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.

But this is the age of business intelligence and analytics! Even taxi drivers can stand to benefit from some careful investigation of the data, guiding them to maximize their profits. 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.

Let's start by taking a look at the data!

In [16]:
```
# 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)
```

```
Rows: 49999 Columns: 7
── Column specification ────────────────────────────────────────────────────────
Delimiter: ","
chr  (1): medallion
dbl  (5): pickup_longitude, pickup_latitude, trip_time_in_secs, fare_amount,...
dttm (1): pickup_datetime

ℹ Use `spec()` to retrieve the full column specification for this data.
ℹ Specify the column types or set `show_col_types = FALSE` to quiet this message.
```

A tibble: 6 × 7

medallion	pickup_datetime	pickup_longitude	pickup_latitude
<chr>	<dttm>	<dbl>	<dbl>
4D24F4D8EF35878595044A52B098DFD2	2013-01-13 10:23:00	-73.94646	40.77273
A49C37EB966E7B05E69523D1CB7BE303	2013-01-13 04:52:00	-73.99827	40.74041
1E4B72A8E623888F53A9693C364AC05A	2013-01-13 10:47:00	-73.95346	40.77586
F7E4E9439C46B8AD5B16AB9F1B3279D7	2013-01-13 11:14:00	-73.98137	40.72473
A9DC75D59E0EA27E1ED328E8BE8CD828	2013-01-13 11:24:00	-73.96800	40.76000
19BF1BB516C4E992EA3FBAEDA73D6262	2013-01-13 10:51:00	-73.98502	40.76341

2. Cleaning the taxi data

As you 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.

Cleaning data is a large part of any data scientist's daily work. It may not seem glamorous, but it makes the difference between a successful model and a failure. The `taxi` dataset needs a bit of polishing before we're ready to use it.

In [17]:
```
taxi <- taxi %>%
   rename(long = pickup_longitude, lat = pickup_latitude) %>%
   filter(fare_amount > 0 | tip_amount > 0) %>%
   mutate(total = log(fare_amount + tip_amount))

taxi
```

A tibble: 49998 × 8

medallion	pickup_datetime	long	lat	trip_time_in_
<chr>	<dttm>	<dbl>	<dbl>	<d
4D24F4D8EF35878595044A52B098DFD2	2013-01-13 10:23:00	-73.94646	40.77273	
A49C37EB966E7B05E69523D1CB7BE303	2013-01-13 04:52:00	-73.99827	40.74041	
1E4B72A8E623888F53A9693C364AC05A	2013-01-13 10:47:00	-73.95346	40.77586	
F7E4E9439C46B8AD5B16AB9F1B3279D7	2013-01-13 11:14:00	-73.98137	40.72473	
A9DC75D59E0EA27E1ED328E8BE8CD828	2013-01-13 11:24:00	-73.96800	40.76000	
19BF1BB516C4E992EA3FBAEDA73D6262	2013-01-13 10:51:00	-73.98502	40.76341	
5F2EFC03B544635C9B0E7A4AA4FF9AC3	2013-01-13 12:53:00	-73.97295	40.79527	
8DEB70907D00AA1D7FF5E2683240549B	2013-01-13 07:59:00	-73.96577	40.76530	
E15F7CCB808DD15E0496D830D3DEDECE	2013-01-13 08:09:00	-73.94768	40.77507	
0B3D3D51C78E944F68DC04209E86D5F7	2013-01-13 12:53:00	-73.98457	40.72488	
3C16CFAD2B12F3508F7211C37F8F8B8F	2013-01-13 13:07:00	-73.97068	40.78506	
5888835B75CF97CBE738A58484B12A6D	2013-01-13 12:31:00	-74.00164	40.74414	
58A836AD639867DB949723BA2A941734	2013-01-13 13:37:00	-73.98800	40.74960	
C01368C76C8DEFA6C678CF0D54AE87F0	2013-01-13 13:31:00	-73.99309	40.74755	
F86299BD8DF9B0C90A4E20AD2A44EAF7	2013-01-13 14:31:00	-73.96832	40.78688	
00D07524C1482FF5A3CB0932BE29003E	2013-01-13 13:02:00	-73.95131	40.81007	1
2232F3F8B6124B1947AEF62509A97DAE	2013-01-13 16:16:00	-73.98534	40.72374	
85045374BBBFA1376CE5D64DED56B010	2013-01-13 13:14:00	-73.91682	40.76967	1
70F5983D94E1BA0B217E9F8F447D4382	2013-01-13 16:30:00	-73.97288	40.78684	
3AC3AB85F6E59CB391253FA638B854AA	2013-01-13 13:24:00	-73.98768	40.71979	

