Shared Car Service Availability Study

Thomas E Linnell

9/28/2019

Introduction

As part of a program to improve the livability of its environs, the city of Tel Aviv, Israel has developed a car sharing service (think Zipcar combined with Blue Bikes) called AutoTel that allows users to take a car from an available location whenever they wish and use it for transportation. When finished, they can park it at the nearest designated spot near their destination that is open. Since this service depends on the users to distribute cars around the city, and since demand patterns frequently shift, it is important to study the usage patterns in order to be able to provide enough cars in the right places to insure that most users can utilize this service when they need it.

This project will use data from a database available on kaggle that contains data on the avaialability of cars in Tel Aviv from mid-December, 2018 to mid-January, 2019. We will examine this data and modify it to make it more useful for our analysis, and will provide some observations on usage patterns.

What this data does not provide, however is any information on the users demand, other than by proxy when a car is moved from a particular location (if someone wants to use a car in a particular neighborhood, but there is no car nearby, we don't see that information). However, we will develop a model, using machine learning techniques, in order to predict the availability of cars.

We will also investigate the performance of an alternative model to provide comparisons to our initial attempt and to see if improvements in accuracy are possible.

Project setup and environment

Check for required packages and install them when necessary.

Data

This project will utilize the dataset *shared_table*, in csv format, which is a sample of a Big Query dataset available on Kaggle provided by Do-It International, and can be found here.

For the purposes of this project, a copy of this dataset is also provided in the github project repository since Kaggle requires an authenticated account in order to access the data.

We will load the data and adjust it into the format that we want to use to analyze the usage patterns.

Download the sample data, unzip the file and convert it for use in R:

```r
dl <- tempfile()

dURL <- "https://github.com/tomcache/data-science-shared-cars/raw/master/datasource/autotel-shared-car-l

download.file(dURL, destfile = dl, method = "wininet")

unzip(dl, overwrite = TRUE, exdir = "autotel")

# This dataset sample contains one months worth of data on parked cars in the format shown below.
```

```
db <- read_csv("autotel//sample_table.csv")
```

```
## Parsed with column specification:
## cols(
##   timestamp = col_character(),
##   latitude = col_double(),
##   longitude = col_double(),
##   total_cars = col_double(),
##   carsList = col_character()
## )
```

Review

We first take a look at the structure, definition and layout of the sample data. We see from the below that we have a total of about 6.7 million observations of parked car locations and times. From the summary, we also see that the number of cars in each location can vary from 0 to 10.

```
glimpse(db)
```

```
## Observations: 6,683,756
## Variables: 5
## $ timestamp  <chr> "2019-01-10 11:45:55.070781 UTC", "2019-01-10 11:45...
## $ latitude   <dbl> 32.09995, 32.06567, 32.06465, 32.05978, 32.05133, 3...
## $ longitude  <dbl> 34.78794, 34.79612, 34.80322, 34.81034, 34.75089, 3...
## $ total_cars <dbl> 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 0, 0, 0, 0, 0, ...
## $ carsList   <chr> "[182]", "[268]", "[106]", "[180]", "[16]", "[72]",...
```

```
summary(db)
```

```
##   timestamp            latitude       longitude       total_cars
##  Length:6683756     Min.   :31.95   Min.   :34.73   Min.   : 0.0000
##  Class :character   1st Qu.:32.06   1st Qu.:34.77   1st Qu.: 0.0000
##  Mode  :character   Median :32.08   Median :34.79   Median : 1.0000
##                     Mean   :32.08   Mean   :34.79   Mean   : 0.6432
##                     3rd Qu.:32.11   3rd Qu.:34.80   3rd Qu.: 1.0000
##                     Max.   :32.15   Max.   :34.95   Max.   :10.0000
##    carsList
##  Length:6683756
##  Class :character
##  Mode  :character
##
##
##
```

We also pull information from the timestamps to get the start and end date of the observations. Since the data is from December 11 through January 10, we expect that there may be effects due to holiday and end of year celebrations.

```
min(db$timestamp)
```

```
## [1] "2018-12-11 15:48:53.592141 UTC"
```

```r
max(db$timestamp)
```

```
## [1] "2019-01-10 12:58:50.134717 UTC"
```

Tidy the data

Our first task will be to arrange the data to make it more useful for our analysis. We want to correct the timestamp format, and we will add columns to allow us to group observations by time and day of the week. We will also adjust the timestamps for local time. Finally, we add columns to allow grouping by location.

In order to facilitate the location analysis, we create a grid structure of approximately 110 x 90m square. We also create a unique key for each grid to allow us to easily index by this metric.

```r
db <- db %>%
  mutate(timestamp = as.POSIXct(timestamp)) %>%
  mutate(timestamp = timestamp + 3600*2, # local time = GMT/UTC + 2
         Hour = lubridate::hour(timestamp),
         Minute = lubridate::minute(timestamp),
         Weekday = lubridate::wday(timestamp)
  )

# We next add a grid identifier, using a grid size of approximately 100m square

db <- db %>%
  mutate(Grid_lat = round(latitude,3),
         Grid_long = round(longitude,3),
         gridKey = (Grid_lat*10000000000 + Grid_long*10000)
 )

# We also can use a calculation of the number of cars in use / available at any given time:

dbCarUse <- db %>%
  group_by(timestamp) %>% summarise(total_cars = sum(total_cars))

dbCarUse <- dbCarUse %>%
  mutate(timestamp = timestamp + 3600*2, # local time = GMT/UTC + 2
         Hour = lubridate::hour(timestamp),
         Minute = lubridate::minute(timestamp),
         label_wday = lubridate::wday(timestamp, label = TRUE, abbr = FALSE)
  )
```

A quick review of the changes that we have made:

```r
glimpse(db)
```

```
## Observations: 6,683,756
## Variables: 11
## $ timestamp  <dttm> 2019-01-10 13:45:55, 2019-01-10 13:45:55, 2019-01-...
## $ latitude   <dbl> 32.09995, 32.06567, 32.06465, 32.05978, 32.05133, 3...
## $ longitude  <dbl> 34.78794, 34.79612, 34.80322, 34.81034, 34.75089, 3...
## $ total_cars <dbl> 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 0, 0, 0, 0, 0, ...
## $ carsList   <chr> "[182]", "[268]", "[106]", "[180]", "[16]", "[72]",...
## $ Hour       <int> 13, 13, 13, 13, 13, 13, 13, 13, 13, 13, 13, 13, 13,...
## $ Minute     <int> 45, 45, 45, 45, 45, 45, 45, 45, 45, 45, 45, 45, 45,...
## $ Weekday    <dbl> 5, 5, 5, 5, 5, 5, 5, 5, 5, 5, 5, 5, 5, 5, 5, 5, 5, ...
## $ Grid_lat   <dbl> 32.100, 32.066, 32.065, 32.060, 32.051, 32.042, 32....
## $ Grid_long  <dbl> 34.788, 34.796, 34.803, 34.810, 34.751, 34.774, 34....
## $ gridKey    <dbl> 321000347880, 320660347960, 320650348030, 320600348...
```

