03 - Analisi e Visualizzazioni

Data Science case study for a Microsoft 213x Project NYC Taxi BigData Analysis
Lorenzo Negri, April 2018
Applications used: Microsoft R Open, Visual Studio, RevoScaleR libraries

Esaminazione

Oltre a chiederci che i dati abbiano un senso logico, spesso è una buona idea controllare anche che possano essere usati per aggiungere valore al business. Ciò può anche aiutarci a rilevare determinati errori, ad esempio dati errati o attribuiti all'insieme di funzioni errate, che se non vengono rilevati, possono avere un impatto negativo sull'analisi generale. Eseguo quindi delle `rxSummary`:

```
system.time(
rxs_all <- rxSummary( ~ ., nyc_xdf)
)
```

```
Rows Processed: 69406520
   user  system elapsed
   0.05    0.02   85.16
```

```
head(rxs_all$sDataFrame)
```

```
                  Name       Mean      StdDev           Min           Max ValidObs
1             VendorID         NA          NA            NA            NA 69406520
2  tpep_pickup_datetime         NA          NA            NA            NA        0
3 tpep_dropoff_datetime         NA          NA            NA            NA        0
4       passenger_count   1.660674    1.310478        0.0000        9.0000 69406520
5         trip_distance   4.850022 4044.503422 -3390583.8000 19072628.8000 69406520
6      pickup_longitude -72.920469    8.763351     -165.0819      118.4089 69406520
  MissingObs
1          0
2          0
3          0
4          0
5          0
6          0
```

```
nhoods_by_borough <- rxCrossTabs( ~ pickup_nhood:pickup_borough, nyc_xdf)

nhoods_by_borough <- nhoods_by_borough$counts[[1]]

nhoods_by_borough <- as.data.frame(nhoods_by_borough)


# ottenere I quartieri

lnbs <- lapply(names(nhoods_by_borough), function(vv) subset(nhoods_by_borough, nhoods_by_borough[ , vv] > 0,
select = vv, drop = FALSE))

lapply(lnbs, head)
```

```
[[1]]

[1] Albany

<0 rows> (or 0-length row.names)


[[2]]

[1] Buffalo

<0 rows> (or 0-length row.names)


[[3]]

             New York City-Bronx

Baychester                   125

Bedford Park                1413

City Island                   52

Country Club                 354

Eastchester                   98

Fordham                     1243


[[4]]

                   New York City-Brooklyn

Bay Ridge                            3378

Bedford-Stuyvesant                  54269

Bensonhurst                          1159

Boerum Hill                         76404

Borough Park                         8762

Brownsville                          2757


[[5]]

              New York City-Manhattan

Battery Park                   643283

Carnegie Hill                  807204

Central Park                   936840

Chelsea                       4599098

Chinatown                      211229

Clinton                       2050545
```

```
[[6]]
                         New York City-Queens
Astoria-Long Island City               303231
Auburndale                                464
Clearview                                 152
College Point                               1
Corona                                   1496
Douglastown-Little Neck                   937

[[7]]
                            New York City-Staten Island
Annandale                                             6
Ardon Heights                                        22
Bloomfield-Chelsea-Travis                            26
Charlestown-Richmond Valley                           7
Clifton                                             525
Ettingville                                          13

[[8]]
[1] Rochester
<0 rows> (or 0-length row.names)

[[9]]
[1] Syracuse
<0 rows> (or 0-length row.names)
```

Siccome la maggior parte dei viaggi in taxi si svolge a Manhattan, focalizziamo la nostra attenzione su Manhattan e ignoriamo gli altri quattro quartieri. A tale scopo, creo due nuove colonne chiamate *pickup_nb* e *dropoff_nb* basate sulle colonne originali *pickup_nhood* e *dropoff_nhood*, tranne per il fatto che i loro livelli di fattore sono limitati ai quartieri di Manhattan (qualsiasi altro livello fattore verrà sostituito con un NA).

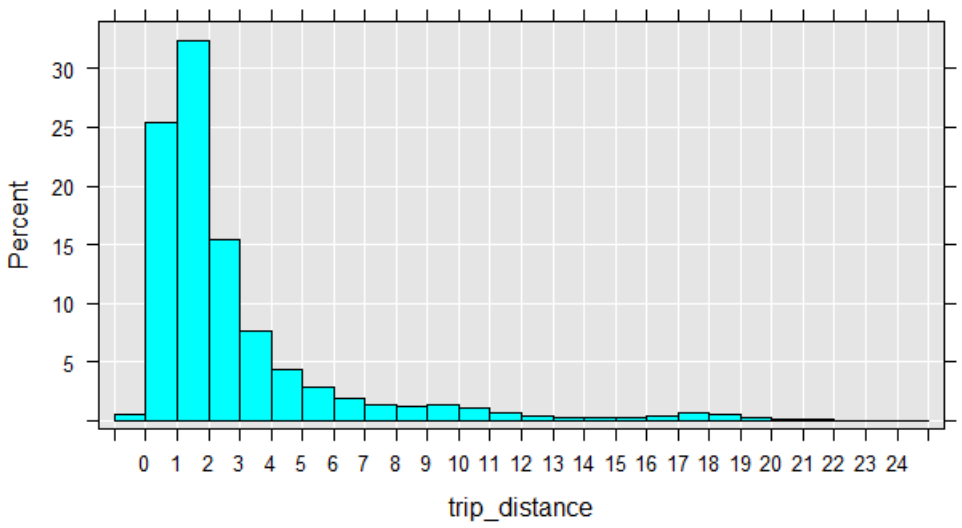
```
manhattan_nhoods <- rownames(nhoods_by_borough)
                    [nhoods_by_borough$`New York City-Manhattan` > 0]


refactor_columns <- function(dataList) {
  dataList$pickup_nb = factor(dataList$pickup_nhood, levels = nhoods_levels)
  dataList$dropoff_nb = factor(dataList$dropoff_nhood, levels = nhoods_levels)
  dataList
}


rxDataStep(nyc_xdf, nyc_xdf,
```

```
            transformFunc = refactor_columns,
            transformObjects = list(nhoods_levels = manhattan_nhoods),
            overwrite = TRUE)

rxs_pickdrop <- rxSummary( ~ pickup_nb:dropoff_nb, nyc_xdf)
head(rxs_pickdrop$categorical[[1]])
```

```
Rows Processed: 69406520

      pickup_nb   dropoff_nb Counts
1  Battery Park Battery Park  19876
2 Carnegie Hill Battery Park   2699
3  Central Park Battery Park   3479
4       Chelsea Battery Park  61024
5     Chinatown Battery Park   3813
6       Clinton Battery Park  23962
```