medallion	pickup_datetime	long	lat	trip_time_in_s
<chr>	<dttm>	<dbl>	<dbl>	<d
60DC999A38D953EE86AB81C23C9480E8	2013-01-13 16:03:00	-73.98846	40.73724	
084ACA045413975E5FB960ECBD9C6588	2013-01-13 15:58:00	-73.96040	40.76165	
AC496252AF3119662DEDBFD24A70E83B	2013-01-13 16:08:00	-73.98025	40.75132	
F2773B3D7BC9C2A18B942644F43DD33D	2013-01-13 14:45:00	-74.01029	40.71158	
9E800FF6BCF49308A6BC0678ED27499D	2013-01-13 17:29:00	-73.97753	40.74235	
ED9B6E969C35E16314E6863A95D83B5F	2013-01-13 17:48:00	-73.99191	40.73557	
4FBA078630428EFA5EEEF2E67A293464	2013-01-13 15:14:00	-73.87447	40.77406	1
FDD65AC3468B2A12AF0D3577D64FBF76	2013-01-13 19:15:00	-73.99110	40.69207	1
135A8EDC6EBD6F4397B7A6D281C75AB0	2013-01-13 16:59:00	-73.97753	40.76359	
131949A57D5717A3E7112404CCDFF27B	2013-01-13 17:36:00	-73.97504	40.75832	
⋮	⋮	⋮	⋮	
0CE65EC42F3ABEB0BCC08F274703DDE7	2013-12-03 11:58:00	-73.98287	40.74546	
126A5559920CCC1F4AA7A7DFB137D328	2013-12-05 22:12:00	-74.00825	40.73770	
221F94E1FB935807635C1045866796CC	2013-12-03 12:31:00	-73.95893	40.77771	
0C4726D4E2AF94BF8FE2D23EFDA20917	2013-12-05 22:03:00	-73.99159	40.74441	1
A6DB36B570BD59E08FBE76086C5EE662	2013-12-03 13:36:00	-73.98994	40.74703	1
60C3AD7179183044E91FF3B9B90C1CAA	2013-12-03 11:50:00	-73.99038	40.72965	
A910CF5A84FFE5F10B8CCBC06CC5F944	2013-12-05 22:42:00	-73.99249	40.74290	
037CB433AB5895649F7A9EF37F767EF1	2013-12-05 22:49:00	-73.96532	40.77145	
798452AE1E4F97CE2B491F9590861FEA	2013-12-03 14:26:00	-73.95112	40.78284	

medallion	pickup_datetime	long	lat	trip_time_in_s
<chr>	<dttm>	<dbl>	<dbl>	<d
204BAB16D3382C5A5711068B28E624C2	2013-12-03 14:12:00	-73.97968	40.76131	
F3BA458FFB70903630ABF3332CB983F1	2013-12-03 14:39:00	-73.94083	40.79271	1
5221BCE45F9FD34B709EB0882885B7AB	2013-12-05 23:21:00	-73.94922	40.80278	1
2B3EFA3719EA953A656D5041750787EB	2013-12-03 13:43:00	-73.78947	40.64633	3
B65ECEC404246E0A0370540B5FE24AAB	2013-12-05 23:40:00	-73.98241	40.75505	2
8220D3BEF8BCF3C5DB50C66F32D9AB61	2013-12-05 23:50:00	-73.98628	40.74044	
87CDD10EC56CB55A6F92D872A16AECCD	2013-12-03 14:32:00	-73.98254	40.76761	2
758FE3823459A49A24051D260415E866	2013-12-05 22:42:00	-73.98413	40.72923	1
554389035D554151A222468199443C39	2013-12-06 00:35:00	-73.94059	40.71187	
1AABA654E5ABF4F25333847479904504	2013-12-03 13:55:00	-73.99451	40.74100	
0C4CEF33F6F0D06E62988C22B6F53983	2013-12-05 22:46:00	-74.01471	40.71081	1
16183FABEA1B2C53821A44CBFA0C1907	2013-12-03 16:10:00	-73.94863	40.77665	
7D5EBEEF1F35996553AE89392DF504C8	2013-12-06 01:30:00	-73.99052	40.75630	
82BA7115B2866F14CC4364621C71D050	2013-12-03 17:24:00	-73.95631	40.76751	
6E2BE266388543BEBD9DAE67DE2EF745	2013-12-03 17:23:00	-74.00461	40.71924	
162A967C11C5A4839059F1B1C9868C33	2013-12-03 15:56:00	-73.96405	40.77710	
FB84C95C217D345556E3B14EA6D63E5C	2013-12-03 16:46:00	-73.99992	40.71914	
1A507DEE7AD80F346106A3016A388038	2013-12-03 17:06:00	-73.99654	40.76344	
D45DCD8D59D2C02E5630FAC6BF9B9F96	2013-12-06 06:53:00	-73.94276	40.79025	
CCDD7C317BBF35D4585CB9BC4F4299A5	2013-12-03 16:53:00	-73.95743	40.78273	1

medallion	pickup_datetime	long	lat	trip_time_in_
<chr>	<dttm>	<dbl>	<dbl>	<d
90D244D17A03926D69448F687C1424A2	2013-12-03 17:37:00	-73.96688	40.79355	

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 let's focus only on trips initiated there.

