02 - Trasformazione e Integrazione Dati

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

Bonifica

Le attività di preparazione dei dati comunemente riguardano la gestione dei valori mancanti, gestione dei valori anomali, determinazione del livello di granularità dei dati, ad esempio le variabili temporali possono essere in secondi, minuti o ore, ecc. - decidere quali funzionalità aggiungere o estrarre in base alle funzionalità esistenti per rendere l'analisi più interessante o più facile da interpretare.

Come prima cosa è buona idea controllare le tipologie delle variabili e accertarsi che nulla di anomalo venga riscontrato. Oltre alla tipologia dei valori in colonna, la funzione `rxGetInfo` mostra anche i minimi e i massimi per tutte le variabili, che possono essere utili ad identificare ulteriormente i valori anomali ed eseguire controlli di integrità. Con l'argomento `numRows = 10`, possiamo esaminare anche le prime 10 righe dei dati.

```
# tipologia di colonna e le prime 10 righe dei dati #
rxGetInfo(nyc_xdf, getVarInfo = TRUE, numRows = 10)
```

```
File name: C:\Data\NYC_taxi\01\yellow_tripdata_2016.xdf
Number of observations: 3467953
Number of variables: 20
Number of blocks: 6
Compression type: zlib
Variable information:
Var 1: VendorID 2 factor levels: 2 1
Var 2: tpep_pickup_datetime, Type: character
Var 3: tpep_dropoff_datetime, Type: character
Var 4: passenger_count, Type: integer, Low/High: (0, 9)
Var 5: trip_distance, Type: numeric, Low/High: (0.0000, 12000004.5000)
Var 6: pickup_longitude, Type: numeric, Low/High: (-121.9333, 0.0000)
Var 7: pickup_latitude, Type: numeric, Low/High: (0.0000, 53.4046)
Var 8: RatecodeID, Type: integer, Low/High: (1, 99)
Var 9: store_and_fwd_flag 2 factor levels: N Y
```

```
Var 10: dropoff_longitude, Type: numeric, Low/High: (-121.9333, 0.0000)
Var 11: dropoff_latitude, Type: numeric, Low/High: (0.0000, 50.7979)
Var 12: payment_type  4 factor levels: 2 1 3 4
Var 13: fare_amount, Type: numeric, Low/High: (-376.0000, 2550.2000)
Var 14: extra, Type: numeric, Low/High: (-4.5000, 50.0100)
Var 15: mta_tax, Type: numeric, Low/High: (-1.0000, 3.0000)
Var 16: tip_amount, Type: numeric, Low/High: (-35.0000, 998.1400)
Var 17: tolls_amount, Type: numeric, Low/High: (-10.5000, 613.5000)
Var 18: improvement_surcharge, Type: numeric, Low/High: (-0.3000, 11.6400)
Var 19: total_amount, Type: numeric, Low/High: (-376.3000, 2551.0000)
Var 20: u, Type: numeric, Low/High: (0.0000, 0.0500)
Data (5 rows starting with row 1):
VendorID tpep_pickup_datetime tpep_dropoff_datetime passenger_count trip_distance
1        2  2016-01-01 00:00:00   2016-01-01 00:00:00               2          1.10
2        2  2016-01-01 00:00:00   2016-01-01 00:00:00               5          4.90
3        2  2016-01-01 00:00:00   2016-01-01 00:00:00               1         10.54
4        2  2016-01-01 00:00:00   2016-01-01 00:00:00               1          4.75
5        2  2016-01-01 00:00:00   2016-01-01 00:00:00               3          1.76
  pickup_longitude pickup_latitude RatecodeID store_and_fwd_flag dropoff_longitude
1        -73.99037        40.73470          1                  N         -73.98184
2        -73.98078        40.72991          1                  N         -73.94447
3        -73.98455        40.67957          1                  N         -73.95027
4        -73.99347        40.71899          1                  N         -73.96224
5        -73.96062        40.78133          1                  N         -73.97726
  dropoff_latitude payment_type fare_amount extra mta_tax tip_amount tolls_amount
1         40.73241            2         7.5   0.5     0.5          0            0
2         40.71668            1        18.0   0.5     0.5          0            0
3         40.78893            1        33.0   0.5     0.5          0            0
4         40.65733            2        16.5   0.0     0.5          0            0
5         40.75851            2         8.0   0.0     0.5          0            0
  improvement_surcharge total_amount
1                   0.3          8.8
2                   0.3         19.3
3                   0.3         34.3
4                   0.3         17.3
5                   0.3          8.8
```

Una volta che i dati sono stati importati e controllati, possiamo iniziare a pensare alle interessanti / rilevanti caratteristiche che rientrano nella nostra analisi. L'obiettivo è principalmente esplorativo: vogliamo raccontare una storia basata sui dati. In questo progetto, qualsiasi informazione contenuta nei dati può essere utile. Inoltre, nuove informazioni (o caratteristiche) possono essere estratte da dati esistenti. Non è solo importante pensare a quali funzionalità estrarre, ma anche a come devono essere categorizzate le colonne, in modo che le analisi successive siano eseguite in modo appropriato e alcune

anomalia vengano escluse. Come una semplice trasformazione, ad esempio, per estrarre la percentuale di mancia che i passeggeri hanno lasciato per il viaggio.