Analysis

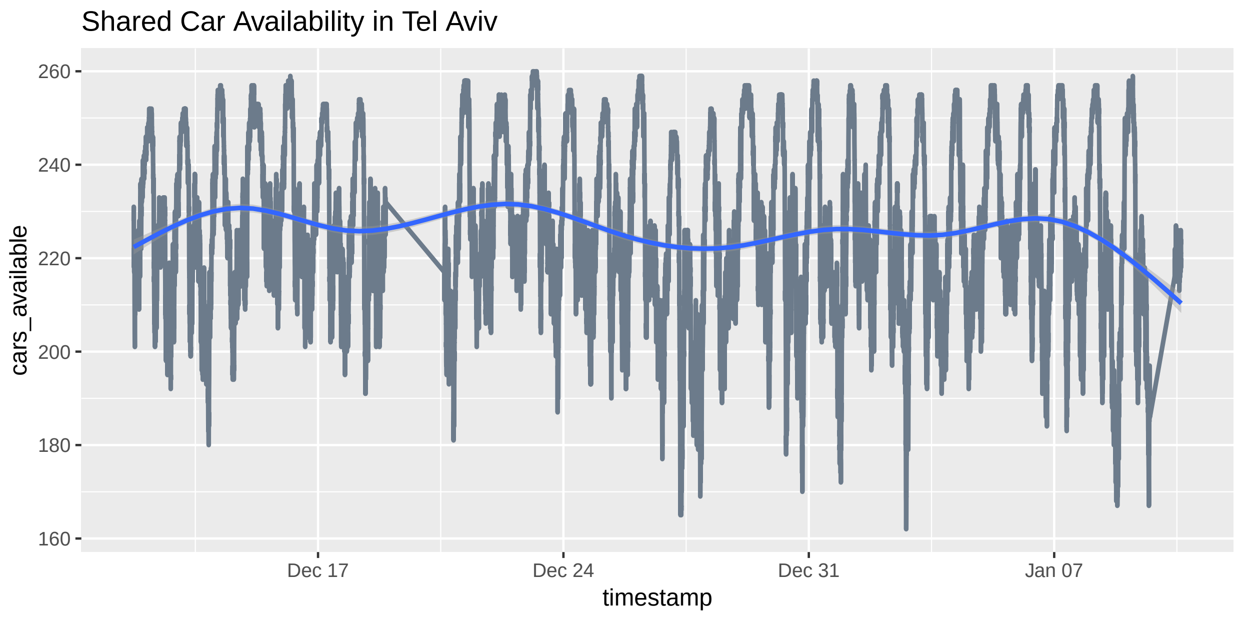
We now want to examine the data in more detail to see what it can tell us about the usage patterns for the shared car service. Since we have the number and location of all of the shared cars that are not in use for each timestamp entry, we can use this to plot the availability of cars across the fleet for the sample time period.

We see several patterns in this data. From this plot, we can see that there is a daily cycle that the demand follows. We also see a dropout around December 18, which as it turns out is a national holiday in Israel.

By looking at the smoothed plot, we can also discern that there is a weekly cycle to car demand as well.

It is also likely that there are effects for other holiday observations during this timeframe which would be more apparent if we had a wider range of samples, but these will not be periodic in nature, as is the case for the previously noted national holiday.

```
## `geom_smooth()` using method = 'gam' and formula 'y ~ s(x, bs = "cs")'
```

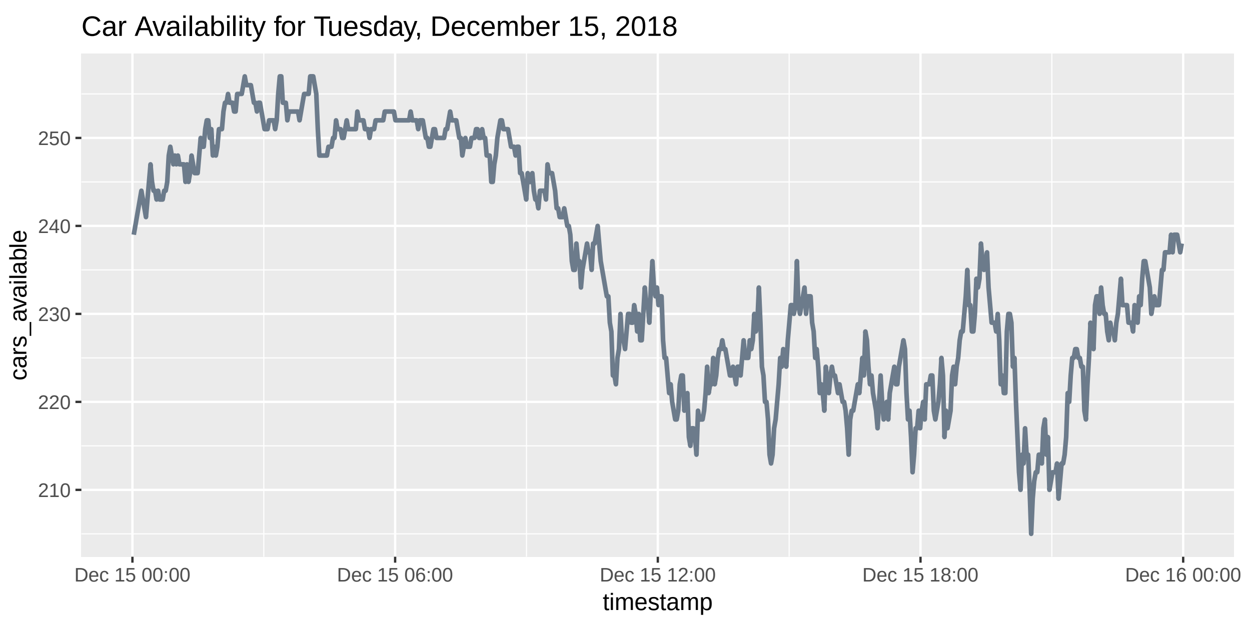


Let’s dive a little further into the details of this data.

By looking at a single 24-hour period for a weekday, we can observe that the demand for cars starts to occur around 6am, has several local peaks during the day, relaxes around lunch and dinnertime, and the peak demand for the day occurs around 9pm before tapering off and going through the cycle again.

