accounting for ~80% of the data. This led to issues with the logistic regression model later on and predicting an individual's employment level, particularly for part-time individuals. We were unable to effectively address this issue within the

timeframe of the project however this is something to consider moving forward and in future work. We created a training and testing set for the model using an 80-20% split and used this data to train and analyze the model. After fitting our Logistic Regression model we obtained a testing accuracy of 79%, however examining the confusion matrix (*fig.11*) we can see where the majority of errors occur. The misclassification rate for full-time was <1% whereas the misclassification rate for part-time was 84.6%. Looking at *fig.12* we can see that there are a good chunk of points clustered around logit(Y) = 0, and this could be helpful in determining further ways to tune the model or attempt to separate the groups with more or different features. From here we selected all the coefficients with a p-value < .05 and calculated their log odds to determine the effects on employment level. In fig.13&14 we can see these effects over activities and individual info. Interestingly enough it appears that any amount of time spent on any of these activities reduces the probability of an individual being classified as part-time (odds < 1). We also found that the housing unit being a hotel/motel(odds=12.48), being a female (odds=3.24), and not being unionized (odds=1.35) were the only features that increased the probability an individual was part-time.

3.5 Ridge/Lasso

Our logistic regression model was originally trained on 419 features with 54 of which being significant. We attempted to reduce the model complexity using lasso regression and were successfully able to reduce the feature size by 94% down to 26 features. Using cross validation to determine the best lambda (fig.15) we first tried using the best lambda (min = .0053) however the model failed to converge after 1000+ iterations so we used the lambda 1 SE away (1se = .0146). The resulting model only increased the general accuracy about 1% to ~81% and improved the accuracy of full and part-time by <1%. *Fig.16* shows the model's reduction of coefficients over log lambda, the red line showing lambda.min and the green line showing lambda.1se (our selection). *Fig.17* shows the most significant and impactful features from the original 26 (odds < 1.1 and > .9) and here we see that education lower than college increases odds of being part-time whereas higher education is the inverse. We can also see that this model keeps whether someone is a male (odds=.45) rather than whether they are a female and also has issues with running errands as increasing the odds of being part-time(odds=1.27). Lasso regression was successful in reducing our model size/complexity and further tuning should be focused on data imbalances and increasing the specificity of the model (part-time predictions).

3.6 Random Forest Regressor

The random forest regressor attempted to predict weekly income based on the time the individual spent during different activities. There were 382 distinct activities in all. Some of these included sleeping, housework, work, education, consumer purchases, travel, and leisure time. There were 500 trees, and 127 variables were tried at each split. It was able to explain 9.99% of variance, and had a root mean squared error of 615.15, which was more than the 577.17 of the linear model, attesting that job flexibility may be a better predictor of weekly income than time spent. *Fig.9* shows the predicted values versus the actual. The random forest tends to underestimate weekly income, predicting between 500 and 1000 for true values of 2000+. The random forest also achieves an R-squared of 0.8768, which is high despite the low variance explained. This likely means the data is highly nonlinear or complex, which follows with the amount of variables used in this analysis.

Building a lasso random forest with the same data gave an out of bag root mean squared error of 608.73, improving upon the normal random forest, but is still behind the linear model. Its R-squared also increases to 0.9260. According to the lasso model the 7 most important variables are time spent working, sleeping, eating and drinking, watching television and movies (not religious), travel related to working, washing and dressing oneself, and reading for personal interest. *fig.10* shows the top 15 most important variables.

4 Conclusion

Our attempts at supervised learning techniques were not as successful as we originally expected. Our Linear, Logistic, and Random Forest models all accounted for much lower variance than expected although this is most likely due to the imbalances within the data and the amount of missing values for many potential features. The data was collected through in person and phone surveys as well as using notes from daily diaries the individuals kept and so the amount of missing data is going to be higher due to the inability to ensure everyone answers everything. PCA and Clustering using the Principal Components was the method that returned the best results. Using K-Means we were able to separate the data into three fairly distinct groups when plotted along the first two principal components. We used logistic regression to classify individuals as part-time or full-time workers, and found that the model struggled with correctly classifying part-time workers due to the data imbalance in our dataset, and lasso regression was able to marginally improve its accuracy. The random forest found that job flexibility may be a better predictor than time spent. The lasso random forest model achieved better results than the normal random forest, but was still behind the linear model in terms of root mean squared error.

