Subway Data Analysis

Introduction

O sistema de ônibus e trens de Nova Iorque - o Metro Transit Authority - fornece seus dados para download através de arquivos csv. Uma das informações disponíveis são os dados das catracas do metrô que contém logs semanais de entradas cumulativas e saídas por catraca por estação de metrô em algum intervalo de tempo.

Neste projeto iremos utilizar apenas os das catraca disponíveis em:
http://web.mta.info/developers/turnstile.html (http://web.mta.info/developers/turnstile.html).

Sobre este projeto

Neste projeto você irá aplicar todos os conhecimentos adquiridos neste primeiro mês de curso. Iremos praticar tarefas básicas de aquisição, limpeza de dados e nesse processo iremos descobrir coisas essenciais sobre os dados utilizando o que foi aprendido no curso de estatística.

O objetivo deste projeto é explorar a relação entre os dados das catracas do metro de Nova Iorque e o clima no dia da coleta. Para isso, além dos dados do metrô, precisaremos os dados de clima da cidade de Nova Iorque.

Os principais pontos que serão verificados neste trabalho:

- Coleta de dados da internet
- Utilização de estatística para análise de dados
- Manipulação de dados e criação de gráficos simples com o `Pandas`

Como conseguir ajuda: Sugerimos que tente os seguintes canais, nas seguintes ordens:

Tipo de dúvida\Canais	**Google**	**Fórum**	**Slack**	**Email**
Programação Pyhon e Pandas	1	2	3	
Requisitos do projeto		1	2	3
Partes específicas do Projeto		1	2	3

Os endereços dos canais são:

- Fórum: https://discussions.udacity.com/c/ndfdsi-project (https://discussions.udacity.com/c/ndfdsi-project)
- Slack: udacity-br.slack.com (https://udacity-br.slack.com/messages/C5MT6E3E1)
- Email: data-suporte@udacity.com

Espera-se que o estudante entregue este relatório com:

- Todos os TODO feitos, pois eles são essenciais para que o código rode corretamente
- O arquivo ipynb exportado como html

Para entregar este projeto, vá a sala de aula (https://coco.udacity.com/nanodegrees/nd111/locale/pt-br/versions/1.0.0/parts/339726/modules/339733/lessons/340886/project) e submeta o seu `.ipynb` e o html, zipados.

Lembretes

Antes de começarmos, alguns lembretes devem ter em mente ao usar os notebooks iPython:

- Lembre-se de que você pode ver do lado esquerdo de uma célula de código quando foi executado pela última vez se houver um número dentro das chaves.
- Quando você inicia uma nova sessão do notebook, certifique-se de executar todas as células até o ponto em que você deixou a última vez. Mesmo que a saída ainda seja visível a partir de quando você executou as células em sua sessão anterior, o kernel começa em um estado novo, então você precisará recarregar os dados, etc. em uma nova sessão.
- O ponto anterior é útil para ter em mente se suas respostas não correspondem ao que é esperado nos questionários da aula. Tente recarregar os dados e execute todas as etapas de processamento um a um para garantir que você esteja trabalhando com as mesmas variáveis e dados que estão em cada fase do questionário.

Seção 1 - Coleta de Dados

Exercicio 1.1

Mãos a obra!! Agora é sua vez de coletar os dados. Escreva abaixo um código python que acesse o link http://web.mta.info/developers/turnstile.html (http://web.mta.info/developers/turnstile.html) e baixe os arquivos do mês de junho de 2017. O arquivo deverá ser salvo com o nome turnstile_100617.txt onde 10/06/17 é a data do arquivo.

Abaixo seguem alguns comandos que poderão te ajudar:

Utilize a biblioteca **urllib** para abrir e resgatar uma página da web. Utilize o comando abaixo onde **url** será o caminho da página da web onde se encontra o arquivo:

```
u = urllib.urlopen(url)
html = u.read()
```

Utilize a biblioteca **BeautifulSoup** para procurar na página pelo link do arquivo que deseja baixar. Utilize o comando abaixo para criar o seu objeto *soup* e procurar por todas as tags 'a'no documento:

```
soup = BeautifulSoup(html, "html.parser")
links = soup.find_all('a')
```

Uma dica para baixar apenas os arquivos do mês de junho é verificar a data no nome do arquivo. Por exemplo, para baixar o arquivo do dia 17/06/2017 verifique se o link termina com *"turnstile_170610.txt"*. Se não fizer isso você baixará todos os arquivos da página. Para fazer isso utilize o comando conforme abaixo:

```
if '1706' in link.get('href'):
```

E a dica final é utilizar o comando abaixo para fazer o download do arquivo txt:

```
urllib.urlretrieve(link_do_arquivo, filename)
```

Lembre-se, primeiro, carregue todos os pacotes e funções que você estará usando em sua análise.