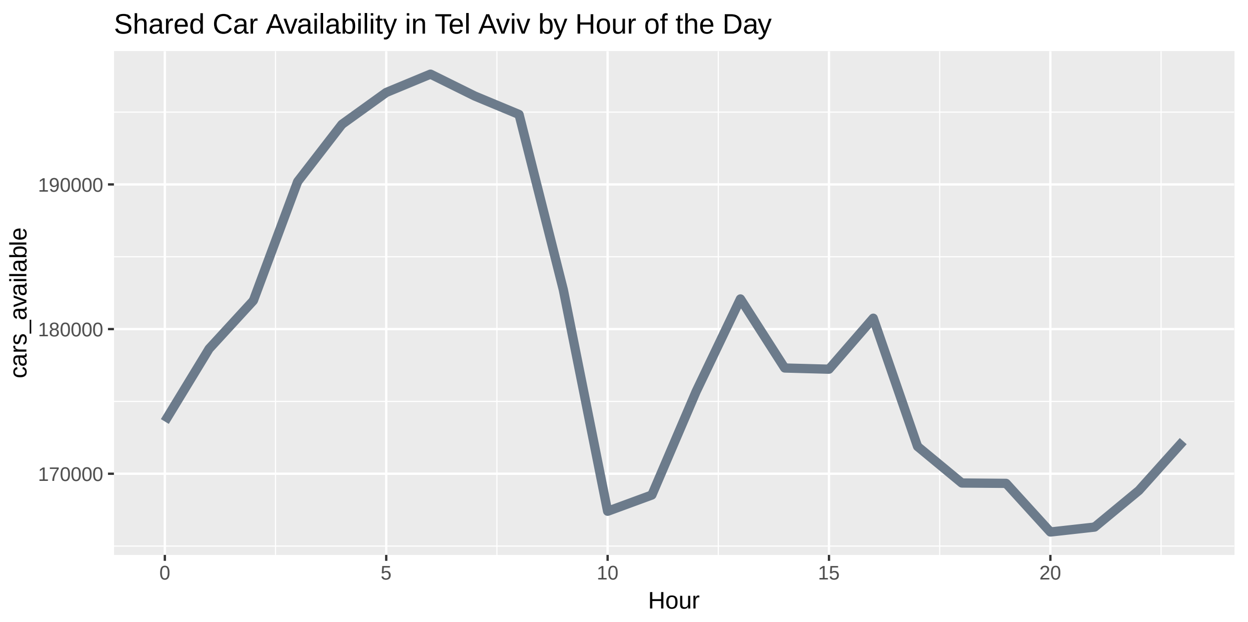
```r
sample_day1 <- ymd("2018-12-15")

db_day1 <- db %>% filter(as_date(timestamp) == sample_day1 )
```

```r
db_day1 %>% group_by(timestamp) %>% summarize(cars_available = sum(total_cars)) %>%
  ggplot(aes(timestamp, cars_available)) + geom_line(color = "slategray4", size = 1.0) +
  ggtitle("Car Availability for Tuesday, December 15, 2018")
```

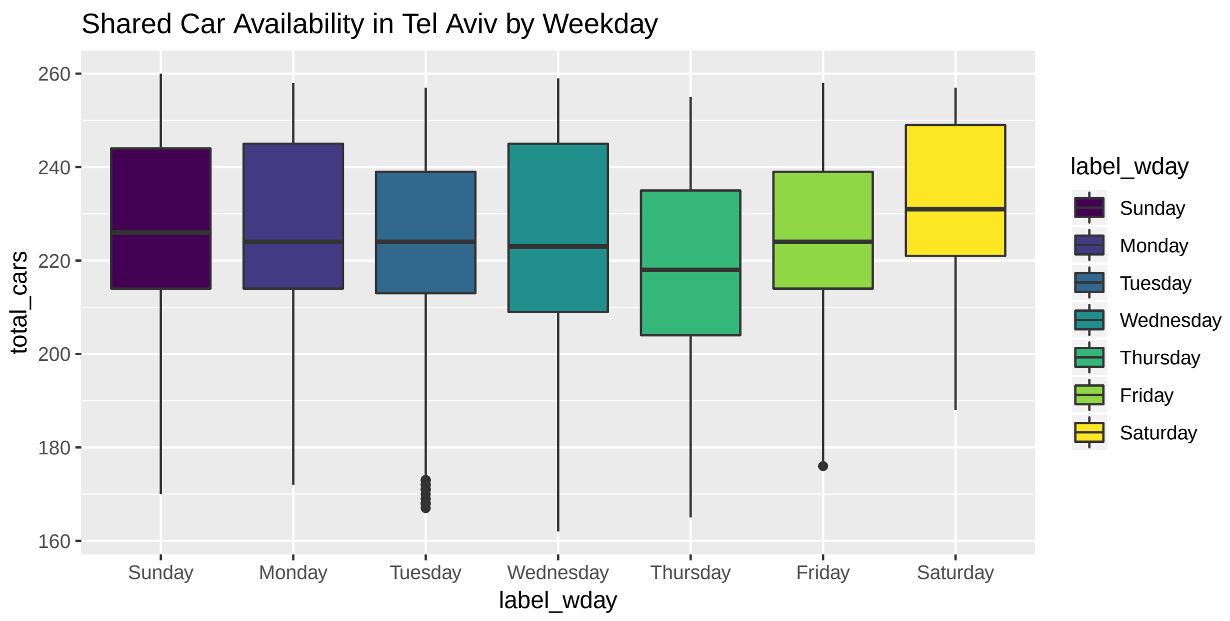


```r
fleet <-  max(all_cars$cars_available)
```

We'll next take a look in more detail at the apparent periodicity in the sample to confirm our intuition. First, we sort out how the availability varies by the time of day across the entire duration of our data set, to see if the pattern observed above fits the general usage pattern. We see that it is very similar to the pattern exhibited in the single day sample outlined above.



We next analyze the variability by weekday. We do see variability in this plot. Although most of the weekdays are very similar to one another, we note that the demand for cars hits a peak on Thursdays, while the slowest activity is recorded on Saturday, which is a day of religious observance in Israel.



We also perform a calculation to understand the amount of utilization of the fleet. We see in the data that there is a total of 260 cars. We can use this information to calculate how much of the car fleet is in use at any given time. We note that the maximum utilization is 37%, which means that at least about two-thirds of the fleet is always idle at any given time. This is likely due to several factors, such as having cars parked in areas where there is little demand at the moment, and the mean time between trip starts from any particular location. In other words, the system will never be highly utilized unless active steps are taken to attempt to match supply with demand at every point.

```
# Size of our car fleet
fleet
```

```
## [1] 260
```