4. fig.s

fig. 1

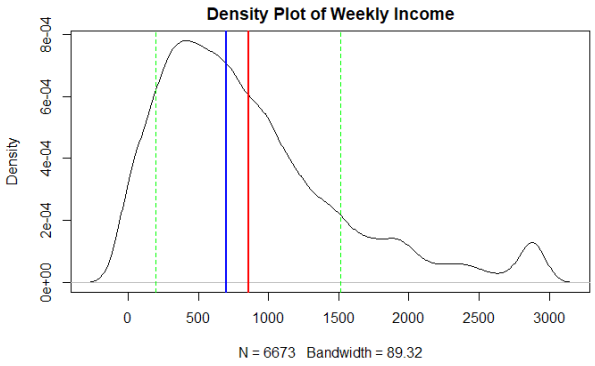


fig. 2

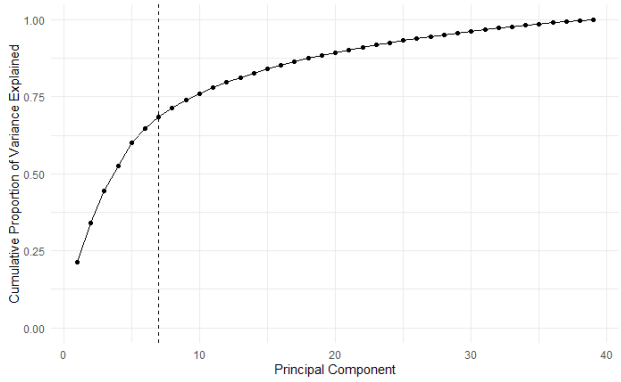


fig. 3

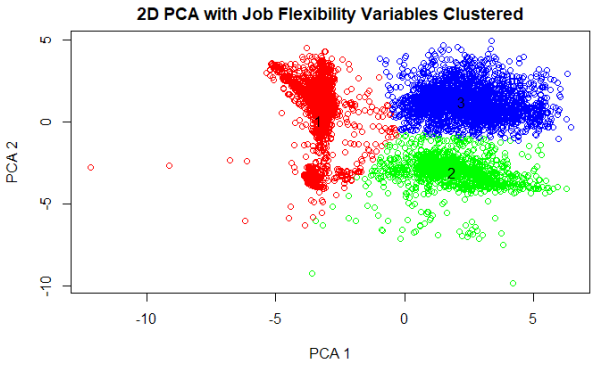


fig. 4

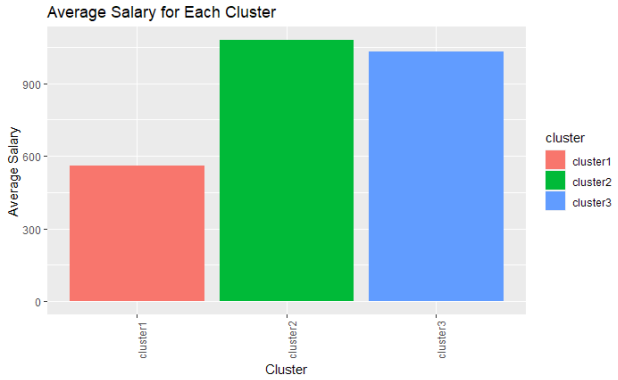


fig. 5

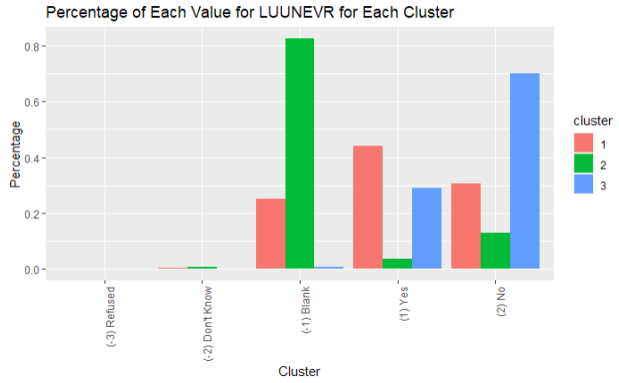


fig. 6

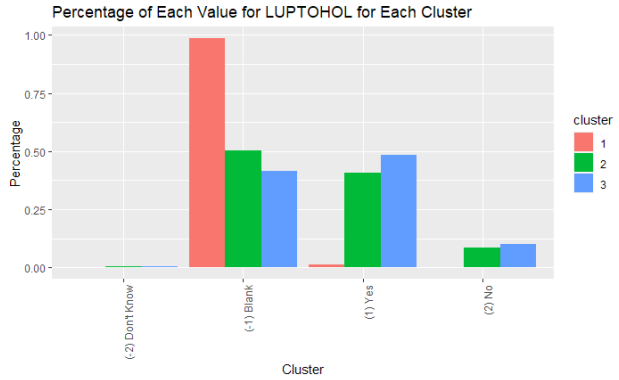


fig. 7

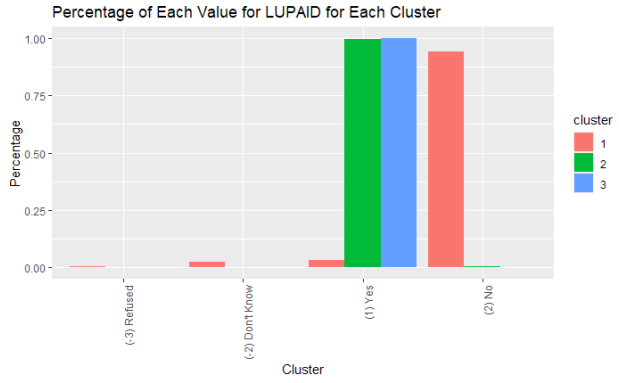


fig. 8

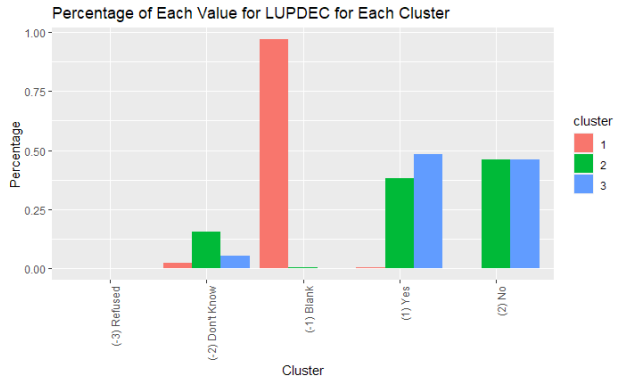


fig. 5

fig. 9

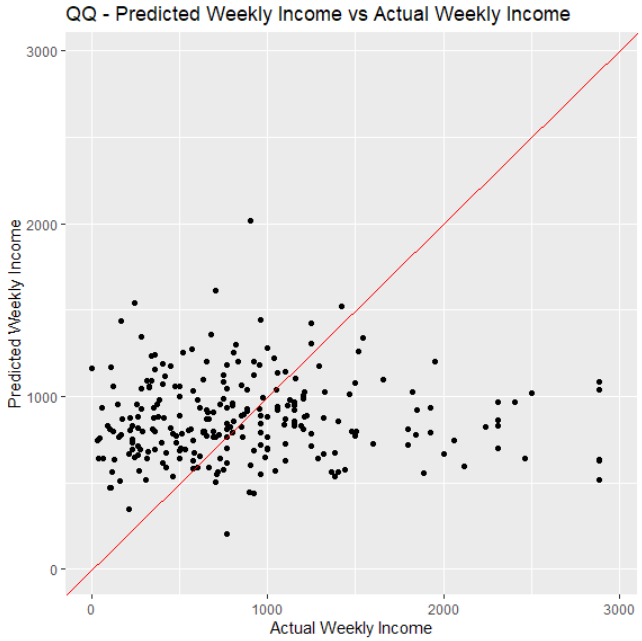


fig. 10

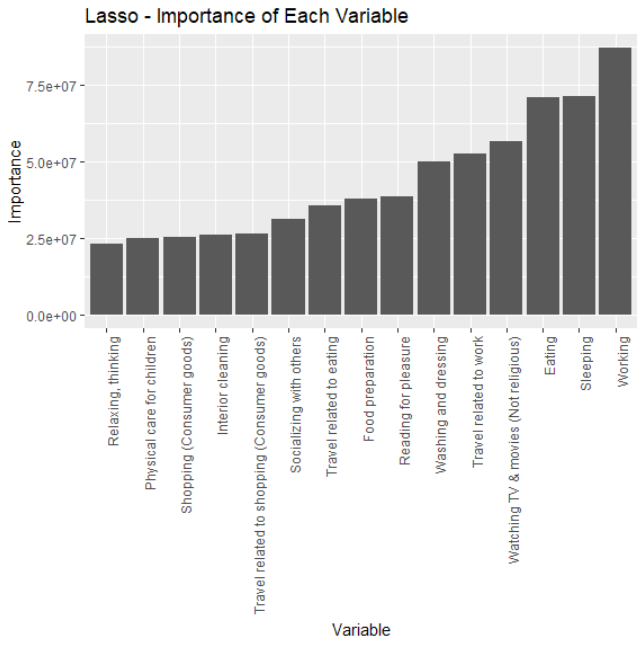
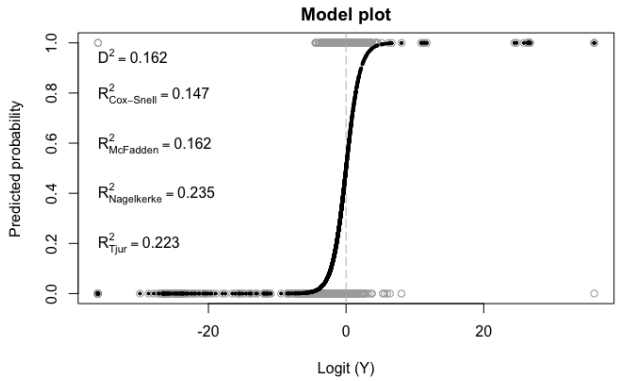


fig.11

Confusion Matrix and Statistics

Prediction	Reference 0	1
0	1154	254
1	59	46

fig12.



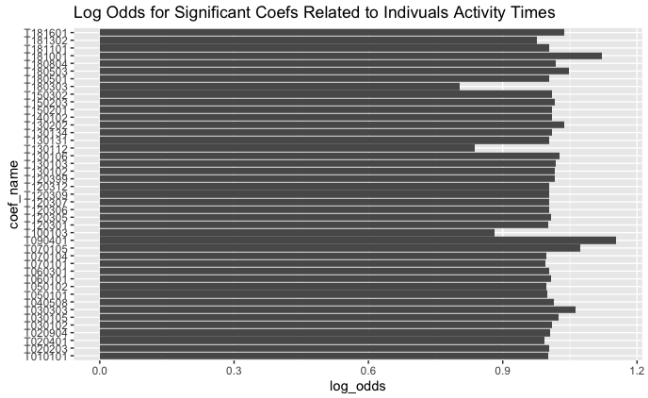


fig13.

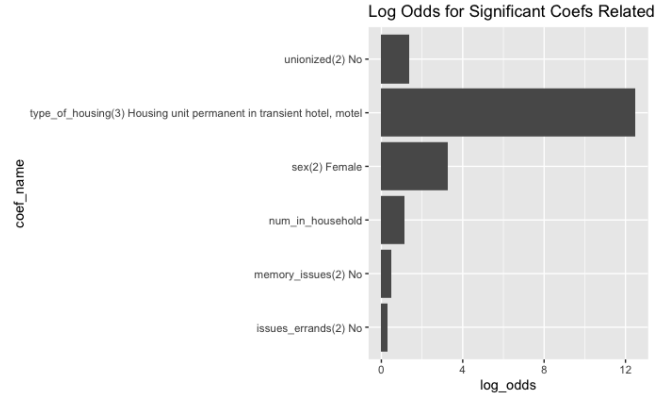


fig14.

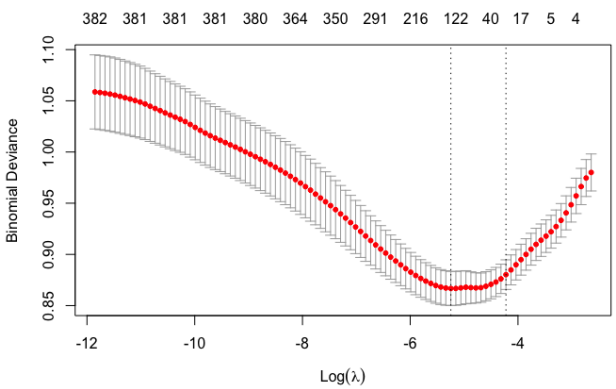
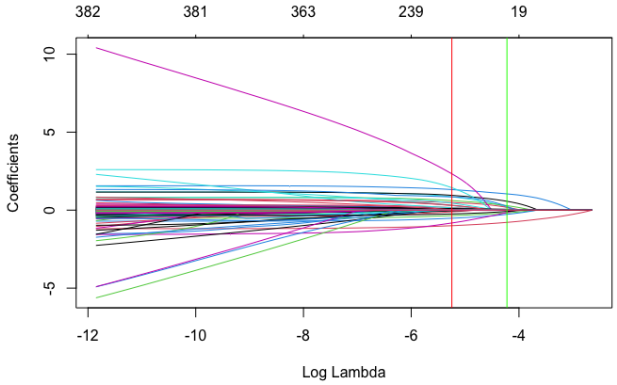


fig15.



fg16.