La trasformazione verrà calcolata su di una nuova colonna denominata *tip_percent* basata sulla seguente logica:

- SE *fare_amount* > zero & *tip_amount* < *fare_amount*, calcoleremo (*tip_amount* / *fare_amount*)* 100. E lo arrotonderemo a un numero intero, che sarà infine il valore di *tip_percent*.
- SE la condizione precedente non è soddisfatta per qualche motivo. Assegneremo NA al valore di *tip_percent*.

```
rxDataStep(nyc_xdf, nyc_xdf,
           transforms = list(tip_percent = ifelse(fare_amount > 0 &
                                tip_amount < fare_amount,
                                round(tip_amount * 100 / fare_amount, 0), NA)),
           overwrite = TRUE)
rxSummary(~tip_percent, nyc_xdf)
```

```
Call:
rxSummary(formula = ~tip_percent, data = nyc_xdf)


Summary Statistics Results for: ~tip_percent
Data: nyc_xdf (RxXdfData Data Source)
File name: yellow_tripdata_2016.xdf
Number of valid observations: 3467953


 Name        Mean     StdDev   Min Max ValidObs MissingObs
 tip_percent 13.96739 11.88536 0   99  3462221  5732
```

Possiamo vedere che in media i clienti hanno pagato circa il 14% del valore della corsa in mancia. La deviazione standard è leggermente alta. Possiamo vedere il minimo e il massimo valore percentuale. E possiamo notare che abbiamo parecchi valori mancanti. Questi sono i valori mancanti che vengono generati dalla nostra formula, e che vanno a escludere quei valori anomali che volevamo evitare.

Ora vogliamo l'interazione tra mese e anno, e ottenere dei conteggi da esso estrapolando i dati dalla colonna che ci interessa. Userò la colonna *pickup_datetime*, che è la prima delle colonne con valori di tipo *character*.

```
rxCrossTabs(~ month:year, nyc_xdf,
            transforms = list(
            date = ymd_hms(tpep_pickup_datetime),
            year = factor(year(date), levels = 2014:2016),
```

```
            month = factor(month(date), levels = 1:12)),
            transformPackages = "lubridate")
```

```
Call:
rxCrossTabs(formula = ~month:year, data = nyc_xdf, transforms = list(date = ymd_hms(tpep_pickup_datetime),
    year = factor(year(date), levels = 2014:2016), month = factor(month(date),
        levels = 1:12)), transformPackages = "lubridate")

Cross Tabulation Results for: ~month:year
Data: nyc_xdf (RxXdfData Data Source)
File name: yellow_tripdata_2016.xdf
Number of valid observations: 3467953
Number of missing observations: 0
Statistic: counts

month:year (counts):
     year
month 2014 2015   2016
   1     0    0 544498
   2     0    0 568348
   3     0    0 610255
   4     0    0 596987
   5     0    0 591985
   6     0    0 555880
   7     0    0      0
   8     0    0      0
   9     0    0      0
   10    0    0      0
   11    0    0      0
   12    0    0      0
```

Per questo progetto sui taxi di New York, siamo interessati a confrontare i viaggi in base al giorno della settimana e all'ora del giorno. Queste due colonne non esistono ancora, ma possiamo estrarle dalla data e ora di partenza e dalla data e ora di arrivo. Per estrarre le funzionalità di cui sopra, utilizziamo il pacchetto *lubridate*, che ha funzioni utili per gestire le colonne di data e ora. Per eseguire queste trasformazioni, usiamo una funzione di trasformazione chiamata `xforms`.

```
xforms <- function(data)
{ # funzione di trasformazione per l'estrazione di parametri di data e ora

  weekday_labels <- c('Sun', 'Mon', 'Tue', 'Wed', 'Thu', 'Fri', 'Sat')
  cut_levels <- c(1, 5, 9, 12, 16, 18, 22)
```

```
  hour_labels <- c('1AM-5AM', '5AM-9AM', '9AM-12PM', '12PM-4PM',
                   '4PM-6PM', '6PM-10PM', '10PM-1AM')

  pickup_datetime <- ymd_hms(data$tpep_pickup_datetime, tz = "UTC")
  pickup_hour <- addNA(cut(hour(pickup_datetime), cut_levels))
  pickup_dow <- factor(wday(pickup_datetime),
                       levels = 1:7,
                       labels = weekday_labels)
  levels(pickup_hour) <- hour_labels

  dropoff_datetime <- ymd_hms(data$tpep_dropoff_datetime, tz = "UTC")
  dropoff_hour <- addNA(cut(hour(dropoff_datetime), cut_levels))
  dropoff_dow <- factor(wday(dropoff_datetime),
                        levels = 1:7,
                        labels = weekday_labels)
  levels(dropoff_hour) <- hour_labels

  data$pickup_hour <- pickup_hour
  data$pickup_dow <- pickup_dow
  data$dropoff_hour <- dropoff_hour
  data$dropoff_dow <- dropoff_dow
  data$trip_duration <- as.integer(as.duration(dropoff_datetime-pickup_datetime))

  data
}
```