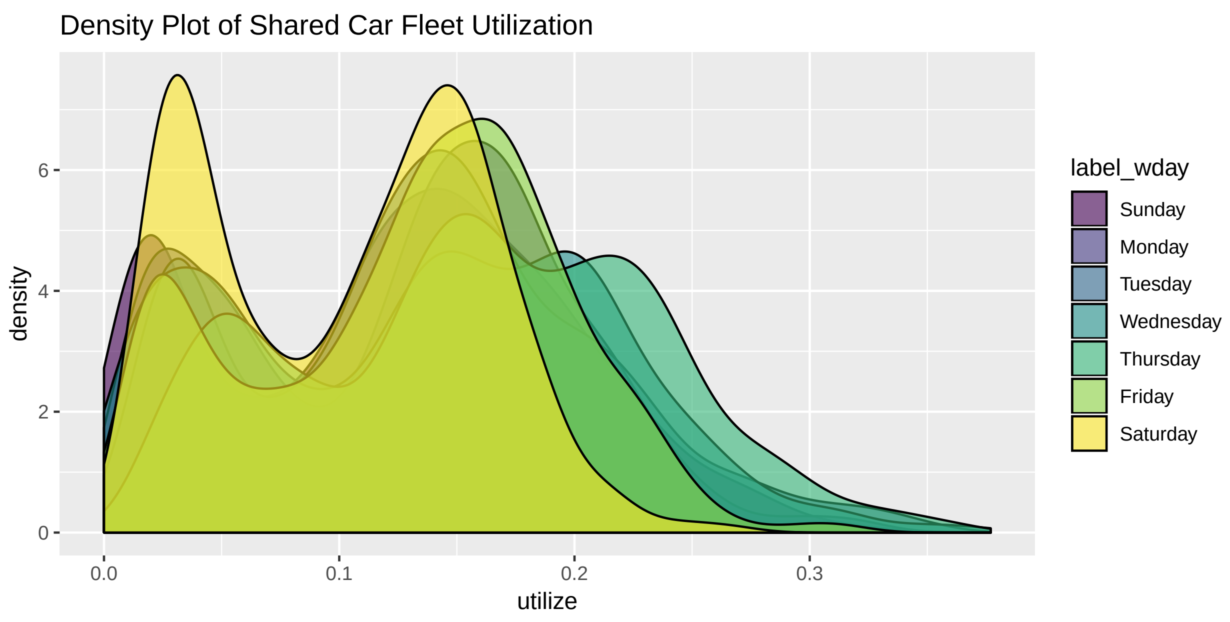
```
# Minimum number of cars in use
(Utilize_min <- min(dbCarUse$utilize))
```

```
## [1] 0
```

```
# Maximum number of cars in use
(Utilize_max <- max(dbCarUse$utilize))
```

```
## [1] 0.3769231
```

By plotting the overlapped density plots by day, we can get a better understanding of the distribution of this demand. We see that the utilization has two peaks at ~5% utilization ,and ~ 15% utilization. The busiest days also exhibit an additional "shoulder" at about 20%. This type of information can be used to plan maintenance cycles, demand-level pricing models, and so on.

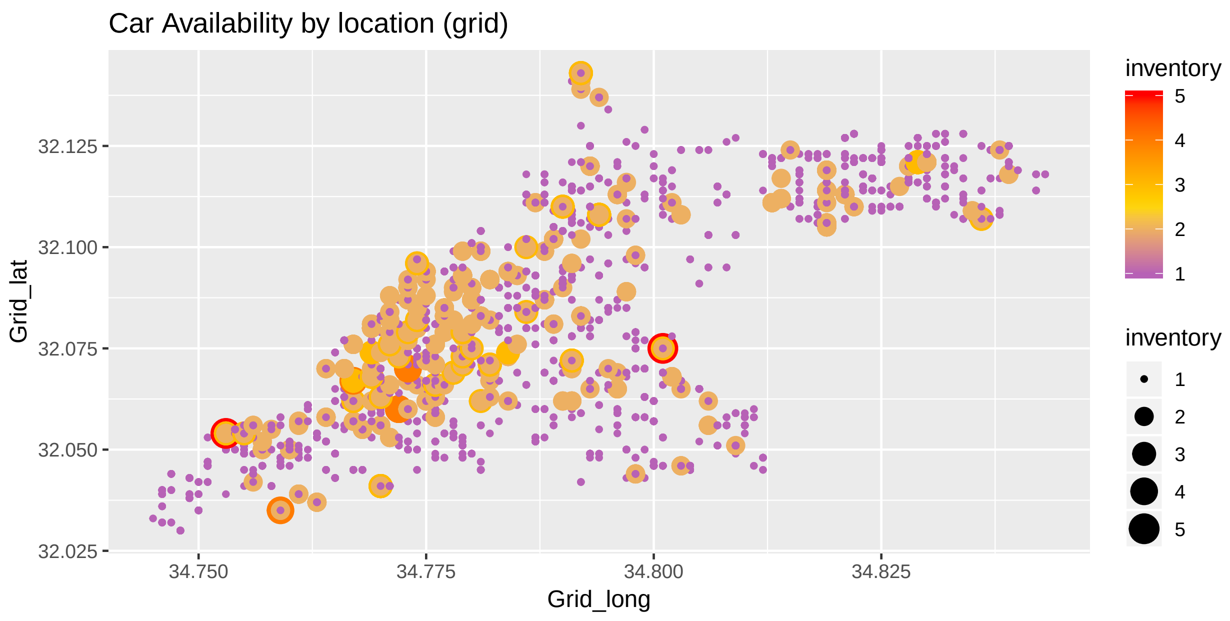


Next we analyze the number of cars that visit each of our defined grids over a one-day period to get an idea of the turnover and demand at various locations around Tel Aviv. To do this, we will need to be able to identify the individual cars that visit each location by utilizing the car identifiers in the $carsList vector. We will tally the total number of unique cars found in each grid over the time period, and then plot that against the geographic coordinates.

```
# split out the car identifiers per observation into distinct columns

dbcars <-  db %>%
  select(carsList) %>%
  distinct() %>%
  mutate(cars = sub(carsList, pattern = "\\[", replacement = "")) %>%
  mutate(cars = sub(cars, pattern = "\\]", replacement = "")) %>%
  separate(cars, sep = ",", into = c("car1", "car2", "car3", "car4", "car5", "car6", "car7", "car8", "c
  mutate_at(2:11, as.numeric)
CarsList <- (1:max(dbcars$car1, na.rm = T))[1:max(dbcars$car1, na.rm = T) %in% dbcars$car1]

# create a list of entries separated by car, and add grid information:

dbByCar <- left_join(db, dbcars) %>%
  gather(CarI, Car, car1:car10)  %>%
  filter(!is.na(Car)) %>%
  group_by(Car) %>%
  arrange(timestamp) %>%
  ungroup()
```