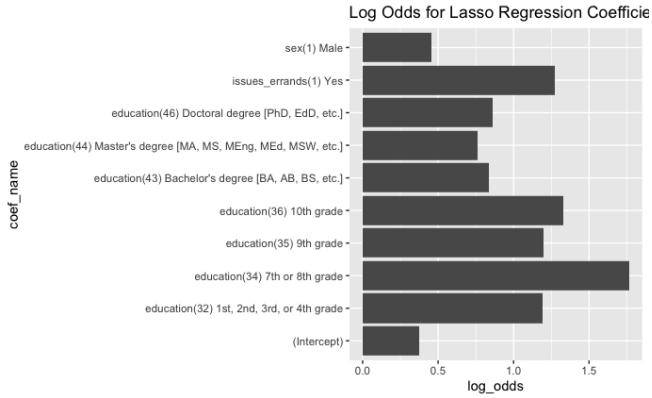


fig17.

````
---
title: "Stat 288 Time Use Analysis"
author: "Nils Dahlin, Gian Cercena"
date: "4/17/2023"
output: html_document
---

# Examining the Time Use of Americans and Predicting Personal Features


## Data Setup
```{r setup, include=FALSE}
knitr::opts_chunk$set(echo = TRUE)

# libraries needed
library(dplyr)
library(tidyverse)
library(rpart)
library(tree)
library(ggplot2)
library(tidyr)
library(randomForest)
library(glmnet)
library(ranger)
library(corrplot)
library(car)
library(pROC)
library(stats)

# activity_dat contains the data/features regarding the
# activities respondents did throughout the week as well
# as related variables
load(file = "34453-0001-Data.rda")
activity_dat_orig <- da34453.0001

# summary_dat contains the data/features regarding the
# individual repsondents, contains features like:
# part/full time status, income, job type, family stats, etc.
load(file = "34453-0008-Data.rda")
````
````

```
summary_dat_orig <- da34453.0008

# surveyq_dat contains responses from individuals to survey questions
# includes questions like: Years of college credit completed?
# Have you looked for work in the past year? How did you get your
# highschool diploma?
load(file = "34453-0004-Data.rda")
surveyq_dat_orig <- da34453.0004

# linear_dat contains the features used in the linear regression
load(file = "34453-0011-Data.rda")
linear_dat_orig <- da34453.0011
```

Pulling Features We Need/Looking at the Raw Data

```{r reducing dataset}
activity_dat <- activity_dat_orig %>%
    select(
        TUCASEID, TUACT_N, TEWHERE, TUACTDUR,
        TUT1CODE, TUT2CODE, TUT3CODE, TRCODE
    )

summary_dat <- summary_dat_orig %>%
    select(
        TUCASEID, TEAGE, TESEX, PEEDUCA, GTMETSTA,
        TELFS, TRDPFTPT, TRERNWA, T010101:T189999
    )

surveyq_dat <- surveyq_dat_orig %>%
    select(
        TUCASEID, HEHOUSUT, HETENURE, HRNUMHOU,
        PEABSRSN, PECYC, PEJHRSN, PEDIPGED, PEAFEVER,
        PEDISREM, PEDISDRS, PEDISEAR, PEDISEYE, PEDISOUT,
        PEDISPHY, PEERNLAB
    )

linear_dat <- linear_dat_orig %>%
    select(
        TUCASEID,
        # LEGNHTH, LEPAIN,
        LRADJ, LUADDY, LUADHR,
        LUADLOC, LULEAVE, LULVHRS, LULVYTD, LUPAID,
        LUPDBRTH, LUPDCC, LUPDEC, LUPDERR, LUPDFMIL,
        LUPDOIL, LUPDVAC, LUPTHOL, LUPTO, LUPTOHOL,
        LUPTOMAT, LUPTOSCK, LUPTPSL, LUPTSCK, LUPTVAC,
        LUUNBRTH, LUUNCC, LUUNEC, LUUNERR, LUUNEVR,
```

```
        LUUNEVR1:LUUNEVR7, LUUNFMIL, LUUNOIL, LUUNPD,
    )

head(activity_dat)
head(summary_dat)
head(surveyq_dat)
head(linear_dat)
```

Renaming Certain Features/Columns
If this code chunk has errors, rerun one above
```{r renaming features}
# in activity_dat the three TUTCODE's are the codes for the
# primary, secondary, and tertiary activity codes which combined
# make the 6 digit activity code for each specific activity measured
# see here https://www.bls.gov/tus/lexicons/lexiconnoex0321.pdf)
activity_dat <- activity_dat %>%
    rename(
        "id" = "TUCASEID",
        "activity_num" = "TUACT_N",
        "location" = "TEWHERE",
        "duration" = "TUACTDUR",
        "code1" = "TUT1CODE",
        "code2" = "TUT2CODE",
        "code3" = "TUT3CODE",
        "activity_code" = "TRCODE"
    )
head(activity_dat)

# renaming non-activity code features
summary_dat <- summary_dat %>%
    rename(
        "id" = "TUCASEID",
        "age" = "TEAGE",
        "sex" = "TESEX",
        "education" = "PEEDUCA",
        "metro_area" = "GTMETSTA",
        "employment_status" = "TELFS",
        "employment_level" = "TRDPFTPT",
        "weekly_income" = "TRERNWA"
    )
head(summary_dat)

surveyq_dat <- surveyq_dat %>%
    rename(
        "id" = "TUCASEID",
        "type_of_housing" = "HEHOUSUT",
```

```
        "ownership_of_home" = "HETENURE",
        "num_in_household" = "HRNUMHOU",
        "missed_work" = "PEABSRSN",
        "yrs_of_collegecred" = "PECYC",
        "how_hs_diploma" = "PEDIPGED",
        "left_last_job" = "PEJHRSN",
        "active_duty" = "PEAFEVER",
        "memory_issues" = "PEDISREM",
        "dressing_issues" = "PEDISDRS",
        "hearing_issues" = "PEDISEAR",
        "eye_issues" = "PEDISEYE",
        "issues_errands" = "PEDISOUT",
        "issues_walking" = "PEDISPHY",
        "unionized" = "PEERNLAB"
    )
head(surveyq_dat)

linear_dat <- linear_dat %>%
    rename(
        "id" = "TUCASEID",
    )

# Add two decimal places to weekly_income
summary_dat$weekly_income <- summary_dat$weekly_income / 100
```
Surveyq Test
```{r testing dat}
dim(surveyq_dat)
surveyq_dat[duplicated(surveyq_dat$TUCASEID), ]

surveyq_dat[]
```


Combining Datasets
need to remove duplicates from surveyq_dat due to yrs_of_collegecred (keep non blank entries for duplicate id's)
```{r combine data}
summary_dat <- summary_dat[!duplicated(summary_dat), ]
surveyq_dat <- surveyq_dat[!duplicated(surveyq_dat), ]
dim(summary_dat)
dim(surveyq_dat)
```


Linear Regression
Set Up

```{r data set up}
# Want to find if flexibility in jobs is correlated with
# the income of the individual

# getting all salary data from summary_dat
salary_dat <- summary_dat %>%
    select(id, weekly_income)

# matching the salary ids and the linear_dat ids
linear_dat <- left_join(linear_dat, salary_dat, by = "id")
dim(linear_dat)

linear_dat <- linear_dat %>%
    select(-id)

# linear_dat$weekly_income <- linear_dat$weekly_income / 100

# removing rows with no income data
linear_dat <- linear_dat[!is.na(linear_dat$weekly_income), ]
dim(linear_dat)

head(linear_dat)
```

```{r}
# making the "(-1) Blank" entries NA
# linear_dat[linear_dat == "(-1) Blank"] <- NA
# linear_dat
# Finding all columns that doesn't have >=2 unique values
# and removing them
# linear_dat <- linear_dat[, sapply(linear_dat, function(x) length(unique(x))) >= 2]
# linear_dat
# # printing the unique values for each column
# for (i in 1:ncol(linear_dat)) {
#     print(unique(linear_dat[, i]))
# }
```

```{r linear train test}
# training/testing data
# setting this seed so the test/train splits in a certain way that will allow
# the linear models to act nicely with categorical data
set.seed(3223232)
lin_train <- sample(1:nrow(linear_dat), 0.8 * nrow(linear_dat))
lin_test <- setdiff(1:nrow(linear_dat), lin_train)

# making training and testing data
```

```
lin_train_dat <- linear_dat[lin_train, ]
lin_test_dat <- linear_dat[lin_test, ]
```

Statistics About the Dataset
```{r}
# average weekly income across entire dataset, as well as
# sd
mean(linear_dat$weekly_income)
median(linear_dat$weekly_income)
sd(linear_dat$weekly_income)
min(linear_dat$weekly_income)
max(linear_dat$weekly_income)
```

```{r}
# density plot of weekly income
plot(density(linear_dat$weekly_income), main = "Density Plot of Weekly Income")
abline(v = mean(linear_dat$weekly_income), col = "red", lwd = 2)
text(mean(linear_dat$weekly_income), 1.2, "Mean", pos = 3, col = "red")
abline(v = median(linear_dat$weekly_income), col = "blue", lwd = 2)
text(median(linear_dat$weekly_income), 0.02, "Median", pos = 3, col = "blue")
sd_val <- sd(linear_dat$weekly_income)
abline(v = mean(linear_dat$weekly_income) + sd_val, col = "green", lty = 2)
text(mean(linear_dat$weekly_income) + sd_val, 0.015, "1 SD", pos = 3, col = "green")
abline(v = mean(linear_dat$weekly_income) - sd_val, col = "green", lty = 2)
text(mean(linear_dat$weekly_income) - sd_val, 0.015, "1 SD", pos = 3, col = "green")
```

Making the Model
```{r}
# making the linear model
head(lin_train_dat)
lm1 <- lm(weekly_income ~ ., data = lin_train_dat)
summary(lm1)
```

```{r}
summary(lm1)$df[1]
```

```{r}
# testing the model
pred <- predict(lm1, lin_test_dat)
mse <- mean((pred - lin_test_dat$weekly_income)^2)
mse
sqrt(mse)
```

```