In [1]:

```
import urllib
from bs4 import BeautifulSoup

#your code here
# definie a url base
baseURL = "http://web.mta.info/developers/"

# le o arquivo e obtem o conteudo
u = urllib.urlopen(baseURL + "turnstile.html")
html = u.read()

# faz parse do html
soup = BeautifulSoup(html, "html.parser")
links = soup.find_all('a')
aLink = None

# obtem o link dos arquivos e faz o download
for link in links:
    aLink = str(link.get('href'))
    if '1706' in aLink:
        link_do_arquivo = baseURL + aLink
        filename = link_do_arquivo[link_do_arquivo.rfind("/")+1:]
        urllib.urlretrieve(link_do_arquivo, filename)
```

Exercicio 1.2

Escreva uma função que pegue a lista de nomes dos arquivos que você baixou no exercicio 1.1 e consolide-os em um único arquivo. Deve existir apenas uma linha de cabeçalho no arquivo de saida.

Por exemplo, se o arquivo_1 tiver: linha 1... linha 2...

e o outro arquivo, arquivo_2 tiver: linha 3... linha 4... linha 5...

Devemos combinar o arquivo_1 com arquivo_2 em um arquivo mestre conforme abaixo:

'C/A, UNIT, SCP, DATEn, TIMEn, DESCn, ENTRIESn, EXITSn' linha 1... linha 2... linha 3... linha 4... linha 5...

In [2]:

```
def create_master_turnstile_file(filenames, output_file):
    with open(output_file, 'w') as master_file:
        master_file.write('C/A,UNIT,SCP,STATION, LINENAME, DIVISION, DATEn,TIMEn,DESC
n,ENTRIESn,EXITSn\n')

        # itera os arquivos para criar um unico arquivo
        for filename in filenames:
            # your code here
            with open(filename, 'r') as file:
                skip = True
                for line in file:
                    # se for a primeira linha pula o cabecalho
                    if skip:
                        skip = False
                        continue
                    master_file.write(line)

filenames = ['turnstile_170603.txt','turnstile_170610.txt','turnstile_170617.txt','tu
rnstile_170624.txt']
outputFile = 'turnstile_data.csv'

create_master_turnstile_file(filenames,outputFile)
```

Exercicio 1.3

Neste exercício, escreva um função que leia o master_file criado no exercicio anterior e carregue-o em um pandas dataframe. Esta função deve filtrar para que o dataframe possua apenas linhas onde a coluna "DESCn" possua o valor "Regular".

Por exemplo, se o data frame do pandas estiver conforme abaixo:

```
,C/A,UNIT,SCP,DATEn,TIMEn,DESCn,ENTRIESn,EXITSn
0,A002,R051,02-00-00,05-01-11,00:00:00,REGULAR,3144312,1088151
1,A002,R051,02-00-00,05-01-11,04:00:00,DOOR,3144335,1088159
2,A002,R051,02-00-00,05-01-11,08:00:00,REGULAR,3144353,1088177
3,A002,R051,02-00-00,05-01-11,12:00:00,DOOR,3144424,1088231
```

O dataframe deverá ficar conforme abaixo depois de filtrar apenas as linhas onde a coluna DESCn possua o valor REGULAR:

```
0,A002,R051,02-00-00,05-01-11,00:00:00,REGULAR,3144312,1088151
2,A002,R051,02-00-00,05-01-11,08:00:00,REGULAR,3144353,1088177
```

In [3]:

```
import pandas as pd

def filter_by_regular(filename):

    turnstile_data = pd.read_csv(filename)
    # more of your code here

    # filtra as linhas quando DESCn = REGULAR
    turnstile_data = turnstile_data[turnstile_data['DESCn'] == 'REGULAR']

    return turnstile_data

df = filter_by_regular('turnstile_data.csv')
df.head(5)
```

Out[3]:

