Road Safety Pilot Project - Forecasting

Hair

Libraries

```
# Needed to load custom scripts
library("here")

# Custom scripts
source(here("functions", "utils.R"))
source(here("functions", "clean_data.R"))
source(here("functions", "install_packages.R"))

# the following function will try to install and load all the required
# packages for this project.
f_load_packages()
```

On Execution time

Due to the methodology employed and the number of models fit as well as several other procedures, running the whole script will take around ~10 minutes in total.

Exploratory Data Analysis

Data Loading and Preparation

The following chunk of code will take care of the main steps:

1. Loading the data
2. Data cleaning and preprocessing
3. Implementing the dummies version of the dataset, necessary for some of the models.
4. Extracting relevant meta-data.

```
############################
### 1. Load the Raw Data ###
############################

# Load the dataset and perform data cleaning
dat_orig <- read.csv(here("data_raw", "data_final.csv"), sep=";")

# store the intersections id_no to rue_1 and rue_2 mapping
inter_names <- subset(dat_orig, select = c(int_no, x, y, rue_1, rue_2))

#########################
### 2. Clean the data ###
#########################

# Create two clean versions of the data
```

```r
dat <- f_clean_data(dat_orig,  # original data
                    group_boroughs = TRUE, # Group the levels in the borough variable
                    drop_borough=TRUE,  # If the borough levels were grouped, drop the original variable
                    drop_year=TRUE,  # Drop the year variable, but weekly and monthly features are kept.
                    standarize = TRUE, # Standarize the numerical variables
                    numerical_categories = FALSE) # Convert some integer-based variables to categorical

# Create a simple version of the data for EDA visualization purposes
dat_no_group <- f_clean_data(dat_orig, group_boroughs = FALSE)

# # Drop the raw data object
# rm(dat_orig); gc(verbose=FALSE);

##################################
### 3. Cleate Dummies Version ###
##################################

## Create an additional version which contains dummies instead of factors

# All the variables should go into the dummies, except `acc`
all_vars <- setdiff(colnames(dat), "acc")

# Create a version with dummies
dat_dum <- f_convert_to_dummies(dat, all_vars)

# Reassign the target variable
dat_dum$acc <- dat$acc

##################################################
### 4. Extract variable names and descriptions ###
##################################################

# Load variable descriptors
result <- f_load_varnames(here("data_raw", "Dictionnaire_final.xlsx"))
varnames <- result$varnames
varnames_dict <- result$varnames_dict
```

Inspecting the raw data

We can preview our dataset, and see that there is a number of columns which are irrelevant, the names are corrupted, variable naming conventions are not clear, etc.

```r
# Data before preprocessing or cleaning
head(dat_orig)
```

```
##   int_no        x       y              rue_1              rue_2
## 1     40 296397.3 5037515 Côte-Saint-Antoine Hampton/Sherbrooke
## 2    263 300293.3 5038407             Centre         Wellington
## 3   1107 301317.1 5039803       Pierre-Dupuy        Habitats 67
## 4   1364 299985.9 5037139         Charlevoix         Wellington
## 5    225 297859.8 5036983          Courcelle         Notre-Dame
## 6   1728 299490.8 5034639               Egan            LaSalle
##             street_1           street_2      date_ all_pedest pi        fi
## 1 C¶te-Saint-Antoine Hampton/Sherbrooke 27/02/2009          0  0  2603.828
## 2             Centre         Wellington 20/11/2007          0  0  7816.052
## 3       Pierre-Dupuy        Habitats 67 09/09/2009          0  0  8896.672
## 4         Charlevoix         Wellington 10/12/2007          0  0 10932.755
## 5          Courcelle         Notre-Dame 15/11/2006          0  0 10479.297
```

```
## 6             Egan          LaSalle 14/11/2007         0  0 14704.115
##         fli         fri       fti cli cri cti acc ln_pi     ln_fi    ln_fli
## 1    0.0000     0.00000 2603.828   0   0   0   0     0 7.864738 0.000000
## 2 1251.5605   777.99707 5786.494   0   0   0   0     0 8.963935 7.132146
## 3  282.2264    54.20242 8560.243   0   0   0   0     0 9.093433 5.642709
## 4 1303.2260 1884.36084 7745.168   0   0   0   0     0 9.299519 7.172598
## 5 1227.4500 1450.41370 7801.433   0   0   0   0     0 9.257157 7.112694
## 6 2974.3826 3367.44238 8362.290   0   0   0   0     0 9.595883 7.997792
##      ln_fri   ln_fti ln_cli ln_cri ln_cti tot_crossw number_of_ avg_crossw
## 1 0.000000 7.864738      0      0      0       77.4          4       19.4
## 2 6.656723 8.663282      0      0      0       51.9          3       17.3
## 3 3.992725 9.054884      0      0      0       27.0          2       13.5
## 4 7.541344 8.954824      0      0      0       48.2          4       12.0
## 5 7.279604 8.962063      0      0      0       57.1          4       14.3
## 6 8.121909 9.031488      0      0      0       32.3          4        8.1
##   tot_road_w median green_stra half_phase any_ped_pr ped_countd lt_protect
## 1       53.7      1          0          1          1          1          1
## 2       40.4      0          0          1          1          1          0
## 3       27.5      0          0          0          0          0          1
## 4       31.0      0          0          0          0          0          0
## 5       44.9      0          0          0          0          0          0
## 6       30.0      0          0          0          0          0          0
##   lt_restric lt_prot_re parking north_veh north_ped east_veh east_ped south_veh
## 1          1          1       1     0.000         0 2603.828        0    0.0000
## 2          1          1       0  1251.561         0 2595.579        0  112.7532
## 3          1          1       0     0.000         0 3762.395        0  336.4288
## 4          0          0       0  1449.128         0 4785.089        0  182.9957
## 5          0          0       1  1758.426         0 4650.059        0 1284.9148
## 6          1          1       0  3461.685         0 4463.872        0 3624.8850
##   south_ped west_veh west_ped total_lane of_exclusi any_exclus commercial
## 1         0    0.000        0          5          0          0          0
## 2         0 3856.159        0          4          0          0          1
## 3         0 4797.848        0          4          0          0          2
## 4         0 4515.542        0          2          0          0          0
## 5         0 2785.898        0          2          0          0          0
## 6         0 3153.673        0          1          0          0          0
##   curb_exten all_red_an new_half_r   distdt ln_distdt traffic_10000 ped_100
## 1          0          0          1 3932.077  8.276923     0.2603828       0
## 2          0          0          1 2097.060  7.648292     0.7816052       0
## 3          1          0          0 2112.164  7.655468     0.8896672       0
## 4          0          0          0 3172.770  8.062361     1.0932755       0
## 5          0          0          0 3525.406  8.167751     1.0479296       0
## 6          0          0          0 5590.807  8.628879     1.4704115       0
##                               borough  X X.1
## 1 C¶te-des-Neiges-Notre-Dame-de-Graces NA
## 2                           Sud-Ouest NA
## 3                         Ville-Marie NA
## 4                           Sud-Ouest NA
## 5                           Sud-Ouest NA
## 6                              Verdun NA
```

The borough names are not very clear and contain strange characters:

```r
unique(dat_orig$borough)
```

```
##  [1] "C¶te-des-Neiges-Notre-Dame-de-Graces"
##  [2] "Sud-Ouest"
##  [3] "Ville-Marie"
##  [4] "Verdun"
```

```
##  [5] "Mercier-Hochelaga-Maisonneuve"
##  [6] "Rosemont-La-Petite-Patrie"
##  [7] "Anjou"
##  [8] "Lasalle"
##  [9] "Plateau-Mont-Royal"
## [10] "Westmount"
## [11] "Saint-Laurent"
## [12] "Villeray-Saint-Michel-Parc-Extension"
## [13] "Pointe-aux-Trembles-RiviÞres-des-Prairies"
## [14] "MontrÚal-Nord"
## [15] "Ahuntsic-Cartierville"
## [16] "MontrÚal-Est"
## [17] "Pierrefonds-Roxboro"
## [18] "St-LÚonard"
## [19] "Outremont"
## [20] "Lachine"
## [21] "Beaconsfield"
## [22] "Hampstead"
## [23] "Dollard-des-Ormeaux"
## [24] "Dorval"
## [25] "C¶te-Saint-Luc"
## [26] "Mont-Royal"
## [27] "?le-Bizard-Sainte-GeneviÞve"
## [28] "Kirkland"
```

```r
str(dat_orig)
```

```
## 'data.frame':    1864 obs. of  61 variables:
##  $ int_no      : int  40 263 1107 1364 225 1728 1623 978 382 944 ...
##  $ x           : num  296397 300293 301317 299986 297860 ...
##  $ y           : num  5037515 5038407 5039803 5037139 5036983 ...
##  $ rue_1       : chr  "Côte-Saint-Antoine" "Centre" "Pierre-Dupuy" "Charlevoix" ...
##  $ rue_2       : chr  "Hampton/Sherbrooke" "Wellington" "Habitats 67" "Wellington" ...
##  $ street_1    : chr  "C¶te-Saint-Antoine" "Centre" "Pierre-Dupuy" "Charlevoix" ...
##  $ street_2    : chr  "Hampton/Sherbrooke" "Wellington" "Habitats 67" "Wellington" ...
##  $ date_       : chr  "27/02/2009" "20/11/2007" "09/09/2009" "10/12/2007" ...
##  $ all_pedest  : int  0 0 0 0 0 0 0 0 0 0 ...
##  $ pi          : num  0 0 0 0 0 0 0 0 0 0 ...
##  $ fi          : num  2604 7816 8897 10933 10479 ...
##  $ fli         : num  0 1252 282 1303 1227 ...
##  $ fri         : num  0 778 54.2 1884.4 1450.4 ...
##  $ fti         : num  2604 5786 8560 7745 7801 ...
##  $ cli         : num  0 0 0 0 0 0 0 0 0 0 ...
##  $ cri         : num  0 0 0 0 0 0 0 0 0 0 ...
##  $ cti         : num  0 0 0 0 0 0 0 0 0 0 ...
##  $ acc         : int  0 0 0 0 0 0 0 0 0 0 ...
##  $ ln_pi       : num  0 0 0 0 0 0 0 0 0 0 ...
##  $ ln_fi       : num  7.86 8.96 9.09 9.3 9.26 ...
##  $ ln_fli      : num  0 7.13 5.64 7.17 7.11 ...
##  $ ln_fri      : num  0 6.66 3.99 7.54 7.28 ...
##  $ ln_fti      : num  7.86 8.66 9.05 8.95 8.96 ...
##  $ ln_cli      : num  0 0 0 0 0 0 0 0 0 0 ...
##  $ ln_cri      : num  0 0 0 0 0 0 0 0 0 0 ...
##  $ ln_cti      : num  0 0 0 0 0 0 0 0 0 0 ...
##  $ tot_crossw  : num  77.4 51.9 27 48.2 57.1 ...
##  $ number_of_  : int  4 3 2 4 4 4 3 3 4 3 ...
##  $ avg_crossw  : num  19.4 17.3 13.5 12 14.3 ...
##  $ tot_road_w  : num  53.7 40.4 27.5 31 44.9 ...
##  $ median      : int  1 0 0 0 0 0 0 1 1 1 ...
```

```
## $ green_stra   : int  0 0 0 0 0 0 0 0 1 0 ...
## $ half_phase   : int  1 1 0 0 0 0 0 0 0 0 ...
## $ any_ped_pr   : int  1 1 0 0 0 0 0 0 1 0 ...
## $ ped_countd   : int  1 1 0 0 0 0 0 0 1 0 ...
## $ lt_protect   : int  1 0 1 0 0 0 0 1 1 1 ...
## $ lt_restric   : int  1 1 1 0 0 1 1 1 1 1 ...
## $ lt_prot_re   : int  1 1 1 0 0 1 1 1 1 1 ...
## $ parking      : num  1 0 0 0 1 0 0 0 0 0 ...
## $ north_veh    : num  0 1252 0 1449 1758 ...
## $ north_ped    : num  0 0 0 0 0 0 0 0 0 0 ...
## $ east_veh     : num  2604 2596 3762 4785 4650 ...
## $ east_ped     : num  0 0 0 0 0 0 0 0 0 0 ...
## $ south_veh    : num  0 113 336 183 1285 ...
## $ south_ped    : num  0 0 0 0 0 0 0 0 0 0 ...
## $ west_veh     : num  0 3856 4798 4516 2786 ...
## $ west_ped     : num  0 0 0 0 0 0 0 0 0 0 ...
## $ total_lane   : int  5 4 4 2 2 1 2 2 4 5 ...
## $ of_exclusi   : int  0 0 0 0 0 0 0 1 1 2 ...
## $ any_exclus   : int  0 0 0 0 0 0 0 1 1 1 ...
## $ commercial   : int  0 1 2 0 0 0 0 0 0 0 ...
## $ curb_exten   : int  0 0 1 0 0 0 0 0 0 0 ...
## $ all_red_an   : int  0 0 0 0 0 0 0 0 0 0 ...
## $ new_half_r   : int  1 1 0 0 0 0 0 0 0 0 ...
## $ distdt       : num  3932 2097 2112 3173 3525 ...
## $ ln_distdt    : num  8.28 7.65 7.66 8.06 8.17 ...
## $ traffic_10000: num  0.26 0.782 0.89 1.093 1.048 ...
## $ ped_100      : num  0 0 0 0 0 0 0 0 0 0 ...
## $ borough      : chr  "C¶te-des-Neiges-Notre-Dame-de-Graces" "Sud-Ouest" "Ville-Marie" "Sud-Ouest" ...
## $ X            : num  NA NA NA NA NA NA NA NA NA NA ...
## $ X.1          : chr  "" "" "" "" ...
```

- We can observe a number of irrelevant columns (e.g. `street_1`, `street_2`, `X`, `X.1`) that we will remove.
- Also, the borough names contains typos, so we will also correct those to have consiten names.
- We also observersome covariates are of the wrong type (e.g. categorical, numerical, etc.)

All of these problems are addressed by the function `f_clean_data`, which does the follwoing:

1. Remove a number of columns such that "street_1", "street_2", "X", "X.1", "int_no", 'rue_1', 'rue_2', "traffic_10000", "ped_100".
2. **TODO** Complete.

Meta-data

We separate some meta-data from the main data for convenience:

```r
head(inter_names)
```

```
##   int_no        x       y               rue_1              rue_2
## 1     40 296397.3 5037515 Côte-Saint-Antoine Hampton/Sherbrooke
## 2    263 300293.3 5038407             Centre         Wellington
## 3   1107 301317.1 5039803        Pierre-Dupuy        Habitats 67
## 4   1364 299985.9 5037139         Charlevoix         Wellington
## 5    225 297859.8 5036983          Courcelle         Notre-Dame
## 6   1728 299490.8 5034639               Egan            LaSalle
```

Data after cleaning

After cleaning and preprocessing, we end up with two versions of the clean dataset:

1. `dat`: The main version of the data, with the following characteristics:
    - Grouped boroughs
    - Standarized numerical variables
    - Removed irrelevant columns
    - Removed the year variable
    - Converted some integer-based variables to categorical
2. `dat_dum`: A version of the data with dummies instead of factors.

```
head(dat)
```

```
##      latitude  longitude all_pedest         pi        fi        fli        fri
## 1 -0.03130934 -1.2403560          0 -0.5725017 -1.690985 -0.8120156 -0.8135587
## 2  0.79609954 -1.0624758          0 -0.5725017 -1.322855 -0.4223036 -0.5862250
## 3  1.01352400 -0.7838334          0 -0.5725017 -1.246532 -0.7241357 -0.7977205
## 4  0.73080782 -1.3154787          0 -0.5725017 -1.102728 -0.4062160 -0.2629413
## 5  0.27928091 -1.3465055          0 -0.5725017 -1.134754 -0.4298112 -0.3897423
## 6  0.62566281 -1.8143016          0 -0.5725017 -0.836363  0.1141502  0.1704207
##         fti        cli        cri        cti acc     ln_pi      ln_fi
## 1 -1.617385 -0.4542889 -0.4680791 -0.5301522   0 -4.062594 -4.0994560
## 2 -1.339887 -0.4542889 -0.4680791 -0.5301522   0 -4.062594 -2.0393157
## 3 -1.098043 -0.4542889 -0.4680791 -0.5301522   0 -4.062594 -1.7966067
## 4 -1.169110 -0.4542889 -0.4680791 -0.5301522   0 -4.062594 -1.4103570
## 5 -1.164204 -0.4542889 -0.4680791 -0.5301522   0 -4.062594 -1.4897515
## 6 -1.115303 -0.4542889 -0.4680791 -0.5301522   0 -4.062594 -0.8549033
##         ln_fli      ln_fri    ln_fti    ln_cli    ln_cri    ln_cti tot_crossw
## 1 -4.88362896 -5.20515130 -2.807246 -3.912127 -4.072173 -5.531152  0.4027162
## 2 -0.08079063 -0.49295344 -1.642394 -3.912127 -4.072173 -5.531152 -0.7378980
## 3 -1.08378829 -2.37875915 -1.071157 -3.912127 -4.072173 -5.531152 -1.8516743
## 4 -0.05354996  0.13325707 -1.217116 -3.912127 -4.072173 -5.531152 -0.9033990
## 5 -0.09388981 -0.05202504 -1.206557 -3.912127 -4.072173 -5.531152 -0.5053023
## 6  0.50214153  0.54423061 -1.105285 -3.912127 -4.072173 -5.531152 -1.6146055
##    number_of_ avg_crossw tot_road_w median green_stra half_phase any_ped_pr
## 1  0.5051574  0.1620242 -0.1766794      1          0          1          1
## 2 -1.2099853 -0.2225428 -0.9705920      0          0          1          1
## 3 -2.9251280 -0.9184258 -1.7406278      0          0          0          0
## 4  0.5051574 -1.1931166 -1.5317034      0          0          0          0
## 5  0.5051574 -0.7719241 -0.7019750      0          0          0          0
## 6  0.5051574 -1.9073123 -1.5913960      0          0          0          0
##    ped_countd lt_protect lt_restric lt_prot_re parking  north_veh  north_ped
## 1          1          1          1          1       2 -0.8442758 -0.4737868
## 2          1          0          1          1       0 -0.6737608 -0.4737868
## 3          0          1          1          1       0 -0.8442758 -0.4737868
## 4          0          0          0          0       0 -0.6468439 -0.4737868
## 5          0          0          0          0       2 -0.6047046 -0.4737868
## 6          0          0          1          1       0 -0.3726493 -0.4737868
##      east_veh   east_ped  south_veh  south_ped   west_veh   west_ped total_lane
## 1 -0.7512007 -0.5598873 -0.8583187 -0.4800757 -1.1616670 -0.5081127  0.2694611
## 2 -0.7525126 -0.5598873 -0.8418498 -0.4800757 -0.5347397 -0.5081127 -0.2987239
## 3 -0.5669462 -0.5598873 -0.8091795 -0.4800757 -0.3816417 -0.5081127 -0.2987239
## 4 -0.4043006 -0.5598873 -0.8315902 -0.4800757 -0.4275385 -0.5081127 -1.4350939
## 5 -0.4257754 -0.5598873 -0.6706426 -0.4800757 -0.7087409 -0.5081127 -1.4350939
## 6 -0.4553858 -0.5598873 -0.3288638 -0.4800757 -0.6489487 -0.5081127 -2.0032789
##    of_exclusi any_exclus  commercial curb_exten all_red_an new_half_r     distdt
## 1 -0.4982195          0 -0.65038725          0          0          1 -0.6286641
## 2 -0.4982195          0  0.05281567          0          0          1 -1.0317585
## 3 -0.4982195          0  0.75601859          1          0          0 -1.0284406
## 4 -0.4982195          0 -0.65038725          0          0          0 -0.7954595
## 5 -0.4982195          0 -0.65038725          0          0          0 -0.7179966
```

```
## 6 -0.4982195          0 -0.65038725          0          0          0 -0.2642942
##    ln_distdt missing_date_ind month       dow
## 1 -0.2767748                0    02    Friday
## 2 -0.9713874                0    11   Tuesday
## 3 -0.9634573                0    09 Wednesday
## 4 -0.5138577                0    12    Monday
## 5 -0.3974054                0    11 Wednesday
## 6  0.1121217                0    11 Wednesday
##                     borough_grouped int_no pi_squared fi_squared
## 1 Côte-des-Neiges-Notre-Dame-de-Grâce     40  0.3277581  2.8594298
## 2                           Sud-Ouest    263  0.3277581  1.7499446
## 3                         Ville-Marie   1107  0.3277581  1.5538432
## 4                           Sud-Ouest   1364  0.3277581  1.2160080
## 5                           Sud-Ouest    225  0.3277581  1.2876677
## 6                               Other   1728  0.3277581  0.6995031
##   distdt_squared distdt_cubed tot_crossw_squared avg_crossw_squared
## 1     0.39521853  -0.24845969          0.1621803         0.02625185
## 2     1.06452557  -1.09833330          0.5444935         0.04952532
## 3     1.05769011  -1.08777147          3.4286978         0.84350604
## 4     0.63275578  -0.50333158          0.8161297         1.42352713
## 5     0.51551918  -0.37014105          0.2553304         0.59586681
## 6     0.06985142  -0.01846133          2.6069510         3.63784039
##   tot_road_w_squared fli_squared fri_squared fti_squared
## 1         0.03121561  0.65936932  0.66187768    2.615933
## 2         0.94204893  0.17834033  0.34365972    1.795297
## 3         3.02978498  0.52437251  0.63635800    1.205698
## 4         2.34611519  0.16501142  0.06913814    1.366817
## 5         0.49276887  0.18473764  0.15189907    1.355371
## 6         2.53254138  0.01303027  0.02904321    1.243900
```

```r
head(dat_dum)
```

```
##      latitude  longitude         pi        fi        fli        fri       fti
## 1 -0.03130934 -1.2403560 -0.5725017 -1.690985 -0.8120156 -0.8135587 -1.617385
## 2  0.79609954 -1.0624758 -0.5725017 -1.322855 -0.4223036 -0.5862250 -1.339887
## 3  1.01352400 -0.7838334 -0.5725017 -1.246532 -0.7241357 -0.7977205 -1.098043
## 4  0.73080782 -1.3154787 -0.5725017 -1.102728 -0.4062160 -0.2629413 -1.169110
## 5  0.27928091 -1.3465055 -0.5725017 -1.134754 -0.4298112 -0.3897423 -1.164204
## 6  0.62566281 -1.8143016 -0.5725017 -0.836363  0.1141502  0.1704207 -1.115303
##          cli        cri        cti     ln_pi      ln_fi      ln_fli      ln_fri
## 1 -0.4542889 -0.4680791 -0.5301522 -4.062594 -4.0994560 -4.88362896 -5.20515130
## 2 -0.4542889 -0.4680791 -0.5301522 -4.062594 -2.0393157 -0.08079063 -0.49295344
## 3 -0.4542889 -0.4680791 -0.5301522 -4.062594 -1.7966067 -1.08378829 -2.37875915
## 4 -0.4542889 -0.4680791 -0.5301522 -4.062594 -1.4103570 -0.05354996  0.13325707
## 5 -0.4542889 -0.4680791 -0.5301522 -4.062594 -1.4897515 -0.09388981 -0.05202504
## 6 -0.4542889 -0.4680791 -0.5301522 -4.062594 -0.8549033  0.50214153  0.54423061
##      ln_fti    ln_cli    ln_cri    ln_cti tot_crossw number_of_ avg_crossw
## 1 -2.807246 -3.912127 -4.072173 -5.531152  0.4027162  0.5051574  0.1620242
## 2 -1.642394 -3.912127 -4.072173 -5.531152 -0.7378980 -1.2099853 -0.2225428
## 3 -1.071157 -3.912127 -4.072173 -5.531152 -1.8516743 -2.9251280 -0.9184258
## 4 -1.217116 -3.912127 -4.072173 -5.531152 -0.9033990  0.5051574 -1.1931166
## 5 -1.206557 -3.912127 -4.072173 -5.531152 -0.5053023  0.5051574 -0.7719241
## 6 -1.105285 -3.912127 -4.072173 -5.531152 -1.6146055  0.5051574 -1.9073123
##   tot_road_w  north_veh  north_ped   east_veh   east_ped  south_veh  south_ped
## 1 -0.1766794 -0.8442758 -0.4737868 -0.7512007 -0.5598873 -0.8583187 -0.4800757
## 2 -0.9705920 -0.6737608 -0.4737868 -0.7525126 -0.5598873 -0.8418498 -0.4800757
## 3 -1.7406278 -0.8442758 -0.4737868 -0.5669462 -0.5598873 -0.8091795 -0.4800757
## 4 -1.5317034 -0.6468439 -0.4737868 -0.4043006 -0.5598873 -0.8315902 -0.4800757
## 5 -0.7019750 -0.6047046 -0.4737868 -0.4257754 -0.5598873 -0.6706426 -0.4800757
```

```
## 6 -1.5913960 -0.3726493 -0.4737868 -0.4553858 -0.5598873 -0.3288638 -0.4800757
##     west_veh   west_ped total_lane of_exclusi  commercial     distdt  ln_distdt
## 1 -1.1616670 -0.5081127  0.2694611 -0.4982195 -0.65038725 -0.6286641 -0.2767748
## 2 -0.5347397 -0.5081127 -0.2987239 -0.4982195  0.05281567 -1.0317585 -0.9713874
## 3 -0.3816417 -0.5081127 -0.2987239 -0.4982195  0.75601859 -1.0284406 -0.9634573
## 4 -0.4275385 -0.5081127 -1.4350939 -0.4982195 -0.65038725 -0.7954595 -0.5138577
## 5 -0.7087409 -0.5081127 -1.4350939 -0.4982195 -0.65038725 -0.7179966 -0.3974054
## 6 -0.6489487 -0.5081127 -2.0032789 -0.4982195 -0.65038725 -0.2642942  0.1121217
##   int_no pi_squared fi_squared distdt_squared distdt_cubed tot_crossw_squared
## 1     40  0.3277581  2.8594298     0.39521853  -0.24845969          0.1621803
## 2    263  0.3277581  1.7499446     1.06452557  -1.09833330          0.5444935
## 3   1107  0.3277581  1.5538432     1.05769011  -1.08777147          3.4286978
## 4   1364  0.3277581  1.2160080     0.63275578  -0.50333158          0.8161297
## 5    225  0.3277581  1.2876677     0.51551918  -0.37014105          0.2553304
## 6   1728  0.3277581  0.6995031     0.06985142  -0.01846133          2.6069510
##   avg_crossw_squared tot_road_w_squared fli_squared fri_squared fti_squared
## 1         0.02625185         0.03121561  0.65936932  0.66187768    2.615933
## 2         0.04952532         0.94204893  0.17834033  0.34365972    1.795297
## 3         0.84350604         3.02978498  0.52437251  0.63635800    1.205698
## 4         1.42352713         2.34611519  0.16501142  0.06913814    1.366817
## 5         0.59586681         0.49276887  0.18473764  0.15189907    1.355371
## 6         3.63784039         2.53254138  0.01303027  0.02904321    1.243900
##   all_pedest_1 median_1 green_stra_1 half_phase_1 any_ped_pr_1 ped_countd_1
## 1            0        1            0            1            1            1
## 2            0        0            0            1            1            1
## 3            0        0            0            0            0            0
## 4            0        0            0            0            0            0
## 5            0        0            0            0            0            0
## 6            0        0            0            0            0            0
##   lt_protect_1 lt_restric_1 lt_prot_re_1 parking_1 parking_2 any_exclus_1
## 1            1            1            1         0         1            0
## 2            0            1            1         0         0            0
## 3            1            1            1         0         0            0
## 4            0            0            0         0         0            0
## 5            0            0            0         0         1            0
## 6            0            1            1         0         0            0
##   curb_exten_1 all_red_an_1 new_half_r_1 missing_date_ind_1 month_02 month_03
## 1            0            0            1                  0        1        0
## 2            0            0            1                  0        0        0
## 3            1            0            0                  0        0        0
## 4            0            0            0                  0        0        0
## 5            0            0            0                  0        0        0
## 6            0            0            0                  0        0        0
##   month_04 month_05 month_06 month_07 month_08 month_09 month_10 month_11
## 1        0        0        0        0        0        0        0        0
## 2        0        0        0        0        0        0        0        1
## 3        0        0        0        0        0        1        0        0
## 4        0        0        0        0        0        0        0        0
## 5        0        0        0        0        0        0        0        1
## 6        0        0        0        0        0        0        0        1
##   month_12 dow_Monday dow_Saturday dow_Sunday dow_Thursday dow_Tuesday
## 1        0          0            0          0            0           0
## 2        0          0            0          0            0           1
## 3        0          0            0          0            0           0
## 4        1          1            0          0            0           0
## 5        0          0            0          0            0           0
## 6        0          0            0          0            0           0
##   dow_Wednesday borough_grouped_Other
## 1             0                     0
## 2             0                     0
```

```
## 3            1                     0
## 4            0                     0
## 5            1                     0
## 6            1                     1
##   borough_grouped_Côte-des-Neiges-Notre-Dame-de-Grâce
## 1                                                   1
## 2                                                   0
## 3                                                   0
## 4                                                   0
## 5                                                   0
## 6                                                   0
##   borough_grouped_Le Plateau-Mont-Royal
## 1                                     0
## 2                                     0
## 3                                     0
## 4                                     0
## 5                                     0
## 6                                     0
##   borough_grouped_Mercier-Hochelaga-Maisonneuve borough_grouped_Montréal-Nord
## 1                                             0                             0
## 2                                             0                             0
## 3                                             0                             0
## 4                                             0                             0
## 5                                             0                             0
## 6                                             0                             0
##   borough_grouped_Pointe-aux-Trembles-Rivières-des-Prairies
## 1                                                         0
## 2                                                         0
## 3                                                         0
## 4                                                         0
## 5                                                         0
## 6                                                         0
##   borough_grouped_Rosemont-La Petite-Patrie borough_grouped_Saint-Laurent
## 1                                         0                             0
## 2                                         0                             0
## 3                                         0                             0
## 4                                         0                             0
## 5                                         0                             0
## 6                                         0                             0
##   borough_grouped_Saint-Léonard borough_grouped_Sud-Ouest
## 1                             0                         0
## 2                             0                         1
## 3                             0                         0
## 4                             0                         1
## 5                             0                         1
## 6                             0                         0
##   borough_grouped_Ville-Marie
## 1                           0
## 2                           0
## 3                           1
## 4                           0
## 5                           0
## 6                           0
##   borough_grouped_Villeray-Saint-Michel-Parc-Extension acc
## 1                                                    0   0
## 2                                                    0   0
## 3                                                    0   0
## 4                                                    0   0
## 5                                                    0   0
## 6                                                    0   0
```

We can observe that the datatypes now are of the correct type for all variables, and the unique names of the borough have also been corrected (and grouped)

```
str(dat)
```

```
## 'data.frame':    1864 obs. of  65 variables:
##  $ latitude          : num [1:1864, 1] -0.0313 0.7961 1.0135 0.7308 0.2793 ...
##   ..- attr(*, "scaled:center")= num 296545
##   ..- attr(*, "scaled:scale")= num 4709
##  $ longitude         : num [1:1864, 1] -1.24 -1.062 -0.784 -1.315 -1.347 ...
##   ..- attr(*, "scaled:center")= num 5043731
##   ..- attr(*, "scaled:scale")= num 5012
##  $ all_pedest        : Factor w/ 2 levels "0","1": 1 1 1 1 1 1 1 1 1 1 ...
##  $ pi                : num [1:1864, 1] -0.573 -0.573 -0.573 -0.573 -0.573 ...
##   ..- attr(*, "scaled:center")= num 2887
##   ..- attr(*, "scaled:scale")= num 5044
##  $ fi                : num [1:1864, 1] -1.69 -1.32 -1.25 -1.1 -1.13 ...
##   ..- attr(*, "scaled:center")= num 26546
##   ..- attr(*, "scaled:scale")= num 14159
##  $ fli               : num [1:1864, 1] -0.812 -0.422 -0.724 -0.406 -0.43 ...
##   ..- attr(*, "scaled:center")= num 2608
##   ..- attr(*, "scaled:scale")= num 3212
##  $ fri               : num [1:1864, 1] -0.814 -0.586 -0.798 -0.263 -0.39 ...
##   ..- attr(*, "scaled:center")= num 2784
##   ..- attr(*, "scaled:scale")= num 3422
##  $ fti               : num [1:1864, 1] -1.62 -1.34 -1.1 -1.17 -1.16 ...
##   ..- attr(*, "scaled:center")= num 21154
##   ..- attr(*, "scaled:scale")= num 11469
##  $ cli               : num [1:1864, 1] -0.454 -0.454 -0.454 -0.454 -0.454 ...
##   ..- attr(*, "scaled:center")= num 1548991
##   ..- attr(*, "scaled:scale")= num 3409706
##  $ cri               : num [1:1864, 1] -0.468 -0.468 -0.468 -0.468 -0.468 ...
##   ..- attr(*, "scaled:center")= num 1926842
##   ..- attr(*, "scaled:scale")= num 4116489
##  $ cti               : num [1:1864, 1] -0.53 -0.53 -0.53 -0.53 -0.53 ...
##   ..- attr(*, "scaled:center")= num 28450590
##   ..- attr(*, "scaled:scale")= num 53664949
##  $ acc               : num  0 0 0 0 0 0 0 0 0 0 ...
##  $ ln_pi             : num [1:1864, 1] -4.06 -4.06 -4.06 -4.06 -4.06 ...
##   ..- attr(*, "scaled:center")= num 6.95
##   ..- attr(*, "scaled:scale")= num 1.71
##  $ ln_fi             : num [1:1864, 1] -4.1 -2.04 -1.8 -1.41 -1.49 ...
##   ..- attr(*, "scaled:center")= num 10.1
##   ..- attr(*, "scaled:scale")= num 0.534
##  $ ln_fli            : num [1:1864, 1] -4.8836 -0.0808 -1.0838 -0.0535 -0.0939 ...
##   ..- attr(*, "scaled:center")= num 7.25
##   ..- attr(*, "scaled:scale")= num 1.48
##  $ ln_fri            : num [1:1864, 1] -5.205 -0.493 -2.379 0.133 -0.052 ...
##   ..- attr(*, "scaled:center")= num 7.35
##   ..- attr(*, "scaled:scale")= num 1.41
##  $ ln_fti            : num [1:1864, 1] -2.81 -1.64 -1.07 -1.22 -1.21 ...
##   ..- attr(*, "scaled:center")= num 9.79
##   ..- attr(*, "scaled:scale")= num 0.686
##  $ ln_cli            : num [1:1864, 1] -3.91 -3.91 -3.91 -3.91 -3.91 ...
##   ..- attr(*, "scaled:center")= num 12.5
##   ..- attr(*, "scaled:scale")= num 3.19
##  $ ln_cri            : num [1:1864, 1] -4.07 -4.07 -4.07 -4.07 -4.07 ...
##   ..- attr(*, "scaled:center")= num 12.7
##   ..- attr(*, "scaled:scale")= num 3.11
##  $ ln_cti            : num [1:1864, 1] -5.53 -5.53 -5.53 -5.53 -5.53 ...
```

```
##   ..- attr(*, "scaled:center")= num 15.8
##   ..- attr(*, "scaled:scale")= num 2.85
##  $ tot_crossw        : num [1:1864, 1] 0.403 -0.738 -1.852 -0.903 -0.505 ...
##   ..- attr(*, "scaled:center")= num 68.4
##   ..- attr(*, "scaled:scale")= num 22.4
##  $ number_of_        : num [1:1864, 1] 0.505 -1.21 -2.925 0.505 0.505 ...
##   ..- attr(*, "scaled:center")= num 3.71
##   ..- attr(*, "scaled:scale")= num 0.583
##  $ avg_crossw        : num [1:1864, 1] 0.162 -0.223 -0.918 -1.193 -0.772 ...
##   ..- attr(*, "scaled:center")= num 18.5
##   ..- attr(*, "scaled:scale")= num 5.46
##  $ tot_road_w        : num [1:1864, 1] -0.177 -0.971 -1.741 -1.532 -0.702 ...
##   ..- attr(*, "scaled:center")= num 56.7
##   ..- attr(*, "scaled:scale")= num 16.8
##  $ median            : Factor w/ 2 levels "0","1": 2 1 1 1 1 1 1 2 2 2 ...
##  $ green_stra        : Factor w/ 2 levels "0","1": 1 1 1 1 1 1 1 1 2 1 ...
##  $ half_phase        : Factor w/ 2 levels "0","1": 2 2 1 1 1 1 1 1 1 1 ...
##  $ any_ped_pr        : Factor w/ 2 levels "0","1": 2 2 1 1 1 1 1 1 2 1 ...
##  $ ped_countd        : Factor w/ 2 levels "0","1": 2 2 1 1 1 1 1 1 2 1 ...
##  $ lt_protect        : Factor w/ 2 levels "0","1": 2 1 2 1 1 1 1 2 2 2 ...
##  $ lt_restric        : Factor w/ 2 levels "0","1": 2 2 2 1 1 2 2 2 2 2 ...
##  $ lt_prot_re        : Factor w/ 2 levels "0","1": 2 2 2 1 1 2 2 2 2 2 ...
##  $ parking           : Factor w/ 3 levels "0","1","2": 3 1 1 1 3 1 1 1 1 1 ...
##  $ north_veh         : num [1:1864, 1] -0.844 -0.674 -0.844 -0.647 -0.605 ...
##   ..- attr(*, "scaled:center")= num 6197
##   ..- attr(*, "scaled:scale")= num 7340
##  $ north_ped         : num [1:1864, 1] -0.474 -0.474 -0.474 -0.474 -0.474 ...
##   ..- attr(*, "scaled:center")= num 839
##   ..- attr(*, "scaled:scale")= num 1771
##  $ east_veh          : num [1:1864, 1] -0.751 -0.753 -0.567 -0.404 -0.426 ...
##   ..- attr(*, "scaled:center")= num 7327
##   ..- attr(*, "scaled:scale")= num 6288
##  $ east_ped          : num [1:1864, 1] -0.56 -0.56 -0.56 -0.56 -0.56 ...
##   ..- attr(*, "scaled:center")= num 640
##   ..- attr(*, "scaled:scale")= num 1144
##  $ south_veh         : num [1:1864, 1] -0.858 -0.842 -0.809 -0.832 -0.671 ...
##   ..- attr(*, "scaled:center")= num 5876
##   ..- attr(*, "scaled:scale")= num 6846
##  $ south_ped         : num [1:1864, 1] -0.48 -0.48 -0.48 -0.48 -0.48 ...
##   ..- attr(*, "scaled:center")= num 765
##   ..- attr(*, "scaled:scale")= num 1594
##  $ west_veh          : num [1:1864, 1] -1.162 -0.535 -0.382 -0.428 -0.709 ...
##   ..- attr(*, "scaled:center")= num 7145
##   ..- attr(*, "scaled:scale")= num 6151
##  $ west_ped          : num [1:1864, 1] -0.508 -0.508 -0.508 -0.508 -0.508 ...
##   ..- attr(*, "scaled:center")= num 643
##   ..- attr(*, "scaled:scale")= num 1265
##  $ total_lane        : num [1:1864, 1] 0.269 -0.299 -0.299 -1.435 -1.435 ...
##   ..- attr(*, "scaled:center")= num 4.53
##   ..- attr(*, "scaled:scale")= num 1.76
##  $ of_exclusi        : num [1:1864, 1] -0.498 -0.498 -0.498 -0.498 -0.498 ...
##   ..- attr(*, "scaled:center")= num 0.41
##   ..- attr(*, "scaled:scale")= num 0.823
##  $ any_exclus        : Factor w/ 2 levels "0","1": 1 1 1 1 1 1 1 2 2 2 ...
##  $ commercial        : num [1:1864, 1] -0.6504 0.0528 0.756 -0.6504 -0.6504 ...
##   ..- attr(*, "scaled:center")= num 0.925
##   ..- attr(*, "scaled:scale")= num 1.42
##  $ curb_exten        : Factor w/ 2 levels "0","1": 1 1 2 1 1 1 1 1 1 1 ...
##  $ all_red_an        : Factor w/ 2 levels "0","1": 1 1 1 1 1 1 1 1 1 1 ...
##  $ new_half_r        : Factor w/ 2 levels "0","1": 2 2 1 1 1 1 1 1 1 1 ...
```

```
##  $ distdt             : num [1:1864, 1] -0.629 -1.032 -1.028 -0.795 -0.718 ...
##   ..- attr(*, "scaled:center")= num 6794
##   ..- attr(*, "scaled:scale")= num 4552
##  $ ln_distdt          : num [1:1864, 1] -0.277 -0.971 -0.963 -0.514 -0.397 ...
##   ..- attr(*, "scaled:center")= num 8.53
##   ..- attr(*, "scaled:scale")= num 0.905
##  $ missing_date_ind  : Factor w/ 2 levels "0","1": 1 1 1 1 1 1 1 1 1 1 ...
##  $ month              : Factor w/ 12 levels "01","02","03",..: 2 11 9 12 11 11 6 11 11 8 ...
##  $ dow                : Factor w/ 7 levels "Friday","Monday",..: 1 6 7 2 7 7 7 7 6 2 ...
##  $ borough_grouped   : Factor w/ 13 levels "Ahuntsic-Cartierville",..: 3 11 12 11 11 2 5 11 12 5 ...
##  $ int_no             : int  40 263 1107 1364 225 1728 1623 978 382 944 ...
##  $ pi_squared         : num [1:1864, 1] 0.328 0.328 0.328 0.328 0.328 ...
##   ..- attr(*, "scaled:center")= num 2887
##   ..- attr(*, "scaled:scale")= num 5044
##  $ fi_squared         : num [1:1864, 1] 2.86 1.75 1.55 1.22 1.29 ...
##   ..- attr(*, "scaled:center")= num 26546
##   ..- attr(*, "scaled:scale")= num 14159
##  $ distdt_squared    : num [1:1864, 1] 0.395 1.065 1.058 0.633 0.516 ...
##   ..- attr(*, "scaled:center")= num 6794
##   ..- attr(*, "scaled:scale")= num 4552
##  $ distdt_cubed      : num [1:1864, 1] -0.248 -1.098 -1.088 -0.503 -0.37 ...
##   ..- attr(*, "scaled:center")= num 6794
##   ..- attr(*, "scaled:scale")= num 4552
##  $ tot_crossw_squared: num [1:1864, 1] 0.162 0.544 3.429 0.816 0.255 ...
##   ..- attr(*, "scaled:center")= num 68.4
##   ..- attr(*, "scaled:scale")= num 22.4
##  $ avg_crossw_squared: num [1:1864, 1] 0.0263 0.0495 0.8435 1.4235 0.5959 ...
##   ..- attr(*, "scaled:center")= num 18.5
##   ..- attr(*, "scaled:scale")= num 5.46
##  $ tot_road_w_squared: num [1:1864, 1] 0.0312 0.942 3.0298 2.3461 0.4928 ...
##   ..- attr(*, "scaled:center")= num 56.7
##   ..- attr(*, "scaled:scale")= num 16.8
##  $ fli_squared        : num [1:1864, 1] 0.659 0.178 0.524 0.165 0.185 ...
##   ..- attr(*, "scaled:center")= num 2608
##   ..- attr(*, "scaled:scale")= num 3212
##  $ fri_squared        : num [1:1864, 1] 0.6619 0.3437 0.6364 0.0691 0.1519 ...
##   ..- attr(*, "scaled:center")= num 2784
##   ..- attr(*, "scaled:scale")= num 3422
##  $ fti_squared        : num [1:1864, 1] 2.62 1.8 1.21 1.37 1.36 ...
##   ..- attr(*, "scaled:center")= num 21154
##   ..- attr(*, "scaled:scale")= num 11469
```

```r
unique(dat$borough)
```

```
##  [1] Côte-des-Neiges-Notre-Dame-de-Grâce
##  [2] Sud-Ouest
##  [3] Ville-Marie
##  [4] Other
##  [5] Mercier-Hochelaga-Maisonneuve
##  [6] Rosemont-La Petite-Patrie
##  [7] Le Plateau-Mont-Royal
##  [8] Saint-Laurent
##  [9] Villeray-Saint-Michel-Parc-Extension
## [10] Pointe-aux-Trembles-Rivières-des-Prairies
## [11] Montréal-Nord
## [12] Ahuntsic-Cartierville
## [13] Saint-Léonard
## 13 Levels: Ahuntsic-Cartierville Other ... Villeray-Saint-Michel-Parc-Extension
```

The variable `Other` contains the boroughts which contained fewer observations (**TODO:** Name which boroughs were removed.)