```{r echo = FALSE results = hide}
# Using stepwise variable selection to make a more interpretable model
# This steps both forwads and back
set.seed()
step_lm1 <- step(lm1, direction = "both")
plot(step_lm1)
```

```{r}
plot(step_lm1$anova$AIC, xlab = "Number of variables", ylab = "AIC")
# abline(v = 21, col = "red", lty = 1)
```

```{r}
step_lm1
# This is the best model according to AIC
lm2 <- lm(
    formula = weekly_income ~ LUADDY + LUADHR + LUADLOC + LUPAID +
        LUPDBRTH + LUPDERR + LUPDOIL + LUPTHOL + LUPTOSCK + LUPTSCK +
        LUPTVAC + LUUNBRTH + LUUNEVR + LUUNEVR4 + LUUNFMIL + LUUNPD,
    data = lin_train_dat
)
```

```{r}
summary(lm2)
summary(lm2)$df[1]
```

```{r}
lm2_tval <- summary(lm2)$coefficients[, "t value"]
lm2_tval_abs <- abs(lm2_tval)
lm2_tval_sorted <- lm2_tval[order(lm2_tval_abs, decreasing = TRUE)]
lm2_tval_sorted
```

```{r}
# testing the model
pred <- predict(lm2, lin_test_dat)
mse <- mean((pred - lin_test_dat$weekly_income)^2)
mse
sqrt(mse)
```

PCA
```

```{r pca}
# turn all data into numeric
lin_pca <- as.data.frame(lapply(linear_dat, as.numeric))
head(lin_pca)
```

```{r}
# standardize the data
set.seed(1)
lin_pca <- scale(lin_pca)
```

```{r}
# run pca over linear_dat to 2 dimensions
set.seed(1)
pca <- prcomp(lin_pca, scale = TRUE)
summary(pca)
```

```{r}
variance_prop <- pca$sdev^2 / sum(pca$sdev^2)
cumulative_prop <- cumsum(variance_prop)
plot_data <- data.frame(
    PC = 1:length(variance_prop),
    Cumulative_prop = cumulative_prop
)

ggplot(plot_data, aes(x = PC, y = Cumulative_prop)) +
    geom_line() +
    geom_point() +
    scale_x_continuous("Principal Component") +
    scale_y_continuous("Cumulative Proportion of Variance Explained",
        limits = c(0, 1)
    ) +
    theme_minimal() +
    geom_vline(xintercept = 7, linetype = "dashed")
```
```{r}
# finding the most important features

# getting the loadings
loadings <- pca$rotation
# getting the absolute value of the loadings
loadings <- abs(loadings)
# getting the sum of the absolute value of the loadings
loadings <- apply(loadings, 1, sum)
# getting the names of the features
```

```
loadings <- data.frame(
    feature = names(loadings),
    loadings = loadings
)

# sorting the loadings
loadings <- loadings[order(-loadings$loadings), ]
loadings
head(loadings)
```

```{r}
# plotting two dimensions
plot(pca$x[, 1], pca$x[, 2],
    main = "Job Flexibility Variables Reduced to Two Dimensions with PCA",
    xlab = "PCA 1",
    ylab = "PCA 2"
)
```

Cluster Analysis on PCA
```{r}
# run cluster analysis on pca
best_seed <- 0
best_kmeans_pca <- NULL
# running a for loop over different set seeds to find the best

for (i in 1:100) {
    set.seed(i)
    kmeans_pca <- kmeans(pca$x[, 1:2], centers = 3)
    if (i == 1) {
        best_kmeans_pca <- kmeans_pca
        best_seed <- i
    } else if (kmeans_pca$tot.withinss < best_kmeans_pca$tot.withinss) {
        best_kmeans_pca <- kmeans_pca
        best_seed <- i
    }
}
best_seed
```

```{r}
set.seed(best_seed)
kmeans_pca <- kmeans(pca$x[, 1:2], centers = 3)
kmeans_pca
```

```{r}
# red yellow green
colors <- c("red", "green", "blue")
# plot the clusters with labels for which cluster they are
plot(pca$x[, 1], pca$x[, 2],
    col = colors[kmeans_pca$cluster],
    main = "2D PCA with Job Flexibility Variables Clustered",
    xlab = "PCA 1",
    ylab = "PCA 2"
)
# putting the labels on the center of the clusters
text(kmeans_pca$centers[, 1], kmeans_pca$centers[, 2], labels = 1:3, col = "black")
```

```{r}
# This and the following code block were originally meant to analyze some data
# but it wasn't meaningful because the data categories weren't ordinal
# The dataset avg_dat_graph is needed though, so these blocks still need to
# be run.

# get the average values in the linear data for each cluster
cluster1 <- linear_dat[kmeans_pca$cluster == 1, ]
cluster2 <- linear_dat[kmeans_pca$cluster == 2, ]
cluster3 <- linear_dat[kmeans_pca$cluster == 3, ]

# convert the cluster columnsn to numeric
cluster1 <- as.data.frame(lapply(cluster1, as.numeric))
cluster2 <- as.data.frame(lapply(cluster2, as.numeric))
cluster3 <- as.data.frame(lapply(cluster3, as.numeric))

# get the average values for each cluster
avg1 <- colMeans(cluster1)
avg2 <- colMeans(cluster2)
avg3 <- colMeans(cluster3)

# combine the averages into a dataframe
avg_dat <- data.frame(avg1, avg2, avg3)

# transpose the dataframe
avg_dat <- t(avg_dat)

# rename the columns
rownames(avg_dat) <- c("cluster1", "cluster2", "cluster3")

avg_dat
```

```{r}
# convert the avg_dat to a dataframe
avg_dat_graph <- as.data.frame(avg_dat)

# add a column for the cluster
avg_dat_graph$cluster <- rownames(avg_dat_graph)

# convert the dataframe to long format
avg_dat_graph <- gather(avg_dat_graph, variable, value, -cluster)

# ggplot(avg_dat_graph, aes(x = variable, y = value, fill = cluster)) +
#   geom_bar(stat = "identity", position = "dodge") +
#       labs(title = "Average Values of Variables for Each Cluster", x =
"Variables", y = "Average Value") +
#     theme(axis.text.x = element_text(angle = 90, hjust = 1))
```

```{r}
# get the average salary for each cluster
avg_sal <- linear_dat %>%
    group_by(kmeans_pca$cluster) %>%
    summarise(avg_sal = mean(weekly_income))

avg_sal

# plot the average salary for each cluster with the colors
ggplot(avg_dat_graph, aes(x = cluster, y = value, fill = cluster)) +
    geom_bar(stat = "identity", position = "dodge") +
    labs(title = "Average Salary for Each Cluster", x = "Cluster", y = "Average
Salary") +
    theme(axis.text.x = element_text(angle = 90, hjust = 1)) +
    scale_color_manual(colors)
```

```{r}
# get average salary across entire dataset
avg_sal_all <- mean(linear_dat$weekly_income)
avg_sal_all
```

Cluster Plotting
```{r}
cluster_1_rows <- linear_dat[kmeans_pca$cluster == 1, ]
cluster_2_rows <- linear_dat[kmeans_pca$cluster == 2, ]
cluster_3_rows <- linear_dat[kmeans_pca$cluster == 3, ]
```

```
linear_dat
```