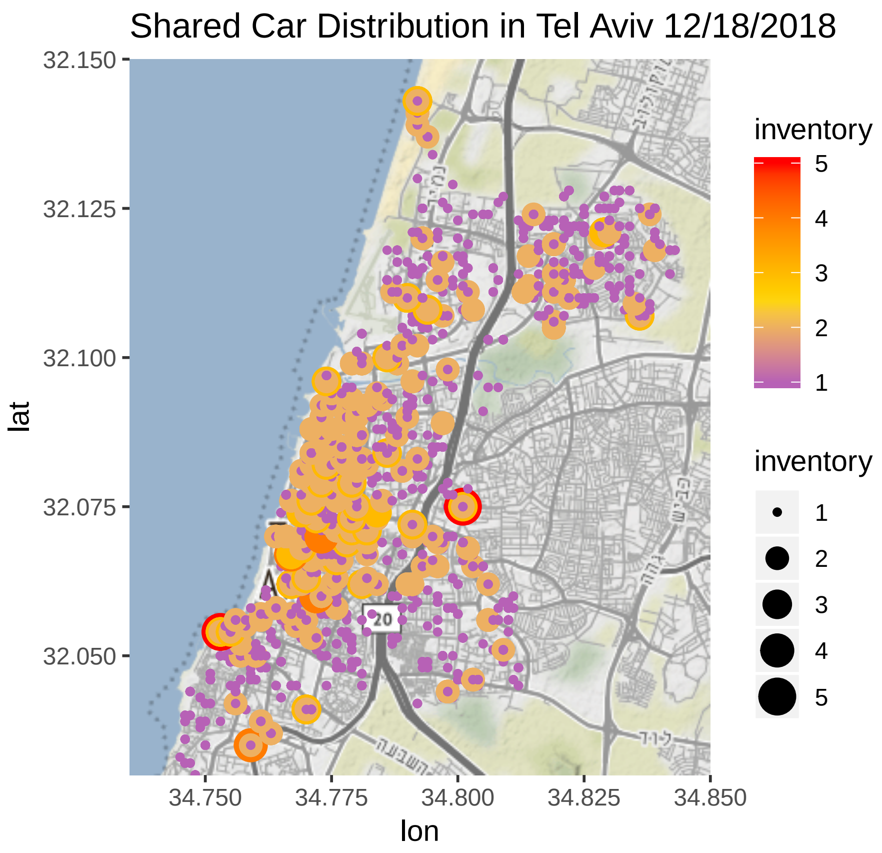
```
## Joining, by = "carsList"
```

```
# We'll limit our data to a single day to keep it from becoming too unweildy. We will also calculate a 


flowByGrid <- dbByCar %>%
  filter(as_date(timestamp) == sample_day1 ) %>%
  group_by(Grid_lat, Grid_long, Hour) %>%
  summarize(inventory = n_distinct(Car)
  )

flowByGrid %>% ggplot(aes(Grid_long, Grid_lat, color = inventory, size = inventory)) +
  geom_point() +
  scale_color_gradient2(midpoint = 2.5, low = "blue",  mid = "gold",
                        high = "red", space = "Lab" ) +
  ggtitle("Car Availability by location (grid)")
```



And through the magic of ggmaps, we can plot these locations over a map of the city to see how these patterns map onto the neighborhoods of the city.

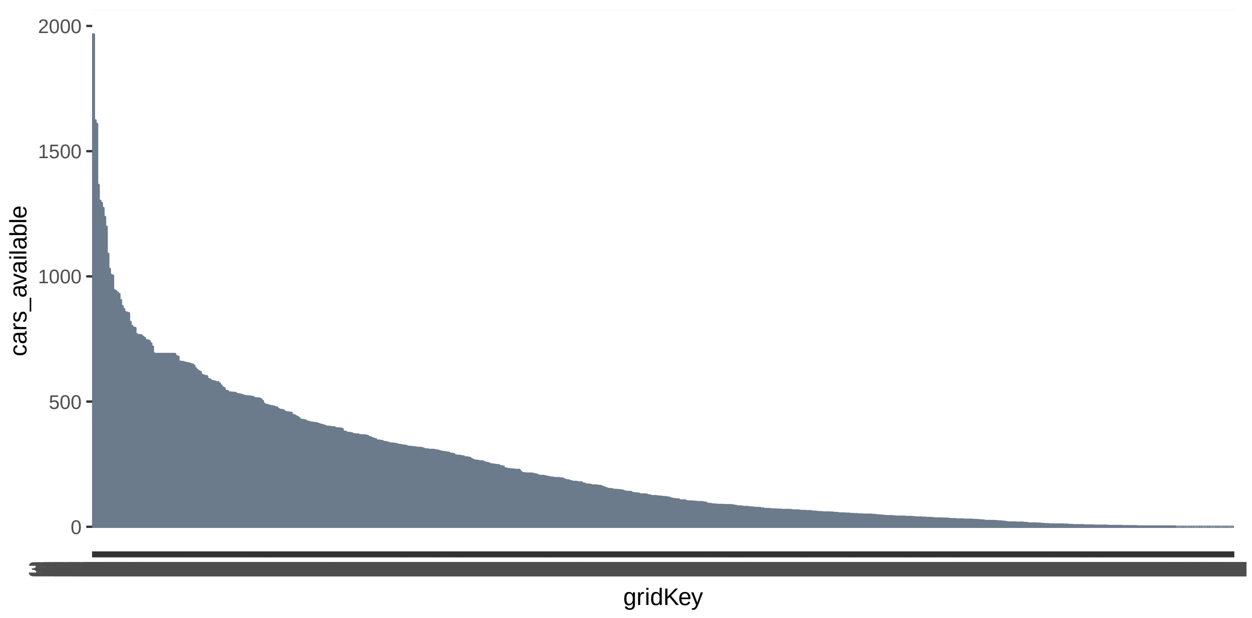
```
## Source : http://tile.stamen.com/terrain/12/2443/1661.png

## Source : http://tile.stamen.com/terrain/12/2444/1661.png

## Source : http://tile.stamen.com/terrain/12/2443/1662.png

## Source : http://tile.stamen.com/terrain/12/2444/1662.png
```



Availability patterns

Let's look at the distribution of the total number of cars by location. We can see from this chart that once we remove the geographic clustering information, the distribution is not random but follows a power law distribution, with some locations having many more cars fill spots than others.

```
db_cars <- db_day1 %>% group_by(gridKey) %>% summarize(cars_available = sum(total_cars))

db_cars$gridKey <- factor(db_cars$gridKey, levels = db_cars$gridKey[order(-db_cars$cars_available)])

ggplot(data = db_cars, aes(x = gridKey, y = cars_available)) +
  geom_bar(stat = "identity", color = "slategray4")
```



Availability Prediction

Now that we have gotten some idea of the usage patterns in the data, we want to build a prediction system for cars in specific locations.