Prima di applicare la trasformazione a tutti i dati, di solito è una buona idea testarlo e assicurarsi che funzioni. A tal fine, mettiamo da parte un campione dei dati come `data.frame`. L'esecuzione della funzione di trasformazione su `nyc_sample_df` dovrebbe restituire i dati originali con le nuove colonne.

```
library(lubridate)
Sys.setenv(TZ = "US/Eastern") # non importante per questo df
head(xforms(nyc_sample_df)) # testa la funzione sul data.frame
```

```
VendorID tpep_pickup_datetime tpep_dropoff_datetime passenger_count trip_distance
1        2  2016-01-01 00:00:00   2016-01-01 00:00:00               2          1.10
2        2  2016-01-01 00:00:00   2016-01-01 00:00:00               5          4.90
3        2  2016-01-01 00:00:00   2016-01-01 00:00:00               1         10.54
4        2  2016-01-01 00:00:00   2016-01-01 00:00:00               1          4.75
5        2  2016-01-01 00:00:00   2016-01-01 00:00:00               3          1.76
6        2  2016-01-01 00:00:00   2016-01-01 00:18:30               2          5.52
```

```
  pickup_longitude pickup_latitude RatecodeID store_and_fwd_flag dropoff_longitude
1        -73.99037        40.73470          1                  N         -73.98184
2        -73.98078        40.72991          1                  N         -73.94447
3        -73.98455        40.67957          1                  N         -73.95027
4        -73.99347        40.71899          1                  N         -73.96224
5        -73.96062        40.78133          1                  N         -73.97726
6        -73.98012        40.74305          1                  N         -73.91349
  dropoff_latitude payment_type fare_amount extra mta_tax tip_amount tolls_amount
1         40.73241            2         7.5   0.5     0.5          0            0
2         40.71668            1        18.0   0.5     0.5          0            0
3         40.78893            1        33.0   0.5     0.5          0            0
4         40.65733            2        16.5   0.0     0.5          0            0
5         40.75851            2         8.0   0.0     0.5          0            0
6         40.76314            2        19.0   0.5     0.5          0            0
  improvement_surcharge total_amount pickup_hour pickup_dow dropoff_hour
1                   0.3          8.8    10PM-1AM        Fri     10PM-1AM
2                   0.3         19.3    10PM-1AM        Fri     10PM-1AM
3                   0.3         34.3    10PM-1AM        Fri     10PM-1AM
4                   0.3         17.3    10PM-1AM        Fri     10PM-1AM
5                   0.3          8.8    10PM-1AM        Fri     10PM-1AM
6                   0.3         20.3    10PM-1AM        Fri     10PM-1AM
  dropoff_dow trip_duration
1         Fri             0
2         Fri             0
3         Fri             0
4         Fri             0
5         Fri             0
6         Fri          1110
```

Tutto sembra funzionare bene. Ciò non garantisce che l'esecuzione della funzione di trasformazione sull'intero set di dati abbia esito positivo, ma rende meno probabile il fallimento soprattutto se l'operazione sui dati richiede diverso tempo di elaborazione. Se la trasformazione funziona sul campione `data.frame`, come sopra, ma fallisce quando la eseguiamo sull'intero *dataset*, di solito è a causa di qualcosa nel set di dati che causa il fallimento (come i valori mancanti) che non erano presenti nei dati di esempio. Ora eseguiamo la trasformazione su tutto il set di dati.

```
st <- Sys.time()
rxDataStep(nyc_xdf, nyc_xdf, overwrite = TRUE, transformFunc = xforms, transformPackages = "lubridate")
Sys.time() - st
```

```
Time difference of 39.28945 secs
```

Esamino le nuove colonne create per assicurarmiche la trasformazione sia più o meno funzionante. Uso la funzione `rxSummary` per ottenere alcuni riepiloghi statistici dei dati. La funzione `rxSummary` è simile alla funzione di `summary` R base (a parte il fatto che il `summary` funziona solo su un `data.frame`):

- Fornisce riepiloghi per colonne numeriche (eccetto per percentili, per i quali uso la funzione `rxQuantile`).
- Fornisce i conteggi per ogni livello delle factor columns.