Top 5 Most Important Features from the PCA
LUUNEVR
```{r}
LUUNEVR_vals <- unique(linear_dat$LUUNEVR)

# get the counts for each value for each cluster
LUUNEVR_1 <- data.frame(table(cluster_1_rows$LUUNEVR))
LUUNEVR_2 <- data.frame(table(cluster_2_rows$LUUNEVR))
LUUNEVR_3 <- data.frame(table(cluster_3_rows$LUUNEVR))

# add the cluster number to each dataframe
LUUNEVR_1 <- mutate(LUUNEVR_1, cluster = 1)
LUUNEVR_2 <- mutate(LUUNEVR_2, cluster = 2)
LUUNEVR_3 <- mutate(LUUNEVR_3, cluster = 3)

# rename the columns
LUUNEVR_1 <- rename(LUUNEVR_1, LUUNEVR = Var1, Freq = Freq)
LUUNEVR_2 <- rename(LUUNEVR_2, LUUNEVR = Var1, Freq = Freq)
LUUNEVR_3 <- rename(LUUNEVR_3, LUUNEVR = Var1, Freq = Freq)

# merge the dataframes
LUUNEVR_dat <- rbind(LUUNEVR_1, LUUNEVR_2, LUUNEVR_3)
LUUNEVR_dat

# change cluster to factor
LUUNEVR_dat$cluster <- as.factor(LUUNEVR_dat$cluster)

# plot adjacent bars for each value for each cluster
ggplot(LUUNEVR_dat, aes(x = LUUNEVR, y = Freq, fill = cluster)) +
    geom_bar(stat = "identity", position = "dodge") +
    labs(title = "Count of Each Value for LUUNEVR for Each Cluster", x = "Cluster",
y = "Count") +
    theme(axis.text.x = element_text(angle = 90, hjust = 1))

# plotting the percentage of each value for LUUNEVR for each cluster
# get the total number of rows for each cluster
LUUNEVR_total_1 <- nrow(cluster_1_rows)
LUUNEVR_total_2 <- nrow(cluster_2_rows)
LUUNEVR_total_3 <- nrow(cluster_3_rows)

# get the percentage of each value for each cluster
LUUNEVR_1 <- mutate(LUUNEVR_1, percent = Freq / LUUNEVR_total_1)
LUUNEVR_2 <- mutate(LUUNEVR_2, percent = Freq / LUUNEVR_total_2)
LUUNEVR_3 <- mutate(LUUNEVR_3, percent = Freq / LUUNEVR_total_3)
```

```
# printing the values to make sure they equal 1
sum(LUUNEVR_1$percent)
sum(LUUNEVR_2$percent)
sum(LUUNEVR_3$percent)

# merge the dataframes
LUUNEVR_dat <- rbind(LUUNEVR_1, LUUNEVR_2, LUUNEVR_3)

# change cluster to factor
LUUNEVR_dat$cluster <- as.factor(LUUNEVR_dat$cluster)

# plot the data
ggplot(LUUNEVR_dat, aes(x = LUUNEVR, y = percent, fill = cluster)) +
    geom_bar(stat = "identity", position = "dodge", group = LUUNEVR_dat$LUUNEVR) +
    labs(title = "Percentage of Each Value for LUUNEVR for Each Cluster", x =
"Cluster", y = "Percentage") +
    theme(axis.text.x = element_text(angle = 90, hjust = 1))
```
LUPTOHOL
```{r}
LUPTOHOL_vals <- unique(linear_dat$LUPTOHOL)

# get the counts for each value for each cluster
LUPTOHOL_1 <- data.frame(table(cluster_1_rows$LUPTOHOL))
LUPTOHOL_2 <- data.frame(table(cluster_2_rows$LUPTOHOL))
LUPTOHOL_3 <- data.frame(table(cluster_3_rows$LUPTOHOL))

# add the cluster number to each dataframe
LUPTOHOL_1 <- mutate(LUPTOHOL_1, cluster = 1)
LUPTOHOL_2 <- mutate(LUPTOHOL_2, cluster = 2)
LUPTOHOL_3 <- mutate(LUPTOHOL_3, cluster = 3)

# rename the columns
LUPTOHOL_1 <- rename(LUPTOHOL_1, LUPTOHOL = Var1, Freq = Freq)
LUPTOHOL_2 <- rename(LUPTOHOL_2, LUPTOHOL = Var1, Freq = Freq)
LUPTOHOL_3 <- rename(LUPTOHOL_3, LUPTOHOL = Var1, Freq = Freq)

# merge the dataframes
LUPTOHOL_dat <- rbind(LUPTOHOL_1, LUPTOHOL_2, LUPTOHOL_3)
LUPTOHOL_dat

# change cluster to factor
LUPTOHOL_dat$cluster <- as.factor(LUPTOHOL_dat$cluster)

# plot adjacent bars for each value for each cluster
ggplot(LUPTOHOL_dat, aes(x = LUPTOHOL, y = Freq, fill = cluster)) +
```

```
    geom_bar(stat = "identity", position = "dodge") +
    labs(title = "Count of Each Value for LUPTOHOL for Each Cluster", x = "Cluster",
y = "Count") +
    theme(axis.text.x = element_text(angle = 90, hjust = 1))

# plotting the percentage of each value for LUPTOHOL for each cluster
# get the total number of rows for each cluster
LUPTOHOL_total_1 <- nrow(cluster_1_rows)
LUPTOHOL_total_2 <- nrow(cluster_2_rows)
LUPTOHOL_total_3 <- nrow(cluster_3_rows)

# get the percentage of each value for each cluster
LUPTOHOL_1 <- mutate(LUPTOHOL_1, percent = Freq / LUPTOHOL_total_1)
LUPTOHOL_2 <- mutate(LUPTOHOL_2, percent = Freq / LUPTOHOL_total_2)
LUPTOHOL_3 <- mutate(LUPTOHOL_3, percent = Freq / LUPTOHOL_total_3)

# printing the values to make sure they equal 1
sum(LUPTOHOL_1$percent)
sum(LUPTOHOL_2$percent)
sum(LUPTOHOL_3$percent)

# merge the dataframes
LUPTOHOL_dat <- rbind(LUPTOHOL_1, LUPTOHOL_2, LUPTOHOL_3)

# change cluster to factor
LUPTOHOL_dat$cluster <- as.factor(LUPTOHOL_dat$cluster)

# plot the data
ggplot(LUPTOHOL_dat, aes(x = LUPTOHOL, y = percent, fill = cluster)) +
    geom_bar(stat = "identity", position = "dodge", group = LUPTOHOL_dat$LUPTOHOL) +
    labs(title = "Percentage of Each Value for LUPTOHOL for Each Cluster", x =
"Cluster", y = "Percentage") +
    theme(axis.text.x = element_text(angle = 90, hjust = 1))
```
LUPAID
```{r}
LUPAID_vals <- unique(linear_dat$LUPAID)

# get the counts for each value for each cluster
LUPAID_1 <- data.frame(table(cluster_1_rows$LUPAID))
LUPAID_2 <- data.frame(table(cluster_2_rows$LUPAID))
LUPAID_3 <- data.frame(table(cluster_3_rows$LUPAID))

# add the cluster number to each dataframe
LUPAID_1 <- mutate(LUPAID_1, cluster = 1)
LUPAID_2 <- mutate(LUPAID_2, cluster = 2)
LUPAID_3 <- mutate(LUPAID_3, cluster = 3)
```

```r
# rename the columns
LUPAID_1 <- rename(LUPAID_1, LUPAID = Var1, Freq = Freq)
LUPAID_2 <- rename(LUPAID_2, LUPAID = Var1, Freq = Freq)
LUPAID_3 <- rename(LUPAID_3, LUPAID = Var1, Freq = Freq)

# merge the dataframes
LUPAID_dat <- rbind(LUPAID_1, LUPAID_2, LUPAID_3)
LUPAID_dat

# change cluster to factor
LUPAID_dat$cluster <- as.factor(LUPAID_dat$cluster)

# plot adjacent bars for each value for each cluster
ggplot(LUPAID_dat, aes(x = LUPAID, y = Freq, fill = cluster)) +
    geom_bar(stat = "identity", position = "dodge") +
    labs(title = "Count of Each Value for LUPAID for Each Cluster", x = "Cluster", y
= "Count") +
    theme(axis.text.x = element_text(angle = 90, hjust = 1))

# plotting the percentage of each value for LUPAID for each cluster
# get the total number of rows for each cluster
LUPAID_total_1 <- nrow(cluster_1_rows)
LUPAID_total_2 <- nrow(cluster_2_rows)
LUPAID_total_3 <- nrow(cluster_3_rows)

# get the percentage of each value for each cluster
LUPAID_1 <- mutate(LUPAID_1, percent = Freq / LUPAID_total_1)
LUPAID_2 <- mutate(LUPAID_2, percent = Freq / LUPAID_total_2)
LUPAID_3 <- mutate(LUPAID_3, percent = Freq / LUPAID_total_3)

# printing the values to make sure they equal 1
sum(LUPAID_1$percent)
sum(LUPAID_2$percent)
sum(LUPAID_3$percent)

# merge the dataframes
LUPAID_dat <- rbind(LUPAID_1, LUPAID_2, LUPAID_3)
LUPAID_dat

# change cluster to factor
LUPAID_dat$cluster <- as.factor(LUPAID_dat$cluster)

# plot the data
ggplot(LUPAID_dat, aes(x = LUPAID, y = percent, fill = cluster)) +
    geom_bar(stat = "identity", position = "dodge", group = LUPAID_dat$LUPAID) +
    labs(title = "Percentage of Each Value for LUPAID for Each Cluster", x =
```

```
"Cluster", y = "Percentage") +
    theme(axis.text.x = element_text(angle = 90, hjust = 1))
```
LUPTSCK
```{r}