	C/A	UNIT	SCP	STATION	LINENAME	DIVISION	DATEn	TIMEn	DESCn
0	A002	R051	02-00-00	59 ST	NQR456W	BMT	05/27/2017	00:00:00	REGULAR
1	A002	R051	02-00-00	59 ST	NQR456W	BMT	05/27/2017	04:00:00	REGULAR
2	A002	R051	02-00-00	59 ST	NQR456W	BMT	05/27/2017	08:00:00	REGULAR
3	A002	R051	02-00-00	59 ST	NQR456W	BMT	05/27/2017	12:00:00	REGULAR
4	A002	R051	02-00-00	59 ST	NQR456W	BMT	05/27/2017	16:00:00	REGULAR

Exercicio 1.4

Os dados do metrô de NY possui dados cumulativos de entradas e saidas por linha. Assuma que você possui um dataframe chamado df que contém apenas linhas para uma catraca em particular (unico SCP, C/A, e UNIT). A função abaixo deve alterar essas entradas cumulativas para a contagem de entradas desde a última leitura (entradas desde a última linha do dataframe).

Mais especificamente, você deverá fazer duas coisas:

1 - Criar uma nova coluna chamada ENTRIESn_hourly 2 - Inserir nessa coluna a diferença enre ENTRIESn da coluna atual e a da coluna anterior. Se a linha possuir alguma NAN, preencha/substitua por 1.

Dica: as funções do pandas shift() e fillna() pode ser úteis nesse exercicio.

Abaixo tem um exemplo de como seu dataframe deve ficar ao final desse exercicio:

```
     C/A  UNIT       SCP     DATEn     TIMEn    DESCn  ENTRIESn    EXITSn  ENTRIE
Sn_hourly
0     A002  R051  02-00-00  05-01-11  00:00:00  REGULAR   3144312   1088151
           1
1     A002  R051  02-00-00  05-01-11  04:00:00  REGULAR   3144335   1088159
          23
2     A002  R051  02-00-00  05-01-11  08:00:00  REGULAR   3144353   1088177
          18
3     A002  R051  02-00-00  05-01-11  12:00:00  REGULAR   3144424   1088231
          71
4     A002  R051  02-00-00  05-01-11  16:00:00  REGULAR   3144594   1088275
         170
5     A002  R051  02-00-00  05-01-11  20:00:00  REGULAR   3144808   1088317
         214
6     A002  R051  02-00-00  05-02-11  00:00:00  REGULAR   3144895   1088328
          87
7     A002  R051  02-00-00  05-02-11  04:00:00  REGULAR   3144905   1088331
          10
8     A002  R051  02-00-00  05-02-11  08:00:00  REGULAR   3144941   1088420
          36
9     A002  R051  02-00-00  05-02-11  12:00:00  REGULAR   3145094   1088753
         153
10    A002  R051  02-00-00  05-02-11  16:00:00  REGULAR   3145337   1088823
         243
```

In [5]:

```
import pandas

def set_ENTRIESn_hourly(dt):
    # calcula a diferenca entre o valor da linha atual e a anterior
    entryn_hourly = dt['ENTRIESn'] - dt['ENTRIESn_Aux']
    return entryn_hourly

def get_hourly_entries(df):

    #your code here

    # cria a coluna nova para receber o valor e a auxiliar para obter o valor da linha
 anterior
    df['ENTRIESn_hourly'] = 0
    df['ENTRIESn_Aux'] = df['ENTRIESn'].shift(1)

    # chama a funcao para calcular a diferenca entre os valores
    df['ENTRIESn_hourly'] = df.apply(set_ENTRIESn_hourly,axis=1)

    # subistitui os valores nulos por 1
    df['ENTRIESn_hourly'].fillna('1', inplace=True)

    # elimina a coluna auxiliar que foi usada para o calculo da diferenca
    df.drop('ENTRIESn_Aux', axis=1, inplace=True)

    return df

get_hourly_entries(df)
df.head(5)
```

Out[5]:

	C/A	UNIT	SCP	STATION	LINENAME	DIVISION	DATEn	TIMEn	DESCn
0	A002	R051	02-00-00	59 ST	NQR456W	BMT	05/27/2017	00:00:00	REGULAR
1	A002	R051	02-00-00	59 ST	NQR456W	BMT	05/27/2017	04:00:00	REGULAR
2	A002	R051	02-00-00	59 ST	NQR456W	BMT	05/27/2017	08:00:00	REGULAR
3	A002	R051	02-00-00	59 ST	NQR456W	BMT	05/27/2017	12:00:00	REGULAR
4	A002	R051	02-00-00	59 ST	NQR456W	BMT	05/27/2017	16:00:00	REGULAR

Exercicio 1.5

Faça o mesmo do exercicio anterior mas agora considerando as saidas, coluna EXITSn. Para isso crie uma coluna chamada de EXITSn_hourly e insira a diferença entre a coluna EXITSn da linha atual versus a linha anterior. Se tiver algum NaN, preencha/substitua por 0.