In [18]:
```
# 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))

taxi
```

A tibble: 45766 × 8

medallion	pickup_datetime	long	lat	trip_time_in_
<chr>	<dttm>	<dbl>	<dbl>	<d
4D24F4D8EF35878595044A52B098DFD2	2013-01-13 10:23:00	-73.94646	40.77273	
A49C37EB966E7B05E69523D1CB7BE303	2013-01-13 04:52:00	-73.99827	40.74041	
1E4B72A8E623888F53A9693C364AC05A	2013-01-13 10:47:00	-73.95346	40.77586	
F7E4E9439C46B8AD5B16AB9F1B3279D7	2013-01-13 11:14:00	-73.98137	40.72473	
A9DC75D59E0EA27E1ED328E8BE8CD828	2013-01-13 11:24:00	-73.96800	40.76000	
19BF1BB516C4E992EA3FBAEDA73D6262	2013-01-13 10:51:00	-73.98502	40.76341	
5F2EFC03B544635C9B0E7A4AA4FF9AC3	2013-01-13 12:53:00	-73.97295	40.79527	
8DEB70907D00AA1D7FF5E2683240549B	2013-01-13 07:59:00	-73.96577	40.76530	
E15F7CCB808DD15E0496D830D3DEDECE	2013-01-13 08:09:00	-73.94768	40.77507	
0B3D3D51C78E944F68DC04209E86D5F7	2013-01-13 12:53:00	-73.98457	40.72488	
3C16CFAD2B12F3508F7211C37F8F8B8F	2013-01-13 13:07:00	-73.97068	40.78506	
5888835B75CF97CBE738A58484B12A6D	2013-01-13 12:31:00	-74.00164	40.74414	
58A836AD639867DB949723BA2A941734	2013-01-13 13:37:00	-73.98800	40.74960	
C01368C76C8DEFA6C678CF0D54AE87F0	2013-01-13 13:31:00	-73.99309	40.74755	
F86299BD8DF9B0C90A4E20AD2A44EAF7	2013-01-13 14:31:00	-73.96832	40.78688	
00D07524C1482FF5A3CB0932BE29003E	2013-01-13 13:02:00	-73.95131	40.81007	1
2232F3F8B6124B1947AEF62509A97DAE	2013-01-13 16:16:00	-73.98534	40.72374	
70F5983D94E1BA0B217E9F8F447D4382	2013-01-13 16:30:00	-73.97288	40.78684	
3AC3AB85F6E59CB391253FA638B854AA	2013-01-13 13:24:00	-73.98768	40.71979	
60DC999A38D953EE86AB81C23C9480E8	2013-01-13 16:03:00	-73.98846	40.73724	

medallion	pickup_datetime	long	lat	trip_time_in_s
<chr>	<dttm>	<dbl>	<dbl>	<d
084ACA045413975E5FB960ECBD9C6588	2013-01-13 15:58:00	-73.96040	40.76165	
AC496252AF3119662DEDBFD24A70E83B	2013-01-13 16:08:00	-73.98025	40.75132	
F2773B3D7BC9C2A18B942644F43DD33D	2013-01-13 14:45:00	-74.01029	40.71158	
9E800FF6BCF49308A6BC0678ED27499D	2013-01-13 17:29:00	-73.97753	40.74235	
ED9B6E969C35E16314E6863A95D83B5F	2013-01-13 17:48:00	-73.99191	40.73557	
135A8EDC6EBD6F4397B7A6D281C75AB0	2013-01-13 16:59:00	-73.97753	40.76359	
131949A57D5717A3E7112404CCDFF27B	2013-01-13 17:36:00	-73.97504	40.75832	
09EE700858482DA94CCBC192D2208B10	2013-01-13 15:36:00	-73.98092	40.75594	
CDC40A1F194AA8D4CA9EF331A266E1FA	2013-01-13 17:39:00	-73.96726	40.76012	
BA2E28EC4C10FA42A04131313286A591	2013-01-13 18:58:00	-73.97436	40.75417	
⋮	⋮	⋮	⋮	
C8980A6501C7D728020B1FC2FC1B64B1	2013-12-03 12:41:00	-73.98369	40.75583	
0CE65EC42F3ABEB0BCC08F274703DDE7	2013-12-03 11:58:00	-73.98287	40.74546	
126A5559920CCC1F4AA7A7DFB137D328	2013-12-05 22:12:00	-74.00825	40.73770	
221F94E1FB935807635C1045866796CC	2013-12-03 12:31:00	-73.95893	40.77771	
0C4726D4E2AF94BF8FE2D23EFDA20917	2013-12-05 22:03:00	-73.99159	40.74441	1
A6DB36B570BD59E08FBE76086C5EE662	2013-12-03 13:36:00	-73.98994	40.74703	1
60C3AD7179183044E91FF3B9B90C1CAA	2013-12-03 11:50:00	-73.99038	40.72965	
A910CF5A84FFE5F10B8CCBC06CC5F944	2013-12-05 22:42:00	-73.99249	40.74290	
037CB433AB5895649F7A9EF37F767EF1	2013-12-05 22:49:00	-73.96532	40.77145	