The dataset `dat_dum` is similar in nature, but contains only numerical variables, since the categorical variables have been converted to one-hot encoded vectors:

```
str(dat_dum)
```

```
## 'data.frame':    1864 obs. of  92 variables:
##  $ latitude                                             : num  -0.0313 0.7961 1.0135 0.7308 0.2793 ...
##  $ longitude                                            : num  -1.24 -1.062 -0.784 -1.315 -1.347 ...
##  $ pi                                                   : num  -0.573 -0.573 -0.573 -0.573 -0.573 ...
##  $ fi                                                   : num  -1.69 -1.32 -1.25 -1.1 -1.13 ...
##  $ fli                                                  : num  -0.812 -0.422 -0.724 -0.406 -0.43 ...
##  $ fri                                                  : num  -0.814 -0.586 -0.798 -0.263 -0.39 ...
##  $ fti                                                  : num  -1.62 -1.34 -1.1 -1.17 -1.16 ...
##  $ cli                                                  : num  -0.454 -0.454 -0.454 -0.454 -0.454 ...
##  $ cri                                                  : num  -0.468 -0.468 -0.468 -0.468 -0.468 ...
##  $ cti                                                  : num  -0.53 -0.53 -0.53 -0.53 -0.53 ...
##  $ ln_pi                                                : num  -4.06 -4.06 -4.06 -4.06 -4.06 ...
##  $ ln_fi                                                : num  -4.1 -2.04 -1.8 -1.41 -1.49 ...
##  $ ln_fli                                               : num  -4.8836 -0.0808 -1.0838 -0.0535 -0.0939
##  $ ln_fri                                               : num  -5.205 -0.493 -2.379 0.133 -0.052 ...
##  $ ln_fti                                               : num  -2.81 -1.64 -1.07 -1.22 -1.21 ...
##  $ ln_cli                                               : num  -3.91 -3.91 -3.91 -3.91 -3.91 ...
##  $ ln_cri                                               : num  -4.07 -4.07 -4.07 -4.07 -4.07 ...
##  $ ln_cti                                               : num  -5.53 -5.53 -5.53 -5.53 -5.53 ...
##  $ tot_crossw                                           : num  0.403 -0.738 -1.852 -0.903 -0.505 ...
##  $ number_of_                                           : num  0.505 -1.21 -2.925 0.505 0.505 ...
##  $ avg_crossw                                           : num  0.162 -0.223 -0.918 -1.193 -0.772 ...
##  $ tot_road_w                                           : num  -0.177 -0.971 -1.741 -1.532 -0.702 ...
##  $ north_veh                                            : num  -0.844 -0.674 -0.844 -0.647 -0.605 ...
##  $ north_ped                                            : num  -0.474 -0.474 -0.474 -0.474 -0.474 ...
##  $ east_veh                                             : num  -0.751 -0.753 -0.567 -0.404 -0.426 ...
##  $ east_ped                                             : num  -0.56 -0.56 -0.56 -0.56 -0.56 ...
##  $ south_veh                                            : num  -0.858 -0.842 -0.809 -0.832 -0.671 ...
##  $ south_ped                                            : num  -0.48 -0.48 -0.48 -0.48 -0.48 ...
##  $ west_veh                                             : num  -1.162 -0.535 -0.382 -0.428 -0.709 ...
##  $ west_ped                                             : num  -0.508 -0.508 -0.508 -0.508 -0.508 ...
##  $ total_lane                                           : num  0.269 -0.299 -0.299 -1.435 -1.435 ...
##  $ of_exclusi                                           : num  -0.498 -0.498 -0.498 -0.498 -0.498 ...
##  $ commercial                                           : num  -0.6504 0.0528 0.756 -0.6504 -0.6504 ...
##  $ distdt                                               : num  -0.629 -1.032 -1.028 -0.795 -0.718 ...
##  $ ln_distdt                                            : num  -0.277 -0.971 -0.963 -0.514 -0.397 ...
##  $ int_no                                               : int  40 263 1107 1364 225 1728 1623 978 382 9
##  $ pi_squared                                           : num  0.328 0.328 0.328 0.328 0.328 ...
##  $ fi_squared                                           : num  2.86 1.75 1.55 1.22 1.29 ...
##  $ distdt_squared                                       : num  0.395 1.065 1.058 0.633 0.516 ...
##  $ distdt_cubed                                         : num  -0.248 -1.098 -1.088 -0.503 -0.37 ...
##  $ tot_crossw_squared                                   : num  0.162 0.544 3.429 0.816 0.255 ...
##  $ avg_crossw_squared                                   : num  0.0263 0.0495 0.8435 1.4235 0.5959 ...
##  $ tot_road_w_squared                                   : num  0.0312 0.942 3.0298 2.3461 0.4928 ...
##  $ fli_squared                                          : num  0.659 0.178 0.524 0.165 0.185 ...
##  $ fri_squared                                          : num  0.6619 0.3437 0.6364 0.0691 0.1519 ...
##  $ fti_squared                                          : num  2.62 1.8 1.21 1.37 1.36 ...
##  $ all_pedest_1                                         : int  0 0 0 0 0 0 0 0 0 0 ...
##  $ median_1                                             : int  1 0 0 0 0 0 0 1 1 1 ...
##  $ green_stra_1                                         : int  0 0 0 0 0 0 0 0 1 0 ...
##  $ half_phase_1                                         : int  1 1 0 0 0 0 0 0 0 0 ...
##  $ any_ped_pr_1                                         : int  1 1 0 0 0 0 0 0 1 0 ...
##  $ ped_countd_1                                         : int  1 1 0 0 0 0 0 0 1 0 ...
```

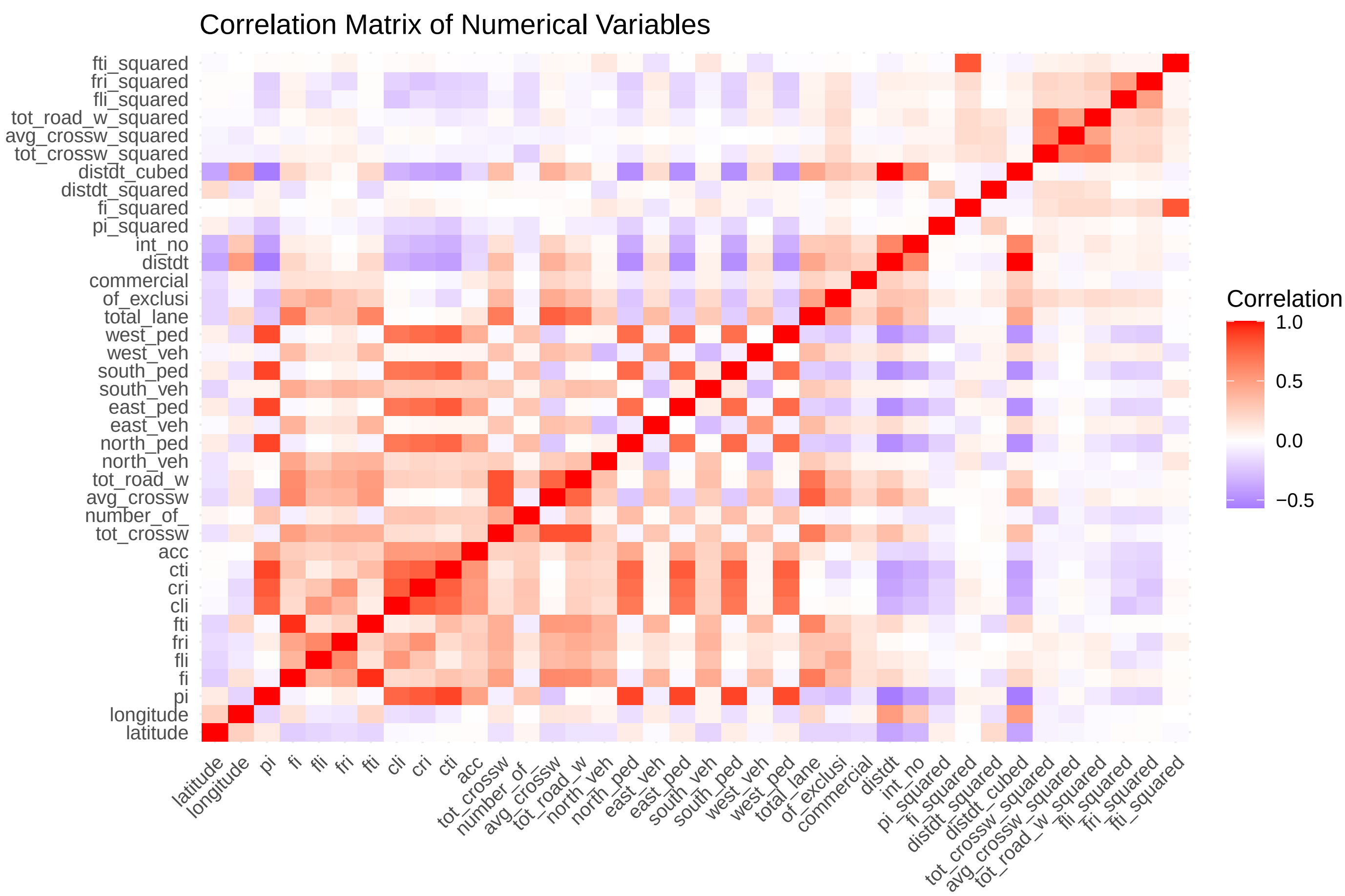
```
##  $ lt_protect_1                                              : int  1 0 1 0 0 0 0 1 1 1 ...
##  $ lt_restric_1                                              : int  1 1 1 0 0 1 1 1 1 1 ...
##  $ lt_prot_re_1                                              : int  1 1 1 0 0 1 1 1 1 1 ...
##  $ parking_1                                                 : int  0 0 0 0 0 0 0 0 0 0 ...
##  $ parking_2                                                 : int  1 0 0 0 1 0 0 0 0 0 ...
##  $ any_exclus_1                                              : int  0 0 0 0 0 0 0 1 1 1 ...
##  $ curb_exten_1                                              : int  0 0 1 0 0 0 0 0 0 0 ...
##  $ all_red_an_1                                              : int  0 0 0 0 0 0 0 0 0 0 ...
##  $ new_half_r_1                                              : int  1 1 0 0 0 0 0 0 0 0 ...
##  $ missing_date_ind_1                                        : int  0 0 0 0 0 0 0 0 0 0 ...
##  $ month_02                                                  : int  1 0 0 0 0 0 0 0 0 0 ...
##  $ month_03                                                  : int  0 0 0 0 0 0 0 0 0 0 ...
##  $ month_04                                                  : int  0 0 0 0 0 0 0 0 0 0 ...
##  $ month_05                                                  : int  0 0 0 0 0 0 0 0 0 0 ...
##  $ month_06                                                  : int  0 0 0 0 0 0 1 0 0 0 ...
##  $ month_07                                                  : int  0 0 0 0 0 0 0 0 0 0 ...
##  $ month_08                                                  : int  0 0 0 0 0 0 0 0 0 1 ...
##  $ month_09                                                  : int  0 0 1 0 0 0 0 0 0 0 ...
##  $ month_10                                                  : int  0 0 0 0 0 0 0 0 0 0 ...
##  $ month_11                                                  : int  0 1 0 0 1 1 0 1 1 0 ...
##  $ month_12                                                  : int  0 0 0 1 0 0 0 0 0 0 ...
##  $ dow_Monday                                                : int  0 0 0 1 0 0 0 0 0 1 ...
##  $ dow_Saturday                                              : int  0 0 0 0 0 0 0 0 0 0 ...
##  $ dow_Sunday                                                : int  0 0 0 0 0 0 0 0 0 0 ...
##  $ dow_Thursday                                              : int  0 0 0 0 0 0 0 0 0 0 ...
##  $ dow_Tuesday                                               : int  0 1 0 0 0 0 0 0 1 0 ...
##  $ dow_Wednesday                                             : int  0 0 1 0 1 1 1 1 0 0 ...
##  $ borough_grouped_Other                                     : int  0 0 0 0 0 1 0 0 0 0 ...
##  $ borough_grouped_Côte-des-Neiges-Notre-Dame-de-Grâce       : int  1 0 0 0 0 0 0 0 0 0 ...
##  $ borough_grouped_Le Plateau-Mont-Royal                     : int  0 0 0 0 0 0 0 0 0 0 ...
##  $ borough_grouped_Mercier-Hochelaga-Maisonneuve             : int  0 0 0 0 0 0 1 0 0 1 ...
##  $ borough_grouped_Montréal-Nord                             : int  0 0 0 0 0 0 0 0 0 0 ...
##  $ borough_grouped_Pointe-aux-Trembles-Rivières-des-Prairies: int  0 0 0 0 0 0 0 0 0 0 ...
##  $ borough_grouped_Rosemont-La Petite-Patrie                 : int  0 0 0 0 0 0 0 0 0 0 ...
##  $ borough_grouped_Saint-Laurent                             : int  0 0 0 0 0 0 0 0 0 0 ...
##  $ borough_grouped_Saint-Léonard                             : int  0 0 0 0 0 0 0 0 0 0 ...
##  $ borough_grouped_Sud-Ouest                                 : int  0 1 0 1 1 0 0 1 0 0 ...
##  $ borough_grouped_Ville-Marie                               : int  0 0 1 0 0 0 0 0 1 0 ...
##  $ borough_grouped_Villeray-Saint-Michel-Parc-Extension      : int  0 0 0 0 0 0 0 0 0 0 ...
##  $ acc                                                       : num  0 0 0 0 0 0 0 0 0 0 ...
```

Correlation Analysis

We can study the correlation among numerical variables and the target `acc`, to get a good idea of potential predictors (this is worked more in depth in the **Variable Selection** section):

```r
# Define the numerical variables from the dat dataset
num_vars <- colnames(dat)[sapply(dat, is.numeric)]

# Filter out 'ln_' prefixed variables from num_vars and ensure 'acc' is included
filtered_num_vars <- num_vars[!grepl("^ln_", num_vars) | num_vars == "acc"]

# Selecting only the correct numerical variables including 'acc'
numeric_dat <- dat %>%
  dplyr::select(all_of(filtered_num_vars))

# Standardize the numerical data (becomes a matrix)
std_num_dat <- f_standardize_data(numeric_dat, auto=TRUE)
```

```
# Re-add the target variable 'acc' to the standardized data (matrix)
std_num_dat <- cbind(std_num_dat, dat$acc)
colnames(std_num_dat)[ncol(std_num_dat)] <- "acc"

# Compute the correlation matrix
cor_matrix <- cor(std_num_dat, use = "complete.obs", method="spearman")

# Convert the correlation matrix to a long format
cor_data <- as.data.frame(as.table(cor_matrix))

# Plotting the correlation matrix
ggplot(cor_data, aes(x = Var1, y = Var2, fill = Freq)) +
  geom_tile() +
  scale_fill_gradient2(low = "blue", high = "red", mid = "white", midpoint = 0) +
  theme_minimal() +
  theme(axis.text.x = element_text(angle = 45, hjust = 1)) +
  labs(fill = "Correlation", x = "", y = "", title = "Correlation Matrix of Numerical Variables")
```



Since the correlation matrix is hard to understand, we look to focus our attention on the most linearly correlated covariates with the target variable `acc`. For this, we use three correlation metrics: Pearson, Spearman and Kendall. In particular, we use as a sorting criterion the Spearman correlation metric, which is the most robust to outliers.

```
# Compute correlations
correlations_with_acc <- f_compute_correlations(dat,
                                                target_var = "acc",
                                                standarize = FALSE,
                                                filtered_num_vars = filtered_num_vars)

# Convert row names to a column
```

```r
correlations_with_acc <- correlations_with_acc %>% rownames_to_column(var = "variable")

# Identifying the top 15 most positively and negatively correlated variables
top_n = 15
top_positively_correlated <- correlations_with_acc %>%
                             arrange(desc(spearman)) %>%
                             head(top_n+1)

top_negatively_correlated <- correlations_with_acc %>%
                             arrange(spearman) %>%
                             head(top_n+1)

# Filter out rows with negative correlation in positive correlated, and vice versa
top_positively_correlated <- top_positively_correlated[top_positively_correlated$spearman > 0, ]
top_negatively_correlated <- top_negatively_correlated[top_negatively_correlated$spearman < 0, ]

# filter the target variable out of the correlations
top_positively_correlated <- top_positively_correlated[-1, ]
top_negatively_correlated <- top_negatively_correlated[-1, ]

# reset the index of both corr tables
rownames(top_positively_correlated) <- seq(1, nrow(top_positively_correlated))
rownames(top_negatively_correlated) <- seq(1, nrow(top_negatively_correlated))

# Add a column containing the description of the variables
top_positively_correlated$description <- sapply(top_positively_correlated$variable,
                                                function(x) f_get_description(x, varnames_dict))
top_negatively_correlated$description <- sapply(top_negatively_correlated$variable,
                                                function(x) f_get_description(x, varnames_dict))

# Convert the description column to a character vector if it's not already
top_positively_correlated$description <- as.character(top_positively_correlated$description)
top_negatively_correlated$description <- as.character(top_negatively_correlated$description)
```

Let's inspect both of these tables:

```r
# Presenting the tables
print(top_positively_correlated)
```

```
##      variable   pearson  spearman   kendall
## 1         cti 0.4861139 0.5419556 0.4134257
## 2         cli 0.3589527 0.5173048 0.3939687
## 3         cri 0.3923028 0.5073740 0.3857580
## 4          pi 0.3694418 0.4763539 0.3577246
## 5   north_ped 0.3215824 0.4450390 0.3337312
## 6   south_ped 0.3287241 0.4420143 0.3295267
## 7    east_ped 0.3256322 0.4355127 0.3260887
## 8    west_ped 0.3141225 0.4106774 0.3078888
## 9  tot_road_w 0.2337136 0.2722756 0.2018397
## 10        fri 0.1379657 0.2653474 0.1995335
## 11         fi 0.2586556 0.2626369 0.1931364
## 12 number_of_ 0.1921787 0.2447570 0.2139843
## 13        fti 0.2439317 0.2430690 0.1786135
## 14 tot_crossw 0.2097814 0.2290229 0.1690196
## 15  south_veh 0.2038303 0.2285701 0.1699891
##
## 1               number of pedestrian-vehicle prohibited interactions over each 15 min intervals during the
## 2   number of pedestrian-vehicle left turning potential interactions over each 15 min intervals during the
## 3  number of pedestrian-vehicle right turning potential interactions over each 15 min intervals during the
```

```
## 4                                                          average annual daily flow for pedest
## 5                                                   average annual daily flow for pedestrians heading n
## 6                                                  average annual daily flow for pedestrians heading sou
## 7                                                    average annual daily flow for pedestrians heading
## 8                                                    average annual daily flow for pedestrians heading
## 9                                             sum of the road widths (outside crosswalks) along each appr
## 10                                                   average annual daily flow for vehicules turning r
## 11                                                          average annual daily flow for vehic
## 12
## 13                                           average annual daily flow for vehicules going through (strai
## 14                                                     sum of the crosswalk widths along each appr
## 15                                                   average annual daily flow for vehicules heading s
```

```r
print(top_negatively_correlated)
```

```
##              variable      pearson     spearman      kendall
## 1         fri_squared  0.029827409 -0.180394812 -0.135549685
## 2              distdt -0.130384802 -0.167872998 -0.122310530
## 3        distdt_cubed -0.079023993 -0.167872998 -0.122310530
## 4         fli_squared  0.050805376 -0.164115665 -0.120890883
## 5          pi_squared  0.155474369 -0.097318432 -0.079427235
## 6  tot_road_w_squared -0.005836319 -0.071042119 -0.052838389
## 7  tot_crossw_squared  0.023425140 -0.064026451 -0.047539349
## 8  avg_crossw_squared  0.007428100 -0.045963327 -0.034173479
## 9          of_exclusi  0.030620512 -0.024994823 -0.021647481
## 10        fti_squared  0.105081306 -0.010964650 -0.008742668
## 11     distdt_squared -0.044071642 -0.009958092 -0.007667958
## 12          longitude -0.002886844 -0.005784827 -0.004372044
##                          description
## 1                               <NA>
## 2               distance from downtown
## 3                               <NA>
## 4                               <NA>
## 5                               <NA>
## 6                               <NA>
## 7                               <NA>
## 8                               <NA>
## 9  number of exclusive left turn lane
## 10                              <NA>
## 11                              <NA>
## 12                              <NA>
```

Visualizations

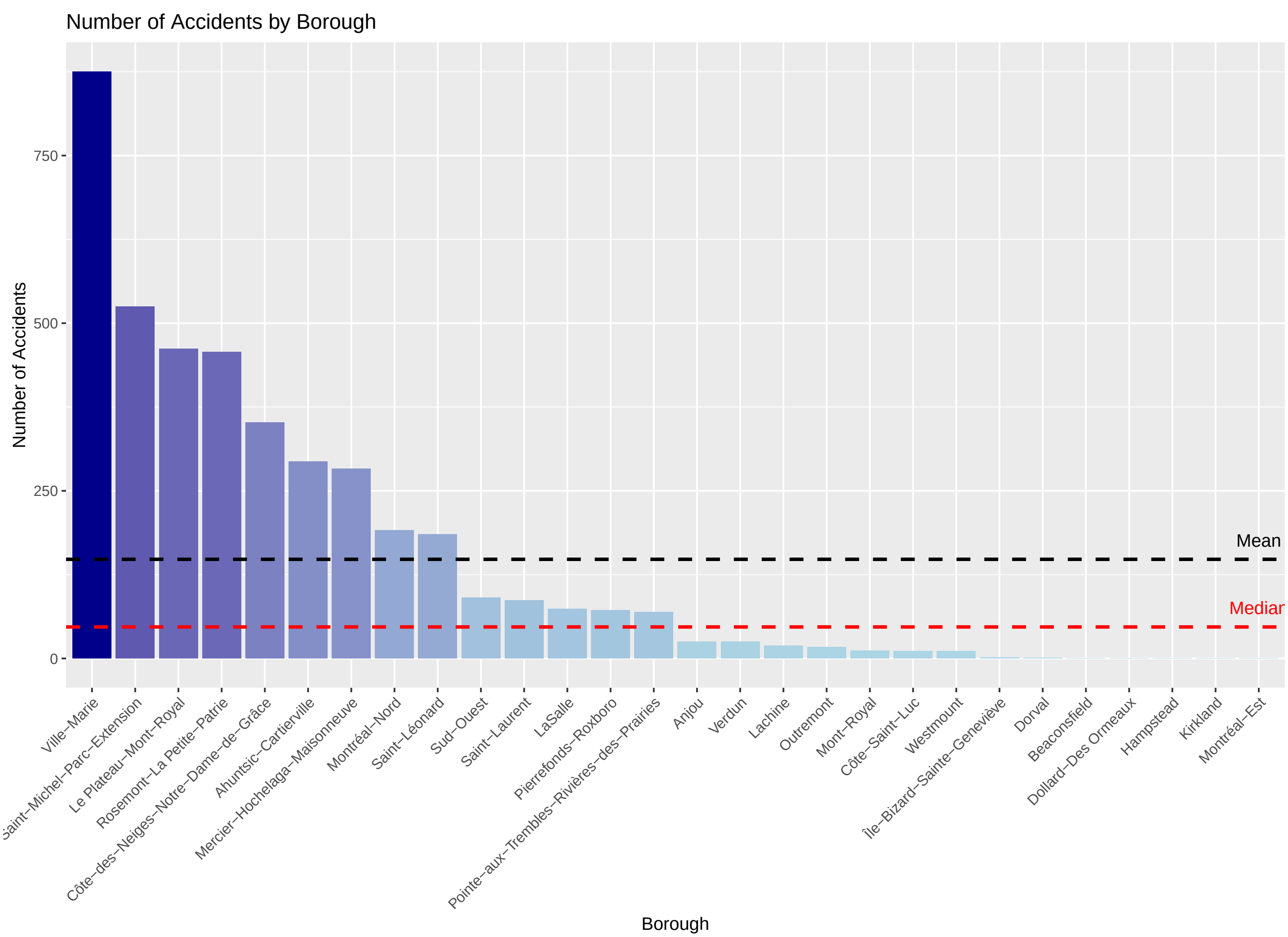
We visualize a number of variables of interest to get further insights into the data.