Training and Test Sets

We start by building a series of training and test sets. Since the number of observations in the full sample is fairly large, we will also create two smaller data sets, one which incorporates data from a single day, and another that includes observations for a single week. We will use these sets to compare the models performance against these different time epochs, as well as provide a more convenient vehicle for experimentation of various ML methods.

```r
set.seed(1, sample.kind="Rounding")
```

```
## Warning in set.seed(1, sample.kind = "Rounding"): non-uniform 'Rounding'
## sampler used
```

```r
# if using R 3.5 or earlier, use `set.seed(1)` instead
test_index <- createDataPartition(y = db$total_cars, times = 1, p = 0.1, list = FALSE)
db_train <- db[-test_index,]
db_test <- db[test_index,]

db_test <- db_test %>%
  semi_join(db_train, by = "gridKey") %>%
  semi_join(db_train, by = "Hour") %>%
  semi_join(db_train, by = "Weekday")

# We will now create a single-day data set, and do the same thing:

sample_day1 <- ymd("2018-12-15")
```

```r
db_day <- db %>% filter(as_date(timestamp) == sample_day1 )

set.seed(1, sample.kind="Rounding")
```

```
## Warning in set.seed(1, sample.kind = "Rounding"): non-uniform 'Rounding'
## sampler used
```

```r
# if using R 3.5 or earlier, use `set.seed(1)` instead
test_index <- createDataPartition(y = db_day$total_cars, times = 1, p = 0.1, list = FALSE)
db_day_train <- db_day[-test_index,]
db_day_test <- db_day[test_index,]

db_day_test <- db_day_test %>%
  semi_join(db_day_train, by = "gridKey") %>%
  semi_join(db_day_train, by = "Hour") %>%
  semi_join(db_day_train, by = "Weekday")

# and a 1-week data set, from 12/20/18 through 12/26/18:
week_day1 <- ymd("2018-12-20")
week_day7 <- ymd("2018-12-26")

db_week <- db %>% filter(timestamp >= week_day1 & timestamp <= week_day7 )

set.seed(1, sample.kind="Rounding")
```

```
## Warning in set.seed(1, sample.kind = "Rounding"): non-uniform 'Rounding'
## sampler used
```

```r
# if using R 3.5 or earlier, use `set.seed(1)` instead
test_index <- createDataPartition(y = db_week$total_cars, times = 1, p = 0.1, list = FALSE)
db_week_train <- db_week[-test_index,]
db_week_test <- db_week[test_index,]

db_week_test <- db_week_test %>%
  semi_join(db_week_train, by = "gridKey") %>%
  semi_join(db_week_train, by = "Hour") %>%
  semi_join(db_week_train, by = "Weekday")
```

Prediction Model

Now that we have our test sets, we will start building up a prediction model.

```r
# function for calculating RMSE of prediction vs. actual

RMSE <- function(true_ratings, predicted_ratings){
  sqrt(mean((true_ratings - predicted_ratings)^2))
}
```

We now will build a predictor, utilizing a linear regression model. Since we have detected variability associated with location, time of day, and day of week, we will select these as factors for our calculation.

Single day model

We begin by building up a model utilizing our one-day sample. We calculate a baseline mean, and then add hour of the day and location (by grid) effects to see how these affect the accuracy of the model.

```r
# build up a model for the day data set, start with a baseline mean:

db_dayTrainMean <- db_day_train %>% summarize(Mean = mean(total_cars))
mu_db_dayTrain <- mean(db_day_train$total_cars)

baseline_day_rmse <- RMSE(db_day_test$total_cars, mu_db_dayTrain)


# Now add the hour effect:


day_hour_avgs <- db_day_train %>%
  group_by(Hour) %>%
  summarize(b_i_day = mean(total_cars - mu_db_dayTrain))

# day_hour_avgs %>% qplot(b_i, geom ="histogram", data = ., color = I("slategray4"))


predicted_ratings_day <- mu_db_dayTrain + db_day_test %>%
  left_join(day_hour_avgs, by='Hour') %>%
  pull(b_i_day)

model_1_day_rmse <- RMSE(predicted_ratings_day, db_day_test$total_cars)

# Then we add the location effect:

day_grid_avgs <- db_day_train %>%
  left_join(day_hour_avgs, by = 'Hour') %>%
  group_by(gridKey) %>%
  summarize(b_u_day = mean(total_cars - mu_db_dayTrain - b_i_day))

predicted_ratings_day <- db_day_test %>%
  left_join(day_hour_avgs, by='Hour') %>%
  left_join(day_grid_avgs, by='gridKey') %>%
  mutate(pred = mu_db_dayTrain + b_i_day + b_u_day) %>%
  pull(pred)


model_2_day_rmse <- RMSE(predicted_ratings_day, db_day_test$total_cars)
```

We now put all of our modeling results into a table for comparison

We start with our analysis for a single day. In these results, we see that the hour of the day factor does not improve the prediction result, but by adding in information about the location we do significantly improve our result.

```r
rmse_day_results <- tibble(method = "One Day Baseline (mean)", RMSE = baseline_day_rmse)

rmse_day_results <- bind_rows(rmse_day_results,
                  tibble(method="One Day Hour Effect Model",
                  RMSE = model_1_day_rmse)
                  )

rmse_day_results <- bind_rows(rmse_day_results,
                  tibble(method="One Day Location + Hour Effects Model",
                  RMSE = model_2_day_rmse)
                  )

rmse_day_results %>% knitr::kable()
```

method	RMSE
One Day Baseline (mean)	0.6564794
One Day Hour Effect Model	0.6559568
One Day Location + Hour Effects Model	0.4397387

Single week model

We will now develop a similar model, expanding our data to a full week of samples. We also add a calculation for the day of week effects to see how significant that might be.

We now look at the results for a full week. Similar to our first set of results, we see that the time of day factor is not very important, while the location remains significant. We do see a slight degradation as compared to the single day, this might be attributable to different usage patterns between weekdays and weekends, for example.

```
rmse_week_results <- tibble(method = "One Week Baseline (mean)", RMSE = baseline_week_rmse)


rmse_week_results <- bind_rows(rmse_week_results,
                  tibble(method="One Week Hour Effect Model",
                  RMSE = model_1_week_rmse)
                  )

rmse_week_results <- bind_rows(rmse_week_results,
                  tibble(method="One Week Location + Hour Effects Model",
                  RMSE = model_2_week_rmse)
                  )

rmse_week_results <- bind_rows(rmse_week_results,
                  tibble(method="One Week Weekday + Location + Hour Effects Model",
                  RMSE = model_3_week_rmse)
                  )

rmse_week_results %>% knitr::kable()
```

method	RMSE
One Week Baseline (mean)	0.6947236
One Week Hour Effect Model	0.6937836
One Week Location + Hour Effects Model	0.5690924
One Week Weekday + Location + Hour Effects Model	0.5689734

Full sample model

We now proceed to expand our model to include the full set of data. This data will include the previously noted holidays as well as several weeks of daily and weekly patterns.