medallion	pickup_datetime	long	lat	trip_time_in_s
<chr>	<dttm>	<dbl>	<dbl>	<d
798452AE1E4F97CE2B491F9590861FEA	2013-12-03 14:26:00	-73.95112	40.78284	
204BAB16D3382C5A5711068B28E624C2	2013-12-03 14:12:00	-73.97968	40.76131	
F3BA458FFB70903630ABF3332CB983F1	2013-12-03 14:39:00	-73.94083	40.79271	1
5221BCE45F9FD34B709EB0882885B7AB	2013-12-05 23:21:00	-73.94922	40.80278	1
B65ECEC404246E0A0370540B5FE24AAB	2013-12-05 23:40:00	-73.98241	40.75505	2
8220D3BEF8BCF3C5DB50C66F32D9AB61	2013-12-05 23:50:00	-73.98628	40.74044	
87CDD10EC56CB55A6F92D872A16AECCD	2013-12-03 14:32:00	-73.98254	40.76761	2
758FE3823459A49A24051D260415E866	2013-12-05 22:42:00	-73.98413	40.72923	1
554389035D554151A222468199443C39	2013-12-06 00:35:00	-73.94059	40.71187	
1AABA654E5ABF4F25333847479904504	2013-12-03 13:55:00	-73.99451	40.74100	
0C4CEF33F6F0D06E62988C22B6F53983	2013-12-05 22:46:00	-74.01471	40.71081	1
16183FABEA1B2C53821A44CBFA0C1907	2013-12-03 16:10:00	-73.94863	40.77665	
7D5EBEEF1F35996553AE89392DF504C8	2013-12-06 01:30:00	-73.99052	40.75630	
82BA7115B2866F14CC4364621C71D050	2013-12-03 17:24:00	-73.95631	40.76751	
6E2BE266388543BEBD9DAE67DE2EF745	2013-12-03 17:23:00	-74.00461	40.71924	
162A967C11C5A4839059F1B1C9868C33	2013-12-03 15:56:00	-73.96405	40.77710	
FB84C95C217D345556E3B14EA6D63E5C	2013-12-03 16:46:00	-73.99992	40.71914	
1A507DEE7AD80F346106A3016A388038	2013-12-03 17:06:00	-73.99654	40.76344	
D45DCD8D59D2C02E5630FAC6BF9B9F96	2013-12-06 06:53:00	-73.94276	40.79025	
CCDD7C317BBF35D4585CB9BC4F4299A5	2013-12-03 16:53:00	-73.95743	40.78273	1

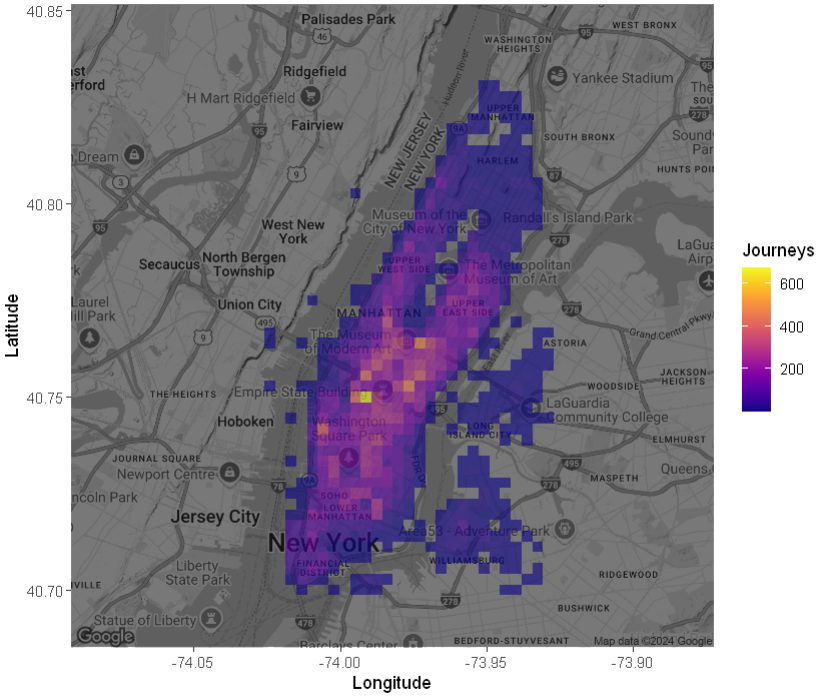
medallion	pickup_datetime	long	lat	trip_time_in_
<chr>	<dttm>	<dbl>	<dbl>	<d
90D244D17A03926D69448F687C1424A2	2013-12-03 17:37:00	-73.96688	40.79355	