Distribution of Accidents per Borough

Question: Which borough has the most accidents?

```r
# Calculate total accidents per borough
borough_accidents <- dat_no_group %>%
  dplyr::group_by(borough) %>%
  dplyr::summarize(total_accidents = sum(acc, na.rm = TRUE)) %>%
  dplyr::arrange(desc(total_accidents))

# Calculate the median and mean number of accidents
```

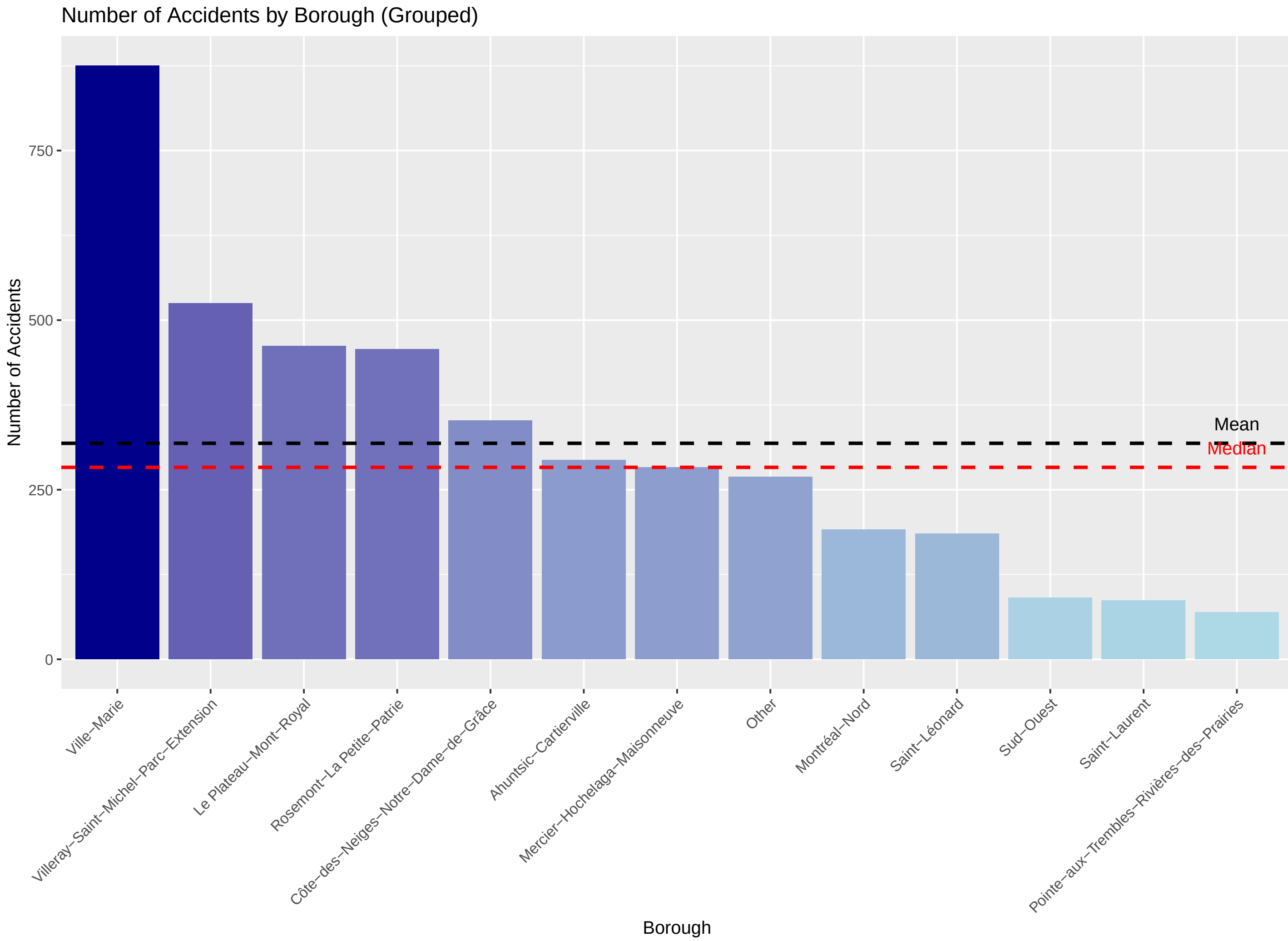
```r
median_accidents <- median(borough_accidents$total_accidents, na.rm = TRUE)
mean_accidents <- mean(borough_accidents$total_accidents, na.rm = TRUE)

# Bar Chart of Accidents by Borough using borough_accidents
ggplot(borough_accidents, aes(
    x = reorder(borough, -total_accidents),
    y = total_accidents
  )) +
  geom_bar(stat = "identity", aes(fill = total_accidents)) +
  geom_hline(yintercept = median_accidents, color = "red", linetype = "dashed", linewidth = 1) +
  geom_hline(yintercept = mean_accidents, color = "black", linetype = "dashed", linewidth = 1) +
  annotate("text", x = nrow(borough_accidents), y = median_accidents,
           label = "Median", vjust = -1, color = "red") +
  annotate("text", x = nrow(borough_accidents), y = mean_accidents,
           label = "Mean", vjust = -1, color = "black") +
  scale_fill_gradient(low = "lightblue", high = "darkblue") +  # Gradient of blues
  ggtitle("Number of Accidents by Borough") +
  xlab("Borough") +
  ylab("Number of Accidents") +
  theme(
    axis.text.x = element_text(angle = 45, hjust = 1),
    legend.position = "none"  # Remove legend and rotate labels
  )
```



An important thing to notice is that some of the boroughts barely contain any data, which demonstrates why we should group the variables.

After grouping, we can repeat this plot for comparison:

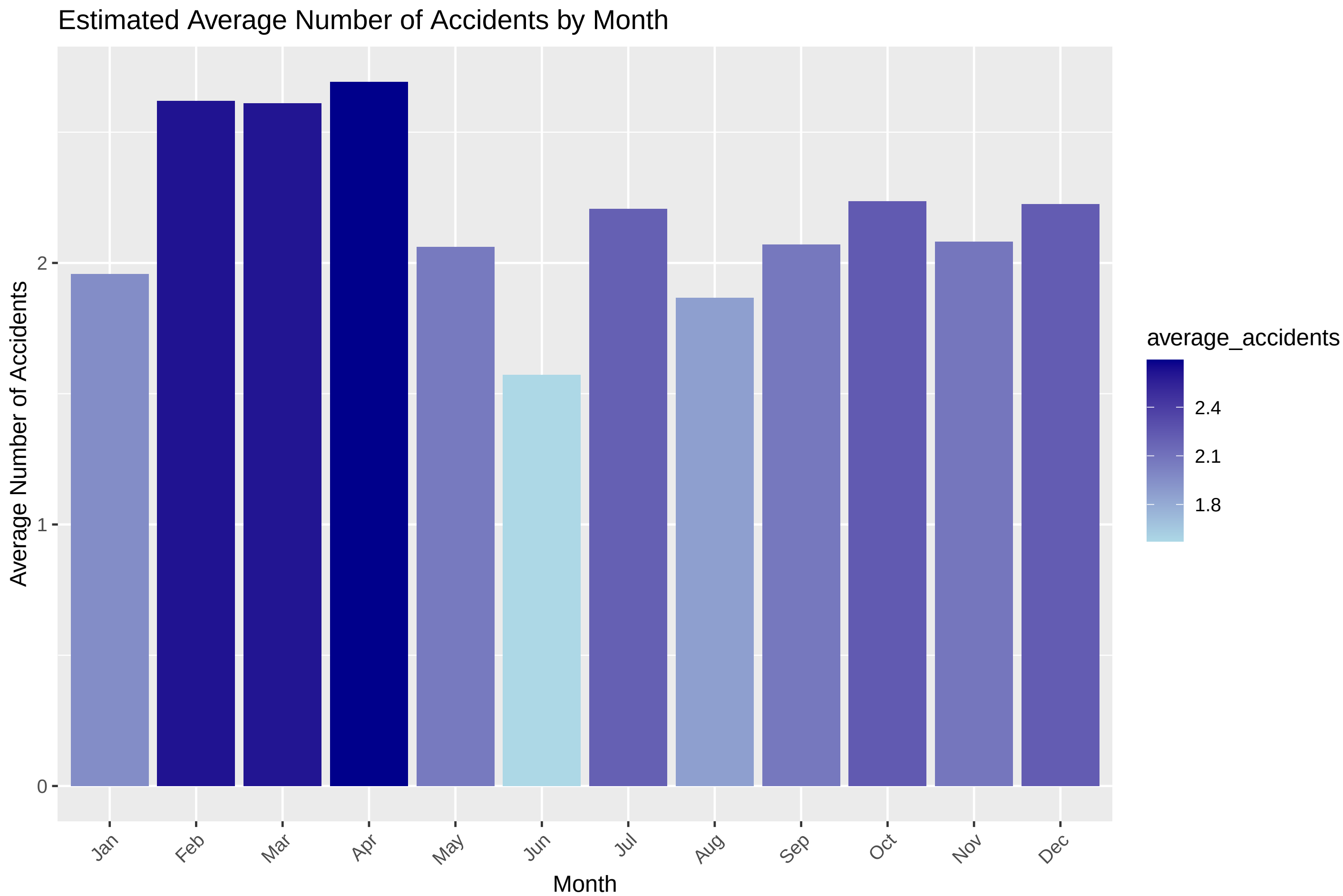
```r
# Calculate total accidents per borough
borough_accidents <- dat %>%
  dplyr::group_by(borough_grouped) %>%
  dplyr::summarize(total_accidents = sum(acc, na.rm = TRUE)) %>%
  dplyr::arrange(desc(total_accidents))

# Calculate the median and mean number of accidents
median_accidents <- median(borough_accidents$total_accidents, na.rm = TRUE)
mean_accidents <- mean(borough_accidents$total_accidents, na.rm = TRUE)

# Bar Chart of Accidents by Borough using borough_accidents
ggplot(borough_accidents, aes(
    x = reorder(borough_grouped, -total_accidents),
    y = total_accidents
  )) +
  geom_bar(stat = "identity", aes(fill = total_accidents)) +
  geom_hline(yintercept = median_accidents, color = "red", linetype = "dashed", linewidth = 1) +
  geom_hline(yintercept = mean_accidents, color = "black", linetype = "dashed", linewidth = 1) +
  annotate("text", x = nrow(borough_accidents), y = median_accidents,
           label = "Median", vjust = -1, color = "red") +
  annotate("text", x = nrow(borough_accidents), y = mean_accidents,
           label = "Mean", vjust = -1, color = "black") +
  scale_fill_gradient(low = "lightblue", high = "darkblue") +  # Gradient of blues
  ggtitle("Number of Accidents by Borough (Grouped)") +
  xlab("Borough") +
  ylab("Number of Accidents") +
  theme(
    axis.text.x = element_text(angle = 45, hjust = 1),
    legend.position = "none"  # Remove legend and rotate labels
  )
```



Aggregated Number of Accidents Month by Month

It is important to understand the seasonality of the accidents, and how the number of accidents varies from month to month.

```r
# Calculate average accidents per month
average_monthly_accidents <- dat %>%
  dplyr::group_by(month) %>%
  dplyr::summarise(average_accidents = mean(acc, na.rm = TRUE)) %>%
  dplyr::arrange(month)

# Create a gradient of blues
blue_gradient <- scale_fill_gradient(low = "lightblue", high = "darkblue")

# Bar Chart of Average Accidents by Month with gradient fill
ggplot(average_monthly_accidents, aes(x = month, y = average_accidents, fill = average_accidents)) +
  geom_bar(stat = "identity") +
  ggtitle("Estimated Average Number of Accidents by Month") +
  xlab("Month") +
  ylab("Average Number of Accidents") +
  scale_x_discrete(labels = c('Jan', 'Feb', 'Mar', 'Apr', 'May', 'Jun', 'Jul', 'Aug', 'Sep', 'Oct', 'Nov', 'De
  theme(axis.text.x = element_text(angle = 45, hjust = 1)) +
  blue_gradient
```

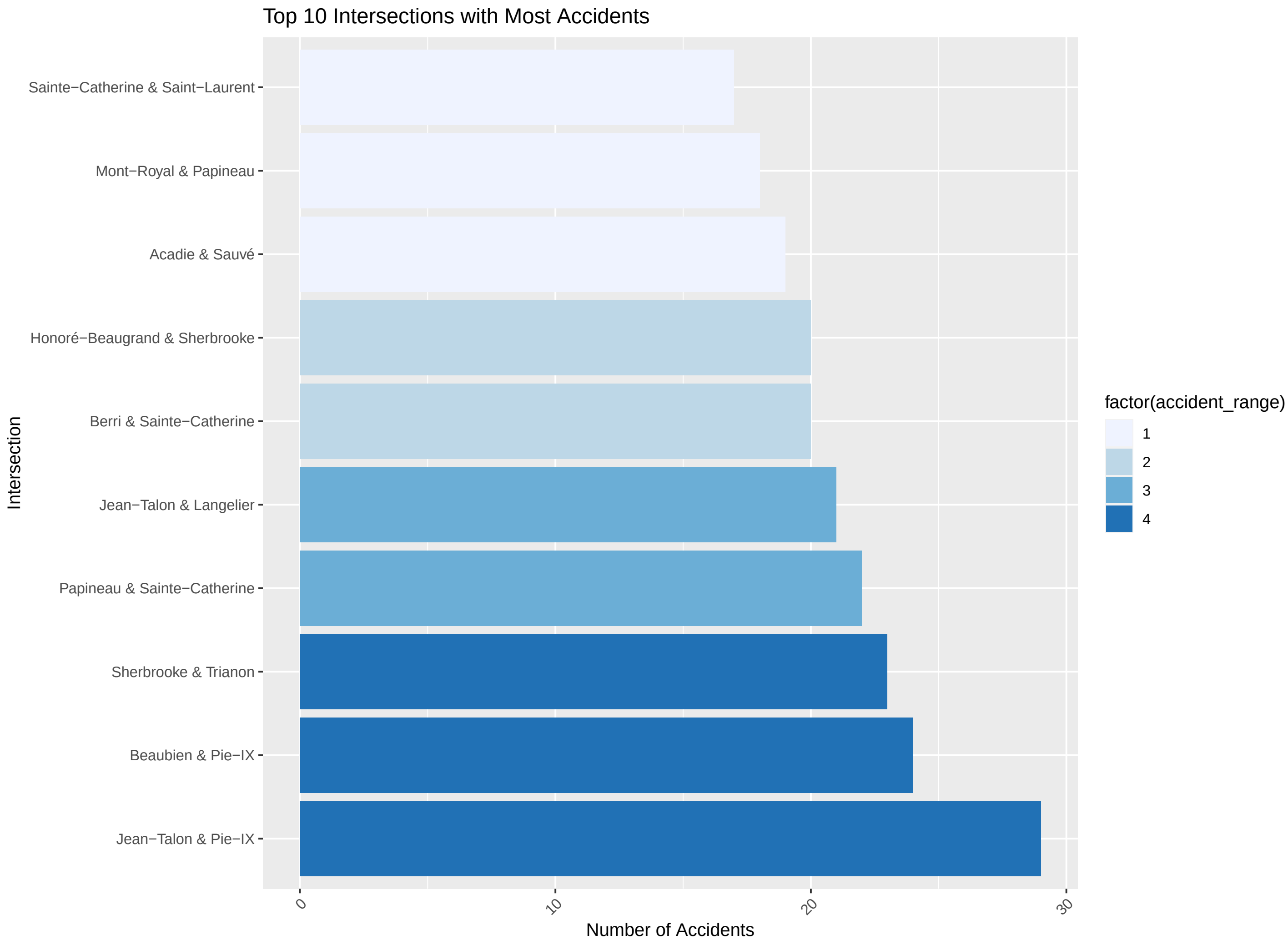


Top and Bottom 10 Intersections with Most Accidents

We can visualize the top 10 intersections with the most accidents, and the bottom 10 intersections with the least accidents.

```r
# Assuming 'dat' and 'inter_names' are your dataframes
# Join 'dat' with 'inter_names' to include 'rue_1' and 'rue_2' based on 'int_no'
dat_with_inter_names <- dplyr::left_join(dat, inter_names, by = "int_no")

# Calculate total accidents per intersection
intersection_accidents <- dat_with_inter_names %>%
  dplyr::group_by(int_no, rue_1, rue_2) %>%
  dplyr::summarise(total_accidents = sum(acc, na.rm = TRUE), .groups = 'drop') %>%
  dplyr::arrange(dplyr::desc(total_accidents))

# Extract the top 10 intersections
top_intersections <- head(intersection_accidents, 10)

# Create a factor variable for accident ranges
top_intersections$accident_range <- cut(
  top_intersections$total_accidents,
  breaks = quantile(top_intersections$total_accidents, probs = seq(0, 1, length.out = 5), na.rm = TRUE),
  include.lowest = TRUE,
  labels = FALSE
)

# Plotting the top 10 intersections with gradient fill
ggplot(top_intersections, aes(x = reorder(paste(rue_1, rue_2, sep = " & "), -total_accidents), y = total_accid
  geom_bar(stat = "identity") +
```

```r
  scale_fill_brewer(palette = "Blues", direction = 1) +
  ggtitle("Top 10 Intersections with Most Accidents") +
  xlab("Intersection") +
  ylab("Number of Accidents") +
  theme(axis.text.x = element_text(angle = 45, hjust = 1)) +
  coord_flip()  # Flip coordinates for horizontal layout
```




```r
# Note: Adjusted to "Top 10" as per the actual data extraction in the code.
```

Additional Data Preparation

Train-validation split

In order to perform variable selection, and be able to compare the different methods, we will split the data into a training and validation set.

```r
# Set seed for reproducibility
set.seed(123)

# Split data - example using dat
trainIndex <- createDataPartition(dat$acc,
                                  p = .8,  # 80% of the data
                                  list = FALSE,
                                  times = 1)
```

```r
dat_train <- dat[ trainIndex,]
dat_val   <- dat[-trainIndex,]

# Print dimensions of the dat_train and dat_val
cat("Dimensions of dat_train:", dim(dat_train), "\n")
```

```
## Dimensions of dat_train: 1493 65
```

```r
cat("Dimensions of dat_val:", dim(dat_val), "\n")
```

```
## Dimensions of dat_val: 371 65
```

```r
# Repeat for dat_dum if necessary
trainIndex_dum <- createDataPartition(dat_dum$acc,
                                      p = .8,
                                      list = FALSE,
                                      times = 1)
dat_dum_train <- dat_dum[ trainIndex_dum,]
dat_dum_val   <- dat_dum[-trainIndex_dum,]

# Print dimensions of the dat_dum_train and dat_dum_val
cat("Dimensions of dat_dum_train:", dim(dat_dum_train), "\n")
```

```
## Dimensions of dat_dum_train: 1493 92
```

```r
cat("Dimensions of dat_dum_val:", dim(dat_dum_val), "\n")
```

```
## Dimensions of dat_dum_val: 371 92
```

Cleanup

```r
# Drop a number of objects which are no longer used in the rest of the script and clean memory
rm(dat_orig, dat_no_group, dat_with_inter_names, intersection_accidents)
gc(verbose=FALSE);
```

```
##           used  (Mb) gc trigger  (Mb) max used  (Mb)
## Ncells 3318080 177.3    5586901 298.4  5586901 298.4
## Vcells 6222409  47.5   12255594  93.6 10146118  77.5
```