In our model with the full set of data, we see the same basic pattern as in the previous two models, with further degradation in the accuracy as more data with presumably additional variation of usage patterns. We noted at the outset that this time period, coming as it does over the end of the year, includes various religious and national holidays, and presumably travel patterns that are not regular.

```
rmse_results <- tibble(method = "Full Sample Baseline (mean)", RMSE = baseline_rmse)


rmse_results <- bind_rows(rmse_results,
                  tibble(method="Full Sample Hour Effect Model",
                  RMSE = model_1_rmse)
                  )

rmse_results <- bind_rows(rmse_results,
                  tibble(method="Full Sample Location + Hour Effects Model",
                  RMSE = model_2_rmse)
                  )

rmse_results <- bind_rows(rmse_results,
                  tibble(method="Full Sample Weekday + Location + Hour Effects Model",
                  RMSE = model_3_rmse)
                  )

rmse_results %>% knitr::kable()
```

method	RMSE
Full Sample Baseline (mean)	0.6988234
Full Sample Hour Effect Model	0.6976902
Full Sample Location + Hour Effects Model	0.6014058
Full Sample Weekday + Location + Hour Effects Model	0.6013389

Prediction Model Using knn

We see from the above that throughout the different datasets we are able to improve our modeling accuracy by including the time of day and location factors. The differences from day to day do not seem to help.

We will now experiment with a knn model. Since the demand for cars is likely to be very similar in the locations immediately adjacent to any given spot, and vary slowly as the spots are traversed, it seems intuitive that this technique may help with accuracy by including some neighbor information in our model.

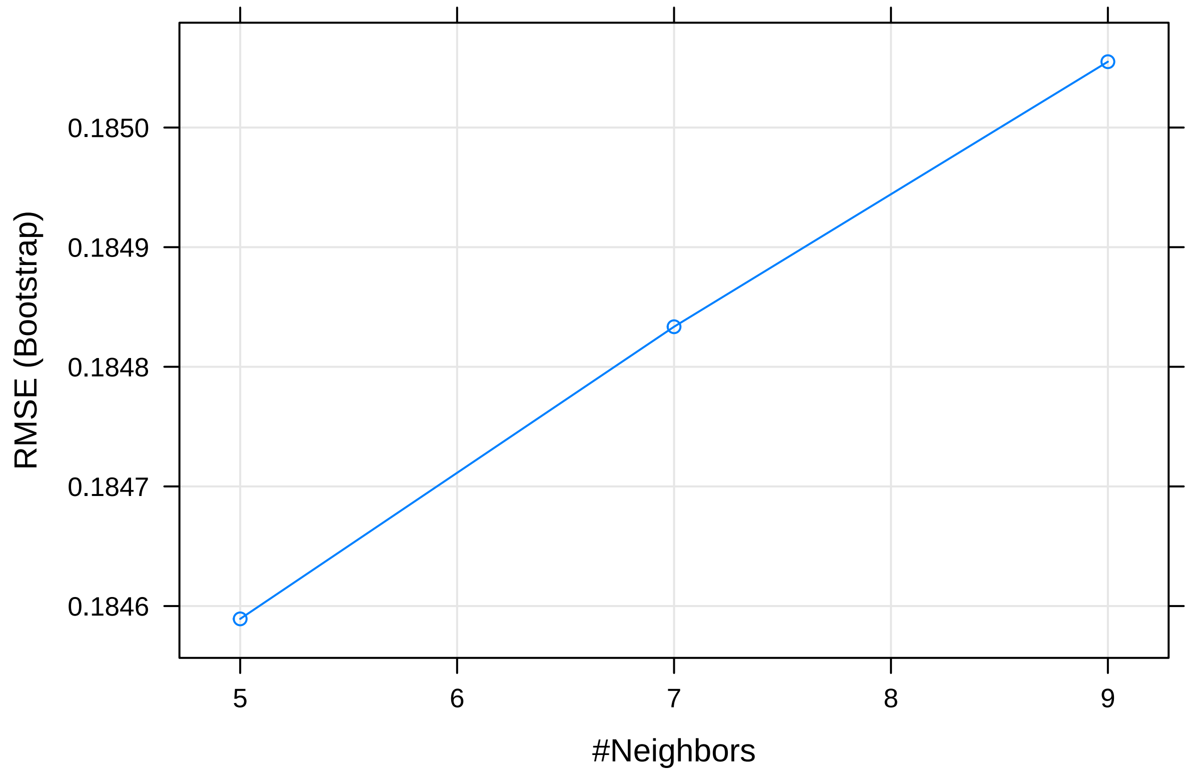
However, it turns out that this model is computationally intensive with large numbers of data points, so for the purposes of this research, we will limit our experiment to the one day sample that we have been using in some of our previous models.

```
# Now develop our knn model and prediction
# Model: day

train_knnD <- train(total_cars ~ Hour + gridKey, method = "knn", data = db_day_train)
```

```r
train_knnD
```

```
## k-Nearest Neighbors
##
## 224381 samples
##      2 predictor
##
## No pre-processing
## Resampling: Bootstrapped (25 reps)
## Summary of sample sizes: 224381, 224381, 224381, 224381, 224381, 224381, ...
## Resampling results across tuning parameters:
##
##   k  RMSE       Rsquared   MAE
##   5  0.1845893  0.9200604  0.05736550
##   7  0.1848335  0.9198506  0.05748226
##   9  0.1850550  0.9196600  0.05759502
##
## RMSE was used to select the optimal model using the smallest value.
## The final value used for the model was k = 5.
```

```r
plot(train_knnD)
```



```r
train_knnD$bestTune
```

```
##   k
## 1 5
```

```r
y_hat_knnD <- predict(train_knnD, db_day_test, type = "raw")

model_knn_day <- RMSE(y_hat_knnD, db_day_test$total_cars)
```

We see that the best performance of the model is with a small number of neighbors. This seems intuitive, as the demand for cars would likely be very similar with adjacent locations, but the similarity would fall off the further away the nighbor becomes. This neighbor information does improve our RMSE significantly, as we see in the table below.