4. Where does the journey begin?

It's time to draw a map! We're going to use the excellent `ggmap` package together with `ggplot2` to visualize where in Manhattan people tend to start their taxi journeys.

In [19]:
```
# Loading in ggmap and viridis for nice colors
library(ggmap)
library(viridis)

register_google(key = "AIzaSyCYm9kyz4Cjbt9JjNxcbUXZHNQvItQkj9c")

# 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) +
    labs(x = 'Longitude', y = 'Latitude', fill = 'Journeys')
```

```
ℹ <https://maps.googleapis.com/maps/api/staticmap?center=manhattan&zoom=12&size=640x640&scale=2&maptype=terrain&language=en-EN&key=xxx>

ℹ <https://maps.googleapis.com/maps/api/geocode/json?address=manhattan&key=xxx>
```

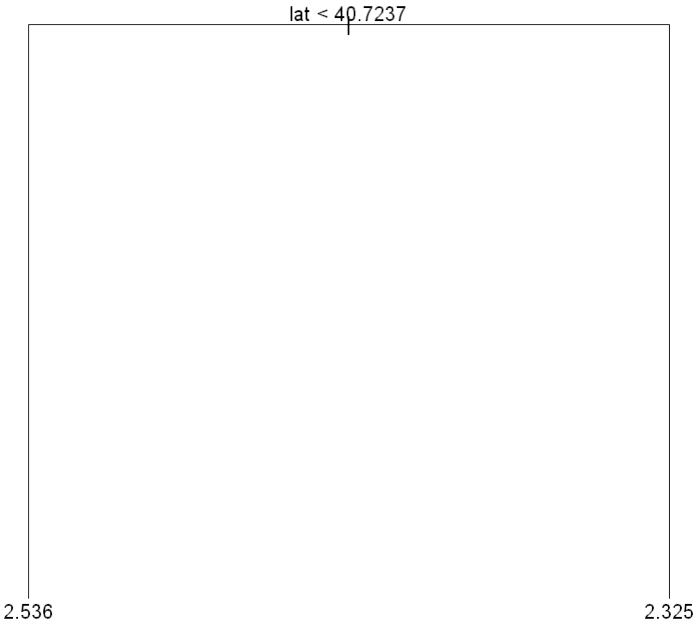


5. Predicting taxi fares using a tree

The map from the previous task showed that the journeys are highly concentrated in the business and tourist areas. 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 [20]:
```
# 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. It's time. 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. But that's it. It didn't even include `long` as `tree` deemed that it didn't improve the predictions. Taxi drivers will need more information than this and any driver paying for your data-driven insights would be disappointed with that. As we know from Robert de Niro, it's best not to upset New York taxi drivers.

Let's start by adding some more predictors related to the *time* the taxi trip was made.

In [21]:
```
# 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))

