1. 49999 New York taxi trips

To drive a yellow New York taxi, you have to hold a "medallion" from the city's *Taxi and Limousine Commission*. Recently, one of those changed hands for over one million dollars, which shows how lucrative the job can be.

In this project, we will analyze a random sample of 49999 New York journeys made in 2013. We will also use regression trees and random forests to build a model that can predict the locations and times when the biggest fares can be earned.

The dataset used in this project is a sample from the complete 2013 NYC taxi data which was originally obtained and published by Chris Whong.

In [1]:
```
# Loading the tidyverse
library(tidyverse)
# Reading in the taxi data
taxi <- read_csv('datasets/taxi.csv')

# Taking a look at the first few rows in taxi
head(taxi)
```

```
-- Attaching packages --------------------------------------- tidyverse 1.3.0 --
v ggplot2 3.2.1     v purrr   0.3.3
v tibble  2.1.3     v dplyr   0.8.3
v tidyr   1.0.0     v stringr 1.4.0
v readr   1.3.1     v forcats 0.4.0
-- Conflicts ------------------------------------------ tidyverse_conflicts() --
x dplyr::filter() masks stats::filter()
x dplyr::lag()    masks stats::lag()
Parsed with column specification:
cols(
  medallion = col_character(),
  pickup_datetime = col_datetime(format = ""),
  pickup_longitude = col_double(),
  pickup_latitude = col_double(),
  trip_time_in_secs = col_double(),
  fare_amount = col_double(),
  tip_amount = col_double()
)
```

medallion	pickup_datetime	pickup_longitude	pickup_latitude	trip_time_in_secs	fare_amount	tip_amount
4D24F4D8EF35878595044A52B098DFD2	2013-01-13 10:23:00	-73.94646	40.77273	600	8.0	2.5
A49C37EB966E7B05E69523D1CB7BE303	2013-01-13 04:52:00	-73.99827	40.74041	840	18.0	0.0
1E4B72A8E623888F53A9693C364AC05A	2013-01-13 10:47:00	-73.95346	40.77586	60	3.5	0.7
F7E4E9439C46B8AD5B16AB9F1B3279D7	2013-01-13 11:14:00	-73.98137	40.72473	720	11.5	2.3
A9DC75D59E0EA27E1ED328E8BE8CD828	2013-01-13 11:24:00	-73.96800	40.76000	240	6.5	0.0
19BF1BB516C4E992EA3FBAEDA73D6262	2013-01-13 10:51:00	-73.98502	40.76341	540	8.5	1.7

2. Cleaning the taxi data

As we can see above, the `taxi` dataset contains the times and price of a large number of taxi trips. Importantly we also get to know the location, the longitude and latitude, where the trip was started. The `taxi` dataset needs to be cleaned before we're ready to use it.

In [2]:
```
# Renaming the location variables,
# dropping any journeys with zero fares and zero tips,
# and creating the total variable as the log sum of fare and tip
taxi <- taxi %>%
     rename(lat = pickup_latitude, long = pickup_longitude) %>%
    filter(fare_amount | tip_amount > 0) %>%
    mutate(total = log(fare_amount + tip_amount))
```

3. Zooming in on Manhattan

While the dataset contains taxi trips from all over New York City, the bulk of the trips are to and from Manhattan, so we are going to focus primarily on trips initiated there.

In [3]:
```
# Reducing the data to taxi trips starting in Manhattan
# Manhattan is bounded by the rectangle with
# latitude from 40.70 to 40.83 and
# longitude from -74.025 to -73.93
taxi <- taxi  %>%
    filter(between(lat, 40.70,40.83) & between(long, -74.025,-73.93))
```

4. Plotting a Density Map

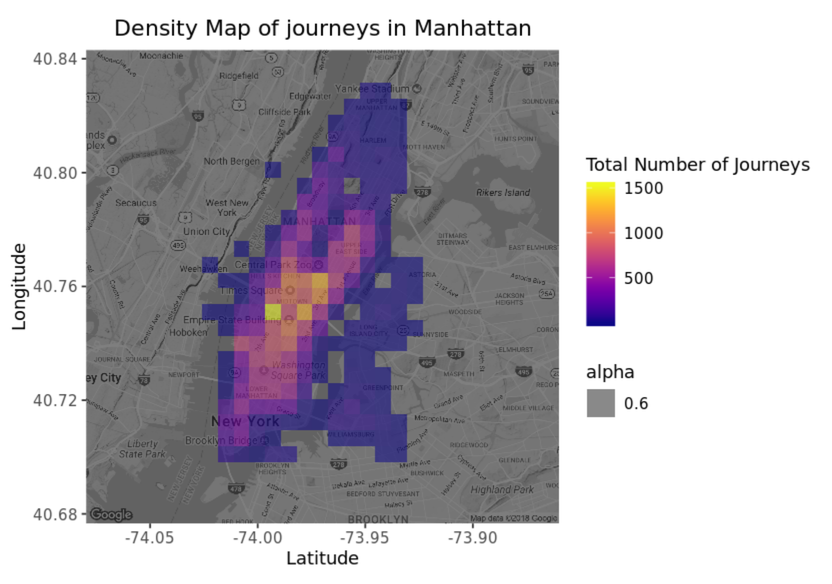
We're going to use the `ggmap` package together with `ggplot2` to visualize where in Manhattan people tend to start their taxi journeys.