Utilizzo la stessa notazione formula usata da molte altre funzioni di modellazione in R o di plotting, per specificare in quali colonne vogliamo i riepiloghi. Ad esempio, qui vogliamo vedere i riepiloghi per *pickup_our* e *pickup_dow* (entrambi i factors) e *trip_duration* (numerico, in secondi).

```
rxs1 <- rxSummary( ~ pickup_hour + pickup_dow + trip_duration, nyc_xdf)
# possiamo aggiungere una colonna per le proporzioni accanto ai conteggi
rxs1$categorical <- lapply(rxs1$categorical, function(x) cbind(x, prop = round(prop.table(x$Counts), 2)))
rxs1
```

```
Call:
rxSummary(formula = ~pickup_hour + pickup_dow + trip_duration,
    data = nyc_xdf)

Summary Statistics Results for: ~pickup_hour + pickup_dow + trip_duration
Data: nyc_xdf (RxXdfData Data Source)
File name: yellow_tripdata_2016.xdf
Number of valid observations: 69406520

 Name          Mean     StdDev   Min        Max      ValidObs MissingObs
 trip_duration 933.9168 119243.5 -631148790 11538803 69406520 0

Category Counts for pickup_hour
Number of categories: 7
Number of valid observations: 69406520
Number of missing observations: 0

 pickup_hour Counts   prop
 1AM-5AM      3801430 0.05
 5AM-9AM     10630653 0.15
 9AM-12PM     9765429 0.14
 12PM-4PM    13473045 0.19
 4PM-6PM      7946899 0.11
 6PM-10PM    16138968 0.23
 10PM-1AM     7650096 0.11
```

```
Category Counts for pickup_dow
Number of categories: 7
Number of valid observations: 69406520
Number of missing observations: 0

 pickup_dow Counts   prop
 Sun         9267881 0.13
 Mon         8938785 0.13
 Tue         9667525 0.14
 Wed         9982769 0.14
 Thu        10398738 0.15
 Fri        10655022 0.15
 Sat        10495800 0.15
```

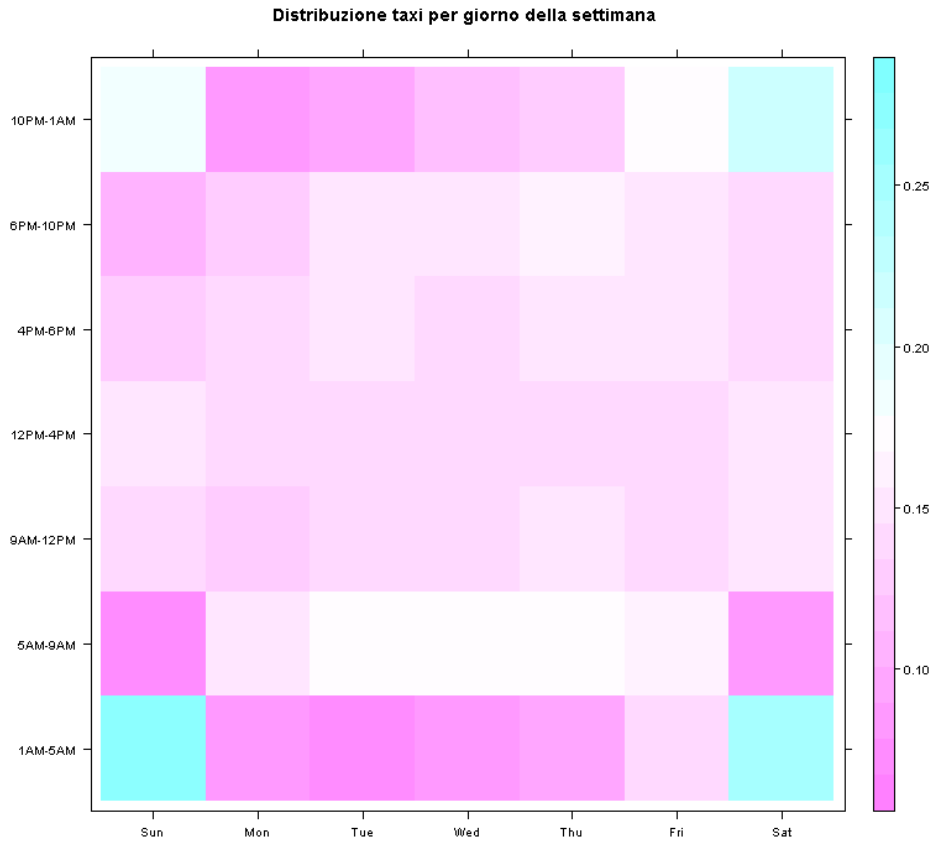
Si possono ottenere dati per ciascuna combinazione dei livelli delle colonne dei due fattori, invece dei soli dati individuali.

```
rxs2 <- rxSummary( ~ pickup_dow:pickup_hour, nyc_xdf)
rxs2 <- tidyr::spread(rxs2$categorical[[1]],
                      key = 'pickup_hour',
                      value = 'Counts')
row.names(rxs2) <- rxs2[ , 1]
rxs2 <- as.matrix(rxs2[ , -1])
rxs2
```

```
    1AM-5AM 5AM-9AM 9AM-12PM 12PM-4PM 4PM-6PM 6PM-10PM 10PM-1AM
Sun 1040233  740157  1396409  1980752 1032434  1697529  1380367
Mon  304474 1630951  1268326  1838143 1133728  2096219   666944
Tue  278407 1840134  1382381  1882356 1151837  2390506   741904
Wed  313809 1854757  1417953  1880896 1142071  2508618   864665
Thu  354646 1871828  1428985  1922502 1165535  2634023  1021219
Fri  553159 1766482  1406979  1922542 1173163  2516285  1316412
Sat  956702  926344  1464396  2045854 1148131  2295788  1658585
```