LUPTSCK_vals <- unique(linear_dat$LUPTSCK)

# get the counts for each value for each cluster
LUPTSCK_1 <- data.frame(table(cluster_1_rows$LUPTSCK))
LUPTSCK_2 <- data.frame(table(cluster_2_rows$LUPTSCK))
LUPTSCK_3 <- data.frame(table(cluster_3_rows$LUPTSCK))

# add the cluster number to each dataframe
LUPTSCK_1 <- mutate(LUPTSCK_1, cluster = 1)
LUPTSCK_2 <- mutate(LUPTSCK_2, cluster = 2)
LUPTSCK_3 <- mutate(LUPTSCK_3, cluster = 3)

# rename the columns
LUPTSCK_1 <- rename(LUPTSCK_1, LUPTSCK = Var1, Freq = Freq)
LUPTSCK_2 <- rename(LUPTSCK_2, LUPTSCK = Var1, Freq = Freq)
LUPTSCK_3 <- rename(LUPTSCK_3, LUPTSCK = Var1, Freq = Freq)

# merge the dataframes
LUPTSCK_dat <- rbind(LUPTSCK_1, LUPTSCK_2, LUPTSCK_3)
LUPTSCK_dat

# change cluster to factor
LUPTSCK_dat$cluster <- as.factor(LUPTSCK_dat$cluster)

# plot adjacent bars for each value for each cluster
ggplot(LUPTSCK_dat, aes(x = LUPTSCK, y = Freq, fill = cluster)) +
    geom_bar(stat = "identity", position = "dodge") +
    labs(title = "Count of Each Value for LUPTSCK for Each Cluster", x = "Cluster",
y = "Count") +
    theme(axis.text.x = element_text(angle = 90, hjust = 1))

# plotting the percentage of each value for LUPTSCK for each cluster
# get the total number of rows for each cluster
LUPTSCK_total_1 <- nrow(cluster_1_rows)
LUPTSCK_total_2 <- nrow(cluster_2_rows)
LUPTSCK_total_3 <- nrow(cluster_3_rows)

# get the percentage of each value for each cluster
LUPTSCK_1 <- mutate(LUPTSCK_1, percent = Freq / LUPTSCK_total_1)
LUPTSCK_2 <- mutate(LUPTSCK_2, percent = Freq / LUPTSCK_total_2)
LUPTSCK_3 <- mutate(LUPTSCK_3, percent = Freq / LUPTSCK_total_3)
```

```r
# printing the values to make sure they equal 1
sum(LUPTSCK_1$percent)
sum(LUPTSCK_2$percent)
sum(LUPTSCK_3$percent)

# merge the dataframes
LUPTSCK_dat <- rbind(LUPTSCK_1, LUPTSCK_2, LUPTSCK_3)

# change cluster to factor
LUPTSCK_dat$cluster <- as.factor(LUPTSCK_dat$cluster)

# plot the data
ggplot(LUPTSCK_dat, aes(x = LUPTSCK, y = percent, fill = cluster)) +
    geom_bar(stat = "identity", position = "dodge", group = LUPTSCK_dat$LUPTSCK) +
    labs(title = "Percentage of Each Value for LUPTSCK for Each Cluster", x =
"Cluster", y = "Percentage") +
    theme(axis.text.x = element_text(angle = 90, hjust = 1))
```
LUUNFMIL
```{r}
LUUNFMIL_vals <- unique(linear_dat$LUUNFMIL)

# get the counts for each value for each cluster
LUUNFMIL_1 <- data.frame(table(cluster_1_rows$LUUNFMIL))
LUUNFMIL_2 <- data.frame(table(cluster_2_rows$LUUNFMIL))
LUUNFMIL_3 <- data.frame(table(cluster_3_rows$LUUNFMIL))

# add the cluster number to each dataframe
LUUNFMIL_1 <- mutate(LUUNFMIL_1, cluster = 1)
LUUNFMIL_2 <- mutate(LUUNFMIL_2, cluster = 2)
LUUNFMIL_3 <- mutate(LUUNFMIL_3, cluster = 3)

# rename the columns
LUUNFMIL_1 <- rename(LUUNFMIL_1, LUUNFMIL = Var1, Freq = Freq)
LUUNFMIL_2 <- rename(LUUNFMIL_2, LUUNFMIL = Var1, Freq = Freq)
LUUNFMIL_3 <- rename(LUUNFMIL_3, LUUNFMIL = Var1, Freq = Freq)

# merge the dataframes
LUUNFMIL_dat <- rbind(LUUNFMIL_1, LUUNFMIL_2, LUUNFMIL_3)
LUUNFMIL_dat

# change cluster to factor
LUUNFMIL_dat$cluster <- as.factor(LUUNFMIL_dat$cluster)

# plot adjacent bars for each value for each cluster
ggplot(LUUNFMIL_dat, aes(x = LUUNFMIL, y = Freq, fill = cluster)) +
    geom_bar(stat = "identity", position = "dodge") +
```

```
    labs(title = "Count of Each Value for LUUNFMIL for Each Cluster", x = "Cluster",
y = "Count") +
    theme(axis.text.x = element_text(angle = 90, hjust = 1))

# plotting the percentage of each value for LUUNFMIL for each cluster
# get the total number of rows for each cluster
LUUNFMIL_total_1 <- nrow(cluster_1_rows)
LUUNFMIL_total_2 <- nrow(cluster_2_rows)
LUUNFMIL_total_3 <- nrow(cluster_3_rows)

# get the percentage of each value for each cluster
LUUNFMIL_1 <- mutate(LUUNFMIL_1, percent = Freq / LUUNFMIL_total_1)
LUUNFMIL_2 <- mutate(LUUNFMIL_2, percent = Freq / LUUNFMIL_total_2)
LUUNFMIL_3 <- mutate(LUUNFMIL_3, percent = Freq / LUUNFMIL_total_3)

# printing the values to make sure they equal 1
sum(LUUNFMIL_1$percent)
sum(LUUNFMIL_2$percent)
sum(LUUNFMIL_3$percent)

# merge the dataframes
LUUNFMIL_dat <- rbind(LUUNFMIL_1, LUUNFMIL_2, LUUNFMIL_3)

# change cluster to factor
LUUNFMIL_dat$cluster <- as.factor(LUUNFMIL_dat$cluster)

# plot the data
ggplot(LUUNFMIL_dat, aes(x = LUUNFMIL, y = percent, fill = cluster)) +
    geom_bar(stat = "identity", position = "dodge", group = LUUNFMIL_dat$LUUNFMIL) +
    labs(title = "Percentage of Each Value for LUUNFMIL for Each Cluster", x =
"Cluster", y = "Percentage") +
    theme(axis.text.x = element_text(angle = 90, hjust = 1))
```
LUPDEC
```{r}
LUPDEC_vals <- unique(linear_dat$LUPDEC)

# get the counts for each value for each cluster
LUPDEC_1 <- data.frame(table(cluster_1_rows$LUPDEC))
LUPDEC_2 <- data.frame(table(cluster_2_rows$LUPDEC))
LUPDEC_3 <- data.frame(table(cluster_3_rows$LUPDEC))

# add the cluster number to each dataframe
LUPDEC_1 <- mutate(LUPDEC_1, cluster = 1)
LUPDEC_2 <- mutate(LUPDEC_2, cluster = 2)
LUPDEC_3 <- mutate(LUPDEC_3, cluster = 3)
```

```r
# rename the columns
LUPDEC_1 <- rename(LUPDEC_1, LUPDEC = Var1, Freq = Freq)
LUPDEC_2 <- rename(LUPDEC_2, LUPDEC = Var1, Freq = Freq)
LUPDEC_3 <- rename(LUPDEC_3, LUPDEC = Var1, Freq = Freq)

# merge the dataframes
LUPDEC_dat <- rbind(LUPDEC_1, LUPDEC_2, LUPDEC_3)
LUPDEC_dat

# change cluster to factor
LUPDEC_dat$cluster <- as.factor(LUPDEC_dat$cluster)

# plot adjacent bars for each value for each cluster
ggplot(LUPDEC_dat, aes(x = LUPDEC, y = Freq, fill = cluster)) +
    geom_bar(stat = "identity", position = "dodge") +
    labs(title = "Count of Each Value for LUPDEC for Each Cluster", x = "Cluster", y
= "Count") +
    theme(axis.text.x = element_text(angle = 90, hjust = 1))

# plotting the percentage of each value for LUPDEC for each cluster
# get the total number of rows for each cluster
LUPDEC_total_1 <- nrow(cluster_1_rows)
LUPDEC_total_2 <- nrow(cluster_2_rows)
LUPDEC_total_3 <- nrow(cluster_3_rows)

# get the percentage of each value for each cluster
LUPDEC_1 <- mutate(LUPDEC_1, percent = Freq / LUPDEC_total_1)
LUPDEC_2 <- mutate(LUPDEC_2, percent = Freq / LUPDEC_total_2)
LUPDEC_3 <- mutate(LUPDEC_3, percent = Freq / LUPDEC_total_3)

# printing the values to make sure they equal 1
sum(LUPDEC_1$percent)
sum(LUPDEC_2$percent)
sum(LUPDEC_3$percent)

# merge the dataframes
LUPDEC_dat <- rbind(LUPDEC_1, LUPDEC_2, LUPDEC_3)

# change cluster to factor
LUPDEC_dat$cluster <- as.factor(LUPDEC_dat$cluster)

# plot the data
ggplot(LUPDEC_dat, aes(x = LUPDEC, y = percent, fill = cluster)) +
    geom_bar(stat = "identity", position = "dodge", group = LUPDEC_dat$LUPDEC) +
    labs(title = "Percentage of Each Value for LUPDEC for Each Cluster", x =
"Cluster", y = "Percentage") +
    theme(axis.text.x = element_text(angle = 90, hjust = 1))
```

```

## Logisitic Regression - Part/Full-Time
### Set Up
```{r glm setup}
set.seed(0)