In [4]:
```
# Loading in ggmap and viridis for nice colors
library(ggmap)
library(viridis)

# Retrieving a stored map object which originally was created by
# manhattan <- get_map("manhattan", zoom = 12, color = "bw")
manhattan <- readRDS("datasets/manhattan.rds")

# Drawing a density map with the number of journey start locations
ggmap(manhattan, darken = 0.5) +
   scale_fill_viridis(option = 'plasma') +
    geom_bin2d(data = taxi, aes(x = long, y = lat, bins = 60, alpha = 0.6)) +
    xlab("Latitude") +
    ylab("Longitude") +
    labs(fill = "Total Number of Journeys")+
    ggtitle("Density Map of journeys in Manhattan") + theme(plot.title = element_text(hjust = 0.5))
```

```
Google's Terms of Service: https://cloud.google.com/maps-platform/terms/.
Please cite ggmap if you use it! See citation("ggmap") for details.
Loading required package: viridisLite
Warning message:
"Ignoring unknown aesthetics: bins"
```



5. Predicting taxi fares using a tree

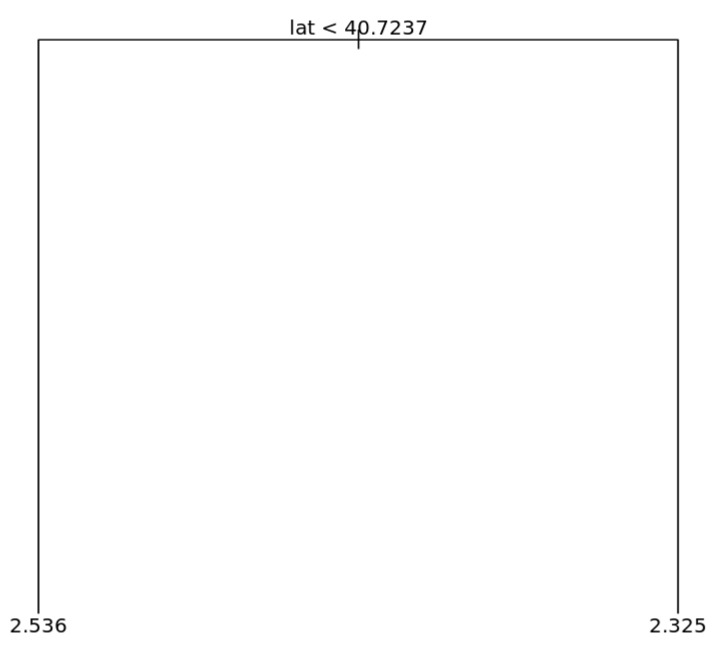
The map shows that the journeys are highly concentrated in the business and tourist areas such as the Times Square and the Central Park. We also see that some taxi trips originating in Brooklyn slipped through, but that's fine.

We're now going to use a regression tree to predict the `total` fare with `lat` and `long` being the predictors. The `tree` algorithm will try to find cutpoints in those predictors that results in the decision tree with the best predictive capability.

In [10]:
```
# Loading in the tree package
library(tree)

# Fitting a tree to lat and long
fitted_tree <- tree(total ~ lat + long, data = taxi)

# Draw a diagram of the tree structure
plot(fitted_tree)
text(fitted_tree)
```



6. Adding more predictors

The tree above looks a bit frugal, it only includes one split: It predicts that trips where `lat < 40.7237` are more expensive, which makes sense as it is downtown Manhattan.Taxi drivers will need more information than this.

We will adding some more predictors related to the *time* the taxi trip was made.

