Project 1

Nic Gagliano

2025-02-25

Here is the GitHub Link:

https://github.com/nicgagliano/STATUN3106-Project-1

The following is the code regardless if I did not setup the GitHub properly. GitHub Page includes the proper write up and other materials required for a final project.

```
library(tidyverse)
```

```
## Warning: package 'ggplot2' was built under R version 4.3.3

## Warning: package 'tidyr' was built under R version 4.3.3

## Warning: package 'dplyr' was built under R version 4.3.3

## -- Attaching core tidyverse packages ------------------------ tidyverse 2.0.0 --
## v dplyr     1.1.4     v readr     2.1.4
## v forcats   1.0.0     v stringr   1.5.0
## v ggplot2   3.5.1     v tibble    3.2.1
## v lubridate 1.9.3     v tidyr     1.3.1
## v purrr     1.0.2
## -- Conflicts ------------------------------------------ tidyverse_conflicts() --
## x dplyr::filter() masks stats::filter()
## x dplyr::lag()    masks stats::lag()
## i Use the conflicted package (<http://conflicted.r-lib.org/>) to force all conflicts to become errors
```

```
library(dplyr)
library(ggplot2)
library(patchwork)
```

```
## Warning: package 'patchwork' was built under R version 4.3.3
```

```
library(tidymodels)
```

```
## Warning: package 'tidymodels' was built under R version 4.3.3

## -- Attaching packages -------------------------------------- tidymodels 1.2.0 --
## v broom        1.0.5     v rsample      1.2.1
## v dials        1.3.0     v tune         1.2.1
## v infer        1.0.7     v workflows    1.1.4
## v modeldata    1.4.0     v workflowsets 1.1.0
## v parsnip      1.2.1     v yardstick    1.3.1
## v recipes      1.1.0
## Warning: package 'dials' was built under R version 4.3.3

## Warning: package 'scales' was built under R version 4.3.3

## Warning: package 'infer' was built under R version 4.3.3

## Warning: package 'modeldata' was built under R version 4.3.3

## Warning: package 'parsnip' was built under R version 4.3.3

## Warning: package 'recipes' was built under R version 4.3.3

## Warning: package 'rsample' was built under R version 4.3.3

## Warning: package 'tune' was built under R version 4.3.3

## Warning: package 'workflows' was built under R version 4.3.3

## Warning: package 'workflowsets' was built under R version 4.3.3

## Warning: package 'yardstick' was built under R version 4.3.3

## -- Conflicts ----------------------------------------- tidymodels_conflicts() --
## x scales::discard() masks purrr::discard()
## x dplyr::filter()   masks stats::filter()
## x recipes::fixed()  masks stringr::fixed()
## x dplyr::lag()      masks stats::lag()
## x yardstick::spec() masks readr::spec()
## x recipes::step()   masks stats::step()
## * Use tidymodels_prefer() to resolve common conflicts.
```

```
library(caret)
```

```
## Warning: package 'caret' was built under R version 4.3.3

## Loading required package: lattice
##
## Attaching package: 'caret'
##
## The following objects are masked from 'package:yardstick':
##
##     precision, recall, sensitivity, specificity
##
## The following object is masked from 'package:purrr':
##
##     lift
```

```
library(rjson)
```

```
## Warning: package 'rjson' was built under R version 4.3.3
```

```r
library(jsonlite)
```

```
##
## Attaching package: ’jsonlite’
##
## The following objects are masked from ’package:rjson’:
##
##     fromJSON, toJSON
##
## The following object is masked from ’package:purrr’:
##
##     flatten
```

```r
library(glmnet)
```

```
## Warning: package ’glmnet’ was built under R version 4.3.3
```

```
## Loading required package: Matrix
##
## Attaching package: ’Matrix’
##
## The following objects are masked from ’package:tidyr’:
##
##     expand, pack, unpack
##
## Loaded glmnet 4.1-8
```

```r
library(e1071)
```

```
## Warning: package ’e1071’ was built under R version 4.3.3
```

```
##
## Attaching package: ’e1071’
##
## The following object is masked from ’package:tune’:
##
##     tune
##
## The following object is masked from ’package:rsample’:
##
##     permutations
##
## The following object is masked from ’package:parsnip’:
##
##     tune
```

```r
library(stringr)
library(lubridate)
```

Project 1: Choose your own adventure

```r
TVC <- read.csv("Traffic_Volume_Counts.csv")
names(TVC)
```

```
##  [1] "ID"             "SegmentID"      "Roadway.Name"   "From"
##  [5] "To"             "Direction"      "Date"           "X12.00.1.00.AM"
##  [9] "X1.00.2.00AM"   "X2.00.3.00AM"   "X3.00.4.00AM"   "X4.00.5.00AM"
## [13] "X5.00.6.00AM"   "X6.00.7.00AM"   "X7.00.8.00AM"   "X8.00.9.00AM"
## [17] "X9.00.10.00AM"  "X10.00.11.00AM" "X11.00.12.00PM" "X12.00.1.00PM"
## [21] "X1.00.2.00PM"   "X2.00.3.00PM"   "X3.00.4.00PM"   "X4.00.5.00PM"
## [25] "X5.00.6.00PM"   "X6.00.7.00PM"   "X7.00.8.00PM"   "X8.00.9.00PM"
## [29] "X9.00.10.00PM"  "X10.00.11.00PM" "X11.00.12.00AM"
```

```r
dim(TVC)
```

```
## [1] 42756    31
```

```r
head(TVC)
```

```
##   ID SegmentID Roadway.Name        From               To Direction       Date
## 1  1     15540 BEACH STREET UNION PLACE VAN DUZER STREET        NB 01/09/2012
## 2  2     15540 BEACH STREET UNION PLACE VAN DUZER STREET        NB 01/10/2012
## 3  3     15540 BEACH STREET UNION PLACE VAN DUZER STREET        NB 01/11/2012
## 4  4     15540 BEACH STREET UNION PLACE VAN DUZER STREET        NB 01/12/2012
## 5  5     15540 BEACH STREET UNION PLACE VAN DUZER STREET        NB 01/13/2012
## 6  6     15540 BEACH STREET UNION PLACE VAN DUZER STREET        NB 01/14/2012
##   X12.00.1.00.AM X1.00.2.00AM X2.00.3.00AM X3.00.4.00AM X4.00.5.00AM
## 1             20           10           11           14           13
## 2             21           16            8            6           13
## 3             27           14            6            5           12
## 4             22            7            7            8           11
## 5             31           17            7            5           13
## 6             42           27           21           18           21
##   X5.00.6.00AM X6.00.7.00AM X7.00.8.00AM X8.00.9.00AM X9.00.10.00AM
## 1           20           34           66          100            52
## 2           13           31           70           67            45
## 3           16           34           75           69            71
## 4           12           33           75           89            66
## 5           28           29           68           84            64
## 6           13           17           18           46            53
##   X10.00.11.00AM X11.00.12.00PM X12.00.1.00PM X1.00.2.00PM X2.00.3.00PM
## 1             68             85            85           94          104
## 2             57             67            73           95          102
## 3             67             70            90           89          115
## 4             70             60           105          103           71
## 5             83             89            88          113          113
## 6             29              0            NA           NA           NA
##   X3.00.4.00PM X4.00.5.00PM X5.00.6.00PM X6.00.7.00PM X7.00.8.00PM X8.00.9.00PM
## 1          105          147          120           91           83           74
## 2           98          133          131           95           73           70
## 3          115          130          143          106           89           68
## 4          127          122          144          122           76           64
## 5          126          133          135          102          106           58
## 6           NA           NA           NA           NA           NA           NA
##   X9.00.10.00PM X10.00.11.00PM X11.00.12.00AM
## 1            49             42             42
## 2            63             42             35
## 3            64             56             43
## 4            58             64             43
## 5            58             55             54
## 6            NA             NA             NA
```