# pull columns from survey data
temp <- surveyq_dat %>% select(-yrs_of_collegecred, -how_hs_diploma, -missed_work,
-left_last_job, -active_duty)

# remove duplicates
temp <- temp[!duplicated(temp), ]

# join data with summary data
temp <- left_join(summary_dat, temp, by = "id")
print("temp size:")
dim(temp)

# grab only rows containing Full and Part Time Individuals
logreg_dat <- temp[temp$employment_level == "(1) Full time" | temp$employment_level
== "(2) Part time", ]

# cleaning log reg data before modeling
logreg_dat$employment_level <- factor(logreg_dat$employment_level)
logreg_dat <- logreg_dat[!duplicated(logreg_dat), ]
cols <- c("memory_issues", "dressing_issues", "hearing_issues", "eye_issues",
"issues_errands", "issues_walking")

logreg_dat <- logreg_dat[logreg_dat[cols] != "(-1) Blank", ]
logreg_dat <- logreg_dat[logreg_dat$unionized != "(-1) Blank", ]

logreg_dat <- logreg_dat %>% select(-weekly_income)
logreg_dat <- logreg_dat[!duplicated(logreg_dat), ]
logreg_dat$employment_level <- ifelse(logreg_dat$employment_level=="(1) Full
time",0,1)
```

Testing/Training Data
```{r glm testing training}
# training/testing data
summ_rows <- nrow(logreg_dat)
train_samp <- sample(1:summ_rows, summ_rows * .8)
train <- logreg_dat[train_samp, ] %>% select(-id)
test <- logreg_dat[-train_samp, ] %>% select(-id)

print("train size")
```
```

```
dim(train)
print("test size")
dim(test)
unique(logreg_dat$employment_level)
```

GLM Model
```{r logreg model}
logreg <- glm(formula = employment_level ~ ., data = train, family = "binomial")

test_pred <- predict(logreg, test[, -6], type = "response")
test_pred <- ifelse(test_pred > .5, 1, 0)
actual <- test$employment_level
confmat <- table(test_pred, actual)
confmat
log_acc <- sum(diag(confmat)) / sum(confmat)
log_acc

full_time_misclass <- confmat[2, 1] / sum(confmat[, 1])
print(paste("Full Time Error Rate:", full_time_misclass))
part_time_misclass <- confmat[1, 2] / sum(confmat[, 2])
print(paste("Part Time Error Rate:", part_time_misclass))
print(paste("prop of full time in test:", sum(test$employment_level == 0) /
nrow(test)))
```

Get Test Predictions and Confusion Matrix
```{r}
test_pred <- predict(logreg, test[, -6], type = "response")
test_pred <- ifelse(test_pred > .3, 1, 0)
confmat <- table(test_pred, actual)
confmat
log_acc <- sum(diag(confmat)) / sum(confmat)
log_acc

full_time_misclass <- confmat[2, 1] / sum(confmat[, 1])
print(paste("Full Time Error Rate:", full_time_misclass))
part_time_misclass <- confmat[1, 2] / sum(confmat[, 2])
print(paste("Part Time Error Rate:", part_time_misclass))
# print(paste("prop of full time in test:",sum(test$employment_level=="(1) Full
time")/nrow(test)))
```

of Features in Mod
```{r}
length(logreg$coefficients)
sum(logreg$beta[,1]!=0)
```

```
```

Significant Features and Their Effects
```{r}
pacman::p_load(jtools)

# use this same code process below for the coeffiecients of the lasso model
coef_p <- summ(logreg)$coeftable[,4]
signif <- as.matrix(coef_p[coef_p<.1])
coef_est <- summ(logreg)$coeftable[,1][coef_p<.1]

coef_name <- names(coef_est)
coef_est <- as.numeric(coef_est)
p_vals <- as.numeric(coef_p[coef_p<.1])
signif_coefs = data.frame(coef_name,coef_est,p_vals)
signif_coefs <- na.omit(signif_coefs)
signif_coefs

# set signif_coefs index to coef_name
rownames(signif_coefs) <- signif_coefs$coef_name
#drop coef_name column
#signif_coefs <- signif_coefs[,-1]

signif_coefs$log_odds <- exp(signif_coefs$coef_est)
```

```{r}
# arrange items in signif_coefs by p-value
signif_coefs <- signif_coefs[order(signif_coefs$p_vals),]

# all coef names starting with T are different activities
# split signif_coefs into two dataframes, one for activities and one for everything
else
activities <- signif_coefs[grepl("^T", signif_coefs$coef_name),]
#remove first 5 rows from activities
activities <- activities[-c(1:5),]

indiv_info <- signif_coefs[!grepl("^T", signif_coefs$coef_name),]

```

Lasso Regression
Setting up Data/Training and Testing
```{r}
lasso_dat <- logreg_dat %>% select(-id)

lasso_train <- lasso_dat[train_samp, ]
```

```
lasso_train <- model.matrix(~.,lasso_train)[,-1]

lasso_test <- lasso_dat[-train_samp, ]
lasso_test <-model.matrix(~.,lasso_test)[,-1]


xtrain <- lasso_train[,-36]
xtest <- lasso_test[,-36]
ytrain <- lasso_train[,36]
ytest <- lasso_test[,36]
```

CV for Best Lambda/Lasso Model
CAUTION - THIS CHUNK TAKES A LITTLE WHILE TO RUN
```{r}
cvlasso <- cv.glmnet(xtrain, ytrain, alpha = 1, family = "binomial")
cvlasso$lambda.min
cvlasso$lambda.1se

lasso <- glmnet(xtrain,ytrain,alpha=1,family = "binomial",lambda =
cvlasso$lambda.1se)

```



```{r}
plot(cvlasso)
print(c(log(.01),log(.001)))
plot(lasso)
```

of Features in Model
```{r}
# count the number of coefficients that are not 0 in the lasso model
sum(lasso$beta[,1]!=0)
```

```{r}

```

Lasso Confusion Matrix
```{r}

confusion.glmnet(lasso, xtest, ytest, family = "binomial")
```

```
lasso_pred <- predict.glmnet(lasso, xtest, type = "response")
lasso_pred <- array(ifelse(lasso_pred>.5,1,0))
lasso_act <- array(ytest)
table(lasso_pred,lasso_act)
caret::confusionMatrix(as.factor(lasso_pred),as.factor(lasso_act))
```

Plot Coefficent Estimates over Weight
```{r}
plot(glmnet(xtrain, ytrain,alpha=1,family = "binomial"),xvar="lambda") +
  abline(v=log(cvlasso$lambda.1se),col="green") +
  abline(v=log(cvlasso$lambda.min),col="red")

plot(glmnet(xtrain, ytrain,alpha=1,family = "binomial"),xvar="dev") +
  abline(v=log(cvlasso$lambda.1se),col="green") +
  abline(v=log(cvlasso$lambda.min),col="red")

```

Significant Coeffeciens and Effects
```{r}
# get the coefficients from the lasso model
lasso_coefs <- as.matrix(coef(lasso))

# make dataframe with name of coefficients and their values
lasso_coefs <- data.frame(coef_name = rownames(lasso_coefs), coef_est =
lasso_coefs[,1])
lasso_coefs <- lasso_coefs[lasso_coefs$coef_est!=0,]

#lasso_coefs <- lasso_coefs %>% select(-coef_name)
lasso_coefs
```

```{r}
# add column to lasso_coefs with the log odds ratios, the column should be named
log_odds and should be e to the power of the coef_est
lasso_coefs$log_odds <- exp(lasso_coefs$coef_est)
lasso_log_odds <- lasso_coefs %>% select(log_odds)
```

```{r}
logreg_logodds<-signif_coefs %>% select(log_odds)
logreg_logodds
lasso_log_odds
```

ROC Curves/Visualizations

```{r}
# using ggplot2 plot the log odds ratios for each coefficient for both the lasso and logreg models ordered highest to lowest with the coefficient name on the y axis and the log odds on the x axis
# use the par function to plot both plots side by side

rownames(indiv_info)[2] ="type_of_housing(3) permanent,hotel/motel"
rownames(lasso_coefs)[19] = "Doctoral Degree"
rownames(lasso_coefs)[20] = "Masters Degree"
rownames(lasso_coefs)[21] = "Bachelors Degree"

# remove rows whose log odds are between .9 and 1.1
lasso_coefs <- lasso_coefs[!(lasso_coefs$log_odds<1.1 & lasso_coefs$log_odds>.9),]

# indiv info
ggplot(data = indiv_info, mapping = aes(x=log_odds,y=coef_name)) +
  geom_bar(stat = "identity") +
  ggtitle("Log Odds for Significant Coefs Related to Indivuals Info")
# activity time use
ggplot(data = activities, mapping = aes(x=log_odds,y=coef_name)) +
  geom_bar(stat = "identity") +
  ggtitle("Log Odds for Significant Coefs Related to Indivuals Activity Times")
#lasso
ggplot(data = lasso_coefs, mapping = aes(x=log_odds,y=coef_name)) +
  geom_bar(stat = "identity") +
  ggtitle("Log Odds for Lasso Regression Coefficients")
```

```{r}
```

```{r}
# plot the roc curve for the lasso model
lasso_roc <- roc(ytest, lasso_pred)
plot(lasso_roc)
```

```{r}
threshold <- c(.1,.2,.3,.4,.5,.6,.7,.8,.9)
# iterate through the threshold values and recalculate lasso_pred plotting the roc curve for each threshold
for (i in threshold){
  lasso_pred <- predict.glmnet(lasso, xtest, type = "response")
  lasso_pred <- array(ifelse(lasso_pred>i,1,0))
  lasso_roc <- roc(ytest, lasso_pred)
  plot(lasso_roc)
}
```

```
```{r}
cfit <- cv.glmnet(xtrain, ytrain, family = "binomial", type.measure = "auc",
                  keep = TRUE)
rocs <- roc.glmnet(cfit$fit.preval, newy = ytrain)

best <- cfit$index["min",]
plot(rocs[[best]], type = "l")
invisible(sapply(rocs, lines, col="grey"))
lines(rocs[[best]], lwd = 2,col = "red")
```