Osservando i risultati ottenuti finora e le istantanee dei dati, cerchiamo ora di identificare anomalie o outliers. Ad esempio, possiamo tracciare un istogramma di *trip_distance* e notare che quasi tutti i viaggi hanno percorso una distanza inferiore alle 20 miglia, con la grande maggioranza a meno di 5 miglia.

```
rxHistogram( ~ trip_distance,
               nyc_xdf,
               startVal = 0,
               endVal = 25,
               histType = "Percent",
               numBreaks = 20)
```



C'è un secondo picco intorno alle corse taxi di 16 e 20 miglia, che vale la pena esaminare ulteriormente. Possiamo verificare ciò osservando in quali quartieri i passeggeri viaggiano da e per.

```
rxs <- rxSummary( ~ pickup_nhood:dropoff_nhood,
                    nyc_xdf,
                    rowSelection = (trip_distance > 15 & trip_distance < 22))
head(arrange(rxs$categorical[[1]], desc(Counts)), 10)
```

```
      pickup_nhood            dropoff_nhood Counts
1          Midtown Gravesend-Sheepshead Bay   2517
2   Upper East Side Gravesend-Sheepshead Bay   1090
3          Midtown  Douglastown-Little Neck   1013
4          Midtown                  Midtown    978
5  Garment District Gravesend-Sheepshead Bay    911
6          Midtown              Bensonhurst    878
7          Gramercy Gravesend-Sheepshead Bay    784
8           Jamaica          Upper West Side    775
9           Chelsea Gravesend-Sheepshead Bay    729
10          Midtown               Bay Ridge    687
```

Come possiamo vedere, *Gravesend-Sheepshead Bay* appare spesso come destinazione ma non come punto di partenza. Possiamo anche vedere viaggi da *Jamaica*, che è il quartiere più vicino all'aeroporto JFK.

Uso `rxDataStep` e il suo `rowSelection` per estrarre tutti i punti di dati che sono valori anomali secondo certi criteri. Con `outFile`, emettiamo il set di dati risultante in un `data.frame` che chiamiamo `odd_trips`. Infine, se siamo troppo pignoli nei nostri criteri di selezione degli outlier, il `data.frame` potrebbe ancora contenere troppe righe (che potrebbero intasare la memoria e rallentare la produzione di grafici e altri riepiloghi). Quindi creiamo una nuova colonna u e la popoliamo con numeri casuali uniformi tra 0 e 1, e aggiungiamo u <.05 ai nostri criteri rowSelection. Possiamo regolare questo valore per alleggerire il `data.frame` (con una soglia più vicina a 0) o un `data.frame` più grande (con soglia più vicina a 1).

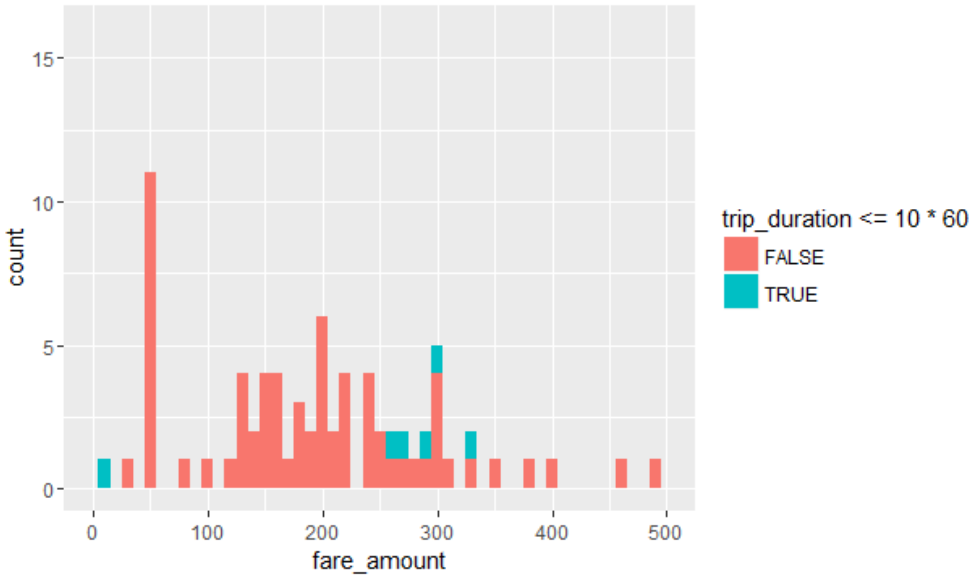
```
odd_trips <- rxDataStep(nyc_xdf, rowSelection = (
  u < .05 & ( # possiamo regolare il valore se vogliamo velocizzare il calcolo
    (trip_distance > 50 | trip_distance <= 0) |
    (passenger_count > 5 | passenger_count == 0) |
    (fare_amount > 5000 | fare_amount <= 0)
)), transforms = list(u = runif(.rxNumRows)))
```

```
print(dim(odd_trips))
```

```
[1] 93750    32
```

Ora limito gli *odd_trips* ai casi in cui è stata percorsa una distanza > 50 miglia, traccio un istogramma dell'importo pagato dal passeggero e coloro in base al fatto che il viaggio sia durato più o meno di 10 minuti.

```
odd_trips %>%
  filter(trip_distance > 50) %>%
  ggplot() -> p
p + geom_histogram(aes(x = fare_amount,
                       fill = trip_duration <= 10*60),
                       binwidth = 10) +
  xlim(0, 500) + coord_fixed(ratio = 25)
```



Come possiamo vedere, la maggior parte dei viaggi che hanno percorso più di 50 miglia non costano nulla o quasi nulla, anche se la maggior parte di questi viaggi ha impiegato 10 minuti o più. Non è chiaro se tali viaggi siano il risultato di errori umani o di errori della telemetria, ma se per esempio questa analisi fosse rivolta alla compagnia che possiede i taxi, questa constatazione meriterebbe ulteriori indagini.