Ma in questo caso, i conteggi individuali non sono così utili come le proporzioni degli stessi, e per fare un confronto tra diversi giorni della settimana, vorrei che le proporzioni siano basate sui totali di ogni colonna, non sull'intera tabella. Posso quindi visualizzare visivamente le proporzioni usando la funzione `levelplot`.

```
levelplot(prop.table(rxs2, 2),
          cuts = 4,
          xlab = "",
          ylab = "",
          main = "Distribuzione taxi per giorno della settimana")
```



Il grafico mostra alcune informazioni interessanti:

1. La mattina presto (tra le 5:00 e le 9:00) le corse in taxi sono prevedibilmente al minimo durante il fine settimana e leggermente basse anche il lunedì.
2. Durante l'orario lavorativo (tra le 9:00 e le 18:00), avviene circa la stessa percentuale di viaggi in taxi (dal 42 al 45% circa) per ogni giorno della settimana, compresi i fine settimana. In altre parole, indipendentemente da quale sia il giorno della settimana, un po' meno della metà di tutti i viaggi avviene tra le 9:00 e le 18:00.
3. Possiamo vedere un picco di viaggi in taxi tra le 18:00 e le 22:00 il giovedì e il venerdì sera, e un picco tra le 22:00 e l'1:00 di venerdì e soprattutto il sabato sera. Viaggi in taxi tra 1AM e 5AM con un picco al sabato (rientri dal venerdì sera) e ancor più la domenica (rientri del sabato sera). Cadono poi bruscamente negli altri giorni, ma leggermente si riprendono al venerdì (in tarda serata). In altre parole, molte persone escono di giovedì ma non restano fuori fino a tardi, altre persone escono il venerdì e rimangono anche più tardi, ma il sabato è il giorno in cui la maggior parte delle persone sceglie per una serata di ore piccole.

Integrazione Dati

Ora aggiungo un altro set di dati: i quartieri di partenza e arrivo. Ottenere informazioni sui quartieri da longitudine e latitudine non è qualcosa che possiamo codificare facilmente, possiamo però usare alcuni pacchetti GIS e uno *shapefile* (per gentile concessione di Zillow). Uno *shapefile* è un file che contiene informazioni geografiche al suo interno, incluse informazioni sui confini che separano le aree geografiche. Il file *ZillowNeighborhoods-NY.shp* contiene informazioni sui quartieri di New York.

Iniziamo disegnando una mappa dei quartieri di Manhattan, così possiamo vedere i confini del quartiere e familiarizzare con i loro nomi.

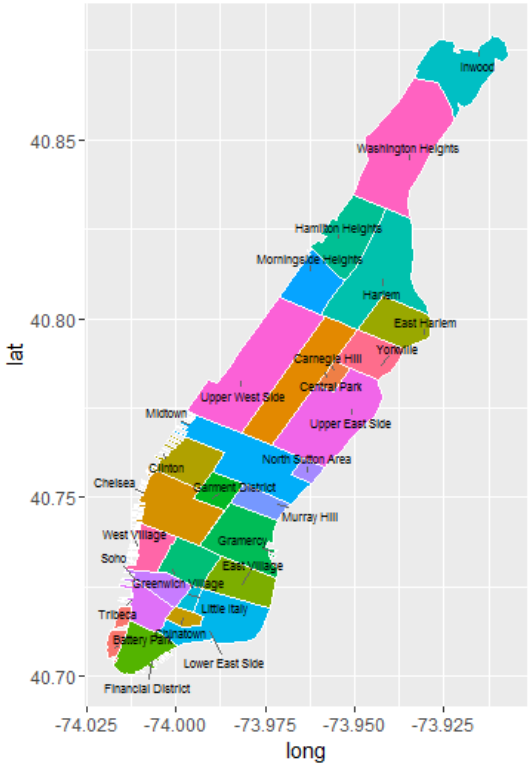
```
library(rgeos)
library(sp)
library(maptools)
library(rgdal)

nyc_shapefile<-readShapePoly('ZillowNeighborhoods-NY/ZillowNeighborhoods-NY.shp')
mht_shapefile <- subset(nyc_shapefile, str_detect(CITY, 'New York City-Manhattan'))

mht_shapefile@data$id <- as.character(mht_shapefile@data$NAME)
mht.points <- fortify(gBuffer(mht_shapefile,
                              byid = TRUE,
                              width = 0), region = "NAME")
mht.df <- inner_join(mht.points, mht_shapefile@data, by = "id")

library(dplyr)
mht.cent <- mht.df %>%
  group_by(id) %>%
  summarize(long = median(long), lat = median(lat))

library(ggrepel)
ggplot(mht.df, aes(long, lat, fill = id)) +
    geom_polygon() +
    geom_path(color = "white") +
    coord_equal() +
    theme(legend.position = "none") +
    geom_text_repel(aes(label = id), data = mht.cent, size = 2)
```