In [6]:

```
import pandas

def set_EXITSn_hourly(dt):
    # calcula a diferenca entre o valor da linha atual e a anterior
    existn_hourly = dt['EXITSn'] - dt['EXITSn_Aux']
    return existn_hourly

def get_hourly_exits(df):

    #your code here

    # cria a coluna nova para receber o valor e a auxiliar para obter o valor da linha
 anterior
    df['EXITSn_hourly'] = 0
    df['EXITSn_Aux'] = df['EXITSn'].shift(1)

    # chama a funcao para calcular a diferenca entre os valores
    df['EXITSn_hourly'] = df.apply(set_EXITSn_hourly,axis=1)

    # subistitui os valores nulos por 1
    df['EXITSn_hourly'].fillna('1', inplace=True)

    # elimina a coluna auxiliar que foi usada para o calculo da diferenca
    df.drop('EXITSn_Aux', axis=1, inplace=True)

    return df

get_hourly_exits(df)
df.head(5)
```

Out[6]:

	C/A	UNIT	SCP	STATION	LINENAME	DIVISION	DATEn	TIMEn	DESCn
0	A002	R051	02-00-00	59 ST	NQR456W	BMT	05/27/2017	00:00:00	REGULAR
1	A002	R051	02-00-00	59 ST	NQR456W	BMT	05/27/2017	04:00:00	REGULAR
2	A002	R051	02-00-00	59 ST	NQR456W	BMT	05/27/2017	08:00:00	REGULAR
3	A002	R051	02-00-00	59 ST	NQR456W	BMT	05/27/2017	12:00:00	REGULAR
4	A002	R051	02-00-00	59 ST	NQR456W	BMT	05/27/2017	16:00:00	REGULAR

Exercicio 1.6

Dado uma variável de entrada que representa o tempo no formato de: "00:00:00" (hora: minutos: segundos) Escreva uma função para extrair a parte da hora do tempo variável de entrada E devolva-o como um número inteiro. Por exemplo: 1) se a hora for 00, seu código deve retornar 0 2) se a hora for 01, seu código deve retornar 1 3) se a hora for 21, seu código deve retornar 21 Por favor, devolva a hora como um número inteiro.

In [7]:

```
def time_to_hour(time):

    # obtem apenas o valor da hora do campo
    hour = int(time[:time.find(":")])
    return hour

print 'A hora é = {:n}.'.format(time_to_hour('21:34:50'))
```

```
A hora é = 21.
```

Exercicio 2 - Análise dos dados

Exercicio 2.1

Para verificar a relação entre o movimento do metrô e o clima, precisaremos complementar os dados do arquivo já baixado com os dados do clima. Nós complementamos para você este arquivo com os dados de clima de Nova Iorque e disponibilizamos na área de materiais do projeto. Você pode acessa-lo pelo link: https://s3.amazonaws.com/content.udacity-data.com/courses/ud359/turnstile_data_master_with_weather.csv (https://s3.amazonaws.com/content.udacity-data.com/courses/ud359/turnstile_data_master_with_weather.csv)

Agora que temos nossos dados em um arquivo csv, escreva um código python que leia este arquivo e salve-o em um data frame do pandas.

Dica:

Utilize o comando abaixo para ler o arquivo:

```
pd.read_csv('output_list.txt', sep=",")
```

In [8]:

```
import pandas as pd

filename = "turnstile_data_master_with_weather.csv"

#your code here

# carrega o arquivo
df_weather = pd.read_csv(filename, sep=",")
df_weather.head(5)
```

Out[8]:

	Unnamed: 0	UNIT	DATEn	TIMEn	Hour	DESCn	ENTRIESn_hourly	EXITSn_hc
0	0	R001	2011-05-01	01:00:00	1	REGULAR	0.0	0.0
1	1	R001	2011-05-01	05:00:00	5	REGULAR	217.0	553.0
2	2	R001	2011-05-01	09:00:00	9	REGULAR	890.0	1262.0
3	3	R001	2011-05-01	13:00:00	13	REGULAR	2451.0	3708.0
4	4	R001	2011-05-01	17:00:00	17	REGULAR	4400.0	2501.0

5 rows × 22 columns

Exercicio 2.2

Agora crie uma função que calcule a quantidade de dias chuvosos, para isso retorne a contagem do numero de dias onde a coluna *"rain"* é igual a 1.