Ora limito il campo concentrando i dati sui viaggi che si sono svolti solo all'interno di Manhattan, e solo a quelli che incontrino criteri "ragionevoli" per un viaggio. Poiché abbiamo aggiunto nuove funzionalità ai dati, possiamo anche eliminare alcune vecchie colonne di variabili in modo che il *dataset* possa essere elaborato più velocemente.

```
input_xdf <- 'yellow_tripdata_2016_manhattan.xdf'
mht_xdf <- RxXdfData(input_xdf)


rxDataStep(nyc_xdf, mht_xdf,
           rowSelection = (
             passenger_count > 0 &
               trip_distance >= 0 & trip_distance < 30 &
               trip_duration > 0 & trip_duration < 60*60*24 &
               str_detect(pickup_borough, 'Manhattan') &
               str_detect(dropoff_borough, 'Manhattan') &
               !is.na(pickup_nb) &
               !is.na(dropoff_nb) &
               fare_amount > 0),
           transformPackages = "stringr",
           varsToDrop = c('extra',
                          'mta_tax',
                          'improvement_surcharge',
                          'total_amount',
                          'pickup_borough',
                          'dropoff_borough',
                          'pickup_nhood',
                          'dropoff_nhood'),
           overwrite = TRUE)
```

E poiché ho limitato i dati, potrebbe essere una buona idea creare un campione dei nuovi dati (come `data.frame`). Il nostro ultimo campione, `nyc_sample_df` non era un buon esempio di dati, siccome prendevamo solo le prime 1000 righe di dati. Questa volta, utilizzo `rxDataStep` per creare un campione casuale dei dati, contenente solo l'1% delle righe dal set originale.

```
mht_sample_df <- rxDataStep(mht_xdf, rowSelection = (u < .01),
                            transforms = list(u = runif(.rxNumRows)))


dim(mht_sample_df)
```

```
Rows Processed: 57493035
WARNING: The number of rows (574832) times the number of columns (24)
exceeds the 'maxRowsByCols' argument (3000000). Rows will be truncated.


> dim(mht_sample_df)
[1] 125000     24
```

Quindi, abbiamo i dati in un formato pulito. Ho creato nuove colonne dati per rendere il set più completo. E ora si può analizzare più seriamente. Possiamo iniziare a farci domande più serie riguardo i dati, possiamo iniziare a guardare più visualizzazioni e vedere che tipo di storie possono raccontarci.

Cerco ora patterns tra i quartieri di partenza e arrivo e altre variabili come l'importo delle tariffe, la distanza percorsa, il traffico e le mance. Per avere una stima del traffico osservando il rapporto tra la durata del viaggio e la distanza del viaggio, supponendo che il traffico sia il motivo più comune per richiedere più tempo di viaggio.

Per l'analisi, uso `rxCube` e `rxCrossTabs` che sono entrambi molto simili a `rxSummary` ma restituiscono meno riepiloghi statistici e quindi funzionano più velocemente. Con `y ~ u: v` come formula, `rxCrossTabs` restituisce i conteggi e le somme e `rxCube` restituisce i conteggi e le medie per la colonna y suddivisi in base a qualsiasi combinazione di colonne `u` e `v`.

Comincio usando `rxCrossTabs` per ottenere somme e conteggi per *trip_distance*, suddivisi in *pickup_nb* e *dropoff_nb*. Posso poi dividere immediatamente le somme per i conteggi e ottenere le medie. Il risultato è una matrice di distanze e può essere inviata alla funzione `seriate` nella libreria di `seriation` per ordinarle in modo che i quartieri più vicini appaiano uno accanto all'altro (all'inizio i quartieri sono ordinati alfabeticamente, che è ciò che R fa di default, a meno che specificato diversamente).

```
rxct <- rxCrossTabs(trip_distance ~ pickup_nb:dropoff_nb, mht_xdf)
res <- rxct$sums$trip_distance / rxct$counts$trip_distance


library(seriation)
res[which(is.nan(res))] <- mean(res, na.rm = TRUE)
nb_order <- seriate(res)
```