Variable Selection

In this section to keep it short, we will perform variable selection using the following methods:

- Stepwise with AIC/BIC
- Lasso selection
- RF Importance Selection
- Top Spearman-correlated covariates with acc

Stepwise with AIC/BIC

A couple of important interactions to consider:

- **Time interactions**: With `month` and day of the week `dow`
- **Traffic Flow and Pedestrian Protection Measures**: Interactions between the average annual daily flow for vehicles and pedestrians (e.g., `fi`, `pi`) and pedestrian protection measures (`any_ped_pr`, `lt_protect`, `lt_restric`, `lt_prot_re`, `ped_countd`, `curb_exten`) could reveal how traffic volume interacts with safety measures.
- **Road Characteristics and Safety Measures:** The presence of medians, exclusive lanes, and the total width of roads (`median`, `any_exclus`, `tot_road_w`) may have different impacts on safety when combined with pedestrian safety measures.
- **Directional Traffic Flow with Specific Safety Measures:** Examining how the flow of traffic in specific directions (e.g., `north_veh`, `east_veh`, `south_veh`, `west_veh`) interacts with pedestrian phases and countdowns could highlight directional risks.
- **Pedestrian and Vehicle Flow Interactions:** The interactions between pedestrian flow (`pi`, `north_ped`, `east_ped`, `south_ped`, `west_ped`) and vehicle flow (`fi`, `north_veh`, `east_veh`, `south_veh`, `west_veh`) could help understand how pedestrian safety is affected by vehicle traffic direction and volume.
- **Distance from Downtown and Safety Measures:** The effect of an intersection's distance from downtown (`distdt`, `ln_distdt`) on the effectiveness of safety measures might vary, considering that downtown areas could have different traffic and pedestrian patterns.
- **Temporal Factors and Traffic Flow:** The interaction between temporal factors (`month`, `dow`) and traffic flow variables (fi, pi) might uncover seasonal or weekly patterns in pedestrian safety.

```r
# Create the initial model with extended interactions
initial_model <- lm(acc ~ .

                    # Existing interactions with month and day of week
                    + month * cli
                    + month * cri
                    + month * cti
                    + month * ln_cli
                    + month * ln_cri
                    + month * ln_cti
                    + dow * cli
                    + dow * cri
                    + dow * cti
                    + dow * ln_cli
                    + dow * ln_cri
                    + dow * ln_cti

                    # Traffic Flow and Pedestrian Protection Measures
                    + fi * any_ped_pr
                    + pi * lt_protect
                    + fi * lt_restric
                    + pi * lt_prot_re
                    + fi * ped_countd
                    + pi * curb_exten

                    # Road Characteristics and Safety Measures
                    + median * any_ped_pr
                    + any_exclus * lt_protect
                    + tot_road_w * curb_exten

                    # Directional Traffic Flow with Specific Safety Measures
                    + north_veh * half_phase
                    + east_veh * ped_countd
                    + south_veh * green_stra
                    + west_veh * any_ped_pr

                    # Pedestrian and Vehicle Flow Interactions
```

```
                    + pi * fi
                    + north_ped * north_veh
                    + east_ped * east_veh
                    + south_ped * south_veh
                    + west_ped * west_veh

                    # Distance from Downtown and Safety Measures
                    + ln_distdt * any_ped_pr
                    + distdt * lt_protect

                    # Temporal Factors and Traffic Flow
                    + month * fi
                    + dow * pi

                    # Interactions with Polynomial Terms
                    + pi_squared * lt_protect
                    + fi_squared * any_ped_pr
                    + distdt_squared * green_stra
                    + distdt_cubed * half_phase
                    + tot_crossw_squared * lt_restric
                    + avg_crossw_squared * ped_countd
                    + tot_road_w_squared * curb_exten
                    + fli_squared * east_veh
                    + fri_squared * west_veh
                    + fti_squared * north_veh

                    # Ignore the index int_no
                    - int_no

                    , data = dat_train)


# Perform stepwise feature selection with interactions using AIC
stepwise_aic <- stepAIC(initial_model, direction = "both", k = 2, trace=FALSE)

# Display the summary of the final model
summary(stepwise_aic)
```

```
##
## Call:
## lm(formula = acc ~ latitude + longitude + pi + fi + fli + cli +
##     cri + cti + ln_pi + ln_fri + ln_cri + ln_cti + number_of_ +
##     tot_road_w + median + green_stra + half_phase + any_ped_pr +
##     ped_countd + lt_protect + lt_restric + lt_prot_re + parking +
##     north_veh + north_ped + east_veh + south_veh + south_ped +
##     west_veh + any_exclus + commercial + curb_exten + distdt +
##     ln_distdt + missing_date_ind + month + dow + borough_grouped +
##     pi_squared + distdt_squared + distdt_cubed + tot_crossw_squared +
##     avg_crossw_squared + tot_road_w_squared + fli_squared + fri_squared +
##     fti_squared + cli:month + cri:month + cti:month + cli:dow +
##     cri:dow + cti:dow + fi:lt_restric + pi:lt_prot_re + fi:ped_countd +
##     south_veh:south_ped + any_ped_pr:ln_distdt + fi:month + pi:dow +
##     lt_protect:pi_squared + green_stra:distdt_squared + half_phase:distdt_cubed +
##     lt_restric:tot_crossw_squared + ped_countd:avg_crossw_squared +
##     east_veh:fli_squared + west_veh:fri_squared + north_veh:fti_squared,
##     data = dat_train)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
```

```
## -7.2714 -1.2360 -0.2162  0.8780 18.7150
##
## Coefficients: (5 not defined because of singularities)
##                                                          Estimate Std. Error
## (Intercept)                                             1.818e+00  5.716e-01
## latitude                                                2.864e-01  1.990e-01
## longitude                                              -2.641e-01  1.819e-01
## pi                                                     -1.402e+00  7.068e-01
## fi                                                      7.739e+05  1.010e+06
## fli                                                     3.038e-01  1.599e-01
## cli                                                     1.339e+00  8.650e-01
## cri                                                     1.671e+00  8.522e-01
## cti                                                     3.326e-01  1.154e+00
## ln_pi                                                   1.071e+00  2.223e-01
## ln_fri                                                  3.263e-01  1.288e-01
## ln_cri                                                 -5.861e-01  1.744e-01
## ln_cti                                                 -4.129e-01  1.516e-01
## number_of_                                             -1.718e-01  9.914e-02
## tot_road_w                                              6.698e-01  1.243e-01
## median1                                                -7.525e-01  1.864e-01
## green_stra1                                             4.818e-01  2.461e-01
## half_phase1                                            -2.537e-01  2.291e-01
## any_ped_pr1                                            -2.920e-02  2.571e-01
## ped_countd1                                             3.137e-01  1.809e-01
## lt_protect1                                            -2.176e-01  1.723e-01
## lt_restric1                                            -3.338e-01  2.578e-01
## lt_prot_re1                                             3.921e-01  3.002e-01
## parking1                                               -1.358e-01  2.081e-01
## parking2                                               -8.335e-01  1.796e-01
## north_veh                                              -4.012e+05  5.235e+05
## north_ped                                               5.856e-01  1.935e-01
## east_veh                                               -3.437e+05  4.485e+05
## south_veh                                              -3.742e+05  4.883e+05
## south_ped                                               2.539e-01  1.746e-01
## west_veh                                               -3.362e+05  4.387e+05
## any_exclus1                                            -4.568e-01  1.978e-01
## commercial                                              1.769e-01  6.954e-02
## curb_exten1                                            -7.154e-01  3.091e-01
## distdt                                                 -1.161e+00  4.994e-01
## ln_distdt                                               2.326e+00  4.895e-01
## missing_date_ind1                                       2.221e+00  9.783e-01
## month02                                                 6.465e-01  4.739e-01
## month03                                                 9.934e-01  4.306e-01
## month04                                                 6.600e-01  4.285e-01
## month05                                                 8.241e-01  4.393e-01
## month06                                                 1.602e-01  4.757e-01
## month07                                                 1.126e-01  5.372e-01
## month08                                                 7.687e-01  5.746e-01
## month09                                                 4.194e-01  4.343e-01
## month10                                                 3.736e-01  4.319e-01
## month11                                                 1.370e-01  4.318e-01
## month12                                                 3.183e-01  4.605e-01
## dowMonday                                              -1.161e-01  2.596e-01
## dowSaturday                                            -5.456e-01  2.543e+00
## dowSunday                                               9.482e+00  4.705e+00
## dowThursday                                             1.315e-01  2.395e-01
## dowTuesday                                              1.273e-01  2.425e-01
## dowWednesday                                           -4.195e-02  2.364e-01
## borough_groupedOther                                   -3.483e-01  4.451e-01
## borough_groupedCôte-des-Neiges-Notre-Dame-de-Grâce     -3.438e-02  4.564e-01
```

```
## borough_groupedLe Plateau-Mont-Royal                       -3.686e-02  5.207e-01
## borough_groupedMercier-Hochelaga-Maisonneuve                5.079e-01  4.845e-01
## borough_groupedMontréal-Nord                                7.347e-01  4.808e-01
## borough_groupedPointe-aux-Trembles-Rivières-des-Prairies -9.166e-01  8.406e-01
## borough_groupedRosemont-La Petite-Patrie                    8.110e-01  3.940e-01
## borough_groupedSaint-Laurent                               -1.012e+00  3.921e-01
## borough_groupedSaint-Léonard                                9.462e-01  4.549e-01
## borough_groupedSud-Ouest                                    3.282e-01  6.417e-01
## borough_groupedVille-Marie                                  6.113e-01  6.017e-01
## borough_groupedVilleray-Saint-Michel-Parc-Extension         9.507e-01  3.308e-01
## pi_squared                                                 -1.314e-01  2.730e-02
## distdt_squared                                              8.358e-01  3.100e-01
## distdt_cubed                                               -1.857e-01  6.942e-02
## tot_crossw_squared                                          9.277e-02  6.327e-02
## avg_crossw_squared                                          2.082e-02  4.589e-02
## tot_road_w_squared                                         -1.075e-01  3.311e-02
## fli_squared                                                -1.947e-01  6.305e-02
## fri_squared                                                -8.979e-02  3.230e-02
## fti_squared                                                -7.575e-02  5.782e-02
## cli:month02                                                -8.595e-01  8.965e-01
## cli:month03                                                -8.961e-01  8.142e-01
## cli:month04                                                 1.813e-01  8.204e-01
## cli:month05                                                 3.300e-01  8.927e-01
## cli:month06                                                -6.015e-01  1.003e+00
## cli:month07                                                -1.065e+00  8.470e-01
## cli:month08                                                 1.725e+00  1.150e+00
## cli:month09                                                -1.023e+00  8.555e-01
## cli:month10                                                 2.012e-01  8.346e-01
## cli:month11                                                -1.404e-01  8.568e-01
## cli:month12                                                -1.312e+00  8.815e-01
## cri:month02                                                 6.730e-01  8.373e-01
## cri:month03                                                 9.492e-01  7.752e-01
## cri:month04                                                -1.246e+00  7.069e-01
## cri:month05                                                -9.662e-01  9.817e-01
## cri:month06                                                -8.520e-01  9.904e-01
## cri:month07                                                 5.939e-02  9.074e-01
## cri:month08                                                 1.237e-01  1.112e+00
## cri:month09                                                -8.555e-01  6.759e-01
## cri:month10                                                -1.207e+00  8.315e-01
## cri:month11                                                -9.395e-02  7.113e-01
## cri:month12                                                -1.748e+00  7.693e-01
## cti:month02                                                 4.737e-01  1.151e+00
## cti:month03                                                 9.843e-02  1.097e+00
## cti:month04                                                 7.622e-01  1.015e+00
## cti:month05                                                 1.127e-01  1.119e+00
## cti:month06                                                 6.532e-01  1.299e+00
## cti:month07                                                 1.374e-01  1.136e+00
## cti:month08                                                -3.922e-01  1.669e+00
## cti:month09                                                 1.408e+00  1.110e+00
## cti:month10                                                 1.066e+00  1.101e+00
## cti:month11                                                -7.981e-01  1.069e+00
## cti:month12                                                 2.672e+00  1.127e+00
## cli:dowMonday                                              -9.507e-01  4.802e-01
## cli:dowSaturday                                                    NA          NA
## cli:dowSunday                                               9.947e+00  1.213e+01
## cli:dowThursday                                            -2.857e-01  5.067e-01
## cli:dowTuesday                                              8.687e-02  4.207e-01
## cli:dowWednesday                                           -6.477e-01  4.808e-01
## cri:dowMonday                                              -1.046e+00  6.948e-01
## cri:dowSaturday                                                    NA          NA
```

```
## cri:dowSunday                          1.408e+00  2.763e+01
## cri:dowThursday                       -4.187e-01  6.532e-01
## cri:dowTuesday                        -1.381e+00  6.340e-01
## cri:dowWednesday                      -1.397e+00  6.301e-01
## cti:dowMonday                          5.679e-01  6.947e-01
## cti:dowSaturday                               NA         NA
## cti:dowSunday                          1.896e+01  2.570e+01
## cti:dowThursday                        1.237e+00  6.894e-01
## cti:dowTuesday                        -3.714e-01  6.702e-01
## cti:dowWednesday                       6.020e-01  6.148e-01
## fi:lt_restric1                        -3.173e-01  1.541e-01
## pi:lt_prot_re1                         4.472e-01  2.468e-01
## fi:ped_countd1                         3.154e-01  1.546e-01
## south_veh:south_ped                   -4.448e-01  1.240e-01
## any_ped_pr1:ln_distdt                 -2.689e-01  1.486e-01
## fi:month02                             1.237e+00  5.460e-01
## fi:month03                             1.640e+00  4.962e-01
## fi:month04                             6.492e-01  4.565e-01
## fi:month05                             7.356e-01  4.852e-01
## fi:month06                             6.425e-01  5.061e-01
## fi:month07                             7.458e-01  5.319e-01
## fi:month08                            -6.528e-01  7.231e-01
## fi:month09                             6.480e-02  4.730e-01
## fi:month10                             1.172e-01  4.665e-01
## fi:month11                             7.563e-01  4.628e-01
## fi:month12                            -2.566e-01  5.067e-01
## pi:dowMonday                           3.117e-01  6.305e-01
## pi:dowSaturday                                NA         NA
## pi:dowSunday                                  NA         NA
## pi:dowThursday                        -2.009e-01  6.381e-01
## pi:dowTuesday                          1.692e+00  6.425e-01
## pi:dowWednesday                        9.393e-01  6.323e-01
## lt_protect1:pi_squared                -6.562e-02  4.558e-02
## green_stra1:distdt_squared             1.650e-01  9.269e-02
## half_phase1:distdt_cubed               4.348e-02  2.715e-02
## lt_restric1:tot_crossw_squared        -1.174e-01  6.936e-02
## ped_countd1:avg_crossw_squared        -9.599e-02  5.758e-02
## east_veh:fli_squared                   3.250e-02  1.319e-02
## west_veh:fri_squared                   2.031e-02  7.005e-03
## north_veh:fti_squared                  5.684e-02  2.586e-02
##                                       t value Pr(>|t|)
## (Intercept)                             3.182 0.001499 **
## latitude                                1.439 0.150405
## longitude                              -1.452 0.146712
## pi                                     -1.984 0.047464 *
## fi                                      0.766 0.443565
## fli                                     1.900 0.057598 .
## cli                                     1.548 0.121933
## cri                                     1.961 0.050100 .
## cti                                     0.288 0.773254
## ln_pi                                   4.818 1.62e-06 ***
## ln_fri                                  2.532 0.011442 *
## ln_cri                                 -3.361 0.000797 ***
## ln_cti                                 -2.723 0.006550 **
## number_of_                             -1.733 0.083292 .
## tot_road_w                              5.387 8.46e-08 ***
## median1                                -4.036 5.74e-05 ***
## green_stra1                             1.958 0.050483 .
## half_phase1                            -1.107 0.268279
## any_ped_pr1                            -0.114 0.909573
```

```
## ped_countd1                                          1.734 0.083190 .
## lt_protect1                                         -1.262 0.207022
## lt_restric1                                         -1.295 0.195618
## lt_prot_re1                                          1.306 0.191709
## parking1                                            -0.653 0.514187
## parking2                                            -4.640 3.83e-06 ***
## north_veh                                           -0.766 0.443565
## north_ped                                            3.026 0.002526 **
## east_veh                                            -0.766 0.443565
## south_veh                                           -0.766 0.443565
## south_ped                                            1.454 0.146291
## west_veh                                            -0.766 0.443565
## any_exclus1                                         -2.309 0.021111 *
## commercial                                           2.544 0.011059 *
## curb_exten1                                         -2.314 0.020802 *
## distdt                                              -2.326 0.020188 *
## ln_distdt                                            4.752 2.23e-06 ***
## missing_date_ind1                                    2.270 0.023336 *
## month02                                              1.364 0.172796
## month03                                              2.307 0.021215 *
## month04                                              1.540 0.123709
## month05                                              1.876 0.060900 .
## month06                                              0.337 0.736410
## month07                                              0.210 0.834044
## month08                                              1.338 0.181152
## month09                                              0.966 0.334430
## month10                                              0.865 0.387153
## month11                                              0.317 0.751076
## month12                                              0.691 0.489589
## dowMonday                                           -0.447 0.654723
## dowSaturday                                         -0.215 0.830151
## dowSunday                                            2.015 0.044071 *
## dowThursday                                          0.549 0.582938
## dowTuesday                                           0.525 0.599652
## dowWednesday                                        -0.177 0.859172
## borough_groupedOther                                -0.783 0.434031
## borough_groupedCôte-des-Neiges-Notre-Dame-de-Grâce  -0.075 0.939973
## borough_groupedLe Plateau-Mont-Royal                -0.071 0.943572
## borough_groupedMercier-Hochelaga-Maisonneuve         1.048 0.294652
## borough_groupedMontréal-Nord                         1.528 0.126714
## borough_groupedPointe-aux-Trembles-Rivières-des-Prairies -1.090 0.275747
## borough_groupedRosemont-La Petite-Patrie             2.058 0.039747 *
## borough_groupedSaint-Laurent                        -2.582 0.009923 **
## borough_groupedSaint-Léonard                         2.080 0.037713 *
## borough_groupedSud-Ouest                             0.512 0.609070
## borough_groupedVille-Marie                           1.016 0.309845
## borough_groupedVilleray-Saint-Michel-Parc-Extension  2.874 0.004122 **
## pi_squared                                          -4.813 1.66e-06 ***
## distdt_squared                                       2.696 0.007106 **
## distdt_cubed                                        -2.675 0.007572 **
## tot_crossw_squared                                   1.466 0.142790
## avg_crossw_squared                                   0.454 0.650142
## tot_road_w_squared                                  -3.248 0.001189 **
## fli_squared                                         -3.088 0.002053 **
## fri_squared                                         -2.780 0.005513 **
## fti_squared                                         -1.310 0.190387
## cli:month02                                         -0.959 0.337867
## cli:month03                                         -1.100 0.271312
## cli:month04                                          0.221 0.825105
## cli:month05                                          0.370 0.711694
```

```
## cli:month06                                    -0.600 0.548819
## cli:month07                                    -1.257 0.208964
## cli:month08                                     1.500 0.133956
## cli:month09                                    -1.196 0.231807
## cli:month10                                     0.241 0.809492
## cli:month11                                    -0.164 0.869825
## cli:month12                                    -1.488 0.136950
## cri:month02                                     0.804 0.421682
## cri:month03                                     1.224 0.221005
## cri:month04                                    -1.763 0.078112 .
## cri:month05                                    -0.984 0.325182
## cri:month06                                    -0.860 0.389804
## cri:month07                                     0.065 0.947829
## cri:month08                                     0.111 0.911441
## cri:month09                                    -1.266 0.205823
## cri:month10                                    -1.452 0.146739
## cri:month11                                    -0.132 0.894944
## cri:month12                                    -2.272 0.023242 *
## cti:month02                                     0.412 0.680673
## cti:month03                                     0.090 0.928553
## cti:month04                                     0.751 0.452697
## cti:month05                                     0.101 0.919766
## cti:month06                                     0.503 0.615165
## cti:month07                                     0.121 0.903749
## cti:month08                                    -0.235 0.814213
## cti:month09                                     1.269 0.204731
## cti:month10                                     0.969 0.332714
## cti:month11                                    -0.747 0.455461
## cti:month12                                     2.371 0.017868 *
## cli:dowMonday                                  -1.980 0.047940 *
## cli:dowSaturday                                    NA       NA
## cli:dowSunday                                   0.820 0.412352
## cli:dowThursday                                -0.564 0.572939
## cli:dowTuesday                                  0.206 0.836441
## cli:dowWednesday                               -1.347 0.178156
## cri:dowMonday                                  -1.505 0.132595
## cri:dowSaturday                                    NA       NA
## cri:dowSunday                                   0.051 0.959355
## cri:dowThursday                                -0.641 0.521627
## cri:dowTuesday                                 -2.179 0.029525 *
## cri:dowWednesday                               -2.217 0.026772 *
## cti:dowMonday                                   0.817 0.413863
## cti:dowSaturday                                    NA       NA
## cti:dowSunday                                   0.738 0.460873
## cti:dowThursday                                 1.794 0.073051 .
## cti:dowTuesday                                 -0.554 0.579636
## cti:dowWednesday                                0.979 0.327645
## fi:lt_restric1                                 -2.059 0.039665 *
## pi:lt_prot_re1                                  1.812 0.070239 .
## fi:ped_countd1                                  2.040 0.041517 *
## south_veh:south_ped                            -3.588 0.000345 ***
## any_ped_pr1:ln_distdt                          -1.809 0.070676 .
## fi:month02                                      2.266 0.023632 *
## fi:month03                                      3.306 0.000972 ***
## fi:month04                                      1.422 0.155184
## fi:month05                                      1.516 0.129774
## fi:month06                                      1.270 0.204462
## fi:month07                                      1.402 0.161102
## fi:month08                                     -0.903 0.366786
## fi:month09                                      0.137 0.891055
```

```
## fi:month10                                    0.251 0.801604
## fi:month11                                    1.634 0.102438
## fi:month12                                   -0.506 0.612609
## pi:dowMonday                                  0.494 0.621115
## pi:dowSaturday                                   NA       NA
## pi:dowSunday                                     NA       NA
## pi:dowThursday                               -0.315 0.752985
## pi:dowTuesday                                 2.634 0.008547 **
## pi:dowWednesday                               1.486 0.137630
## lt_protect1:pi_squared                       -1.439 0.150263
## green_stra1:distdt_squared                    1.780 0.075229 .
## half_phase1:distdt_cubed                      1.602 0.109495
## lt_restric1:tot_crossw_squared               -1.693 0.090783 .
## ped_countd1:avg_crossw_squared               -1.667 0.095727 .
## east_veh:fli_squared                          2.465 0.013833 *
## west_veh:fri_squared                          2.899 0.003808 **
## north_veh:fti_squared                         2.198 0.028105 *
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 2.282 on 1342 degrees of freedom
## Multiple R-squared:  0.5425, Adjusted R-squared:  0.4914
## F-statistic: 10.61 on 150 and 1342 DF,  p-value: < 2.2e-16
```

```r
# Create a list object which will contain all the predictors for different methods

# Extract the model formula
model_formula <- formula(stepwise_aic)

# Extract terms from the formula
model_terms <- labels(terms(model_formula))

# Since the first term is usually the response variable (left of ~), we remove it to get only predictors
selected_vars <- model_terms[-1] # Removes the first element, which is the response variable 'acc'

# Pack into a list under the key "stepwise_bic"
selected_covariates <- list(stepwise_aic = selected_vars)

# Print the list to see the selected variables
print(selected_covariates)
```

```
## $stepwise_aic
##  [1] "longitude"                "pi"
##  [3] "fi"                       "fli"
##  [5] "cli"                      "cri"
##  [7] "cti"                      "ln_pi"
##  [9] "ln_fri"                   "ln_cri"
## [11] "ln_cti"                   "number_of_"
## [13] "tot_road_w"               "median"
## [15] "green_stra"               "half_phase"
## [17] "any_ped_pr"               "ped_countd"
## [19] "lt_protect"               "lt_restric"
## [21] "lt_prot_re"               "parking"
## [23] "north_veh"                "north_ped"
## [25] "east_veh"                 "south_veh"
## [27] "south_ped"                "west_veh"
## [29] "any_exclus"               "commercial"
## [31] "curb_exten"               "distdt"
## [33] "ln_distdt"                "missing_date_ind"
```

```
## [35] "month"                          "dow"
## [37] "borough_grouped"                "pi_squared"
## [39] "distdt_squared"                 "distdt_cubed"
## [41] "tot_crossw_squared"             "avg_crossw_squared"
## [43] "tot_road_w_squared"             "fli_squared"
## [45] "fri_squared"                    "fti_squared"
## [47] "cli:month"                      "cri:month"
## [49] "cti:month"                      "cli:dow"
## [51] "cri:dow"                        "cti:dow"
## [53] "fi:lt_restric"                  "pi:lt_prot_re"
## [55] "fi:ped_countd"                  "south_veh:south_ped"
## [57] "any_ped_pr:ln_distdt"           "fi:month"
## [59] "pi:dow"                         "lt_protect:pi_squared"
## [61] "green_stra:distdt_squared"      "half_phase:distdt_cubed"
## [63] "lt_restric:tot_crossw_squared" "ped_countd:avg_crossw_squared"
## [65] "east_veh:fli_squared"           "west_veh:fri_squared"
## [67] "north_veh:fti_squared"
```