Possiamo visualizzare quindi la mappa di Manhattan, possiamo vedere i quartieri che sono codificati in diversi colori con il nome del quartiere che mostra più o meno il quartiere stesso. Tutte queste informazioni erano codificate nello *shapefile* che ho appena caricato. In questo caso, abbiamo usato le informazioni per disegnare una mappa. Adesso bisogna codificare una trasformazione complessa che andrà a interagire con le coordinate per la partenza e arrivo, che abbiamo nei nostri dati. E troverà così il quartiere corrispondente al *pickup* e il *drop-off*.

Ho quindi archiviato lo *shapefile* per Manhattan in un oggetto chiamato `mht_shapefile`, che abbiamo usato per tracciare una mappa dei quartieri di Manhattan. Vediamo ora come possiamo usare lo stesso *shapefile* per individuare i quartieri di *pick-up* e *drop-off* per ogni corsa. Possiamo farlo con un `data.frame` (utilizzando il *sample* dei dati dei taxi di New York che abbiamo preso in precedenza).

Per aggiungere la colonna dei quartieri di *pick-up* a `nyc_sample_df`, dobbiamo procedere come segue: - sostituire gli `NA` in *pickup_latitude* e *pickup_longitude* con `0`, altrimenti otterremo un errore – utilizzando poi la funzione coordinate per specificare che le due colonne precedenti rappresentano le coordinate geografiche in dati: uso poi la funzione `over` per ottenere quartieri in base alle coordinate specificate sopra, che restituirà un `data.frame` con informazioni sul quartiere – verranno allegati poi i risultati ai dati originali usando `cbind` dopo aver assegnato loro i nomi corretti.

```
# prendere solo le colonne con coordinate e sostituisce NA con 0
data_coords <- transmute(nyc_sample_df,
  long = ifelse(is.na(pickup_longitude), 0, pickup_longitude),
  lat = ifelse(is.na(pickup_latitude), 0, pickup_latitude)
)
```

```
# specifichiamo le colonne che corrispondono alle coordinate
coordinates(data_coords) <- c('long', 'lat')
head(data_coords)

# restituisce il nome dei quartieri in base alle coordinate
nhoods <- over(data_coords, mht_shapefile)
head(nhoods)

# rinomina le colonne in nhoods
names(nhoods) <- paste('pickup', tolower(names(nhoods)), sep = '_')
# combina le informazioni sul quartiere con i dati originali
nyc_sample_df <- cbind(nyc_sample_df, nhoods[, grep('name|city', names(nhoods))])
head(nyc_sample_df)
```

```
SpatialPoints:
          long      lat
[1,] -73.99406 40.71999
[2,] -73.98166 40.77374
[3,] -73.96375 40.77116
[4,] -73.97602 40.74456
[5,] -73.94185 40.82948
[6,] -73.97367 40.78452
Coordinate Reference System (CRS) arguments: NA
  STATE   COUNTY                   CITY            NAME REGIONID              id
1    NY New York New York City-Manhattan Lower East Side   270875 Lower East Side
2    NY New York New York City-Manhattan Upper West Side   270958 Upper West Side
3    NY New York New York City-Manhattan Upper East Side   270957 Upper East Side
4    NY New York New York City-Manhattan        Gramercy   273860        Gramercy
5    NY New York New York City-Manhattan          Harlem   195267          Harlem
6    NY New York New York City-Manhattan Upper West Side   270958 Upper West Side
  VendorID tpep_pickup_datetime tpep_dropoff_datetime passenger_count
1        2  2016-01-01 00:00:00   2016-01-01 00:00:00               2
2        2  2016-01-01 00:00:00   2016-01-01 00:00:00               5
3        2  2016-01-01 00:00:00   2016-01-01 00:00:00               1
4        2  2016-01-01 00:00:00   2016-01-01 00:00:00               1
5        2  2016-01-01 00:00:00   2016-01-01 00:00:00               3
6        2  2016-01-01 00:00:00   2016-01-01 00:18:30               2
  trip_distance pickup_longitude pickup_latitude RatecodeID store_and_fwd_flag
1          1.10        -73.99037        40.73470          1                  N
2          4.90        -73.98078        40.72991          1                  N
3         10.54        -73.98455        40.67957          1                  N
4          4.75        -73.99347        40.71899          1                  N
```

```
5          1.76        -73.96062        40.78133          1                  N
6          5.52        -73.98012        40.74305          1                  N
  dropoff_longitude dropoff_latitude payment_type fare_amount extra mta_tax
1         -73.98184         40.73241            2         7.5   0.5     0.5
2         -73.94447         40.71668            1        18.0   0.5     0.5
3         -73.95027         40.78893            1        33.0   0.5     0.5
4         -73.96224         40.65733            2        16.5   0.0     0.5
5         -73.97726         40.75851            2         8.0   0.0     0.5
6         -73.91349         40.76314            2        19.0   0.5     0.5
  tip_amount tolls_amount improvement_surcharge total_amount
1          0            0                   0.3          8.8
2          0            0                   0.3         19.3
3          0            0                   0.3         34.3
4          0            0                   0.3         17.3
5          0            0                   0.3          8.8
6          0            0                   0.3         20.3
              pickup_city       pickup_name
1 New York City-Manhattan Greenwich Village
2 New York City-Manhattan      East Village
3                    <NA>              <NA>
4 New York City-Manhattan   Lower East Side
5 New York City-Manhattan   Upper East Side
6 New York City-Manhattan          Gramercy
```