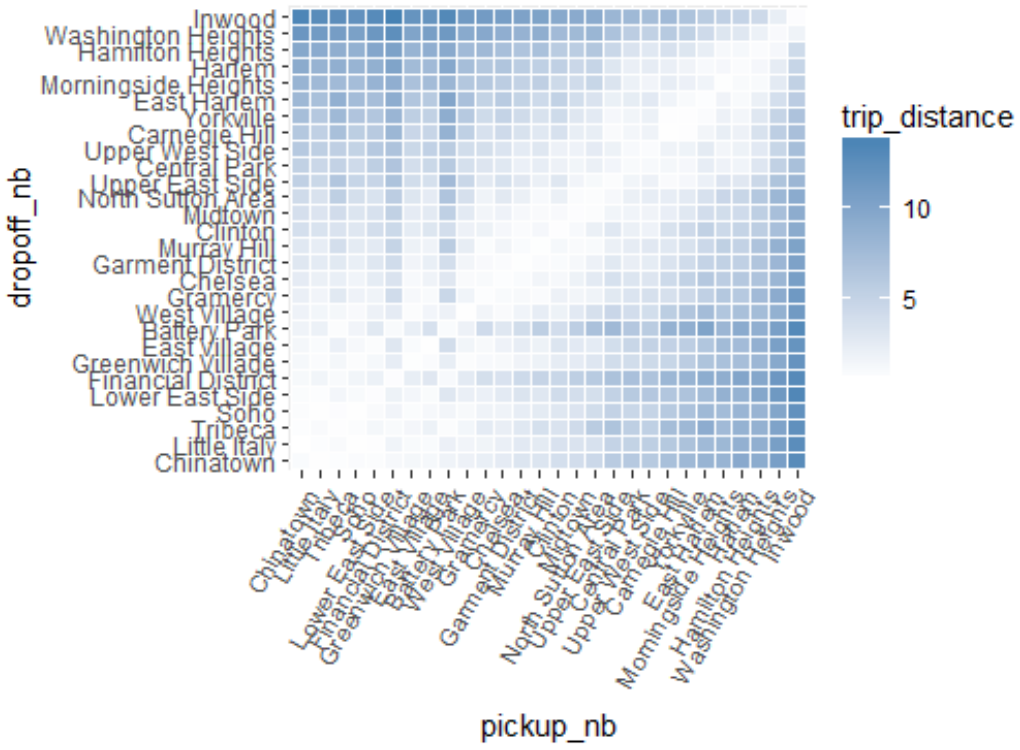
Ora uso `rxCube` per ottenere invece un `data.frame`, dal momento che intendiamo usarlo per il plottaggio con `ggplot2`, che è più facile da codificare utilizzando un `data.frame` lungo come input rispetto a un ampia matrice.

```
rxc1 <- rxCube(trip_distance ~ pickup_nb:dropoff_nb, mht_xdf)
rxc2 <- rxCube(minutes_per_mile ~ pickup_nb:dropoff_nb,
               mht_xdf,
               transforms=list(minutes_per_mile=
                               (trip_duration/60)/trip_distance))
rxc3 <- rxCube(tip_percent ~ pickup_nb:dropoff_nb, mht_xdf)
res <- bind_cols(list(rxc1, rxc2, rxc3))
res <- res[ , c('pickup_nb',
                'dropoff_nb',
                'trip_distance',
```

```
                'minutes_per_mile', 'tip_percent')]
head(res)
```

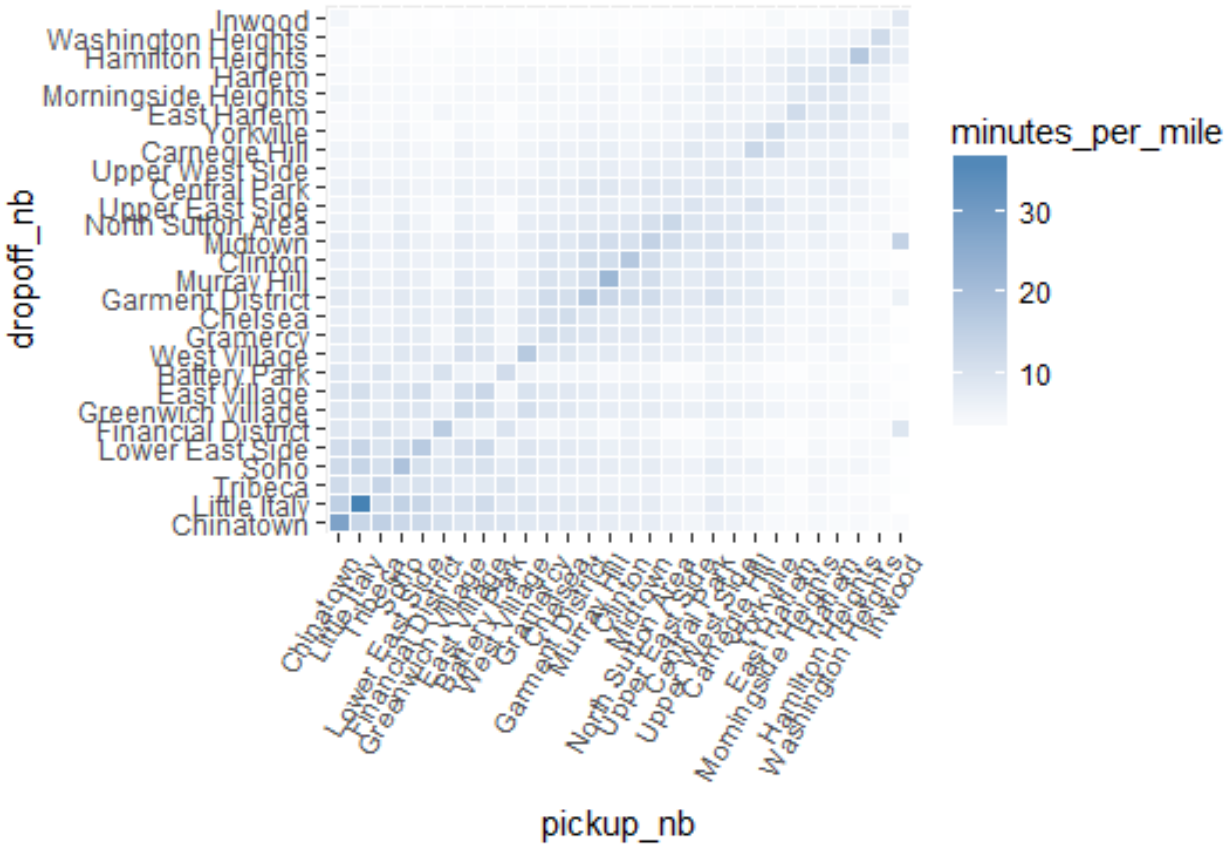
```
      pickup_nb   dropoff_nb trip_distance minutes_per_mile tip_percent
         <fctr>       <fctr>         <dbl>            <dbl>       <dbl>
1  Battery Park Battery Park      1.015857        11.579629   11.394900
2 Carnegie Hill Battery Park      8.570623         3.944350   12.391030
3  Central Park Battery Park      6.277666         5.243241   10.326531
4       Chelsea Battery Park      2.995946         5.169887   11.992151
5     Chinatown Battery Park      1.771597         9.001305   10.292683
6       Clinton Battery Park      3.993806         4.839858    9.794098
```

```
# ordino il grafico in maniera più leggibile
newlevs <- levels(res$pickup_nb)[unlist(nb_order)]
res$pickup_nb <- factor(res$pickup_nb, levels = unique(newlevs))
res$dropoff_nb <- factor(res$dropoff_nb, levels = unique(newlevs))

# visualizzo il grafico a matrice per mostrare alcune tendenze interessanti.
library(ggplot2)
ggplot(res, aes(pickup_nb, dropoff_nb)) +
  geom_tile(aes(fill = trip_distance), colour = "white") +
  theme(axis.text.x = element_text(angle = 60, hjust = 1)) +
  scale_fill_gradient(low = "white", high = "steelblue") +
  coord_fixed(ratio = .9)
```