Stepwise BIC

```r
# Perform stepwise feature selection with interactions using BIC
stepwise_bic <- stepAIC(initial_model, direction = "both", k = log(nrow(dat_train)), trace=FALSE)

# Extract the model formula
model_formula <- formula(stepwise_bic)

# Extract terms from the formula
model_terms <- labels(terms(model_formula))

# Since the first term is usually the response variable (left of ~), we remove it to get only predictors
selected_vars <- model_terms[-1] # Removes the first element, which is the response variable 'acc'

# Pack into a list under the key "stepwise_bic"
selected_covariates$stepwise_bic <- selected_vars

# Print the list to see the selected variables
print(selected_covariates)
```

```
## $stepwise_aic
##  [1] "longitude"                      "pi"
##  [3] "fi"                             "fli"
##  [5] "cli"                            "cri"
##  [7] "cti"                            "ln_pi"
##  [9] "ln_fri"                         "ln_cri"
## [11] "ln_cti"                         "number_of_"
## [13] "tot_road_w"                     "median"
## [15] "green_stra"                     "half_phase"
## [17] "any_ped_pr"                     "ped_countd"
## [19] "lt_protect"                     "lt_restric"
## [21] "lt_prot_re"                     "parking"
## [23] "north_veh"                      "north_ped"
## [25] "east_veh"                       "south_veh"
## [27] "south_ped"                      "west_veh"
## [29] "any_exclus"                     "commercial"
## [31] "curb_exten"                     "distdt"
## [33] "ln_distdt"                      "missing_date_ind"
## [35] "month"                          "dow"
## [37] "borough_grouped"                "pi_squared"
```

```
## [39] "distdt_squared"                 "distdt_cubed"
## [41] "tot_crossw_squared"             "avg_crossw_squared"
## [43] "tot_road_w_squared"             "fli_squared"
## [45] "fri_squared"                    "fti_squared"
## [47] "cli:month"                      "cri:month"
## [49] "cti:month"                      "cli:dow"
## [51] "cri:dow"                        "cti:dow"
## [53] "fi:lt_restric"                  "pi:lt_prot_re"
## [55] "fi:ped_countd"                  "south_veh:south_ped"
## [57] "any_ped_pr:ln_distdt"           "fi:month"
## [59] "pi:dow"                         "lt_protect:pi_squared"
## [61] "green_stra:distdt_squared"      "half_phase:distdt_cubed"
## [63] "lt_restric:tot_crossw_squared" "ped_countd:avg_crossw_squared"
## [65] "east_veh:fli_squared"           "west_veh:fri_squared"
## [67] "north_veh:fti_squared"
##
## $stepwise_bic
##  [1] "pi"                    "cli"                   "cti"
##  [4] "ln_pi"                 "ln_cti"                "tot_road_w"
##  [7] "median"                "green_stra"            "parking"
## [10] "north_veh"             "north_ped"             "south_veh"
## [13] "south_ped"             "west_veh"              "commercial"
## [16] "ln_distdt"             "missing_date_ind"      "dow"
## [19] "pi_squared"            "tot_road_w_squared"    "fri_squared"
## [22] "north_veh:north_ped"   "south_veh:south_ped"   "pi:dow"
## [25] "west_veh:fri_squared"
```

Lasso

```r
# Combine the response and predictor variables into a matrix for the training data
X_train_lasso <- model.matrix(acc ~ . - int_no, data = dat_dum_train)[, -1]  # Remove intercept column
y_train_lasso <- dat_dum_train$acc

# Set up a Lasso model with cross-validation on the training set
lasso_cv_model <- cv.glmnet(X_train_lasso, y_train_lasso, alpha = 1)  # alpha = 1 for Lasso

# # Plot the cross-validated mean squared error (CV MSE) as a function of log(lambda)
# plot(lasso_cv_model)

# Identify the lambda value that minimizes the CV MSE
best_lambda <- lasso_cv_model$lambda.min

# Display the selected lambda and the cross-validated mean squared error (CV MSE)
cat("Selected Lambda (lambda.min):", best_lambda, "\n")
```

```
## Selected Lambda (lambda.min): 0.03487193
```

```r
cat("Cross-validated Mean Squared Error (CV MSE):", min(lasso_cv_model$cvm), "\n")
```

```
## Cross-validated Mean Squared Error (CV MSE): 6.987473
```

```r
# Fit the final Lasso model with the selected lambda on the training set
final_lasso_model <- glmnet(X_train_lasso, y_train_lasso, alpha = 1, lambda = best_lambda)

# Extract coefficients from the final model
selected_features <- coef(final_lasso_model)
```

```r
# Display the selected features
print(selected_features)
```

```
## 91 x 1 sparse Matrix of class "dgCMatrix"
##                                   s0
## (Intercept)              2.850882969
## latitude                 0.222310023
## longitude               -0.084315721
## pi                       .
## fi                       0.232840666
## fli                      0.083413068
## fri                      .
## fti                      .
## cli                      0.222389427
## cri                      .
## cti                      0.861594715
## ln_pi                    0.520770939
## ln_fi                    .
## ln_fli                   0.062329115
## ln_fri                   0.008831163
## ln_fti                   .
## ln_cli                   .
## ln_cri                  -0.044169993
## ln_cti                  -0.099261000
## tot_crossw               .
## number_of_               .
## avg_crossw               .
## tot_road_w               0.448965521
## north_veh                0.154454845
## north_ped                0.151507964
## east_veh                 .
## east_ped                 .
## south_veh                .
## south_ped                0.335497413
## west_veh                 .
## west_ped                 0.095998046
## total_lane               0.102581599
## of_exclusi               .
## commercial               0.244813469
## distdt                   .
## ln_distdt                0.290792668
## pi_squared              -0.065810620
## fi_squared               0.031908213
## distdt_squared           .
## distdt_cubed             .
## tot_crossw_squared       .
## avg_crossw_squared      -0.016647991
## tot_road_w_squared      -0.042638831
## fli_squared              .
## fri_squared             -0.033555572
## fti_squared              .
## all_pedest_1            -0.439067683
## median_1                -0.507668222
## green_stra_1             0.654338723
## half_phase_1            -0.573974580
## any_ped_pr_1             .
## ped_countd_1             0.202512935
## lt_protect_1            -0.067110918
## lt_restric_1            -0.313833909
```

```
## lt_prot_re_1                                                .
## parking_1                                                   .
## parking_2                                                  -0.516605940
## any_exclus_1                                               -0.289966513
## curb_exten_1                                               -0.768114094
## all_red_an_1                                               -0.198550489
## new_half_r_1                                                .
## missing_date_ind_1                                          1.995504119
## month_02                                                    0.244251857
## month_03                                                    0.167754327
## month_04                                                    0.181675344
## month_05                                                    0.082623694
## month_06                                                   -0.035262673
## month_07                                                    .
## month_08                                                    0.111521967
## month_09                                                    .
## month_10                                                    .
## month_11                                                    .
## month_12                                                    .
## dow_Monday                                                  .
## dow_Saturday                                                .
## dow_Sunday                                                  .
## dow_Thursday                                                0.114762220
## dow_Tuesday                                                 0.141604245
## dow_Wednesday                                               .
## borough_grouped_Other                                       .
## `borough_grouped_Côte-des-Neiges-Notre-Dame-de-Grâce`       .
## `borough_grouped_Le Plateau-Mont-Royal`                    -0.101706185
## `borough_grouped_Mercier-Hochelaga-Maisonneuve`             0.097006899
## `borough_grouped_Montréal-Nord`                             0.379762689
## `borough_grouped_Pointe-aux-Trembles-Rivières-des-Prairies` -0.394567324
## `borough_grouped_Rosemont-La Petite-Patrie`                 0.349226495
## `borough_grouped_Saint-Laurent`                            -0.777275291
## `borough_grouped_Saint-Léonard`                             0.371278860
## `borough_grouped_Sud-Ouest`                                 0.013315213
## `borough_grouped_Ville-Marie`                               .
## `borough_grouped_Villeray-Saint-Michel-Parc-Extension`      0.259275937
```

```r
# Extract the selected features from the Lasso model
selected_vars_lasso <- rownames(selected_features[selected_features[, 1] != 0, , drop = FALSE])[-1]

# Pack into a list under the key "lasso"
selected_covariates$lasso <- selected_vars_lasso
print(selected_covariates$lasso)
```

```
##  [1] "latitude"
##  [2] "longitude"
##  [3] "fi"
##  [4] "fli"
##  [5] "cli"
##  [6] "cti"
##  [7] "ln_pi"
##  [8] "ln_fli"
##  [9] "ln_fri"
## [10] "ln_cri"
## [11] "ln_cti"
## [12] "tot_road_w"
## [13] "north_veh"
## [14] "north_ped"
```

```
## [15] "south_ped"
## [16] "west_ped"
## [17] "total_lane"
## [18] "commercial"
## [19] "ln_distdt"
## [20] "pi_squared"
## [21] "fi_squared"
## [22] "avg_crossw_squared"
## [23] "tot_road_w_squared"
## [24] "fri_squared"
## [25] "all_pedest_1"
## [26] "median_1"
## [27] "green_stra_1"
## [28] "half_phase_1"
## [29] "ped_countd_1"
## [30] "lt_protect_1"
## [31] "lt_restric_1"
## [32] "parking_2"
## [33] "any_exclus_1"
## [34] "curb_exten_1"
## [35] "all_red_an_1"
## [36] "missing_date_ind_1"
## [37] "month_02"
## [38] "month_03"
## [39] "month_04"
## [40] "month_05"
## [41] "month_06"
## [42] "month_08"
## [43] "dow_Thursday"
## [44] "dow_Tuesday"
## [45] "`borough_grouped_Le Plateau-Mont-Royal`"
## [46] "`borough_grouped_Mercier-Hochelaga-Maisonneuve`"
## [47] "`borough_grouped_Montréal-Nord`"
## [48] "`borough_grouped_Pointe-aux-Trembles-Rivières-des-Prairies`"
## [49] "`borough_grouped_Rosemont-La Petite-Patrie`"
## [50] "`borough_grouped_Saint-Laurent`"
## [51] "`borough_grouped_Saint-Léonard`"
## [52] "`borough_grouped_Sud-Ouest`"
## [53] "`borough_grouped_Villeray-Saint-Michel-Parc-Extension`"
```

Random Forest Importance

```r
# Ensure that 'dat' contains only the variables in 'all_vars' plus the target variable 'acc'
predictors <- setdiff(colnames(dat), c("acc", "int_no"))

# Train the random forest model
set.seed(123)  # for reproducibility
rf_model <- randomForest(acc ~ . - int_no, data = dat_train, importance = TRUE)

# Extract variable importance
importance_rf <- importance(rf_model)

# Create a data frame of variables and their importance
variable_importance <- data.frame(Variable = rownames(importance_rf),
                                  Importance = importance_rf[, "%IncMSE"])
variable_importance <- variable_importance[order(variable_importance$Importance,
                                                 decreasing = TRUE), ]
# variable_importance$description <- sapply(variable_importance$Variable,
```

```r
#                                          function(x) f_get_description(x, varnames_dict))
rownames(variable_importance) <- NULL

# Calculate the total importance and cumulative importance
total_importance <- sum(variable_importance$Importance)
variable_importance$CumulativeImportance <- cumsum(variable_importance$Importance) / total_importance

# Print the sorted variable importance
print(variable_importance)
```

```
##              Variable Importance CumulativeImportance
## 1                 cti 13.5106944           0.04521165
## 2              ln_cti 12.8970203           0.08836973
## 3            latitude 12.2498962           0.12936229
## 4      borough_grouped 11.2617551           0.16704818
## 5           south_ped  9.4171097           0.19856122
## 6           longitude  9.3184613           0.22974415
## 7              ln_cri  9.0941486           0.26017645
## 8           north_ped  8.5630083           0.28883136
## 9                 cri  7.9218657           0.31534077
## 10         tot_road_w  7.4037614           0.34011643
## 11             ln_fti  7.2974264           0.36453624
## 12           west_ped  7.1942488           0.38861079
## 13             distdt  7.0970237           0.41235999
## 14          ln_distdt  7.0782950           0.43604651
## 15         tot_crossw  7.0052322           0.45948854
## 16                fti  6.9339819           0.48269214
## 17          north_veh  6.8674707           0.50567318
## 18                 pi  6.6562330           0.52794733
## 19              ln_pi  6.4931430           0.54967572
## 20             ln_fli  6.4130981           0.57113626
## 21                fri  5.8278973           0.59063850
## 22             ln_fri  5.7790084           0.60997715
## 23   missing_date_ind  5.3670205           0.62793713
## 24        distdt_cubed  5.3184552           0.64573460
## 25           east_ped  5.2733972           0.66338129
## 26                cli  5.1170305           0.68050471
## 27                fli  5.0225129           0.69731185
## 28                 fi  4.9608141           0.71391252
## 29             ln_cli  4.8210571           0.73004551
## 30              month  4.7814962           0.74604612
## 31              ln_fi  4.7646612           0.76199040
## 32           west_veh  4.7550097           0.77790237
## 33         pi_squared  4.5800027           0.79322871
## 34     distdt_squared  4.1704247           0.80718445
## 35         lt_restric  4.0526457           0.82074607
## 36        fti_squared  3.9833519           0.83407580
## 37          south_veh  3.8457060           0.84694491
## 38            parking  3.7443254           0.85947477
## 39         avg_crossw  3.5141911           0.87123452
## 40        fli_squared  3.4145624           0.88266088
## 41         total_lane  3.2752624           0.89362109
## 42 avg_crossw_squared  3.2440732           0.90447693
## 43         number_of_  3.2329761           0.91529563
## 44           east_veh  2.8340849           0.92477950
## 45                dow  2.5733424           0.93339083
## 46         fi_squared  2.2767078           0.94100951
## 47         green_stra  2.1506512           0.94820637
## 48         of_exclusi  2.1042224           0.95524785
```

```
## 49 tot_crossw_squared  1.9875881          0.96189904
## 50         commercial  1.9764456          0.96851294
## 51         lt_prot_re  1.9169109          0.97492761
## 52 tot_road_w_squared  1.5946819          0.98026399
## 53        fri_squared  1.4502790          0.98511715
## 54         ped_countd  1.2623197          0.98934132
## 55             median  1.2459853          0.99351084
## 56         all_pedest  1.0681529          0.99708526
## 57         all_red_an  1.0010015          1.00043497
## 58         lt_protect  0.6857642          1.00272979
## 59         curb_exten  0.3671536          1.00395842
## 60         any_ped_pr -0.1128970          1.00358062
## 61         new_half_r -0.1182913          1.00318478
## 62         any_exclus -0.1888328          1.00255288
## 63         half_phase -0.7628812          1.00000000
```

In this step, the variable selection is all variables such that the cumulative importance is around 95%. This means that adding more variables will not add much to the model.

```r
# Determine a cutoff for cumulative importance, e.g., 95%
cutoff_threshold <- 0.95

# Select variables with cumulative importance below the threshold
selected_variables <- variable_importance[variable_importance$CumulativeImportance <= cutoff_threshold, ]
selected_variables <- selected_variables$Variable

# Pack into a list under the key "rf_importance"
selected_covariates$rf_importance <- selected_variables

# Display selected variables
print(selected_covariates$rf_importance)
```

```
##  [1] "cti"               "ln_cti"            "latitude"
##  [4] "borough_grouped"   "south_ped"         "longitude"
##  [7] "ln_cri"            "north_ped"         "cri"
## [10] "tot_road_w"        "ln_fti"            "west_ped"
## [13] "distdt"            "ln_distdt"         "tot_crossw"
## [16] "fti"               "north_veh"         "pi"
## [19] "ln_pi"             "ln_fli"            "fri"
## [22] "ln_fri"            "missing_date_ind"  "distdt_cubed"
## [25] "east_ped"          "cli"               "fli"
## [28] "fi"                "ln_cli"            "month"
## [31] "ln_fi"             "west_veh"          "pi_squared"
## [34] "distdt_squared"    "lt_restric"        "fti_squared"
## [37] "south_veh"         "parking"           "avg_crossw"
## [40] "fli_squared"       "total_lane"        "avg_crossw_squared"
## [43] "number_of_"        "east_veh"          "dow"
## [46] "fi_squared"        "green_stra"
```