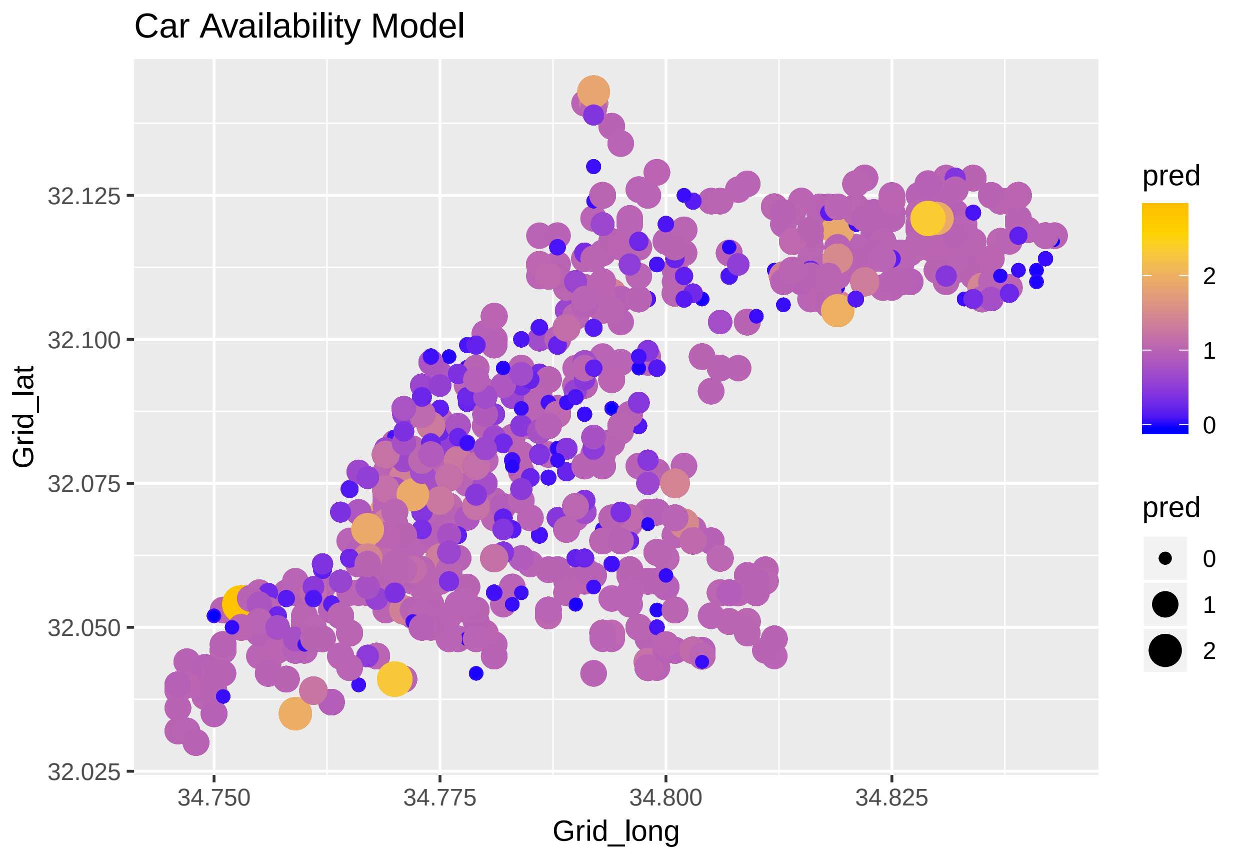
```r
rmse_knn_results <- tibble(method="One Day knn Model",
                                     RMSE = model_knn_day)


rmse_knn_results %>% knitr::kable()
```

method	RMSE
One Day knn Model	0.1799048

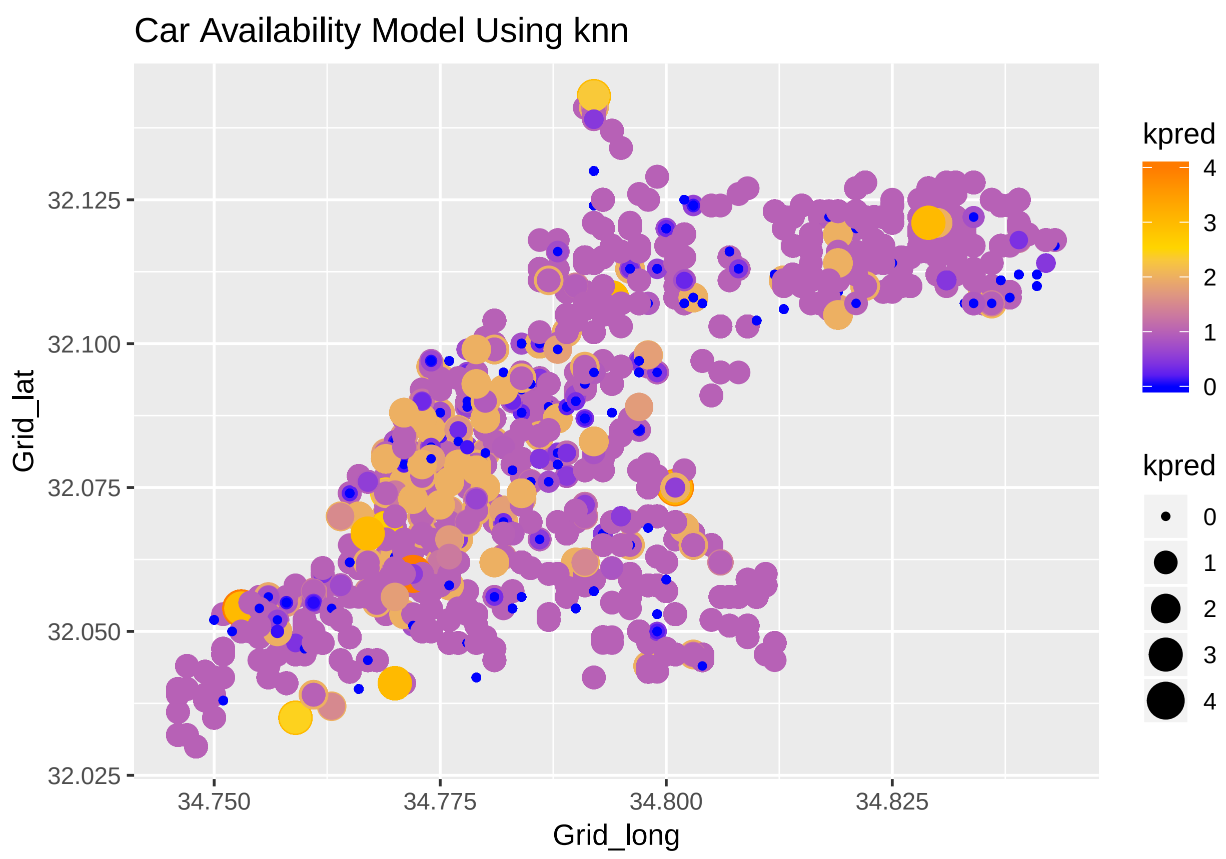
Plotting the predicted results

Now that we have our predictions, let's compare a plot of our results and compare that to our earlier plot of availability.

With our first model, we can see the patterns of usage similar to the actual data beginning to emerge. However the range of predicted available cars is more restricted than in the actual data.



When we look at the knn model data, however, we see that the same clustering patterns that we saw in our data analysis are now more visible in the plot, and the relative scale of availability more closely matches the actual data. Overall the knn model provides the best RMSE and most accurate representation of the pattern of usage that we are attempting to predict. More work remains in data reduction techniques in order to be able to generalize this work for larger samples of data.



Car Availability Model Using knn
Grid_lat
Grid_long
32.125
32.100
32.075
32.050
32.025
34.750
34.775
34.800
34.825
kpred
4
3
2
1
0
kpred
0
1
2
3
4