Per eseguire la trasformazione sopra riportata a tutti i dati, dobbiamo prendere il codice sopra riportato e inserirlo in una funzione di trasformazione da passare a `rxDataStep` attraverso l'argomento `transfromFunc`. Chiamo poi la funzione di trasformazione `find_nhoods`. La funzione di trasformazione aggiungerà sia il quartiere di *pick-up* che il quartiere di *drop-off*.

```
find_nhoods <- function(data) {

  # estrae pick-up lat e long e trova i il quartiere corrispondente
  pickup_longitude<-ifelse(is.na(data$pickup_longitude),0,data$pickup_longitude)
  pickup_latitude<-ifelse(is.na(data$pickup_latitude),0,data$pickup_latitude)
  data_coords <- data.frame(long = pickup_longitude, lat = pickup_latitude)
  coordinates(data_coords) <- c('long', 'lat')
  nhoods <- over(data_coords, shapefile)

  ## aggiunge i quartieri di partenza e quartieri della città ai dati  data$pickup_nhood <- nhoods$NAME
  data$pickup_borough <- nhoods$CITY

  # estrae il drop-off lat e long e trova il quartiere corrispondente
```

```
  dropoff_longitude <- ifelse(is.na(data$dropoff_longitude),
                                    0, data$dropoff_longitude)
  dropoff_latitude <- ifelse(is.na(data$dropoff_latitude),
                                   0, data$dropoff_latitude)
  data_coords <- data.frame(long = dropoff_longitude, lat = dropoff_latitude)
  coordinates(data_coords) <- c('long', 'lat')
  nhoods <- over(data_coords, shapefile)

  ## aggiunge i quartieri di arrivo e quartieri della città ai dati
  data$dropoff_nhood <- nhoods$NAME
  data$dropoff_borough <- nhoods$CITY

  ## restituisce i dati con le nuove colonne aggiunte
  data
}
```

Ora che abbiamo la nostra funzione, è il momento di testarla. Possiamo farlo eseguendo `rxDataStep` sui dati di esempio `nyc_sample_df`. Questo è un buon modo per assicurarsi che la trasformazione funzioni prima di eseguirla sul file XDF `nyc_xdf`. A volte i messaggi di errore che otteniamo sono più informativi quando applichiamo la trasformazione a un `data.frame`, ed è più facile rintracciarli e debuggarli.

```
# testa la funzione su un data.frame usando rxDataStep
head(rxDataStep(nyc_sample_df, transformFunc = find_nhoods, transformPackages = c("sp", "maptools"),
                transformObjects = list(shapefile = mht_shapefile)))
```

```
VendorID tpep_pickup_datetime tpep_dropoff_datetime passenger_count trip_distance
1        2  2016-01-01 00:00:00   2016-01-01 00:00:00               2          1.10
2        2  2016-01-01 00:00:00   2016-01-01 00:00:00               5          4.90
3        2  2016-01-01 00:00:00   2016-01-01 00:00:00               1         10.54
4        2  2016-01-01 00:00:00   2016-01-01 00:00:00               1          4.75
5        2  2016-01-01 00:00:00   2016-01-01 00:00:00               3          1.76
6        2  2016-01-01 00:00:00   2016-01-01 00:18:30               2          5.52
  pickup_longitude pickup_latitude RatecodeID store_and_fwd_flag dropoff_longitude
1        -73.99037        40.73470          1                  N         -73.98184
2        -73.98078        40.72991          1                  N         -73.94447
3        -73.98455        40.67957          1                  N         -73.95027
4        -73.99347        40.71899          1                  N         -73.96224
5        -73.96062        40.78133          1                  N         -73.97726
6        -73.98012        40.74305          1                  N         -73.91349
  dropoff_latitude payment_type fare_amount extra mta_tax tip_amount tolls_amount
1         40.73241            2         7.5   0.5     0.5          0            0
```

```
2         40.71668            1        18.0   0.5     0.5          0            0
3         40.78893            1        33.0   0.5     0.5          0            0
4         40.65733            2        16.5   0.0     0.5          0            0
5         40.75851            2         8.0   0.0     0.5          0            0
6         40.76314            2        19.0   0.5     0.5          0            0
  improvement_surcharge total_amount      pickup_nhood          pickup_borough
1                   0.3          8.8 Greenwich Village New York City-Manhattan
2                   0.3         19.3      East Village New York City-Manhattan
3                   0.3         34.3       Boerum Hill  New York City-Brooklyn
4                   0.3         17.3   Lower East Side New York City-Manhattan
5                   0.3          8.8   Upper East Side New York City-Manhattan
6                   0.3         20.3          Gramercy New York City-Manhattan
             dropoff_nhood         dropoff_borough
1                Gramercy New York City-Manhattan
2                    <NA>                   <NA>
3               Yorkville New York City-Manhattan
4                    <NA>                   <NA>
5                 Midtown New York City-Manhattan
6 Astoria-Long Island City    New York City-Queens
```