Poiché le distanze di viaggio rimangono fisse, ma le durate dipendono in gran parte dalla quantità di traffico, possiamo tracciare un grafico per la colonna *minutes_per_mile*, che ci darà un'idea di quali quartieri hanno il maggior traffico tra di loro.

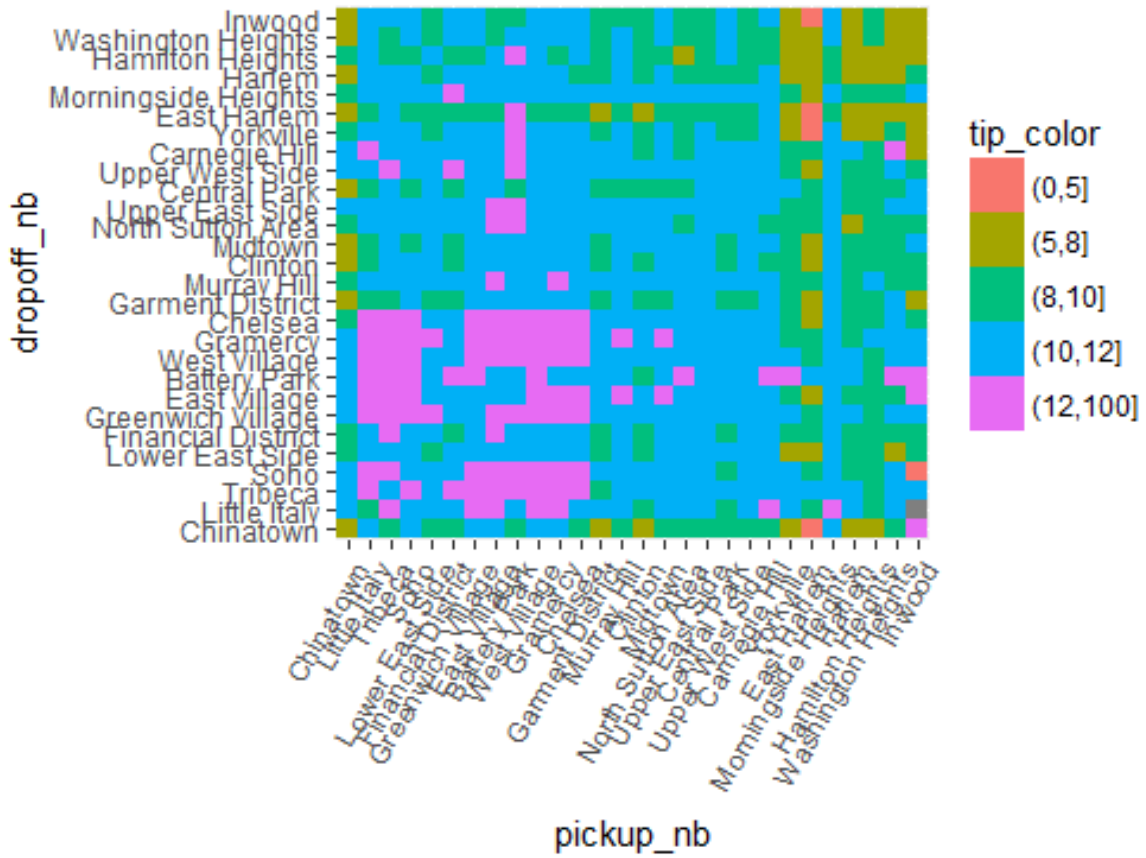
```
ggplot(res, aes(pickup_nb, dropoff_nb)) +
  geom_tile(aes(fill = minutes_per_mile), colour = "white") +
  theme(axis.text.x = element_text(angle = 60, hjust = 1)) +
  scale_fill_gradient(low = "white", high = "steelblue") +
  coord_fixed(ratio = .9)
```



Come mostrano i grafici, molto traffico avviene tra quartieri vicini l'uno all'altro. Questo non è molto sorprendente dato che i viaggi tra quartieri lontani possono essere fatti utilizzando percorsi periferici che bypassano la maggior parte del traffico del centro città.

Un'altra questione interessante da prendere in considerazione è la relazione tra l'importo della corsa e quanta mancia i passeggeri lasciano in base tra quali quartieri viaggiano. Creiamo un altro grafico simile a quelli sopra, mostrando l'importo della tariffa su una scala di colori di sfondo grigio e mostrando quanta mancia in media lasciano per il viaggio. Per rendere più semplice la visualizzazione, codifico con colori la mancia media in base al fatto che sia superiore al 12%, inferiore al 12%, inferiore al 10%, inferiore all'8% e inferiore al 5%.

```
res %>%
  mutate(tip_color = cut(tip_percent, c(0, 5, 8, 10, 12, 100))) %>%
  ggplot(aes(pickup_nb, dropoff_nb)) +
  geom_tile(aes(fill = tip_color)) +
  theme(axis.text.x = element_text(angle = 60, hjust = 1)) +
  coord_fixed(ratio = .9)
```



Alcuni considerazioni interessanti:

- I viaggi che partono da Battery Park o il quartiere finanziario che va a Midtown o nei quartieri periferici, sembrano costare (in mancia) un po' più di quanto sembra giustificato, e vale lo stesso per i viaggi che lasciano il Greenwich Village andando a Chinatown.
- I viaggi dentro e fuori Chinatown sono costantemente bassi (sotto il 10%), specialmente se si viaggia o viene da quartieri alti.
- I più generosi (circa il 12%) sono quelli che viaggiano tra i quartieri del centro (ad eccezione di Chinatown). I successivi più generosi (circa l'11%) sono quelli che viaggiano tra quartieri centrali e quartieri del centro in entrambe le direzioni. I peggiori (quelle che lasciano meno valore di mancia) sono quelli che viaggiano tra quartieri alti.