In [11]:
```
# Loading in the lubridate package
library(lubridate)

# Generate the three new time variables
taxi <- taxi %>%
    mutate(hour = hour(pickup_datetime), wday = wday(pickup_datetime, label = TRUE), month = month(pickup_datetime,
 label = TRUE))
```

```
Attaching package: 'lubridate'

The following object is masked from 'package:base':

    date
```

7. Fitting a new regression tree

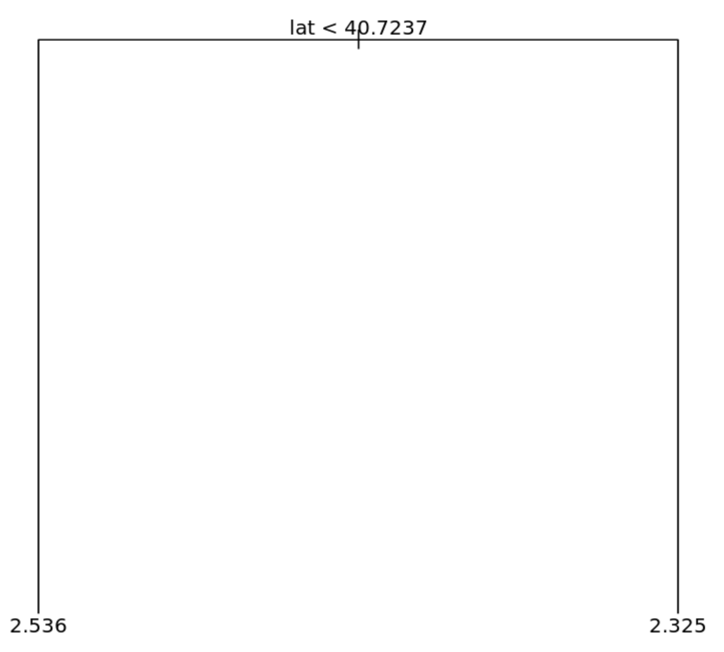
Let's try fitting a new regression tree where we include the new time variables.

In [12]:
```
# Fitting a tree with total as the outcome and
# lat, long, hour, wday, and month as predictors
fitted_tree <- tree(total ~ lat + long + hour + wday + month, data = taxi)

# draw a diagram of the tree structure
plot(fitted_tree)
text(fitted_tree)

# Summarizing the performance of the tree
summary(fitted_tree)
```

```
Regression tree:
tree(formula = total ~ lat + long + hour + wday + month, data = taxi)
Variables actually used in tree construction:
[1] "lat"
Number of terminal nodes:  2
Residual mean deviance:  0.3041 = 13910 / 45760
Distribution of residuals:
    Min.  1st Qu.   Median     Mean  3rd Qu.     Max.
-1.61900 -0.37880 -0.04244  0.00000  0.32660  2.69900
```



8. One tree is not enough

The regression tree has not changed after including the three time variables. This is likely because latitude is still the most promising first variable to split the data on, and after that split, the other variables are not informative enough to be included. A random forest model, where many different trees are fitted to subsets of the data, may well include the other variables in some of the trees that make it up.

In [13]:
```
# Loading in the randomForest package
library(randomForest)

# Fitting a random forest
fitted_forest <- randomForest(total ~ lat + long + hour + wday + month, data = taxi, ntree = 80, sampsize = 10000)

# Printing the fitted_forest object
summary(fitted_forest)
```

```
randomForest 4.6-14
Type rfNews() to see new features/changes/bug fixes.

Attaching package: 'randomForest'

The following object is masked from 'package:dplyr':

    combine

The following object is masked from 'package:ggplot2':

    margin
```

```
                Length Class  Mode
call                5  -none- call
type                1  -none- character
predicted       45766  -none- numeric
mse                80  -none- numeric
rsq                80  -none- numeric
oob.times       45766  -none- numeric
importance          5  -none- numeric
importanceSD        0  -none- NULL
localImportance     0  -none- NULL
proximity           0  -none- NULL
ntree               1  -none- numeric
mtry                1  -none- numeric
forest             11  -none- list
coefs               0  -none- NULL
y               45766  -none- numeric
test                0  -none- NULL
inbag               0  -none- NULL
terms               3  terms  call
```