taxi
```

A tibble: 45766 × 11

medallion	pickup_datetime	long	lat	trip_time_in_
<chr>	<dttm>	<dbl>	<dbl>	<d
4D24F4D8EF35878595044A52B098DFD2	2013-01-13 10:23:00	-73.94646	40.77273	
A49C37EB966E7B05E69523D1CB7BE303	2013-01-13 04:52:00	-73.99827	40.74041	
1E4B72A8E623888F53A9693C364AC05A	2013-01-13 10:47:00	-73.95346	40.77586	
F7E4E9439C46B8AD5B16AB9F1B3279D7	2013-01-13 11:14:00	-73.98137	40.72473	
A9DC75D59E0EA27E1ED328E8BE8CD828	2013-01-13 11:24:00	-73.96800	40.76000	
19BF1BB516C4E992EA3FBAEDA73D6262	2013-01-13 10:51:00	-73.98502	40.76341	
5F2EFC03B544635C9B0E7A4AA4FF9AC3	2013-01-13 12:53:00	-73.97295	40.79527	
8DEB70907D00AA1D7FF5E2683240549B	2013-01-13 07:59:00	-73.96577	40.76530	
E15F7CCB808DD15E0496D830D3DEDECE	2013-01-13 08:09:00	-73.94768	40.77507	
0B3D3D51C78E944F68DC04209E86D5F7	2013-01-13 12:53:00	-73.98457	40.72488	
3C16CFAD2B12F3508F7211C37F8F8B8F	2013-01-13 13:07:00	-73.97068	40.78506	
5888835B75CF97CBE738A58484B12A6D	2013-01-13 12:31:00	-74.00164	40.74414	
58A836AD639867DB949723BA2A941734	2013-01-13 13:37:00	-73.98800	40.74960	
C01368C76C8DEFA6C678CF0D54AE87F0	2013-01-13 13:31:00	-73.99309	40.74755	
F86299BD8DF9B0C90A4E20AD2A44EAF7	2013-01-13 14:31:00	-73.96832	40.78688	
00D07524C1482FF5A3CB0932BE29003E	2013-01-13 13:02:00	-73.95131	40.81007	1
2232F3F8B6124B1947AEF62509A97DAE	2013-01-13 16:16:00	-73.98534	40.72374	
70F5983D94E1BA0B217E9F8F447D4382	2013-01-13 16:30:00	-73.97288	40.78684	
3AC3AB85F6E59CB391253FA638B854AA	2013-01-13 13:24:00	-73.98768	40.71979	
60DC999A38D953EE86AB81C23C9480E8	2013-01-13 16:03:00	-73.98846	40.73724	

medallion	pickup_datetime	long	lat	trip_time_in_s
<chr>	<dttm>	<dbl>	<dbl>	<d
084ACA045413975E5FB960ECBD9C6588	2013-01-13 15:58:00	-73.96040	40.76165	
AC496252AF3119662DEDBFD24A70E83B	2013-01-13 16:08:00	-73.98025	40.75132	
F2773B3D7BC9C2A18B942644F43DD33D	2013-01-13 14:45:00	-74.01029	40.71158	
9E800FF6BCF49308A6BC0678ED27499D	2013-01-13 17:29:00	-73.97753	40.74235	
ED9B6E969C35E16314E6863A95D83B5F	2013-01-13 17:48:00	-73.99191	40.73557	
135A8EDC6EBD6F4397B7A6D281C75AB0	2013-01-13 16:59:00	-73.97753	40.76359	
131949A57D5717A3E7112404CCDFF27B	2013-01-13 17:36:00	-73.97504	40.75832	
09EE700858482DA94CCBC192D2208B10	2013-01-13 15:36:00	-73.98092	40.75594	
CDC40A1F194AA8D4CA9EF331A266E1FA	2013-01-13 17:39:00	-73.96726	40.76012	
BA2E28EC4C10FA42A04131313286A591	2013-01-13 18:58:00	-73.97436	40.75417	
⋮	⋮	⋮	⋮	
C8980A6501C7D728020B1FC2FC1B64B1	2013-12-03 12:41:00	-73.98369	40.75583	
0CE65EC42F3ABEB0BCC08F274703DDE7	2013-12-03 11:58:00	-73.98287	40.74546	
126A5559920CCC1F4AA7A7DFB137D328	2013-12-05 22:12:00	-74.00825	40.73770	
221F94E1FB935807635C1045866796CC	2013-12-03 12:31:00	-73.95893	40.77771	
0C4726D4E2AF94BF8FE2D23EFDA20917	2013-12-05 22:03:00	-73.99159	40.74441	1
A6DB36B570BD59E08FBE76086C5EE662	2013-12-03 13:36:00	-73.98994	40.74703	1
60C3AD7179183044E91FF3B9B90C1CAA	2013-12-03 11:50:00	-73.99038	40.72965	
A910CF5A84FFE5F10B8CCBC06CC5F944	2013-12-05 22:42:00	-73.99249	40.74290	
037CB433AB5895649F7A9EF37F767EF1	2013-12-05 22:49:00	-73.96532	40.77145	

medallion	pickup_datetime	long	lat	trip_time_in_s
<chr>	<dttm>	<dbl>	<dbl>	<d
798452AE1E4F97CE2B491F9590861FEA	2013-12-03 14:26:00	-73.95112	40.78284	
204BAB16D3382C5A5711068B28E624C2	2013-12-03 14:12:00	-73.97968	40.76131	
F3BA458FFB70903630ABF3332CB983F1	2013-12-03 14:39:00	-73.94083	40.79271	1
5221BCE45F9FD34B709EB0882885B7AB	2013-12-05 23:21:00	-73.94922	40.80278	1
B65ECEC404246E0A0370540B5FE24AAB	2013-12-05 23:40:00	-73.98241	40.75505	2
8220D3BEF8BCF3C5DB50C66F32D9AB61	2013-12-05 23:50:00	-73.98628	40.74044	
87CDD10EC56CB55A6F92D872A16AECCD	2013-12-03 14:32:00	-73.98254	40.76761	2
758FE3823459A49A24051D260415E866	2013-12-05 22:42:00	-73.98413	40.72923	1
554389035D554151A222468199443C39	2013-12-06 00:35:00	-73.94059	40.71187	
1AABA654E5ABF4F25333847479904504	2013-12-03 13:55:00	-73.99451	40.74100	
0C4CEF33F6F0D06E62988C22B6F53983	2013-12-05 22:46:00	-74.01471	40.71081	1
16183FABEA1B2C53821A44CBFA0C1907	2013-12-03 16:10:00	-73.94863	40.77665	
7D5EBEEF1F35996553AE89392DF504C8	2013-12-06 01:30:00	-73.99052	40.75630	
82BA7115B2866F14CC4364621C71D050	2013-12-03 17:24:00	-73.95631	40.76751	
6E2BE266388543BEBD9DAE67DE2EF745	2013-12-03 17:23:00	-74.00461	40.71924	
162A967C11C5A4839059F1B1C9868C33	2013-12-03 15:56:00	-73.96405	40.77710	
FB84C95C217D345556E3B14EA6D63E5C	2013-12-03 16:46:00	-73.99992	40.71914	
1A507DEE7AD80F346106A3016A388038	2013-12-03 17:06:00	-73.99654	40.76344	
D45DCD8D59D2C02E5630FAC6BF9B9F96	2013-12-06 06:53:00	-73.94276	40.79025	
CCDD7C317BBF35D4585CB9BC4F4299A5	2013-12-03 16:53:00	-73.95743	40.78273	1

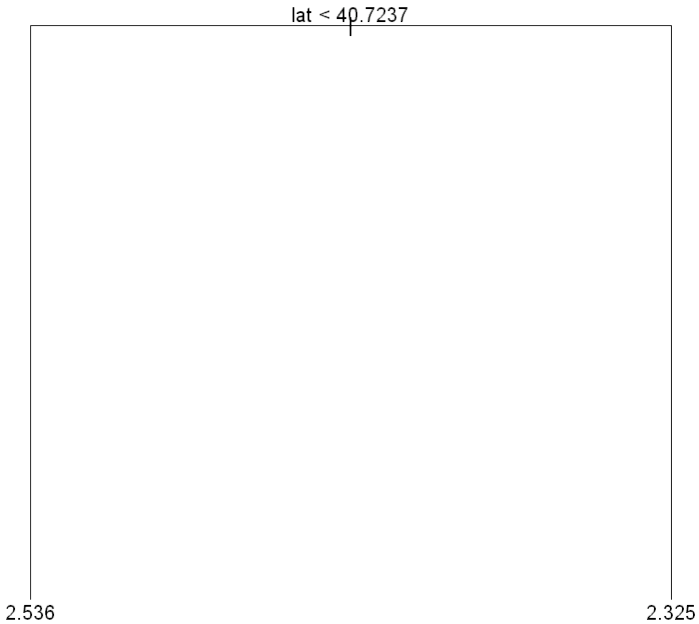
medallion	pickup_datetime	long	lat	trip_time_in_
<chr>	<dttm>	<dbl>	<dbl>	<d
90D244D17A03926D69448F687C1424A2	2013-12-03 17:37:00	-73.96688	40.79355	

7. One more tree!

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

In [22]:
```
# Fitting a tree to lat and long
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 [23]:
```
# 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
fitted_forest
```