Abbiamo modificato (solo una parte del *dataset*) l'ordine dei livelli dei fattori per *pickup_nb* e *dropoff_nb* per disegnare i grafici sopra. Tuttavia, questo cambiamento parziale non giova all'analisi

stessa, perché ogni volta che tracciamo qualcosa che coinvolge *pickup_nb* o *dropoff_nb* avremo bisogno di cambiare l'ordine dei livelli dei fattori. Quindi vado a ora modificare l'intero *dataset*.

```
rxDataStep(inData = mht_xdf, outFile = mht_xdf,
           transforms = list(pickup_nb = factor(pickup_nb, levels = newlevels),
                             dropoff_nb = factor(dropoff_nb, levels = newlevels)),
           transformObjects = list(newlevels = unique(newlevs)),
           overwrite = TRUE)
```

Ora vorrei analizzare tra quali quartieri si verificano maggiormente i viaggi dei taxi. Per farlo devo trovare la distribuzione (o la proporzione) di viaggi tra due quartieri, prima come percentuale del numero totale di viaggi, poi come percentuale di viaggi in partenza da un determinato quartiere e infine come percentuale di viaggi diretti a un determinato quartiere.

```
rxc <- rxCube( ~ pickup_nb:dropoff_nb, mht_xdf)
rxc <- as.data.frame(rxc)

library(dplyr)
rxc %>%
  filter(Counts > 0) %>%
  mutate(pct_all = Counts/sum(Counts) * 100) %>%
  group_by(pickup_nb) %>%
  mutate(pct_by_pickup_nb = Counts/sum(Counts) * 100) %>%
  group_by(dropoff_nb) %>%
  mutate(pct_by_dropoff_nb = Counts/sum(Counts) * 100) %>%
  group_by() %>%
  arrange(desc(Counts)) -> rxcs

head(rxcs)
```

```
         pickup_nb      dropoff_nb  Counts  pct_all pct_by_pickup_nb
            <fctr>          <fctr>   <dbl>    <dbl>            <dbl>
1  Upper East Side Upper East Side 3299324 5.738650         36.88840
2          Midtown         Midtown 2216184 3.854700         21.84268
3  Upper West Side Upper West Side 1924205 3.346849         35.14494
4          Midtown Upper East Side 1646843 2.864422         16.23127
5  Upper East Side         Midtown 1607925 2.796730         17.97756
6 Garment District         Midtown 1072732 1.865847         28.94205
  pct_by_dropoff_nb
              <dbl>
1          38.28066
2          22.41298
3          35.15770
4          19.10762
5          16.26146
6          10.84888
```

Sulla base della prima riga, possiamo vedere che i viaggi dall'Upper East Side all'Upper East Side costituiscono circa il 5% di tutti i viaggi a Manhattan. Tra tutti i viaggi che sono partiti dall'Upper East Side, circa il 36% arriva nell'Upper East Side. Di tutti i viaggi che sono arrivati nell'Upper East Side, il 37% sono anche partiti dall'Upper East Side.

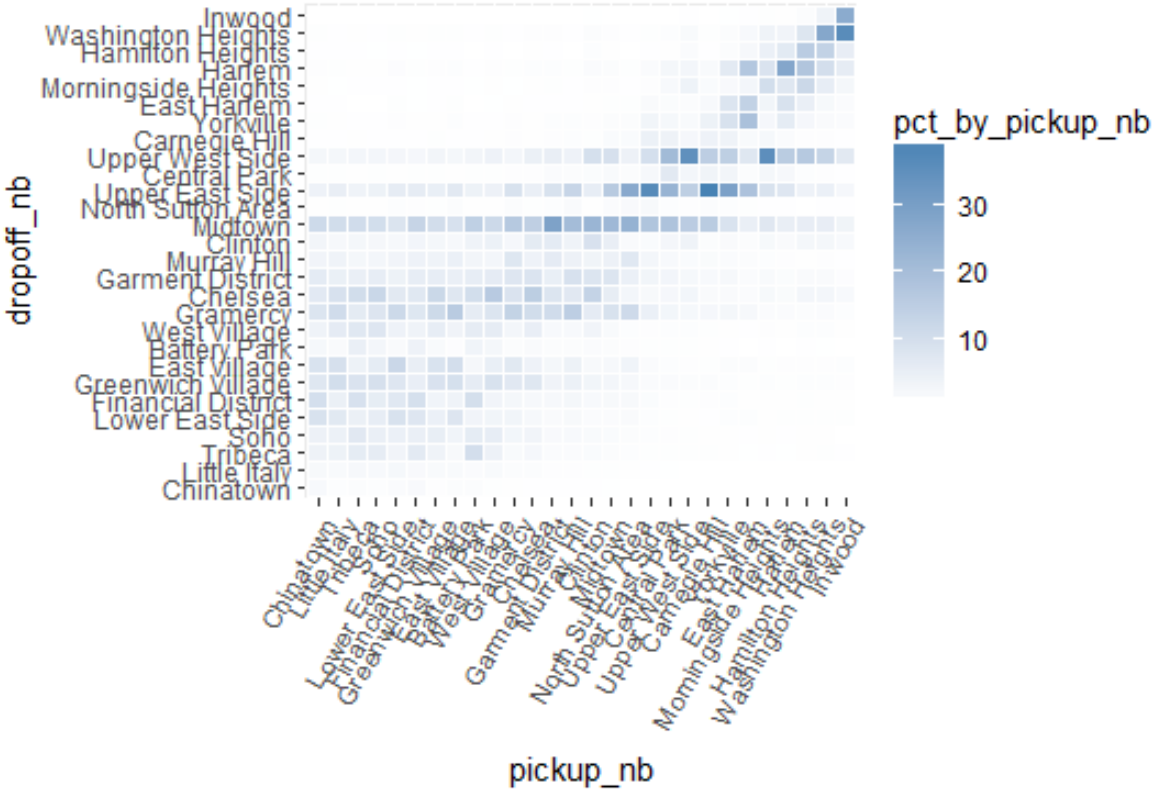
Possiamo ora utilizzare i dati sopra e visualizzarli in grafico a matrice per rendere più facile la lettura che mostra come vengono distribuiti i viaggi in taxi tra ogni coppia di quartieri.

```
ggplot(rxcs, aes(pickup_nb, dropoff_nb)) +
  geom_tile(aes(fill = pct_all), colour = "white") +
  theme(axis.text.x = element_text(angle = 60, hjust = 1)) +
  scale_fill_gradient(low = "white", high = "black") +
  coord_fixed(ratio = .9)
```