After reading in the dataset we make adjustments to the data columns and rows. I changed the name of some columns, specifically I mapped all of the time columns to be just the hour it pertains to, to use more effectively later. I changed the formatting of multiple columns as well for simplicity's sake. I also removed multiple columns that do not have much affect in the work we plan to do, such as "To", "From", and "Direction".

```
TVC <- read.csv("Traffic_Volume_Counts.csv")

TVC <- TVC %>%
  rename(Road = Roadway.Name)
TVC$Date <- as.Date(TVC$Date, format="%m/%d/%Y")

time_map <- c(
  "X12.00.1.00.AM" = "12AM", "X1.00.2.00AM" = "1AM", "X2.00.3.00AM" = "2AM",
  "X3.00.4.00AM" = "3AM", "X4.00.5.00AM" = "4AM", "X5.00.6.00AM" = "5AM",
  "X6.00.7.00AM" = "6AM", "X7.00.8.00AM" = "7AM", "X8.00.9.00AM" = "8AM",
  "X9.00.10.00AM" = "9AM", "X10.00.11.00AM" = "10AM", "X11.00.12.00PM" = "11AM",
  "X12.00.1.00PM" = "12PM", "X1.00.2.00PM" = "1PM", "X2.00.3.00PM" = "2PM",
  "X3.00.4.00PM" = "3PM", "X4.00.5.00PM" = "4PM", "X5.00.6.00PM" = "5PM",
  "X6.00.7.00PM" = "6PM", "X7.00.8.00PM" = "7PM", "X8.00.9.00PM" = "8PM",
  "X9.00.10.00PM" = "9PM", "X10.00.11.00PM" = "10PM", "X11.00.12.00AM" = "11PM"
)
names(TVC) <- recode(names(TVC), !!!time_map)

TVC <- TVC %>%
  mutate(across(8:31, ~replace(as.integer(.), is.na(.), 0)))
TVC[8:31] <- lapply(TVC[8:31], as.integer)
TVC <- TVC %>%
  select(-From, -To, -Direction)

names(TVC)
```

```
##  [1] "ID"        "SegmentID" "Road"      "Date"      "12AM"      "1AM"
##  [7] "2AM"       "3AM"       "4AM"       "5AM"       "6AM"       "7AM"
## [13] "8AM"       "9AM"       "10AM"      "11AM"      "12PM"      "1PM"
## [19] "2PM"       "3PM"       "4PM"       "5PM"       "6PM"       "7PM"
## [25] "8PM"       "9PM"       "10PM"      "11PM"
```

```
head(TVC)
```

```
##   ID SegmentID         Road       Date 12AM 1AM 2AM 3AM 4AM 5AM 6AM 7AM 8AM 9AM
## 1  1      15540 BEACH STREET 2012-01-09   20   10   11   14   13   20   34   66 100   52
## 2  2      15540 BEACH STREET 2012-01-10   21   16    8    6   13   13   31   70   67   45
## 3  3      15540 BEACH STREET 2012-01-11   27   14    6    5   12   16   34   75   69   71
## 4  4      15540 BEACH STREET 2012-01-12   22    7    7    8   11   12   33   75   89   66
## 5  5      15540 BEACH STREET 2012-01-13   31   17    7    5   13   28   29   68   84   64
## 6  6      15540 BEACH STREET 2012-01-14   42   27   21   18   21   13   17   18   46   53
##   10AM 11AM 12PM 1PM 2PM 3PM 4PM 5PM 6PM 7PM 8PM 9PM 10PM 11PM
## 1   68   85   85  94 104 105 147 120  91  83  74  49   42   42
## 2   57   67   73  95 102  98 133 131  95  73  70  63   42   35
## 3   67   70   90  89 115 115 130 143 106  89  68  64   56   43
## 4   70   60  105 103  71 127 122 144 122  76  64  58   64   43
## 5   83   89   88 113 113 126 133 135 102 106  58  58   55   54
## 6   29    0    0   0   0   0   0   0   0   0   0   0    0    0
```

###Different Strategies

Strategy 1: Grouping by month I wanted to see if it would be more effective to look at the roads if they were analyzed by month and year rather than each individual date. This would avoid almost every NA value present, which is not exactly ideal, but we can evaluate fluctuations by the scaled differences instead of 0's.