Correlation Importance (numerical only)

```r
# subset only the correct numerical variables
numerical_dat <- dat_dum_train[, sapply(dat_dum_train, is.numeric)]
numerical_dat <- numerical_dat[, !names(numerical_dat) %in% c('int_no')]

# Compute correlations
correlations_with_acc <- f_compute_correlations(numerical_dat, "acc", standarize = FALSE)
```

```r
correlations_with_acc <- correlations_with_acc[rownames(correlations_with_acc) != "acc", ]

# Convert row names to a column
correlations_with_acc <- correlations_with_acc %>% rownames_to_column(var = "variable")

# Identifying the top n most positively and negatively correlated variables
top_n = 50
top_positively_correlated <- correlations_with_acc %>%
                             arrange(desc(spearman)) %>%
                             head(top_n+1)

top_negatively_correlated <- correlations_with_acc %>%
                             arrange(spearman) %>%
                             head(top_n+1)


# Filter out rows with negative correlation in positive correlated, and vice versa
top_positively_correlated <- top_positively_correlated[top_positively_correlated$spearman > 0, ]
top_negatively_correlated <- top_negatively_correlated[top_negatively_correlated$spearman < 0, ]

# filter the target variable out of the correlations
top_positively_correlated <- top_positively_correlated[-1, ]
top_negatively_correlated <- top_negatively_correlated[-1, ]

# reset the index of both corr tables
rownames(top_positively_correlated) <- seq(1, nrow(top_positively_correlated))
rownames(top_negatively_correlated) <- seq(1, nrow(top_negatively_correlated))

# Add a column containing the description of the variables
top_positively_correlated$description <- sapply(top_positively_correlated$variable,
                                         function(x) f_get_description(x, varnames_dict))
top_negatively_correlated$description <- sapply(top_negatively_correlated$variable,
                                         function(x) f_get_description(x, varnames_dict))

# Convert the description column to a character vector if it's not already
top_positively_correlated$description <- as.character(top_positively_correlated$description)
top_negatively_correlated$description <- as.character(top_negatively_correlated$description)

# Presenting the tables
print(top_positively_correlated)
```

```
##                                               variable      pearson
## 1                                                  cti  0.488153657
## 2                                                  cli  0.358660332
## 3                                               ln_cli  0.323181290
## 4                                                  cri  0.388743103
## 5                                               ln_cri  0.313454518
## 6                                                   pi  0.367480615
## 7                                                ln_pi  0.378929856
## 8                                            south_ped  0.339910147
## 9                                            north_ped  0.317816943
## 10                                            east_ped  0.320061571
## 11                                            west_ped  0.303655664
## 12                                                 fri  0.134972954
## 13                                              ln_fri  0.203224534
## 14                                          tot_road_w  0.261410632
## 15                                               ln_fi  0.250707228
## 16                                                  fi  0.262391252
## 17                                           number_of_  0.199780468
```

```
## 18                                              green_stra_1  0.298071004
## 19                                                       fti  0.252634215
## 20                                                    ln_fti  0.198821243
## 21                                                tot_crossw  0.220005342
## 22                                                       fli  0.115678495
## 23                                                    ln_fli  0.178675815
## 24                                                 south_veh  0.202166219
## 25                                                 north_veh  0.213872401
## 26                                                total_lane  0.155441546
## 27                                                avg_crossw  0.119439276
## 28 borough_grouped_Villeray-Saint-Michel-Parc-Extension  0.063228498
## 29                                                 parking_1  0.071231095
## 30                                                commercial  0.075968483
## 31                                              any_ped_pr_1  0.123939202
## 32                              borough_grouped_Ville-Marie  0.101807322
## 33                borough_grouped_Rosemont-La Petite-Patrie  0.042182223
## 34                                         missing_date_ind_1  0.085225780
## 35                                                  west_veh  0.075732853
## 36                            borough_grouped_Montréal-Nord  0.047571250
## 37                                              ped_countd_1  0.108234574
## 38                                                  month_04  0.051464434
## 39                                                  east_veh  0.053894025
## 40                 borough_grouped_Le Plateau-Mont-Royal  0.013705280
## 41                                                  month_02  0.040416570
## 42  borough_grouped_Côte-des-Neiges-Notre-Dame-de-Grâce  0.005432838
## 43                                                dow_Sunday  0.038690034
## 44                                              dow_Saturday  0.006091342
## 45                                                  month_09 -0.017296826
## 46                                                fi_squared  0.106973044
## 47                                                dow_Monday  0.008604849
## 48                                                  month_03  0.039178228
## 49                                               dow_Tuesday  0.011837395
## 50                                                  latitude  0.056476689
##       spearman     kendall
## 1  0.536459887 0.409681738
## 2  0.519605254 0.396404353
## 3  0.519605254 0.396404353
## 4  0.505896687 0.385887820
## 5  0.505896687 0.385887820
## 6  0.472154604 0.355633827
## 7  0.472154136 0.355632977
## 8  0.442473309 0.330922065
## 9  0.440280476 0.330617774
## 10 0.424026257 0.318207330
## 11 0.407804472 0.306433147
## 12 0.286326272 0.216147198
## 13 0.286326272 0.216147198
## 14 0.285345767 0.212520448
## 15 0.261066228 0.192029845
## 16 0.261064043 0.192028750
## 17 0.251057105 0.219773995
## 18 0.248990750 0.221369780
## 19 0.238754529 0.175771623
## 20 0.238754529 0.175771623
## 21 0.236307828 0.175000522
## 22 0.233541437 0.174166747
## 23 0.233541437 0.174166747
## 24 0.226947267 0.169108536
## 25 0.224376197 0.166182052
## 26 0.127978400 0.101639605
```

```
## 27 0.105267748 0.078338988
## 28 0.094821414 0.084302712
## 29 0.094726797 0.084218591
## 30 0.084158190 0.070027505
## 31 0.083628305 0.074351274
## 32 0.074337967 0.066091529
## 33 0.071166956 0.063272284
## 34 0.057950231 0.051521713
## 35 0.055756482 0.041610682
## 36 0.054347228 0.048318397
## 37 0.053184602 0.047284743
## 38 0.049028154 0.043589377
## 39 0.047917020 0.036274432
## 40 0.045803461 0.040722405
## 41 0.038058584 0.033836680
## 42 0.035377364 0.031452892
## 43 0.027012816 0.024016238
## 44 0.022857030 0.020321461
## 45 0.022242170 0.019774808
## 46 0.020715260 0.014352706
## 47 0.019458723 0.017300134
## 48 0.018855234 0.016763590
## 49 0.008148930 0.007244955
## 50 0.001826449 0.002143522
##
## 1              number of pedestrian-vehicle prohibited interactions over each 15 min intervals during the
## 2   number of pedestrian-vehicle left turning potential interactions over each 15 min intervals during the
## 3
## 4  number of pedestrian-vehicle right turning potential interactions over each 15 min intervals during the
## 5
## 6                                                                        average annual daily flow for pedest
## 7                                                                                                 log of p
## 8                                                     average annual daily flow for pedestrians heading sou
## 9                                                       average annual daily flow for pedestrians heading n
## 10                                                      average annual daily flow for pedestrians heading
## 11                                                      average annual daily flow for pedestrians heading
## 12                                                       average annual daily flow for vehicules turning r
## 13
## 14                                              sum of the road widths (outside crosswalks) along each appr
## 15
## 16                                                                       average annual daily flow for vehic
## 17
## 18
## 19                                             average annual daily flow for vehicules going through (strai
## 20
## 21                                                                  sum of the crosswalk widths along each appr
## 22                                                       average annual daily flow for vehicules turning l
## 23
## 24                                                       average annual daily flow for vehicules heading s
## 25                                                       average annual daily flow for vehicules heading n
## 26                                                     sum of the number of lanes flowing into the intersec
## 27                                                                         average crosswalk width per  appr
## 28
## 29
## 30                                                                                 number of entrances/exits
## 31
## 32
## 33
## 34
## 35                                                        average annual daily flow for vehicules heading
```

```
## 36
## 37
## 38
## 39                                                          average annual daily flow for vehicules heading
## 40
## 41
## 42
## 43
## 44
## 45
## 46
## 47
## 48
## 49
## 50
```

```r
print(top_negatively_correlated)
```

```
##                                                      variable      pearson
## 1                                                 fli_squared  0.054485006
## 2                                                      distdt -0.123943267
## 3                                                   ln_distdt -0.135995273
## 4                                                distdt_cubed -0.075749138
## 5                                  borough_grouped_Saint-Laurent -0.100856864
## 6                                                all_pedest_1 -0.105984912
## 7                                                  pi_squared  0.158607844
## 8                                                lt_restric_1 -0.090684576
## 9                                                lt_protect_1 -0.089872677
## 10                                               curb_exten_1 -0.087692181
## 11                                               lt_prot_re_1 -0.074014080
## 12                                               all_red_an_1 -0.051123241
## 13                                           tot_road_w_squared  0.010829663
## 14                                           tot_crossw_squared  0.012331776
## 15                                        borough_grouped_Other -0.068504591
## 16 borough_grouped_Pointe-aux-Trembles-Rivières-des-Prairies -0.062605071
## 17                                               half_phase_1 -0.081560388
## 18             borough_grouped_Mercier-Hochelaga-Maisonneuve -0.024159306
## 19                                     borough_grouped_Sud-Ouest -0.064527884
## 20                                               any_exclus_1 -0.031734250
## 21                                               new_half_r_1 -0.072641294
## 22                                                 of_exclusi  0.018527610
## 23                                           avg_crossw_squared -0.009907987
## 24                                                   month_06 -0.054580645
## 25                                               distdt_squared -0.048155308
## 26                                                   month_05 -0.011296171
## 27                                               dow_Thursday  0.019176810
## 28                                                  longitude -0.012474775
## 29                                                   median_1  0.022522781
## 30                                                   month_10  0.009027455
## 31                                                fti_squared  0.126752034
## 32                                                   month_12 -0.010921601
## 33                                                   month_08 -0.017348916
## 34                                              dow_Wednesday -0.034422155
## 35                                                   month_11 -0.027054337
## 36                                                   month_07 -0.002170631
## 37                                 borough_grouped_Saint-Léonard  0.032353939
## 38                                                  parking_2 -0.058574261
##        spearman      kendall                                   description
## 1  -0.177556366 -0.131618515                                          <NA>
```

```
## 2  -0.156611216 -0.114806406            distance from downtown
## 3  -0.156611216 -0.114806406                       log of disdt
## 4  -0.156611216 -0.114806406                               <NA>
## 5  -0.138811935 -0.123413290                               <NA>
## 6  -0.134384917 -0.119477368                               <NA>
## 7  -0.111040338 -0.090217251                               <NA>
## 8  -0.105435082 -0.093738988                               <NA>
## 9  -0.097749318 -0.086905819                               <NA>
## 10 -0.095549836 -0.084950330                               <NA>
## 11 -0.094987623 -0.084450483                               <NA>
## 12 -0.078914053 -0.070159982                               <NA>
## 13 -0.074966984 -0.055917820                               <NA>
## 14 -0.074284208 -0.055124422                               <NA>
## 15 -0.073812733 -0.065624561                               <NA>
## 16 -0.066860554 -0.059443599                               <NA>
## 17 -0.064462008 -0.057311127                               <NA>
## 18 -0.059581356 -0.052971894                               <NA>
## 19 -0.058753040 -0.052235464                               <NA>
## 20 -0.057290781 -0.050935417                               <NA>
## 21 -0.050037392 -0.044486658                               <NA>
## 22 -0.043931194 -0.038026808 number of exclusive left turn lane
## 23 -0.041965953 -0.031229287                               <NA>
## 24 -0.041169022 -0.036602072                               <NA>
## 25 -0.032368436 -0.023749256                               <NA>
## 26 -0.028182486 -0.025056155                               <NA>
## 27 -0.023329978 -0.020741943                               <NA>
## 28 -0.017402458 -0.013170136                               <NA>
## 29 -0.017248788 -0.015335351                               <NA>
## 30 -0.016075323 -0.014292060                               <NA>
## 31 -0.014578776 -0.011645043                               <NA>
## 32 -0.014424855 -0.012824681                               <NA>
## 33 -0.012577138 -0.011181935                               <NA>
## 34 -0.009864582 -0.008770287                               <NA>
## 35 -0.009338414 -0.008302488                               <NA>
## 36 -0.006373101 -0.005666122                               <NA>
## 37 -0.004130372 -0.003672183                               <NA>
## 38 -0.001455316 -0.001293875                               <NA>
```

Finally,we perform variable selection with respect to the top correlated (in absolute value) variables which have at least 30% Spearman correlation with the target variable `acc`.

```r
# Combine the positively and negatively correlated variables into one data frame
combined_correlations <- rbind(top_positively_correlated, top_negatively_correlated)

# Filter for variables with an absolute Spearman correlation of at least 30%
significant_correlations <- combined_correlations[abs(combined_correlations$spearman) >= 0.15, ]

# Extract the variable names into the selected_covariates list
selected_covariates$spearman <- significant_correlations$variable

# Print the selected covariates
print(selected_covariates)
```

```
## $stepwise_aic
##  [1] "longitude"                     "pi"
##  [3] "fi"                            "fli"
##  [5] "cli"                           "cri"
##  [7] "cti"                           "ln_pi"
##  [9] "ln_fri"                        "ln_cri"
```

```
## [11] "ln_cti"                        "number_of_"
## [13] "tot_road_w"                    "median"
## [15] "green_stra"                    "half_phase"
## [17] "any_ped_pr"                    "ped_countd"
## [19] "lt_protect"                    "lt_restric"
## [21] "lt_prot_re"                    "parking"
## [23] "north_veh"                     "north_ped"
## [25] "east_veh"                      "south_veh"
## [27] "south_ped"                     "west_veh"
## [29] "any_exclus"                    "commercial"
## [31] "curb_exten"                    "distdt"
## [33] "ln_distdt"                     "missing_date_ind"
## [35] "month"                         "dow"
## [37] "borough_grouped"               "pi_squared"
## [39] "distdt_squared"                "distdt_cubed"
## [41] "tot_crossw_squared"            "avg_crossw_squared"
## [43] "tot_road_w_squared"            "fli_squared"
## [45] "fri_squared"                   "fti_squared"
## [47] "cli:month"                     "cri:month"
## [49] "cti:month"                     "cli:dow"
## [51] "cri:dow"                       "cti:dow"
## [53] "fi:lt_restric"                 "pi:lt_prot_re"
## [55] "fi:ped_countd"                 "south_veh:south_ped"
## [57] "any_ped_pr:ln_distdt"          "fi:month"
## [59] "pi:dow"                        "lt_protect:pi_squared"
## [61] "green_stra:distdt_squared"     "half_phase:distdt_cubed"
## [63] "lt_restric:tot_crossw_squared" "ped_countd:avg_crossw_squared"
## [65] "east_veh:fli_squared"          "west_veh:fri_squared"
## [67] "north_veh:fti_squared"
##
## $stepwise_bic
##  [1] "pi"                  "cli"                 "cti"
##  [4] "ln_pi"               "ln_cti"              "tot_road_w"
##  [7] "median"              "green_stra"          "parking"
## [10] "north_veh"           "north_ped"           "south_veh"
## [13] "south_ped"           "west_veh"            "commercial"
## [16] "ln_distdt"           "missing_date_ind"    "dow"
## [19] "pi_squared"          "tot_road_w_squared"  "fri_squared"
## [22] "north_veh:north_ped" "south_veh:south_ped" "pi:dow"
## [25] "west_veh:fri_squared"
##
## $lasso
##  [1] "latitude"
##  [2] "longitude"
##  [3] "fi"
##  [4] "fli"
##  [5] "cli"
##  [6] "cti"
##  [7] "ln_pi"
##  [8] "ln_fli"
##  [9] "ln_fri"
## [10] "ln_cri"
## [11] "ln_cti"
## [12] "tot_road_w"
## [13] "north_veh"
## [14] "north_ped"
## [15] "south_ped"
## [16] "west_ped"
## [17] "total_lane"
## [18] "commercial"
```

```
## [19] "ln_distdt"
## [20] "pi_squared"
## [21] "fi_squared"
## [22] "avg_crossw_squared"
## [23] "tot_road_w_squared"
## [24] "fri_squared"
## [25] "all_pedest_1"
## [26] "median_1"
## [27] "green_stra_1"
## [28] "half_phase_1"
## [29] "ped_countd_1"
## [30] "lt_protect_1"
## [31] "lt_restric_1"
## [32] "parking_2"
## [33] "any_exclus_1"
## [34] "curb_exten_1"
## [35] "all_red_an_1"
## [36] "missing_date_ind_1"
## [37] "month_02"
## [38] "month_03"
## [39] "month_04"
## [40] "month_05"
## [41] "month_06"
## [42] "month_08"
## [43] "dow_Thursday"
## [44] "dow_Tuesday"
## [45] "`borough_grouped_Le Plateau-Mont-Royal`"
## [46] "`borough_grouped_Mercier-Hochelaga-Maisonneuve`"
## [47] "`borough_grouped_Montréal-Nord`"
## [48] "`borough_grouped_Pointe-aux-Trembles-Rivières-des-Prairies`"
## [49] "`borough_grouped_Rosemont-La Petite-Patrie`"
## [50] "`borough_grouped_Saint-Laurent`"
## [51] "`borough_grouped_Saint-Léonard`"
## [52] "`borough_grouped_Sud-Ouest`"
## [53] "`borough_grouped_Villeray-Saint-Michel-Parc-Extension`"
##
## $rf_importance
##  [1] "cti"                "ln_cti"             "latitude"
##  [4] "borough_grouped"    "south_ped"          "longitude"
##  [7] "ln_cri"             "north_ped"          "cri"
## [10] "tot_road_w"         "ln_fti"             "west_ped"
## [13] "distdt"             "ln_distdt"          "tot_crossw"
## [16] "fti"                "north_veh"          "pi"
## [19] "ln_pi"              "ln_fli"             "fri"
## [22] "ln_fri"             "missing_date_ind"   "distdt_cubed"
## [25] "east_ped"           "cli"                "fli"
## [28] "fi"                 "ln_cli"             "month"
## [31] "ln_fi"              "west_veh"           "pi_squared"
## [34] "distdt_squared"     "lt_restric"         "fti_squared"
## [37] "south_veh"          "parking"            "avg_crossw"
## [40] "fli_squared"        "total_lane"         "avg_crossw_squared"
## [43] "number_of_"         "east_veh"           "dow"
## [46] "fi_squared"         "green_stra"
##
## $spearman
##  [1] "cti"          "cli"          "ln_cli"       "cri"          "ln_cri"
##  [6] "pi"           "ln_pi"        "south_ped"    "north_ped"    "east_ped"
## [11] "west_ped"     "fri"          "ln_fri"       "tot_road_w"   "ln_fi"
## [16] "fi"           "number_of_"   "green_stra_1" "fti"          "ln_fti"
## [21] "tot_crossw"   "fli"          "ln_fli"       "south_veh"    "north_veh"
```

```
## [26] "fli_squared"  "distdt"       "ln_distdt"    "distdt_cubed"
```

Model Selection

Based on the selected variables, we would like to select the "best" model by fitting a model again to the training data

1. **Benchmark:** A model that predicts the mean of the target variable.
2. **Basic Linear Regression:** A simple linear regression model with no variable selection.
3. **Stepwise OLS:** A linear regression model with variable selection using stepwise regression.
4. **Stepwise variables:** Linear/Ridge Regression
5. **Lasso variables:** Linear/Lasso Regression
6. **Random Forest Importance variables:** Random Forest Model
7. **Spearman Correlation variables:** Linear/Ridge Regression
8. **No selection:** This will be used for pure random forest on top.

We will then compare the performance of the models on the validation set.

```r
# Create a dataframe to store performances
model_performance <- data.frame(
  Model_Name = character(),
  MSE = numeric()
)
```

Benchmark

```r
# Predicting the mean of acc which is very close to zero
mse = mean((mean(dat$acc) - dat_val$acc)^2)
model_performance <- rbind(model_performance, list(Model_Name = "Baseline", MSE = mse))
model_performance
```

```
##   Model_Name      MSE
## 1   Baseline 9.523514
```

Basic Linear Regression with no variable selection

```r
# Fit linear regression model using all predictors
lm_model <- lm(acc ~ ., data = dat_train)

# Make predictions on the validation set
predictions <- predict(lm_model, newdata = dat_val)

# Calculate mean squared error
mse <- mean((predictions - dat_val$acc)^2)

# Print the mean squared error
print(paste("Mean Squared Error (MSE) on validation set:", mse))
```

```
## [1] "Mean Squared Error (MSE) on validation set: 5.95087039563261"
```

```
# Store the model performance in dataset
model_performance <- rbind(model_performance, list(Model_Name = "OLS", MSE = mse))
```

```
model_performance
```

```
##   Model_Name      MSE
## 1   Baseline 9.523514
## 2        OLS 5.950870
```

Stepwise AIC OLS

```
# Test performance on validation set
predictions <- predict(stepwise_aic, newdata = dat_val)
```

```
## Warning in predict.lm(stepwise_aic, newdata = dat_val): prediction from
## rank-deficient fit; attr(*, "non-estim") has doubtful cases
```

```
actual <- dat_val$acc
mse <- mean((predictions - actual)^2)

print(paste("Mean Squared Error (MSE):", mse))
```

```
## [1] "Mean Squared Error (MSE): 7.81679576362225"
```

```
# Store the performance in dataset
model_performance <- rbind(model_performance, list(Model_Name = "Stepwise-AIC + OLS", MSE = mse))
```

```
model_performance
```

```
##           Model_Name      MSE
## 1           Baseline 9.523514
## 2                OLS 5.950870
## 3 Stepwise-AIC + OLS 7.816796
```

Stepwise AIC OLS

```
# Test performance on validation set
predictions <- predict(stepwise_bic, newdata = dat_val)
```

```
## Warning in predict.lm(stepwise_bic, newdata = dat_val): prediction from
## rank-deficient fit; attr(*, "non-estim") has doubtful cases
```

```
actual <- dat_val$acc
mse <- mean((predictions - actual)^2)

print(paste("Mean Squared Error (MSE):", mse))
```

```
## [1] "Mean Squared Error (MSE): 6.89351087239669"
```

```r
# Store the performance in dataset
model_performance <- rbind(model_performance, list(Model_Name = "Stepwise-BIC + OLS", MSE = mse))
```

```r
model_performance
```

```
##           Model_Name      MSE
## 1           Baseline 9.523514
## 2                OLS 5.950870
## 3 Stepwise-AIC + OLS 7.816796
## 4 Stepwise-BIC + OLS 6.893511
```

Stepwise-AIC and Lasso

```r
# Create formula with interactions
interaction_formula <- as.formula(paste("acc ~", paste(selected_covariates$stepwise_aic, collapse = " + ")))

# Generate model matrix for train and val
model_matrix_train <- model.matrix(interaction_formula, data = dat_train)
model_matrix_val <- model.matrix(interaction_formula, data = dat_val)

# Extract response variable
y <- dat_train$acc

# Fit Lasso model
lasso_model <- glmnet(model_matrix_train, y, alpha = 1)

# Make predictions on the validation set
predictions <- predict(lasso_model, newx =model_matrix_val)

# Calculate mean squared error
mse <- mean((predictions - dat_val$acc)^2)

# Print the mean squared error
print(paste("Mean Squared Error (MSE):", mse))
```

```
## [1] "Mean Squared Error (MSE): 6.87307565510956"
```

```r
# Store the performance in dataset
model_performance <- rbind(model_performance, list(Model_Name = "Stepwise AIC + Lasso", MSE = mse))
```

Stepwise-BIC and Lasso

```r
# Create formula with interactions
interaction_formula <- as.formula(paste("acc ~", paste(selected_covariates$stepwise_bic, collapse = " + ")))

# Generate model matrix for train and val
model_matrix_train <- model.matrix(interaction_formula, data = dat_train)
model_matrix_val <- model.matrix(interaction_formula, data = dat_val)

# Extract response variable
y <- dat_train$acc

# Fit Lasso model
lasso_model <- glmnet(model_matrix_train, y, alpha = 1)
```

```r
# Make predictions on the validation set
predictions <- predict(lasso_model, newx =model_matrix_val)

# Calculate mean squared error
mse <- mean((predictions - dat_val$acc)^2)

# Print the mean squared error
print(paste("Mean Squared Error (MSE):", mse))
```

```
## [1] "Mean Squared Error (MSE): 6.74301708910752"
```

```r
# Store the performance in dataset
model_performance <- rbind(model_performance, list(Model_Name = "Stepwise BIC + Lasso", MSE = mse))
```

```r
model_performance
```

```
##             Model_Name      MSE
## 1             Baseline 9.523514
## 2                  OLS 5.950870
## 3   Stepwise-AIC + OLS 7.816796
## 4   Stepwise-BIC + OLS 6.893511
## 5 Stepwise AIC + Lasso 6.873076
## 6 Stepwise BIC + Lasso 6.743017
```