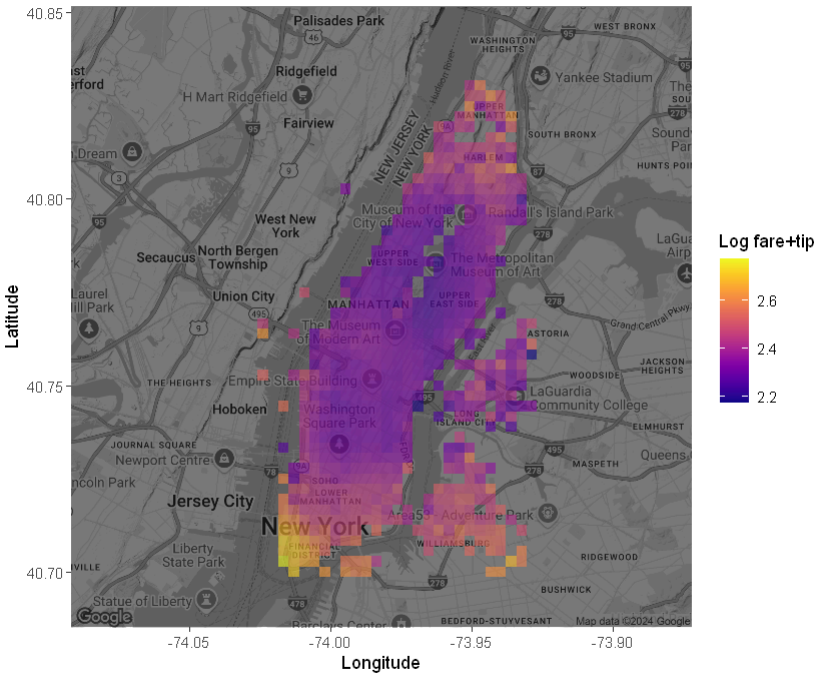
```
Call:
 randomForest(formula = total ~ lat + long + hour + wday + month,      data = tax
i, ntree = 80, sampsize = 10000)
               Type of random forest: regression
                     Number of trees: 80
No. of variables tried at each split: 1

          Mean of squared residuals: 0.3000435
                    % Var explained: 2.69
```