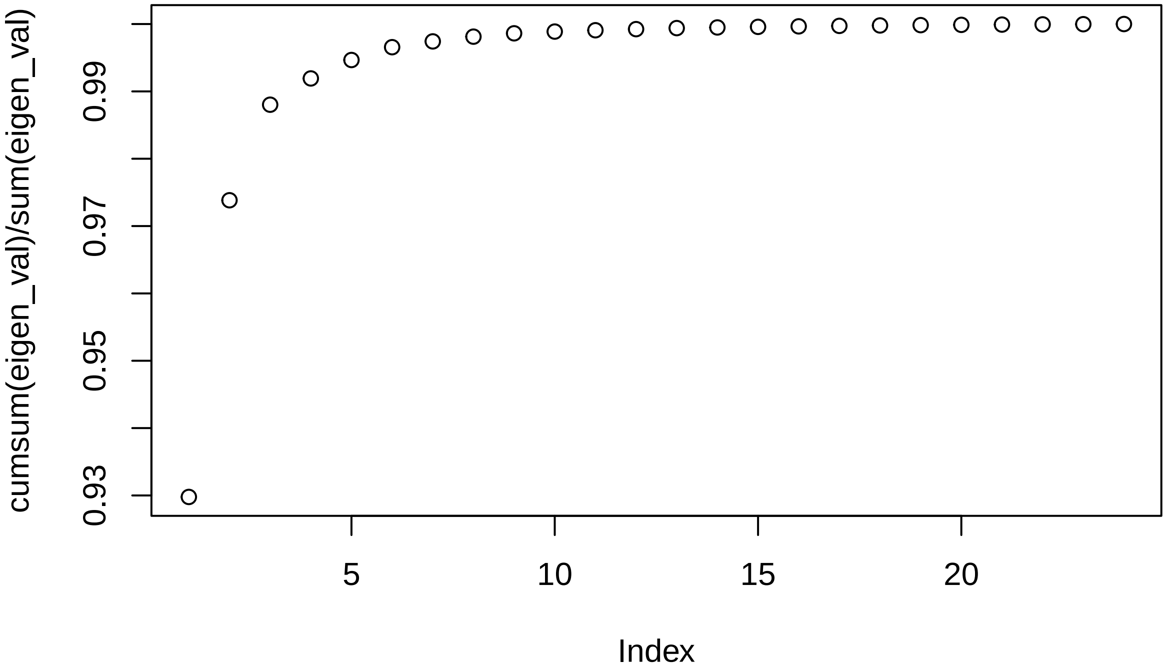
```
TVC_grouped <- TVC
TVC_grouped$YearMonth <- format(TVC_grouped$Date, "%Y-%m")

TVC_grouped <- TVC_grouped %>%
  select(-ID, -Road) %>%
  group_by(SegmentID, YearMonth) %>%
  summarise(across(`12AM`:`11PM`, \(x) sum(x, na.rm = TRUE))) %>%
  arrange(SegmentID)
```

```
## ‘summarise()‘ has grouped output by 'SegmentID'. You can override using the
## ‘.groups‘ argument.
```

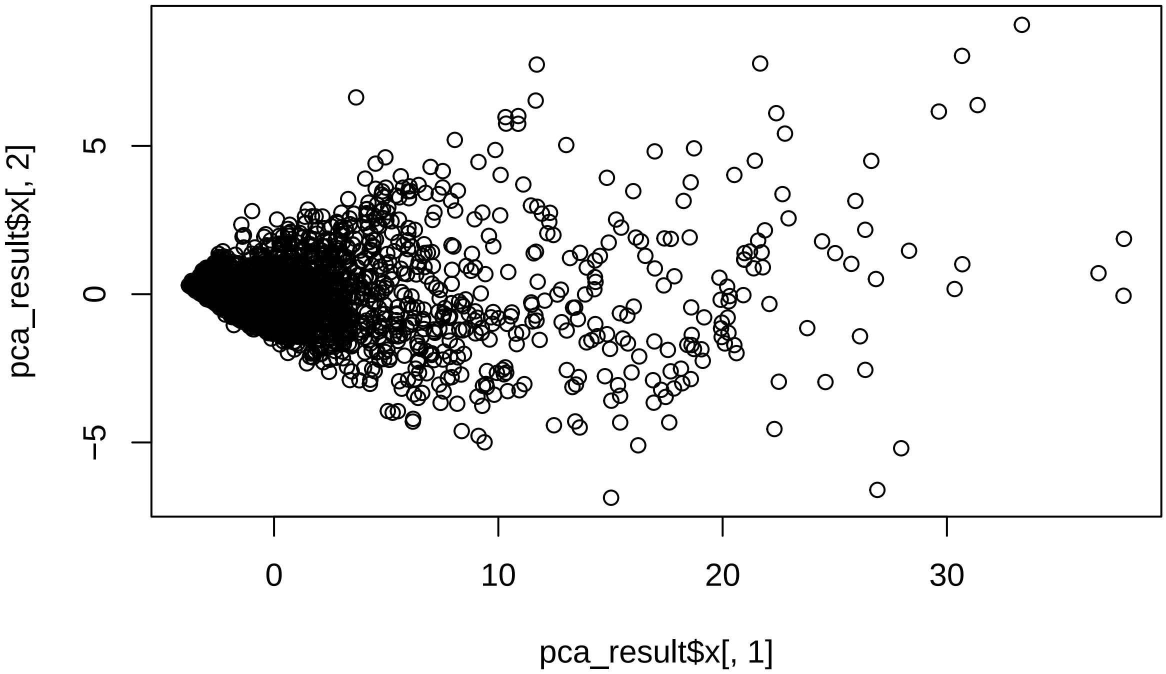
```
TVC_grouped
```

```
## # A tibble: 4,091 x 26
## # Groups:   SegmentID [1,956]
##    SegmentID YearMonth ‘12AM‘ ‘1AM‘ ‘2AM‘ ‘3AM‘ ‘4AM‘ ‘5AM‘ ‘6AM‘ ‘7AM‘ ‘8AM‘
##        <int> <chr>      <int> <int> <int> <int> <int> <int> <int> <int> <int>
##  1       202 2014-10      305   222   137   138   114   155   283   355   485
##  2       646 2012-01      226   203   152   131   211   415   778  1074  1462
##  3      1416 2015-10      892   523   297   251   435   942  2034  3776  4142
##  4      1416 2015-11      219   122    52    62    40    30    90   135   290
##  5      1421 2012-01      430   242   174   152   201   440  1212  2524  2827
##  6      1883 2012-01      114    61    38    13    39   105   269   845   704
##  7      1883 2015-10      875   550   319   196   334   834  2043  4401  4023
##  8      1883 2015-11      230   157    51    46    37    29    99   195   327
##  9      1883 2020-11      580   321   164   154   204   425  1309  2421  3475
## 10      1884 2012-01      105    57    47    18    35   107   290   790   663
## # i 4,081 more rows
## # i 15 more variables: ‘9AM‘ <int>, ‘10AM‘ <int>, ‘11AM‘ <int>, ‘12PM‘ <int>,
## #   ‘1PM‘ <int>, ‘2PM‘ <int>, ‘3PM‘ <int>, ‘4PM‘ <int>, ‘5PM‘ <int>,
## #   ‘6PM‘ <int>, ‘7PM‘ <int>, ‘8PM‘ <int>, ‘9PM‘ <int>, ‘10PM‘ <int>,
## #   ‘11PM‘ <int>
```