Il grafico mostra che da e per l'Upper East Side costituiscono la maggior parte dei viaggi, un risultato un po 'inaspettato. Inoltre, in generale, la maggior parte dei viaggi è da e per l'Upper East Side e l'Upper West Side e i quartieri di midtown (con molti viaggi di questa categoria che hanno Midtown come origine o destinazione). Un altro fatto sorprendente dei dati visualizzati nel grafico, è la simmetria di vicinato, il che suggerisce che forse la maggior parte dei passeggeri usa i taxi per un "viaggio di andata e ritorno", cioè che prendono un taxi per raggiungere la loro destinazione e lo stesso per il viaggio di ritorno. Questo punto merita un'ulteriore indagine (forse coinvolgendo l'ora del giorno nell'analisi) ma per ora non approfondiamo ulteriormente.

Poi analizziamo come i viaggi lasciano un determinato quartiere (un punto sull'asse x nella grafico seguente), "si riversano" poi in altri quartieri (mostrato dal gradiente di colore verticale lungo l'asse y in ogni punto dell'asse x).

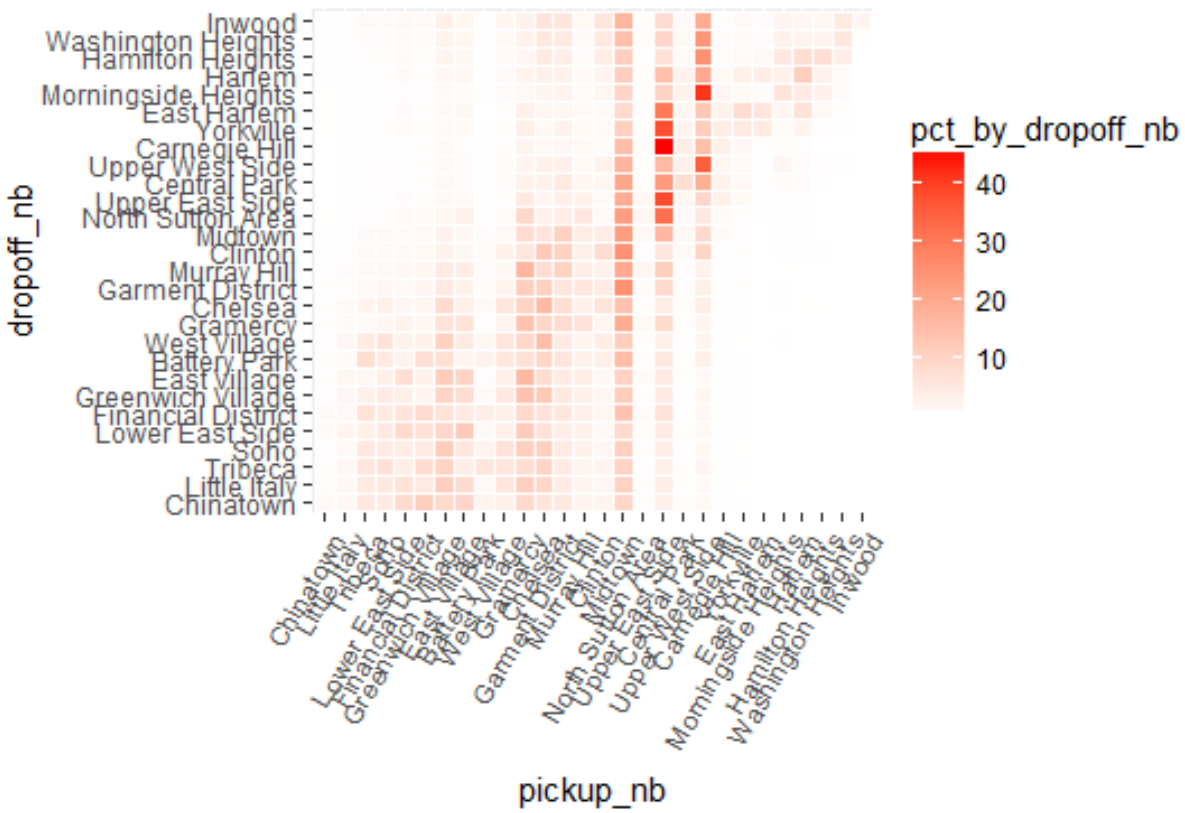
```
ggplot(rxcs, aes(pickup_nb, dropoff_nb)) +
  geom_tile(aes(fill = pct_by_pickup_nb), colour = "white") +
  theme(axis.text.x = element_text(angle = 60, hjust = 1)) +
  scale_fill_gradient(low = "white", high = "steelblue") +
  coord_fixed(ratio = .9)
```



Possiamo vedere come la maggior parte dei viaggi dal centro sono verso altri quartieri del centro o ai quartieri vicino midtown (in particolare Midtown). Midtown e Upper East Side sono destinazioni comuni da qualsiasi quartiere, e l'Upper West Side è una destinazione comune per la maggior parte dei quartieri residenziali.