9. Plotting the predicted fare

In the output of `fitted_forest` you should see the `Mean of squared residuals`, that is, the average of the squared errors the model makes. If you scroll up and check the `summary` of `fitted_tree` you'll find `Residual mean deviance` which is the same number. If you compare these numbers, you'll 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 [24]:
```
# Extracting the prediction from forest_fit
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, aes(x = long, y = lat, z = pred_total),
                    fun = mean, alpha = 0.6, bins = 60) +
    labs(x = 'Longitude', y = 'Latitude', fill = 'Log fare+tip')
```

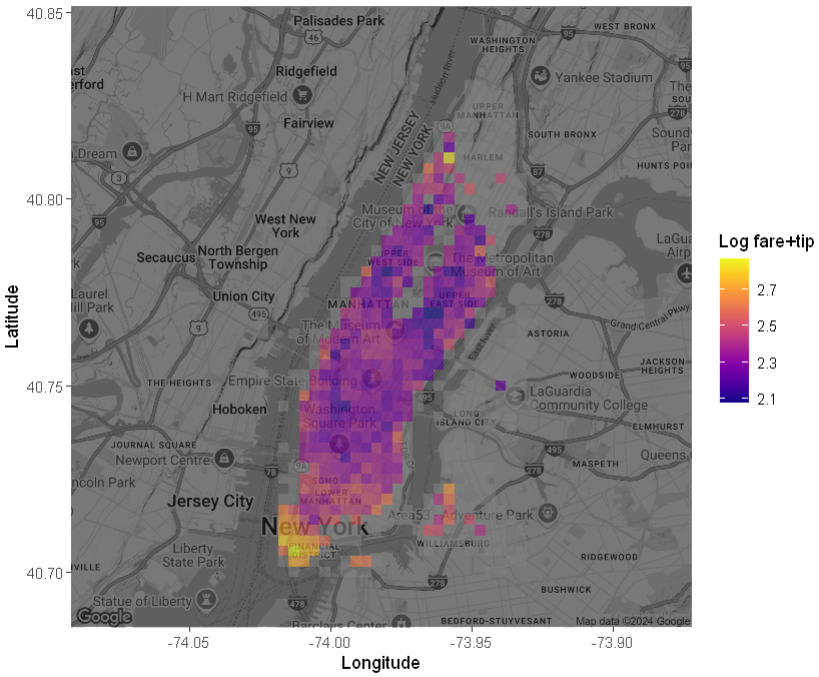


10. Plotting the actual fare

Looking at the map with the predicted fares we see that fares in downtown Manhattan are predicted to be high, while midtown is lower. 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, but we'll leave that for another time.

For now, let's compare the map with the predicted fares with a new map showing the mean fares according to the data.

In [25]:
```
# 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) +
    stat_summary_2d(data=taxi, aes(x = long, y = lat, z = total),
                    fun = mean_if_enough_data,
                    alpha = 0.6, bins = 60) +
  scale_fill_viridis(option = 'plasma') +
  labs(x = 'Longitude', y = 'Latitude', fill = 'Log fare+tip')
```



11. Where do people spend the most?

So it looks like the random forest model captured some of the patterns in our data. At this point in the analysis, there are many more things we could do that we haven't done. We could add more predictors if we have the data. We could try to fine-tune the parameters of `randomForest`. And we should definitely test the model on a hold-out test dataset. But for now, let's be happy with what we have achieved!

So, if you are a taxi driver in NYC, where in Manhattan would you expect people to spend the most on a taxi ride?

In [26]:
```
# Where are people spending the most on their taxi trips?
spends_most_on_trips <- "downtown" # "uptown" or "downtown"
```