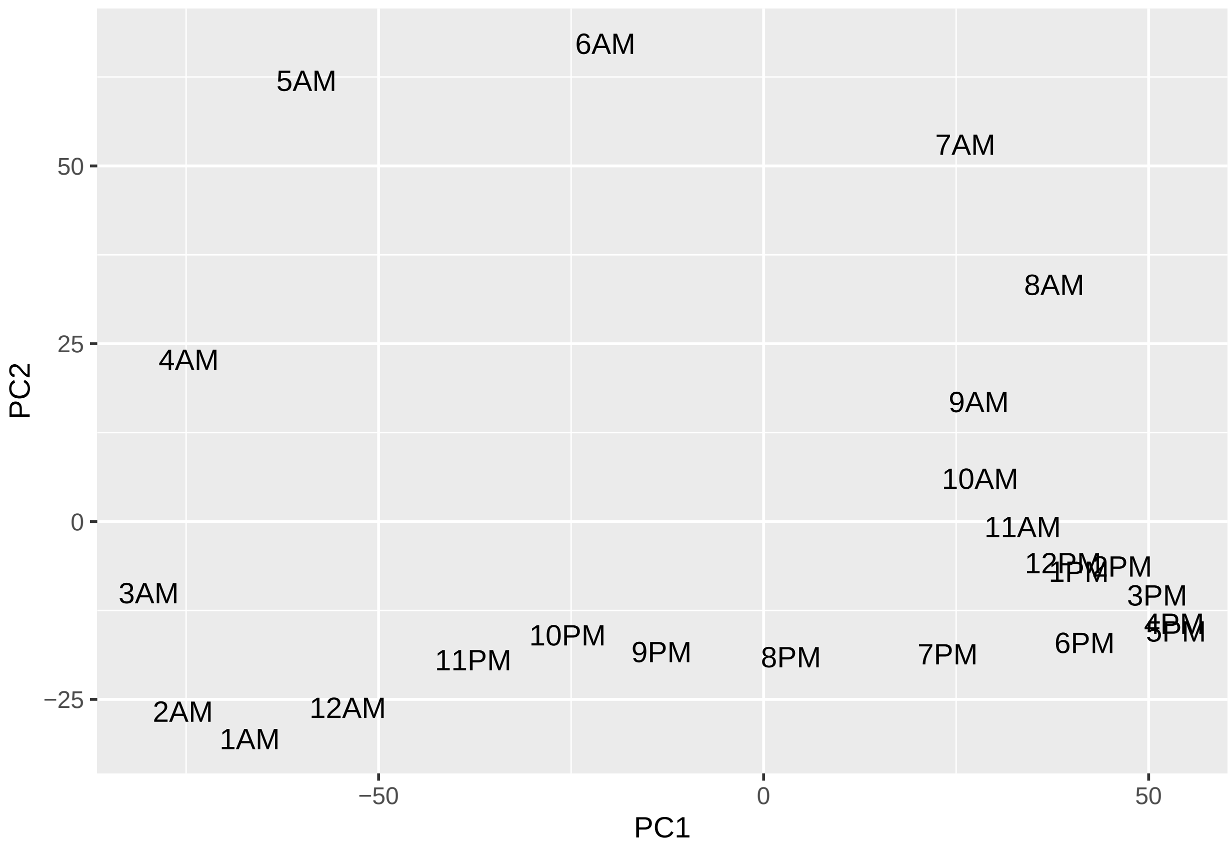
```
scaled_data <- scale(TVC_grouped[, (ncol(TVC_grouped) - 23):ncol(TVC_grouped)])

pca_result <- prcomp(scaled_data, center = TRUE, scale. = TRUE)

eigen_val <- pca_result$sdev^2
plot(cumsum(eigen_val) / sum(eigen_val))
abline(h=.9)
```



```
plot(pca_result$x[,1], pca_result$x[,2])
```



```
pca_result <- prcomp(t(scaled_data), center = TRUE, scale. = TRUE)
pca.data <- data.frame(Sample = rownames(pca_result$x), X = pca_result$x[,1], Y = pca_result$x[,2])

ggplot(data = pca.data, aes(x = X, y = Y, label = Sample)) +
  geom_text() +
  xlab(paste("PC1")) +
  ylab(paste("PC2"))
```



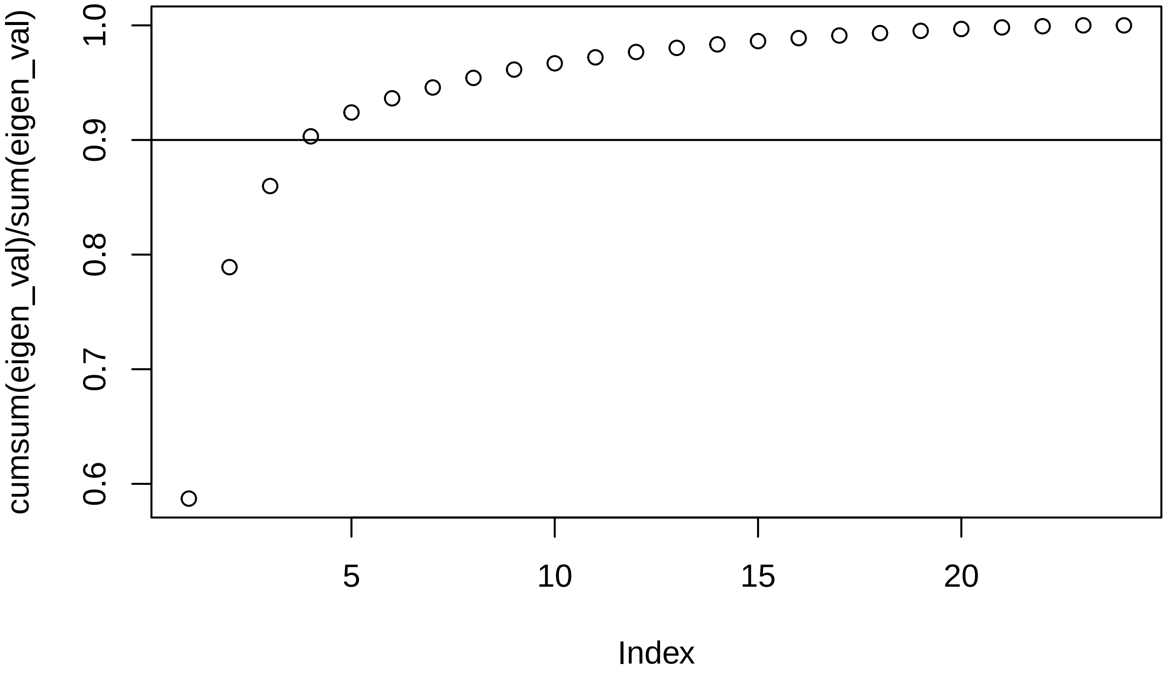
Now this graph, and every other graph preceding this, does not do what I want it to do. This is comparing times to other times rather than roads. Obviously rush hour is so compactly together while slowly moving away is every hour preceding it. I want this to be showing roads. There are a lot of unique road segments however. Making this visually work would be extremely difficult.