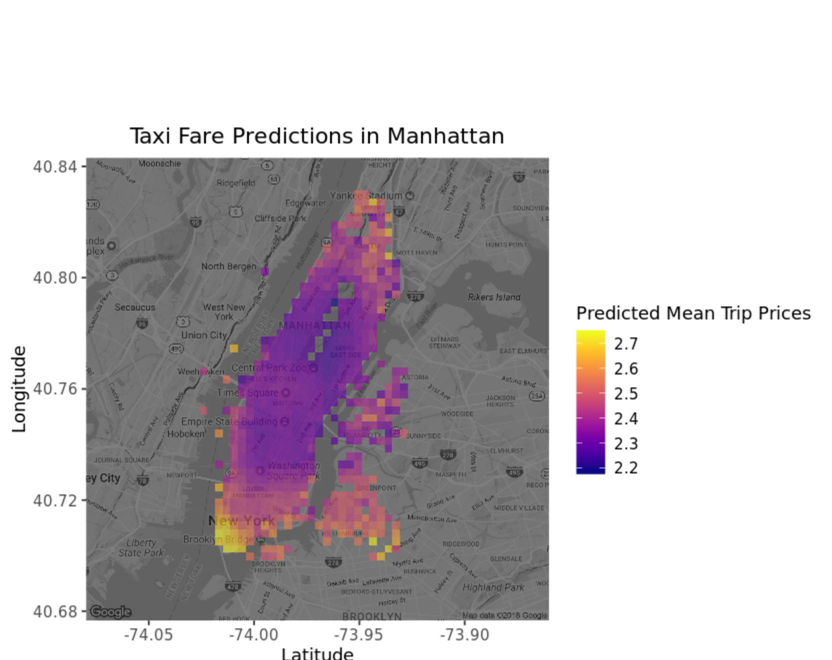
9. Plotting the predicted fare

In the output of `fitted_forest` we should be able to see the `Mean of squared residuals`, that is, the average of the squared errors the model makes. If we check the `summary` of `fitted_tree` we will find `Residual mean deviance` which is the same number. If we compare these numbers, we will see that `fitted_forest` has a slightly lower error. Neither predictive model is *that* good, in statistical terms, they explain only about 3% of the variance.

Now, let's take a look at the predictions of `fitted_forest` projected back onto Manhattan.

In [14]:
```
# Extracting the prediction from fitted_forest
taxi$pred_total <- fitted_forest$predicted

# Plotting the predicted mean trip prices from according to the random forest
ggmap(manhattan, darken = 0.5) +
   scale_fill_viridis(option = 'plasma') +
    stat_summary_2d(data = taxi, fun = "mean", aes(x = long, y = lat, z = pred_total) , bins = 60, alpha = 0.6) +
    xlab("Latitude") +
    ylab("Longitude") +
    labs(fill = "Predicted Mean Trip Prices")+
    ggtitle("Taxi Fare Predictions in Manhattan") + theme(plot.title = element_text(hjust = 0.5))
```



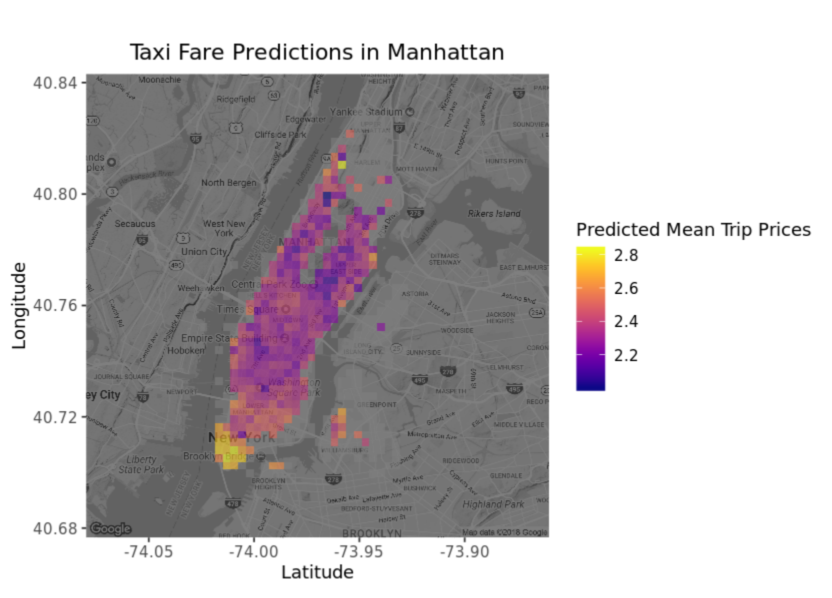
10. Plotting the actual fare

Looking at the map with the predicted fares we see that fares in downtown Manhattan are predicted to be high, while midtown is lower. Note that this map only shows the prediction as a function of `lat` and `long`, but we could also plot the predictions over time, or a combination of time and space.

For now, we will compare the map with the predicted fares with a new map showing the mean fares according to the data.

In [16]:
```
# Function that returns the mean *if* there are 15 or more datapoints
mean_if_enough_data <- function(x) {
    ifelse( length(x) >= 15, mean(x), NA)
}

# Plotting the mean trip prices from the data
ggmap(manhattan, darken = 0.5) +
   scale_fill_viridis(option = 'plasma') +
    stat_summary_2d(data = taxi, fun = mean_if_enough_data, aes(x = long, y = lat, z = total) , bins = 60, alpha =
0.6) +
    xlab("Latitude") +
    ylab("Longitude") +
    labs(fill = "Predicted Mean Trip Prices")+
      ggtitle("Taxi Fare Predictions in Manhattan") + theme(plot.title = element_text(hjust = 0.5))
```



11. Conclusion

So it looks like the random forest model captured some of the patterns in our data. Based on this model, people in downtown Manhattan are likely to spend more on their trips.