Lasso + OLS

```r
# Assuming dat_dum_train and dat_val are your training and validation datasets respectively
# and selected_covariates$lasso contains the names of the variables selected by Lasso

# Create a formula for the linear model using the variables selected by Lasso
selected_vars_formula <- paste("acc ~", paste(selected_covariates$lasso, collapse = " + "))

# Fit the linear model on the training data using only the selected variables
refit_lasso_lm_model <- lm(as.formula(selected_vars_formula), data = dat_dum_train)

# Predictions on the validation set
predictions_lm <- predict(refit_lasso_lm_model, newdata = dat_dum_val)

# Calculate Mean Squared Error (MSE) on the validation set
mse_lm <- mean((predictions_lm - dat_val$acc)^2)

# Print the MSE of the refitted linear model
print(paste("Mean Squared Error (MSE) with Refitted LM:", mse_lm))
```

```
## [1] "Mean Squared Error (MSE) with Refitted LM: 4.98808912371592"
```

```r
# Store the performance in dataset for the refitted model
model_performance <- rbind(model_performance, list(Model_Name = "Lasso + OLS", MSE = mse_lm))
```

```r
# Print the model performance
print(model_performance)
```

```
##             Model_Name      MSE
## 1             Baseline 9.523514
```

```
## 2                  OLS 5.950870
## 3   Stepwise-AIC + OLS 7.816796
## 4   Stepwise-BIC + OLS 6.893511
## 5 Stepwise AIC + Lasso 6.873076
## 6 Stepwise BIC + Lasso 6.743017
## 7          Lasso + OLS 4.988089
```

Lasso + Lasso

```r
# Create formula with Lasso-selected variables
selected_formula <- as.formula(paste("acc ~", paste(selected_covariates$lasso, collapse = " + ")))

# Generate model matrix for train and validation datasets based on the Lasso-selected variables
X_train_selected <- model.matrix(selected_formula, data = dat_dum_train)[, -1]  # Remove intercept column
X_val_selected <- model.matrix(selected_formula, data = dat_dum_val)[, -1]  # Remove intercept column

# Extract response variable for training dataset
y_train_selected <- dat_dum_train$acc

# Fit Lasso model to the Lasso-selected variables
final_lasso_model_selected <- cv.glmnet(X_train_selected, y_train_selected, alpha = 1)

# Identify the lambda value that minimizes the CV MSE for the new Lasso model
best_lambda_selected <- final_lasso_model_selected$lambda.min

# Also get the 1-SE rule lambda
best_lambda_selected_1se <- final_lasso_model_selected$lambda.1se

# Make predictions on the validation set using the newly fitted Lasso model
predictions_selected <- predict(final_lasso_model_selected, newx = X_val_selected, s = best_lambda_selected)

# Also make predictions with the 1-SE rule lambda
predictions_selected_1se <- predict(final_lasso_model_selected, newx = X_val_selected, s = best_lambda_selecte

# Calculate mean squared error for the validation set for both predictions
mse_selected <- mean((predictions_selected - dat_dum_val$acc)^2)
mse_selected_1se <- mean((predictions_selected_1se - dat_dum_val$acc)^2)

# Print the mean squared error for the new Lasso model
print(paste("Mean Squared Error (MSE) with Lasso-selected variables:", mse_selected))
```

```
## [1] "Mean Squared Error (MSE) with Lasso-selected variables: 5.64794611289063"
```

```r
print(paste("Mean Squared Error (MSE) with Lasso-selected variables (1-SE rule):", mse_selected_1se))
```

```
## [1] "Mean Squared Error (MSE) with Lasso-selected variables (1-SE rule): 6.09498470911413"
```

```r
# Store the performance of the new Lasso model in the dataset for both mse
model_performance <- rbind(model_performance, list(Model_Name = "Lasso + Lasso", MSE = mse_selected))
model_performance <- rbind(model_performance, list(Model_Name = "Lasso + Lasso (1-SE rule)", MSE = mse_selecte
```

```r
# Print the updated model performance
print(model_performance)
```

```
##                Model_Name      MSE
## 1                Baseline 9.523514
```

```
## 2                         OLS 5.950870
## 3          Stepwise-AIC + OLS 7.816796
## 4          Stepwise-BIC + OLS 6.893511
## 5        Stepwise AIC + Lasso 6.873076
## 6        Stepwise BIC + Lasso 6.743017
## 7                 Lasso + OLS 4.988089
## 8               Lasso + Lasso 5.647946
## 9 Lasso + Lasso (1-SE rule) 6.094985
```

RF Features + Ridge

- RF Features + Ridge Model

```r
# Convert training data to matrix with selected covariates from random forest importance
X_train_rf <- as.matrix(dat_train[, selected_covariates$rf_importance])
y_train_rf <- dat_train$acc

# Fit Ridge regression model with cross-validation to select lambda
ridge_model <- cv.glmnet(X_train_rf, y_train_rf, alpha = 0)

# Print the selected lambda value
best_lambda <- ridge_model$lambda.min
cat("Selected lambda:", best_lambda, "\n")
```

```
## Selected lambda: 1.104624
```

```r
# Fit the final Ridge regression model using the selected lambda
final_ridge_model <- glmnet(X_train_rf, y_train_rf, alpha = 0, lambda = best_lambda)

# Convert validation data to matrix with selected covariates from random forest importance
X_val <- as.matrix(dat_val[, selected_covariates$rf_importance])

# Make predictions using the fitted Ridge model
predictions <- predict(final_ridge_model, newx = X_val)

# Calculate mean squared error
mse <- mean((predictions - dat_val$acc)^2)

# Print the mean squared error
print(paste("Mean Squared Error (MSE):", mse))
```

```
## [1] "Mean Squared Error (MSE): 6.00443793806498"
```

```r
# Store the performance in dataset
model_performance <- rbind(model_performance, list(Model_Name = "RF + Ridge", MSE = mse))
```

```r
model_performance
```

```
##                      Model_Name      MSE
## 1                      Baseline 9.523514
## 2                           OLS 5.950870
## 3            Stepwise-AIC + OLS 7.816796
## 4            Stepwise-BIC + OLS 6.893511
## 5          Stepwise AIC + Lasso 6.873076
## 6          Stepwise BIC + Lasso 6.743017
## 7                   Lasso + OLS 4.988089
## 8                 Lasso + Lasso 5.647946
## 9   Lasso + Lasso (1-SE rule) 6.094985
## 10                   RF + Ridge 6.004438
```

Basic Random Forest

- No feature selection
- NO hyperparm tuning
- Just basics RF

```
# Fit Random Forest model using all predictors
rf_model <- randomForest(acc ~ ., data = dat_train)

# Make predictions on the validation set
predictions <- predict(rf_model, newdata = dat_val)

# Calculate mean squared error
mse <- mean((predictions - dat_val$acc)^2)

# Print the mean squared error
print(paste("Mean Squared Error (MSE):", mse))
```

```
## [1] "Mean Squared Error (MSE): 5.20817610011381"
```

```
# Store the performance in dataset
model_performance <- rbind(model_performance, list(Model_Name = "RandomForest", MSE = mse))
```

```
model_performance
```

```
##                    Model_Name      MSE
## 1                    Baseline 9.523514
## 2                         OLS 5.950870
## 3          Stepwise-AIC + OLS 7.816796
## 4          Stepwise-BIC + OLS 6.893511
## 5        Stepwise AIC + Lasso 6.873076
## 6        Stepwise BIC + Lasso 6.743017
## 7                 Lasso + OLS 4.988089
## 8               Lasso + Lasso 5.647946
## 9  Lasso + Lasso (1-SE rule) 6.094985
## 10                 RF + Ridge 6.004438
## 11               RandomForest 5.208176
```

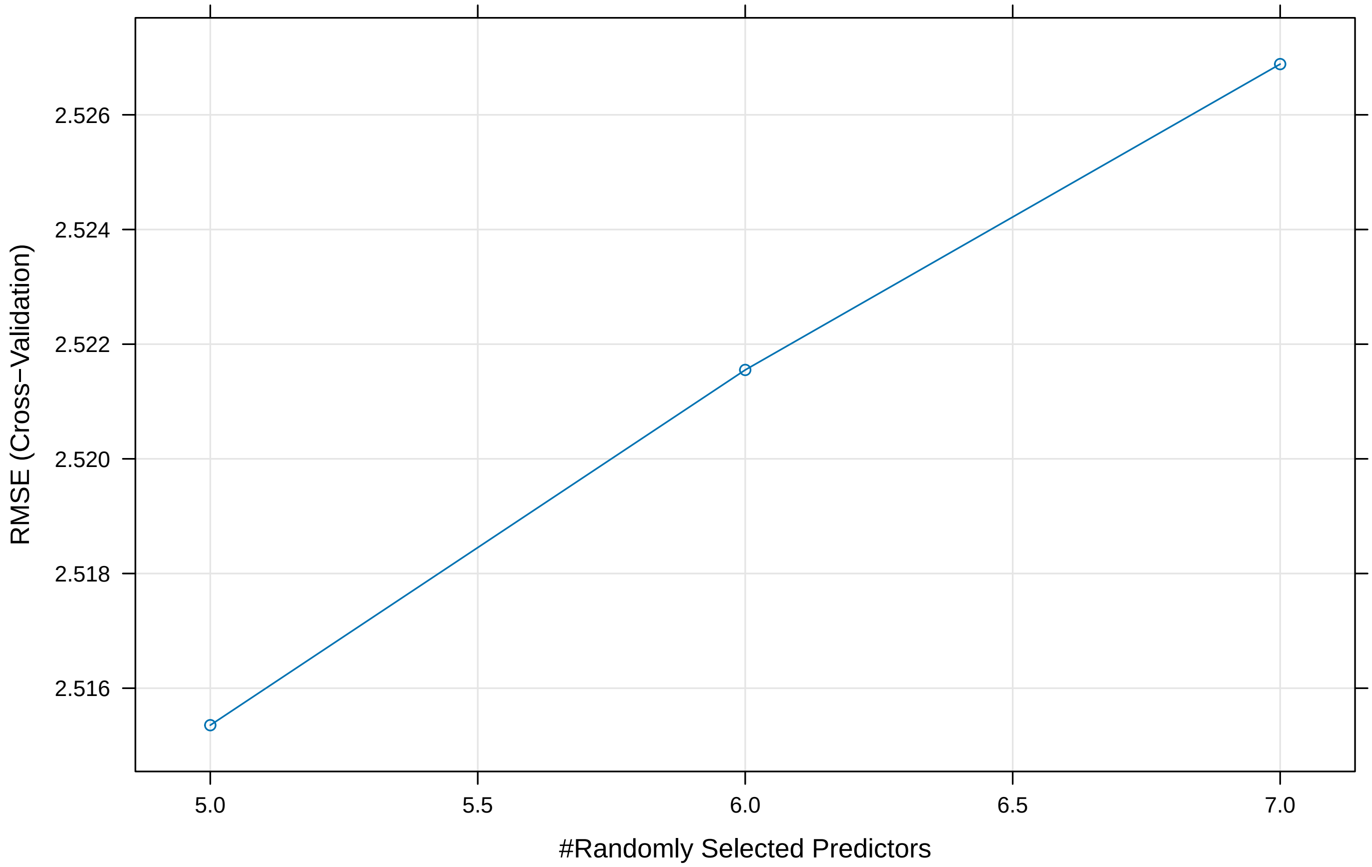
Tuned RF

- Hyper tuned RF with basic features

```
# Define train control
ctrl <- trainControl(method = "cv", number = 5, verbose = TRUE)

# Define the grid for tuning mtry and nodesize parameters
grid <- expand.grid(mtry = 5:7)


# Perform grid search to tune both mtry and nodesize
rf_model_tuned <- train(acc ~ .,
                        data = dat_train[, !(names(dat_train) %in% c("int_no"))],
                        method = "rf",
                        trControl = ctrl,
                        tuneGrid = grid,
                        ntree = 500)
```

```
## + Fold1: mtry=5
## - Fold1: mtry=5
## + Fold1: mtry=6
## - Fold1: mtry=6
## + Fold1: mtry=7
## - Fold1: mtry=7
## + Fold2: mtry=5
## - Fold2: mtry=5
## + Fold2: mtry=6
## - Fold2: mtry=6
## + Fold2: mtry=7
## - Fold2: mtry=7
## + Fold3: mtry=5
## - Fold3: mtry=5
## + Fold3: mtry=6
## - Fold3: mtry=6
## + Fold3: mtry=7
## - Fold3: mtry=7
## + Fold4: mtry=5
## - Fold4: mtry=5
## + Fold4: mtry=6
## - Fold4: mtry=6
## + Fold4: mtry=7
## - Fold4: mtry=7
## + Fold5: mtry=5
## - Fold5: mtry=5
## + Fold5: mtry=6
## - Fold5: mtry=6
## + Fold5: mtry=7
## - Fold5: mtry=7
## Aggregating results
## Selecting tuning parameters
## Fitting mtry = 5 on full training set
```

```r
# Plot the tuning results for both mtry and nodesize
plot(rf_model_tuned)
```




```r
# Make predictions on the validation set using the tuned model
preds_rf_tuned <- predict(rf_model_tuned, newdata = dat_val[, !(names(dat_val) %in% c("int_no"))])

# Calculate mean squared error with tuned parameters
mse_rf_tuned <- mean((preds_rf_tuned - dat_val$acc)^2)
print(mse_rf_tuned)
```

```
## [1] 5.28402
```

```r
# Store the performance in dataset
model_performance <- rbind(model_performance, list(Model_Name = "Tuned RF", MSE = mse_rf_tuned))
```

```r
model_performance
```

```
##                   Model_Name      MSE
## 1                   Baseline 9.523514
## 2                        OLS 5.950870
## 3          Stepwise-AIC + OLS 7.816796
## 4          Stepwise-BIC + OLS 6.893511
## 5        Stepwise AIC + Lasso 6.873076
## 6        Stepwise BIC + Lasso 6.743017
## 7                Lasso + OLS 4.988089
## 8              Lasso + Lasso 5.647946
## 9  Lasso + Lasso (1-SE rule) 6.094985
## 10                RF + Ridge 6.004438
## 11              RandomForest 5.208176
## 12                  Tuned RF 5.284020
```

Spearman Based Variables

```
# Fit Ordinary Least Squares (OLS) model using selected covariates based on Spearman correlation
lm_model <- lm(dat_dum_train$acc ~ ., data = dat_dum_train[, selected_covariates$spearman])

# Make predictions on the validation set
predictions <- predict(lm_model, newdata = dat_dum_val)

# Calculate mean squared error
mse <- mean((predictions - dat_val$acc)^2)

# Print the mean squared error
print(paste("Mean Squared Error (MSE) on validation set:", mse))
```

```
## [1] "Mean Squared Error (MSE) on validation set: 5.5948231626233"
```

```
# Store the performance in dataset
model_performance <- rbind(model_performance, list(Model_Name = "OLS+Corr", MSE = mse))
```

XGBoost

Note: The following was the code used to tune the XGBoost model. However, it was not run in this notebook due to the time it takes to run.

```
# param to run the code
run_xgb_tuning = FALSE

# Tuning XGBoost model
if(run_xgb_tuning){
  # Define train control
  ctrl <- trainControl(method = "cv", number = 5, allowParallel = TRUE, verbose = TRUE)

  # Define the grid for tuning parameters
  grid <- expand.grid(nrounds = c(100, 200, 300),
                      max_depth = c(3, 6, 9),
                      eta = c(0.01, 0.1, 0.3),
                      gamma = c(0, 1, 5),
                      colsample_bytree = c(0.5, 0.7, 1),
                      min_child_weight = c(3, 5),
                      subsample = c(0.5, 0.7))

  # Perform grid search to tune parameters
  xgb_model_tuned <- train(acc ~ .,
                           data = dat_train[, !(names(dat_train) %in% c("acc", "int_no"))],
                           method = "xgbTree",
                           trControl = ctrl,
                           tuneGrid = grid)

  # Make predictions on the validation set using the tuned model
  preds_xgb_tuned <- predict(xgb_model_tuned, newdata = dat_val[, !(names(dat_val) %in% c("acc", "int_no"))])

  # Calculate mean squared error with tuned parameters
  mse_xgb_tuned <- mean((preds_xgb_tuned - dat_val$acc)^2)
  print(mse_xgb_tuned)

  # Store the performance in dataset
  model_performance <- rbind(model_performance, list(Model_Name = "Tuned XGBoost", MSE = mse_xgb_tuned))
}
```

FIt the XGBoost model with optimal hypertuned parameters:

```r
library(xgboost)

# Define the train and val matrices without acc or int_no
train_matrix <- as.matrix(dat_dum_train[, !(names(dat_dum_train) %in% c("acc", "int_no"))])
val_matrix <- as.matrix(dat_dum_val[, !(names(dat_dum_val) %in% c("acc", "int_no"))])

# Define parameters for the XGBoost model
xgb_params <- list(max_depth = 6,
                   eta = 0.01,
                   gamma = 5,
                   colsample_bytree = 1,
                   min_child_weight = 5,
                   subsample = 0.5)

# Train XGBoost model with tuned parameters
xgb_model_tuned <- xgboost(data = train_matrix,
                           label = dat_dum_train$acc,
                           params = xgb_params,
                           nrounds = 300,  # Specify nrounds directly
                           nthread = 1,    # Use only one core
                           verbose = 0)    # Suppress XGBoost warnings

# Make predictions on the validation set
preds_xgb_tuned <- predict(xgb_model_tuned, newdata = val_matrix)

# Calculate mean squared error
mse_xgb_tuned <- mean((preds_xgb_tuned - dat_dum_val$acc)^2)

# Print MSE
print(paste("Mean Squared Error (MSE) for Tuned XGBoost:", mse_xgb_tuned))
```

```
## [1] "Mean Squared Error (MSE) for Tuned XGBoost: 5.10763446489669"
```

```r
# Store the performance in the dataset
model_performance <- rbind(model_performance, list(Model_Name = "Tuned XGBoost", MSE = mse_xgb_tuned))
```

All performances

```r
# Add the RMSE by taking the square root of the MSE
model_performance$RMSE <- sqrt(model_performance$MSE)

# display overall performance
print(model_performance[order(model_performance$RMSE), ])
```

```
##                     Model_Name      MSE     RMSE
## 7                  Lasso + OLS 4.988089 2.233403
## 14               Tuned XGBoost 5.107634 2.260008
## 11                RandomForest 5.208176 2.282143
## 12                    Tuned RF 5.284020 2.298700
## 13                    OLS+Corr 5.594823 2.365338
## 8                Lasso + Lasso 5.647946 2.376541
## 2                          OLS 5.950870 2.439441
## 10                  RF + Ridge 6.004438 2.450395
## 9  Lasso + Lasso (1-SE rule) 6.094985 2.468802
## 6       Stepwise BIC + Lasso 6.743017 2.596732
```

```
## 5       Stepwise AIC + Lasso 6.873076 2.621655
## 4         Stepwise-BIC + OLS 6.893511 2.625550
## 3         Stepwise-AIC + OLS 7.816796 2.795853
## 1                   Baseline 9.523514 3.086019
```

Ranking the Intersections

To perform the ranking, we use the best model based on MSE.

```
# Refit the linear model on the full data using only the selected variables
refit_lasso_lm_full_model <- lm(as.formula(selected_vars_formula), data = dat_dum)

# Predictions on the full dataset
predictions_full_lm <- predict(refit_lasso_lm_full_model, newdata = dat_dum)

# Create a dataframe with predictions, 'int_no', and original 'acc'
predictions_df <- data.frame(int_no = dat_dum$int_no,
                             acc = dat_dum$acc,
                             predicted_acc = predictions_full_lm)

# Perform the join with inter_names on int_no
final_df <- merge(predictions_df, inter_names, by = "int_no")

# Sort final_df by predicted_acc in descending order
final_df <- final_df[order(final_df$predicted_acc, decreasing = TRUE), ]

# Add ranking column to final_df
final_df$ranking <- seq_len(nrow(final_df))

# Remae x to latitude and y to longitude
final_df <- final_df %>% rename(latitude = x, longitude = y)

# Print the final dataframe
head(final_df, 10)
```

```
##      int_no acc predicted_acc latitude longitude          rue_1
## 515     601  17      17.33797 298322.6   5042675     Mont-Royal
## 958    1092  12      15.65736 295863.1   5044271     Jean-Talon
## 308     386  12      14.28682 299700.9   5040100 La Gauchetière
## 338     419   8      14.21700 299405.5   5040023      Mansfield
## 248     317  10      12.67126 298716.7   5039460            Guy
## 340     421  10      11.15602 299303.9   5039871           Peel
## 743     863  12      11.07337 299088.8   5045677         Masson
## 1114   1249  19      10.49632 291866.3   5043984         Acadie
## 217     284   5      10.11414 298196.1   5038826       Atwater
## 329     410   1      10.05484 299611.2   5040350    Beaver Hall
##                rue_2 ranking
## 515      Saint-Denis       1
## 958      Saint-Denis       2
## 308       University       3
## 338     René-Lévesque       4
## 248  Sainte-Catherine       5
## 340     René-Lévesque       6
## 743     Saint-Michel       7
## 1114           Sauvé       8
## 217  Sainte-Catherine       9
## 329     René-Lévesque      10
```

```r
dim(final_df)
```

```
## [1] 1866    8
```

Create the json file with the rankings

```r
# Subset the final_df to include only int_no and ranking
risk_rank_df <- final_df[, c("int_no", "ranking")]

# Save the dataframe to a .csv file
write.csv(risk_rank_df, here("data_clean", "intersection_risk_rank.csv"), row.names = FALSE)

# display the saved file
head(risk_rank_df)
```

```
##     int_no ranking
## 515    601       1
## 958   1092       2
## 308    386       3
## 338    419       4
## 248    317       5
## 340    421       6
```