Strategy 2: Filtering to Zero Dates What I mean by this strategy title is this set of practice is finding all data that a majority are 0's in the row. Then, grabbing the dates of those and filtering the original dataset to only include those dates. This does get us closer to our goal in terms of showing relations between closed roads and other non-zero roads on the same day. However again, this is still showing the relationship of time instead of road.

```r
traffic_columns <- names(TVC)[(ncol(TVC) - 23):ncol(TVC)]

TVC_clean <- TVC
TVC_clean$zero_count <- rowSums(TVC_clean[traffic_columns] == 0)

threshold <- 0.75 * length(traffic_columns)
rows_with_zeros <- TVC_clean %>%
  filter(zero_count > threshold)

dates_with_zeros <- rows_with_zeros$Date

TVC_zeros_dates <- TVC_clean %>%
  filter(Date %in% dates_with_zeros)

TVC_zeros_dates <- TVC_zeros_dates %>%
  mutate(Closed_Road = ifelse(zero_count > threshold, 1, 0)) %>%
  select(ID, SegmentID, Road, Date, Closed_Road, everything())

closed_roads_data <- TVC_zeros_dates %>% filter(Closed_Road == 1)
merged_data <- TVC %>%
  filter(Date %in% closed_roads_data$Date)
traffic_data <- merged_data[(ncol(TVC) - 23):ncol(TVC)]
traffic_matrix <- as.matrix(traffic_data)
rownames(traffic_matrix) <- merged_data$ID

traffic_matrix_clean_rows <- traffic_matrix[apply(traffic_matrix, 1, function(row) sum(row != 0) > 0), ]
traffic_matrix_clean <- traffic_matrix_clean_rows[, apply(traffic_matrix_clean_rows, 2, function(col) su

pca_result <- prcomp(t(traffic_matrix_clean), scale. = TRUE)
summary(pca_result)
```

```
## Importance of components:
##                            PC1     PC2     PC3     PC4     PC5     PC6     PC7
## Standard deviation     26.5430 15.5673 9.21435 7.21379 4.99951 3.84939 3.36919
## Proportion of Variance  0.5871  0.2019 0.07075 0.04337 0.02083 0.01235 0.00946
## Cumulative Proportion   0.5871  0.7891 0.85981 0.90318 0.92401 0.93636 0.94581
##                            PC8     PC9    PC10    PC11    PC12   PC13    PC14
## Standard deviation     3.16598 2.95326 2.55818 2.50409 2.36558 2.0789 1.91657
## Proportion of Variance 0.00835 0.00727 0.00545 0.00523 0.00466 0.0036 0.00306
## Cumulative Proportion  0.95417 0.96144 0.96689 0.97211 0.97678 0.9804 0.98344
##                           PC15    PC16    PC17    PC18    PC19    PC20    PC21
## Standard deviation     1.84485 1.76914 1.64608 1.59147 1.53148 1.40356 1.27873
## Proportion of Variance 0.00284 0.00261 0.00226 0.00211 0.00195 0.00164 0.00136
## Cumulative Proportion  0.98628 0.98888 0.99114 0.99325 0.99521 0.99685 0.99821
##                           PC22    PC23      PC24
## Standard deviation     1.10905 0.95662 7.836e-15
## Proportion of Variance 0.00102 0.00076 0.000e+00
## Cumulative Proportion  0.99924 1.00000 1.000e+00
```

```
eigen_val <- pca_result$sdev^2
plot(cumsum(eigen_val) / sum(eigen_val))
abline(h=.9)
```