Le ultime quattro colonne corrispondono ora ai quartieri che volevamo. Quindi la trasformazione non dovrebbe avere problemi ad essere applicata anche su `nyc_xdf`.

```
st <- Sys.time()
rxDataStep(nyc_xdf, nyc_xdf,
           overwrite = TRUE,
           transformFunc = find_nhoods,
           transformPackages = c("sp", "maptools", "rgeos"),
           transformObjects = list(shapefile = mht_shapefile))
Sys.time() - st
rxGetInfo(nyc_xdf, numRows = 5)
```

```
Time difference of 30.77251 mins

File name: C:\Data\NYC_taxi\yellow_tripdata_2016.xdf
Number of observations: 69406520
Number of variables: 29
Number of blocks: 141
Compression type: zlib
Data (5 rows starting with row 1):
  VendorID tpep_pickup_datetime tpep_dropoff_datetime passenger_count trip_distance
1        2  2016-01-01 00:00:00   2016-01-01 00:00:00               2          1.10
2        2  2016-01-01 00:00:00   2016-01-01 00:00:00               5          4.90
```

```
3        2  2016-01-01 00:00:00   2016-01-01 00:00:00               1         10.54
4        2  2016-01-01 00:00:00   2016-01-01 00:00:00               1          4.75
5        2  2016-01-01 00:00:00   2016-01-01 00:00:00               3          1.76
  pickup_longitude pickup_latitude RatecodeID store_and_fwd_flag dropoff_longitude
1        -73.99037        40.73470          1                  N         -73.98184
2        -73.98078        40.72991          1                  N         -73.94447
3        -73.98455        40.67957          1                  N         -73.95027
4        -73.99347        40.71899          1                  N         -73.96224
5        -73.96062        40.78133          1                  N         -73.97726
  dropoff_latitude payment_type fare_amount extra mta_tax tip_amount tolls_amount
1         40.73241            2         7.5   0.5     0.5          0            0
2         40.71668            1        18.0   0.5     0.5          0            0
3         40.78893            1        33.0   0.5     0.5          0            0
4         40.65733            2        16.5   0.0     0.5          0            0
5         40.75851            2         8.0   0.0     0.5          0            0
  improvement_surcharge total_amount tip_percent pickup_hour pickup_dow
1                   0.3          8.8           0    10PM-1AM        Fri
2                   0.3         19.3           0    10PM-1AM        Fri
3                   0.3         34.3           0    10PM-1AM        Fri
4                   0.3         17.3           0    10PM-1AM        Fri
5                   0.3          8.8           0    10PM-1AM        Fri
  dropoff_hour dropoff_dow trip_duration      pickup_nhood          pickup_borough
1     10PM-1AM         Fri             0 Greenwich Village New York City-Manhattan
2     10PM-1AM         Fri             0      East Village New York City-Manhattan
3     10PM-1AM         Fri             0       Boerum Hill  New York City-Brooklyn
4     10PM-1AM         Fri             0   Lower East Side New York City-Manhattan
5     10PM-1AM         Fri             0   Upper East Side New York City-Manhattan
  dropoff_nhood         dropoff_borough
1      Gramercy New York City-Manhattan
2          <NA>                   <NA>
3     Yorkville New York City-Manhattan
4          <NA>                   <NA>
5       Midtown New York City-Manhattan
```

La funzione ha eseguito il suo lavoro su tutto il *dataset* in poco più di 30 minuti.