Per una corsa che termina in un determinato quartiere (rappresentato da un punto sull'asse y) ora guardiamo alla distribuzione da cui il viaggio ha avuto origine (il gradiente di colore orizzontale lungo l'asse x per ogni punto sull'asse y).

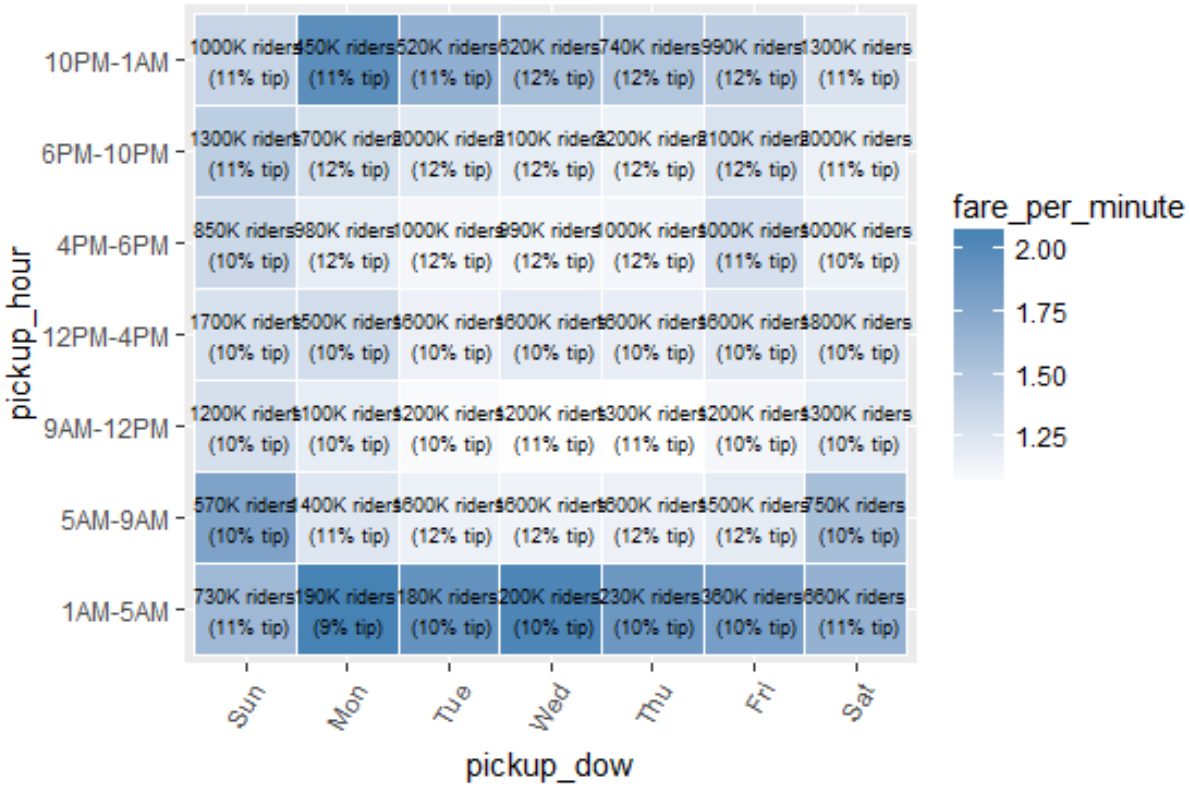
```
ggplot(rxcs, aes(pickup_nb, dropoff_nb)) +
  geom_tile(aes(fill = pct_by_dropoff_nb), colour = "white") +
  theme(axis.text.x = element_text(angle = 60, hjust = 1)) +
  scale_fill_gradient(low = "white", high = "red") +
  coord_fixed(ratio = .9)
```



Come possiamo vedere, molte corse richiamano Midtown indipendentemente da dove sono partite. L'Upper East Side e Upper West Side sono anche loro origini comunemente utilizzate per i viaggi che scendono in uno dei quartieri alti.

Vediamo ora informazioni possono essere ricavate dalle colonne temporali che abbiamo estratto dai dati, vale a dire il giorno della settimana e l'ora in cui il passeggero è stato prelevato.

```
res1 <- rxCube(tip_percent ~ pickup_dow:pickup_hour, mht_xdf)
res2 <- rxCube(fare_amount/(trip_duration/60) ~ pickup_dow:pickup_hour, mht_xdf)
names(res2)[3] <- 'fare_per_minute'
res <- bind_cols(list(res1, res2))
res <- res[ , c('pickup_dow',
                'pickup_hour',
                'fare_per_minute',
                'tip_percent',
                'Counts')]

library(ggplot2)
ggplot(res, aes(pickup_dow, pickup_hour)) +
  geom_tile(aes(fill = fare_per_minute), colour = "white") +
  theme(axis.text.x = element_text(angle = 60, hjust = 1)) +
  scale_fill_gradient(low = "white", high = "steelblue") +
  geom_text(aes(label = sprintf('%dK riders\n (%d%% tip)',
                                signif(Counts/1000, 2),
                                 round(tip_percent, 0))),
                                 size = 2.5) +
  coord_fixed(ratio = .9)
```



Dalla grafico a matrice sopra possiamo vedere che una corsa in taxi costa di più al fine settimana che in un giorno feriale se è presa tra le 5:00 e le 22:00 e viceversa dalle 22:00 alle 5:00. Il grafico suggerisce anche che i passeggeri lascino più mancia nei giorni feriali e soprattutto subito dopo l'orario di ufficio. La questione delle mance dovrebbe essere guardata più da vicino, soprattutto dal momento che la percentuale è influenzata dall'uso di contanti o carta, che finora non ho preso in considerazione.