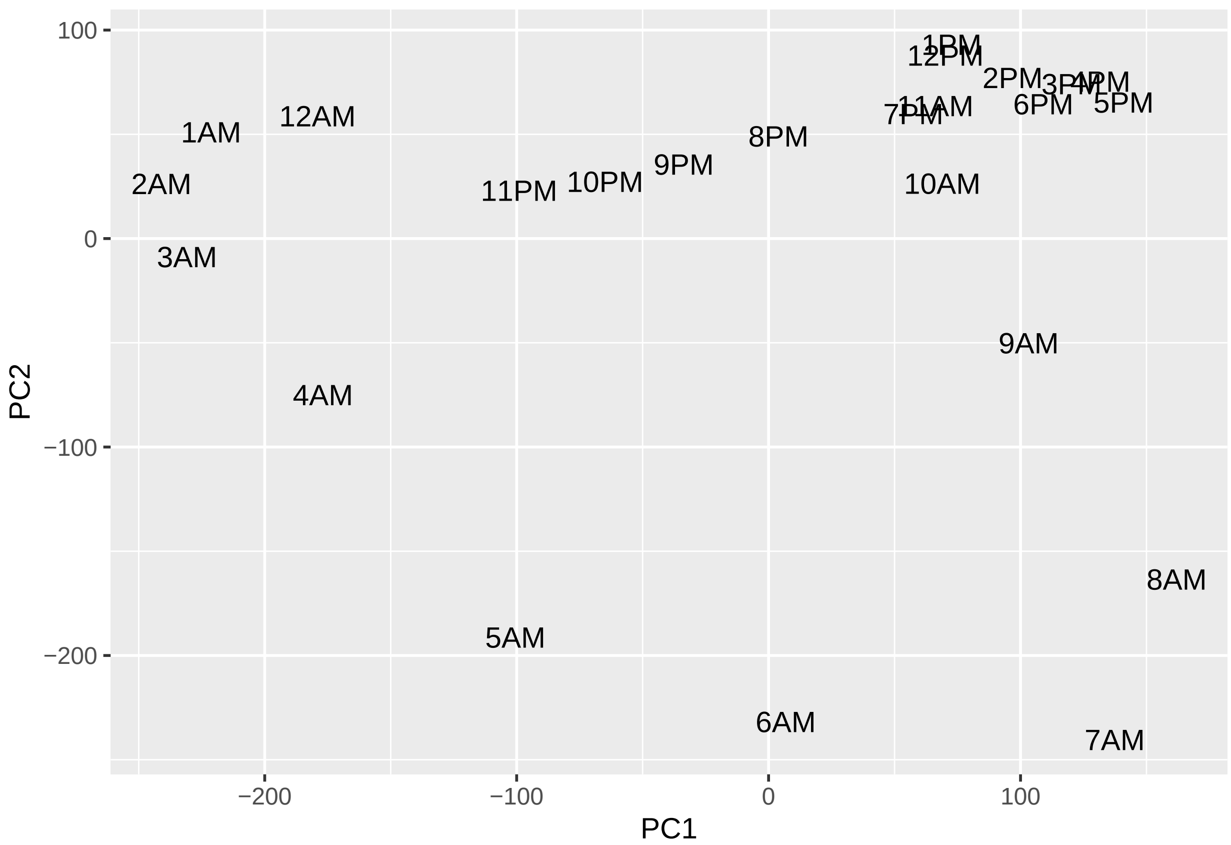
Strategy 3: Simple Correlation Matrix I wanted to see what the basic correlation matrix of this data would look like. Again this is still between times instead of roads, but we can see this definitely would not work out to show per road, as it is already a massive matrix for just the time 24 time slots.

```r
scaled_data <- scale(TVC[, (ncol(TVC) - 23):ncol(TVC)])

closed_roads <- rowSums(TVC[, (ncol(TVC) - 23):ncol(TVC)] == 0) > 0

cor_matrix <- cor(cbind(scaled_data, closed_roads))
cor_matrix
```

```
##                     12AM         1AM        2AM        3AM         4AM
## 12AM          1.00000000  0.97469407  0.9339893  0.9123543  0.86413468
## 1AM           0.97469407  1.00000000  0.9772992  0.9520508  0.86257068
## 2AM           0.93398927  0.97729919  1.0000000  0.9781703  0.87811049
## 3AM           0.91235426  0.95205084  0.9781703  1.0000000  0.93410260
## 4AM           0.86413468  0.86257068  0.8781105  0.9341026  1.00000000
## 5AM           0.70143182  0.65211641  0.6427073  0.7217249  0.89453785
## 6AM           0.66712973  0.59653922  0.5711865  0.6322058  0.80269670
## 7AM           0.67820636  0.60320574  0.5721327  0.6168150  0.76553382
## 8AM           0.72636629  0.65201003  0.6150973  0.6509062  0.77924707
## 9AM           0.79483973  0.72500744  0.6817802  0.7119001  0.81864309
## 10AM          0.83909255  0.77421701  0.7306602  0.7527553  0.83731921
## 11AM          0.85455159  0.79234826  0.7484548  0.7656903  0.83759006
## 12PM          0.85602406  0.79349240  0.7497973  0.7658259  0.83388192
## 1PM           0.85307853  0.78879564  0.7439193  0.7596115  0.82976683
## 2PM           0.83799158  0.76969566  0.7232766  0.7395189  0.81434522
## 3PM           0.82462432  0.75541169  0.7086610  0.7239542  0.79849431
## 4PM           0.81514678  0.74592079  0.6992087  0.7143935  0.78970497
## 5PM           0.81205089  0.74306236  0.6960858  0.7103966  0.78493622
## 6PM           0.82819420  0.76016266  0.7132060  0.7267654  0.79815373
## 7PM           0.85788797  0.79193794  0.7446331  0.7573179  0.82261882
## 8PM           0.89149530  0.82965724  0.7817508  0.7915980  0.84713031
## 9PM           0.91105798  0.85333564  0.8056455  0.8134853  0.86109603
## 10PM          0.92190475  0.86877081  0.8216783  0.8267327  0.86527593
## 11PM          0.93062603  0.88442494  0.8385232  0.8370439  0.85728732
## closed_roads -0.07644587 -0.07415399 -0.0753054 -0.0703446 -0.07046845
##                      5AM         6AM         7AM         8AM         9AM
## 12AM          0.70143182  0.66712973  0.67820636  0.72636629  0.79483973
## 1AM           0.65211641  0.59653922  0.60320574  0.65201003  0.72500744
## 2AM           0.64270729  0.57118645  0.57213266  0.61509729  0.68178018
## 3AM           0.72172492  0.63220580  0.61681498  0.65090618  0.71190007
## 4AM           0.89453785  0.80269670  0.76553382  0.77924707  0.81864309
## 5AM           1.00000000  0.94279923  0.87905724  0.86541998  0.87377147
## 6AM           0.94279923  1.00000000  0.96651394  0.94142764  0.91915077
## 7AM           0.87905724  0.96651394  1.00000000  0.98043914  0.93992958
## 8AM           0.86541998  0.94142764  0.98043914  1.00000000  0.97164675
## 9AM           0.87377147  0.91915077  0.93992958  0.97164675  1.00000000
## 10AM          0.85335496  0.87825863  0.89425878  0.93424218  0.98151643
## 11AM          0.83291906  0.85100425  0.86789229  0.91252590  0.96547922
## 12PM          0.82254293  0.83905009  0.85723122  0.90208799  0.95566221
## 1PM           0.82183509  0.84009114  0.85884186  0.90231313  0.95292122
## 2PM           0.81615791  0.84404813  0.86881053  0.91010551  0.95173372
## 3PM           0.80392311  0.83708706  0.86704929  0.91009787  0.94495472
## 4PM           0.79726416  0.83158930  0.86316672  0.90565510  0.93939130
## 5PM           0.79578992  0.83233097  0.86548272  0.90710849  0.93922409
## 6PM           0.80450638  0.83798091  0.86710349  0.90804921  0.94293925
## 7PM           0.81593623  0.83968835  0.86253066  0.90357744  0.94409327
## 8PM           0.82011802  0.83141470  0.84984189  0.89094391  0.93714377
## 9PM           0.82113813  0.82493424  0.83987409  0.87963990  0.92691264
## 10PM          0.81297911  0.81272003  0.82562349  0.86499964  0.91293986
## 11PM          0.78135091  0.77438149  0.78972571  0.82956440  0.87741344
## closed_roads -0.06598521 -0.07100886 -0.07824962 -0.08365954 -0.09609481
##                     10AM       11AM       12PM        1PM        2PM        3PM
## 12AM           0.8390925  0.8545516  0.8560241  0.8530785  0.8379916  0.8246243
## 1AM            0.7742170  0.7923483  0.7934924  0.7887956  0.7696957  0.7554117
## 2AM            0.7306602  0.7484548  0.7497973  0.7439193  0.7232766  0.7086610
## 3AM            0.7527553  0.7656903  0.7658259  0.7596115  0.7395189  0.7239542
## 4AM            0.8373192  0.8375901  0.8338819  0.8297668  0.8143452  0.7984943
## 5AM            0.8533550  0.8329191  0.8225429  0.8218351  0.8161579  0.8039231