```{r}
# use plotGLM to plot the logreg model
modEvA::plotGLM(logreg)
```

```{r}
modEvA::plotGLM(lasso)
```



```{r}
# plot the logreg test data along with the logreg model line
plot(predict(logreg,test,type = "response"), ytest)
abline(logreg)
```

Random Forest
Set up
```{r}
# Remove all NA's and -1 from weekly_income
rf_dat <- summary_dat %>%
    filter(weekly_income != -0.01) %>%
    filter(!is.na(weekly_income))
```

Random Forest Regression

```{r}
# Removing column 1 - 7 from rf_dat
rfr_dat <- rf_dat %>%
    select(-c(1:7))

# Creating the training and testing data sets
set.seed(456)
```
```

```
train_index <- sample(1:nrow(rfr_dat), size = 0.75 * nrow(rfr_dat))
rfr_train <- rfr_dat[train_index, ]
rfr_test <- rfr_dat[-train_index, ]
```

```{r}
# Creating the random forest classifier
rfr_model <- randomForest(weekly_income ~ ., data = rfr_train, ntree = 500)
rfr_model
sqrt(378404.2)
```

```{r}
rfr_importance <- randomForest::importance(rfr_model)
# order the importance
rfr_importance <- rfr_importance[order(rfr_importance[, 1], decreasing = TRUE), ]
length(rfr_importance)
```

```{r}
# Predicting the income of the test data set
rfr_pred <- predict(rfr_model, rfr_test)
# rf_pred
```

```{r}
# Calculate MSE
rfr_mse <- mean((rfr_test$weekly_income - rfr_pred)^2)
rfr_mse
rfr_rmse <- sqrt(rfr_mse)
rfr_rmse

# rfr_test
rfr_sst <- sum((rfr_test$weekly_income - mean(rfr_test$weekly_income))^2)
rfr_sst
rfr_ssr <- sum((rfr_pred - mean(rfr_test$weekly_income))^2)
rfr_ssr
# Calculate R^2
rfr_r2 <- (rfr_ssr / rfr_sst)
rfr_r2
```

```{r}
sampl_num <- 250
# sampling sampl_num random rows from the test data set
# and plotting the predicted weekly income vs the actual weekly income
```

```
rfr_sample <- rfr_test[sample(1:nrow(rfr_test), sampl_num), ]
rfr_sample <- rfr_sample %>%
    mutate(pred = rfr_pred[1:sampl_num])
rfr_sample

# finds the max to set bounds for the plot and increases it by a bit
sampl_max <- max(rfr_sample$weekly_income, rfr_sample$pred) * 1.025
```

```{r, fig.width=6,fig.height=6}
# Plotting the predicted weekly income vs the actual weekly income
ggplot(rfr_sample, aes(x = weekly_income, y = pred)) +
    geom_point() +
    geom_abline(intercept = 0, slope = 1, color = "red") +
    labs(
        title = "QQ - Predicted Weekly Income vs Actual Weekly Income",
        x = "Actual Weekly Income", y = "Predicted Weekly Income"
    ) +
    xlim(c(0, sampl_max)) +
    ylim(c(0, sampl_max))
```

Lasso Regression

```{r}
# Using same training and testing data sets as random forest regression
# Creating the lasso random forest classifer with ranger
lasso_model <- ranger(weekly_income ~ .,
    data = rfr_train, num.trees = 500,
    importance = "impurity", verbose = TRUE
)
lasso_model
sqrt(370554.9)
```

```{r}
# Predicting the income of the test data set
lasso_pred <- predict(lasso_model, rfr_test)$predictions
# lasso_pred
```

```{r}
# Calculate MSE
lasso_mse <- mean((rfr_test$weekly_income - lasso_pred)^2)
lasso_mse
lasso_rmse <- sqrt(lasso_mse)
lasso_rmse
```

```r
# rfr_test
lasso_sst <- sum((rfr_test$weekly_income - mean(rfr_test$weekly_income))^2)
lasso_ssr <- sum((lasso_pred - mean(rfr_test$weekly_income))^2)
# Calculate R^2
lasso_r2 <- 1 - (lasso_ssr / lasso_sst)
lasso_r2
```

```{r}
# importance of each variable
lasso_importance <- lasso_model$variable.importance
# c('double', 'numeric')
# save to csv to sort in python cause I can't figure out how to do it in R
write.csv(lasso_importance, "python/lasso_importance.csv")
```

```{r}
# read in the sorted csv
lasso_importance <- read.csv("python/lasso_importance_sorted.csv")
lasso_importance
```

```{r fig.width=6,fig.height=6}
lasso_importance_20 <- lasso_importance[1:15, ]
lasso_importance_20
# renaming the feature column
renames <- c(
    "Working", "Sleeping", "Eating", "Watching TV & movies (Not religious)",
    "Travel related to work", "Washing and dressing", "Reading for pleasure",
    "Food preparation", "Travel related to eating", "Socializing with others",
    "Travel related to shopping (Consumer goods)",
    "Interior cleaning", "Shopping (Consumer goods)", "Physical care for children",
    "Relaxing, thinking"
)

lasso_importance_20$feature <- renames

# plotting the importance of each variable
ggplot(lasso_importance_20, aes(x = reorder(feature, importance), y = importance)) +
    geom_bar(stat = "identity") +
    labs(
        title = "Lasso - Importance of Each Variable",
        x = "Variable", y = "Importance"
    ) +
    theme(axis.text.x = element_text(angle = 90, hjust = 1))
```

RF Classifier

```{r random forest classifier creating column}
# Creating income brackets
# Dividing weekly_income into 5 brackets
brackets <- quantile(rf_dat$weekly_income, probs = c(0, 0.2, 0.4, 0.6, 0.8, 1))
brackets
```

```{r}
# Add income bracket to rf_dat
rfc_dat <- rf_dat %>%
    mutate(income_bracket = case_when(
        weekly_income <= brackets[2] ~ "very low",
        weekly_income > brackets[2] & weekly_income <= brackets[3] ~ "low",
        weekly_income > brackets[3] & weekly_income <= brackets[4] ~ "medium",
        weekly_income > brackets[4] & weekly_income <= brackets[5] ~ "high",
        weekly_income > brackets[5] ~ "very high"
    ))
# Moving income_bracket to the front of the data frame
rfc_dat <- rfc_dat %>%
    select(income_bracket, everything())
# convert income_bracket to factor
rfc_dat$income_bracket <- as.factor(rfc_dat$income_bracket)
# Order the factor levels for income
rfc_dat$income_bracket <- ordered(rfc_dat$income_bracket,
    levels =
        c("very low", "low", "medium", "high", "very high")
)
# removing columns 2-8
rfc_dat <- rfc_dat %>%
    select(-c(2:8))
```

```{r}
# Print the income bracket types and their factor as well as their counts
table(rfc_dat$income_bracket)
# print total number of observations
nrow(rfc_dat)
# find the total number of variables
ncol(rfc_dat) - 1
```

```{r}
# Remove weekly_income from rfc_dat
rfc_dat <- rfc_dat %>%
```

```
    select(-weekly_income)
```

```{r}
# Creating the training and testing data sets
# set.seed(456)
train_index <- sample(1:nrow(rfc_dat), size = 0.75 * nrow(rfc_dat))
rfc_train <- rfc_dat[train_index, ]
rfc_test <- rfc_dat[-train_index, ]
```

```{r}
# Creating the random forest classifier
rfc_model <- randomForest(income_bracket ~ ., data = rfc_train, ntree = 500)
# rfc_model
```

```{r}
rfc_importance <- randomForest::importance(rfc_model)
# order the importance
rfc_importance <- rfc_importance[order(rfc_importance[, 1], decreasing = TRUE), ]
rfc_importance
```

```{r}
# getting the values from the model
rfc_model$confusion
```

```{r}
plot(rfc_model)
# you can find out which line is which by matching the error rate at the end
# with the classification error
```

```{r}
# Predicting the income bracket of the test data set
rfc_pred <- predict(rfc_model, rfc_test)
# rfc_pred
```

```{r}
# import confusionMatrix
library(caret)
```

```{r}
```

```
# Confusion matrix
conf_mat_rfc <- confusionMatrix(rfc_pred, rfc_test$income_bracket)
conf_mat_rfc
```

```{r}
```

```{r}
rfc_model$confusion
```

```{r}
# find most important features and make plot bigger
varImpPlot(rfc_model, main = "Random Forest Classifier", n.var = 10)
```

```{r}
# Find the average values of each feature for each income bracket
rfc_dat_avg <- rfc_dat %>%
    group_by(income_bracket) %>%
    summarise_all(mean)
rfc_dat_avg
```

```{r}
# export to csv to perform row analysis in python
# since I can't figure it out in R
write.csv(rfc_dat_avg, "rfc_dat_avg.csv")
```

Logistic Regression
```{r Log Reg setup}
# PRFTLF
# Reading in the 4th file
load(file = "34453-0004-Data.rda")
employ_dat_orig <- da34453.0004

employ_dat <- employ_dat_orig %>%
    select(TUCASEID, PRFTLF)



activity_dat <- activity_dat %>%
    select(
        TUCASEID, TUACT_N, TEWHERE, TUACTDUR,
        TUT1CODE, TUT2CODE, TUT3CODE, TRCODE
```

```
    )

summary_dat <- summary_dat %>%
    select(
        TUCASEID, TEAGE, TESEX, PEEDUCA, GTMETSTA,
        TELFS, TRDPFTPT, TRERNWA, T010101:T189999
    )

head(activity_dat)
head(summary_dat)
```