## 6AM            0.8782586  0.8510042  0.8390501  0.8400911  0.8440481  0.8370871
## 7AM            0.8942588  0.8678923  0.8572312  0.8588419  0.8688105  0.8670493
## 8AM            0.9342422  0.9125259  0.9020880  0.9023131  0.9101055  0.9100979
## 9AM            0.9815164  0.9654792  0.9556622  0.9529212  0.9517337  0.9449547
## 10AM           1.0000000  0.9904390  0.9818578  0.9775093  0.9705688  0.9590169
## 11AM           0.9904390  1.0000000  0.9916171  0.9867664  0.9777645  0.9649564
## 12PM           0.9818578  0.9916171  1.0000000  0.9929089  0.9834676  0.9703570
## 1PM            0.9775093  0.9867664  0.9929089  1.0000000  0.9902848  0.9769795
## 2PM            0.9705688  0.9777645  0.9834676  0.9902848  1.0000000  0.9881302
## 3PM           0.9590169  0.9649564  0.9703570  0.9769795  0.9881302  1.0000000
## 4PM           0.9515860  0.9567508  0.9621928  0.9682953  0.9792115  0.9909914
## 5PM           0.9501128  0.9542758  0.9584074  0.9642366  0.9747676  0.9838253
## 6PM           0.9554203  0.9583361  0.9618355  0.9665990  0.9748313  0.9805192
## 7PM           0.9594943  0.9624055  0.9652314  0.9688208  0.9729115  0.9722363
## 8PM           0.9572582  0.9615833  0.9635768  0.9653561  0.9632794  0.9569739
## 9PM           0.9475326  0.9513598  0.9516732  0.9525996  0.9470948  0.9379284
## 10PM          0.9332618  0.9366956  0.9358131  0.9358430  0.9284951  0.9174589
## 11PM          0.8979948  0.9030467  0.9023292  0.9015427  0.8923808  0.8813057
## closed_roads -0.1044117 -0.1094570 -0.1215520 -0.1239433 -0.1259897 -0.1280670
##                     4PM        5PM        6PM        7PM        8PM        9PM
## 12AM          0.8151468  0.8120509  0.8281942  0.8578880  0.8914953  0.9110580
## 1AM           0.7459208  0.7430624  0.7601627  0.7919379  0.8296572  0.8533356
## 2AM           0.6992087  0.6960858  0.7132060  0.7446331  0.7817508  0.8056455
## 3AM           0.7143935  0.7103966  0.7267654  0.7573179  0.7915980  0.8134853
## 4AM           0.7897050  0.7849362  0.7981537  0.8226188  0.8471303  0.8610960
## 5AM           0.7972642  0.7957899  0.8045064  0.8159362  0.8201180  0.8211381
## 6AM           0.8315893  0.8323310  0.8379809  0.8396884  0.8314147  0.8249342
## 7AM           0.8631667  0.8654827  0.8671035  0.8625307  0.8498419  0.8398741
## 8AM           0.9056551  0.9071085  0.9080492  0.9035774  0.8909439  0.8796399
## 9AM           0.9393913  0.9392241  0.9429393  0.9440933  0.9371438  0.9269126
## 10AM          0.9515860  0.9501128  0.9554203  0.9594943  0.9572582  0.9475326
## 11AM          0.9567508  0.9542758  0.9583361  0.9624055  0.9615833  0.9513598
## 12PM          0.9621928  0.9584074  0.9618355  0.9652314  0.9635768  0.9516732
## 1PM           0.9682953  0.9642366  0.9665990  0.9688208  0.9653561  0.9525996
## 2PM           0.9792115  0.9747676  0.9748313  0.9729115  0.9632794  0.9470948
## 3PM           0.9909914  0.9838253  0.9805192  0.9722363  0.9569739  0.9379284
## 4PM           1.0000000  0.9915007  0.9842618  0.9720329  0.9535176  0.9328193
## 5PM           0.9915007  1.0000000  0.9895472  0.9750824  0.9548483  0.9336799
## 6PM           0.9842618  0.9895472  1.0000000  0.9869515  0.9687458  0.9489524
## 7PM           0.9720329  0.9750824  0.9869515  1.0000000  0.9869215  0.9704966
## 8PM           0.9535176  0.9548483  0.9687458  0.9869215  1.0000000  0.9894637
## 9PM           0.9328193  0.9336799  0.9489524  0.9704966  0.9894637  1.0000000
## 10PM          0.9105150  0.9119948  0.9285253  0.9523197  0.9742714  0.9892017
## 11PM          0.8735794  0.8747593  0.8919908  0.9184391  0.9456321  0.9660675
## closed_roads -0.1304982 -0.1311431 -0.1271745 -0.1218706 -0.1157256 -0.1116852
##                    10PM       11PM closed_roads
## 12AM          0.9219048  0.9306260  -0.07644587
## 1AM           0.8687708  0.8844249  -0.07415399
## 2AM           0.8216783  0.8385232  -0.07530540
## 3AM           0.8267327  0.8370439  -0.07034460
## 4AM           0.8652759  0.8572873  -0.07046845
## 5AM           0.8129791  0.7813509  -0.06598521
## 6AM           0.8127200  0.7743815  -0.07100886
## 7AM           0.8256235  0.7897257  -0.07824962
## 8AM           0.8649996  0.8295644  -0.08365954
## 9AM           0.9129399  0.8774134  -0.09609481
## 10AM          0.9332618  0.8979948  -0.10441175
## 11AM          0.9366956  0.9030467  -0.10945697
## 12PM          0.9358131  0.9023292  -0.12155203
## 1PM           0.9358430  0.9015427  -0.12394334
## 2PM           0.9284951  0.8923808  -0.12598968
## 3PM           0.9174589  0.8813057  -0.12806705
## 4PM           0.9105150  0.8735794  -0.13049817
## 5PM          0.9119948  0.8747593  -0.13114312
## 6PM          0.9285253  0.8919908  -0.12717453
## 7PM          0.9523197  0.9184391  -0.12187062
## 8PM          0.9742714  0.9456321  -0.11572560
## 9PM          0.9892017  0.9660675  -0.11168518
## 10PM         1.0000000  0.9826034  -0.10732738
## 11PM         0.9826034  1.0000000  -0.10294205
## closed_roads -0.1073274 -0.1029420   1.00000000
```

Strategy 4: Transpose Graph I tried getting the transpose of the matrix to work, since that would in theory, switch from analyzing the times from each other to the roads, the graph however is still printing out only the times against one another.

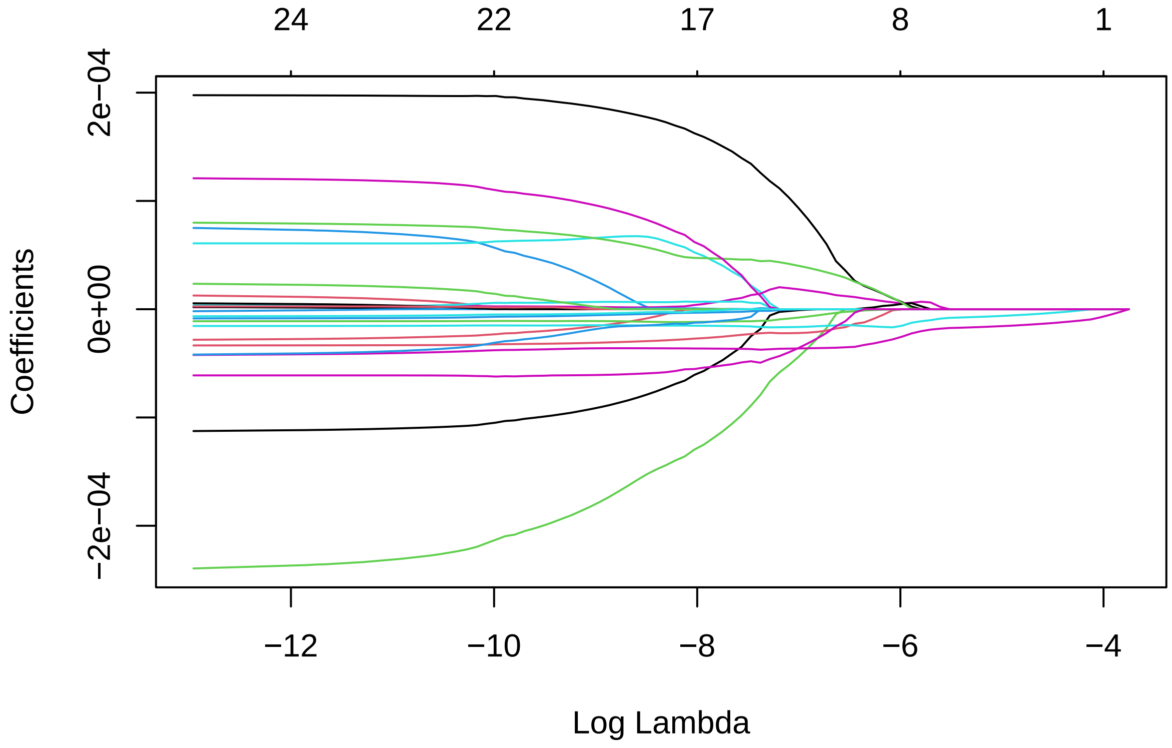
```
pca_result <- prcomp(t(scaled_data), center = TRUE, scale. = TRUE)
pca.data <- data.frame(Sample = rownames(pca_result$x), X = pca_result$x[,1], Y = pca_result$x[,2])

ggplot(data = pca.data, aes(x = X, y = Y, label = Sample)) +
  geom_text() +
  xlab(paste("PC1")) +
  ylab(paste("PC2"))
```



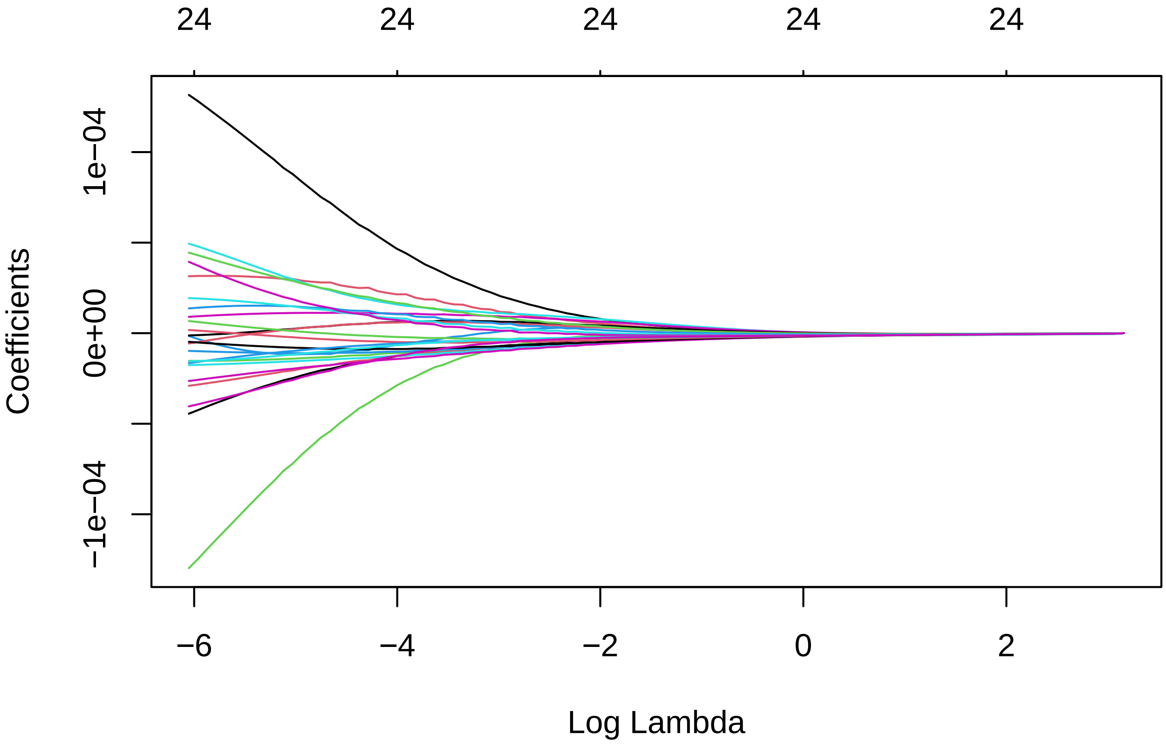
Strategy 5: Lasso and Ridge (EXTRA) This idea is straight from ChatGPT, I wish for it not to be considered when reviewing Project 1. I just wanted to include the code from what I gathered. It is something I would like to see if I can make it work for the roads insetad of times, or if you find the graphs and what

they are showing interesting or unique. But again, not to be considered with the rest of the work done to start the final project.

```
closed_roads <- rowSums(TVC[, (ncol(TVC) - 23):ncol(TVC)] == 0) > 0
model_data <- cbind(TVC[, (ncol(TVC) - 23):ncol(TVC)], closed_roads)
colnames(model_data) <- c(names(TVC)[(ncol(TVC) - 23):ncol(TVC)], "ClosedRoads")

lasso_model <- glmnet(as.matrix(model_data[, -ncol(model_data)]), model_data$ClosedRoads, alpha = 1)
ridge_model <- glmnet(as.matrix(model_data[, -ncol(model_data)]), model_data$ClosedRoads, alpha = 0)

plot(lasso_model, xvar = "lambda")
```



```
plot(ridge_model, xvar = "lambda")
```


24
24
24
24
24
Coefficients
1e−04
0e+00
−1e−04
−6
−4
−2
0
2
Log Lambda