Dica: Você também pode achar que a interpretação de números como números inteiros ou float pode não funcionar inicialmente. Para contornar esta questão, pode ser útil converter esses números para números inteiros. Isso pode ser feito escrevendo cast (coluna como inteiro). Então, por exemplo, se queríamos lançar a coluna maxtempi como um número inteiro, nós devemos escrever algo como cast (maxtempi as integer) = 76, em oposição a simplesmente onde maxtempi = 76.

In [11]:

```
def num_rainy_days(df):

    #your code here

    num_rainy = 0
    # filtra os registros quando a coluna rain = 1
    df_filtered = df[(df['rain'] == 1)]

    # retorna a quantidade de registros encontrados
    num_rainy = df_filtered['rain'].count()
    return num_rainy

print 'A quantidade de dias chuvosos é = {:n}.'.format(num_rainy_days(df_weather))
```

```
A quantidade de dias chuvosos é = 44104.
```

Exercicio 2.3

Calcule se estava nebuloso ou não (0 ou 1) e a temperatura máxima para fog (isto é, a temperatura máxima para dias nebulosos).

In [12]:

```
def max_temp_aggregate_by_fog(df):

    max_temp = 0
    #your code here

    # filtra os registros quando a coluna fog = 1
    df = df[df['fog'] ==1]

    # retorna a temperatura maxima dos registros encontrados
    max_temp = df['maxtempi'].max()

    return max_temp


print 'A temperatura máxima para dias nebulosos é = {:n}.'.format(max_temp_aggregate_by
_fog(df_weather))
```

```
A temperatura máxima para dias nebulosos é = 81.
```

*Exercicio 2.4

Calcule agora a média de 'meantempi' nos dias que são sábado ou domingo (finais de semana):

In [13]:

```
from datetime import datetime

def set_weekday(dt):
    # obtem o dia da semana da data
    week_day = datetime.strptime(dt['DATEn'], '%Y-%m-%d').weekday()
    return week_day

def avg_weekend_temperature(filename):

    # carrega o arquivo
    avg_weekend_temp = pd.read_csv(filename, sep=",")

    # cria coluna para o dia da semana
    avg_weekend_temp['WEEKDAYn'] = -1

    # chama a funcao para obter o dia da semana
    avg_weekend_temp['WEEKDAYn'] = avg_weekend_temp.apply(set_weekday,axis=1)

    # filtra os registros que sejam sabado e domingo
    avg_weekend_temp = avg_weekend_temp[(avg_weekend_temp['WEEKDAYn'] == 5) | (avg_week
end_temp['WEEKDAYn'] == 6)]

    # retorna a media dos registros encontrados
    mean_temp_weekends = avg_weekend_temp['meantempi'].mean()

    return mean_temp_weekends

print 'A média de \'meantempi\' aos finais de semana é = {:.2f}.'.format(avg_weekend_te
mperature('turnstile_data_master_with_weather.csv'))
```

```
A média de 'meantempi' aos finais de semana é = 65.10.
```

*Exercicio 2.5

Calcule a média da temperatura mínima 'mintempi' nos dias chuvosos onde da temperatura mínima foi maior que do 55 graus:

In [14]:

```
def avg_min_temperature(filename):

    # carrregar arquivo csv
    avg_min_temp = pd.read_csv(filename, sep=",")

    avg_min_temp_rainy = 0

    # filtra os registros onde rain =1 e mintempi > 55
    avg_min_temp = avg_min_temp[(avg_min_temp['rain'] == 1) & (avg_min_temp['mintempi']
 > 55)]

    # retorna a media para os registros encontrados
    avg_min_temp_rainy = avg_min_temp['mintempi'].mean()

    return avg_min_temp_rainy

print 'A média da temperatura mínima nos dias chuvosos onde a temperatura mínima > 55 g
raus é = {:0.2f}.'.format(avg_min_temperature('turnstile_data_master_with_weather.csv'
))
```

```
A média da temperatura mínima nos dias chuvosos onde a temperatura mínima
> 55 graus é = 61.24.
```

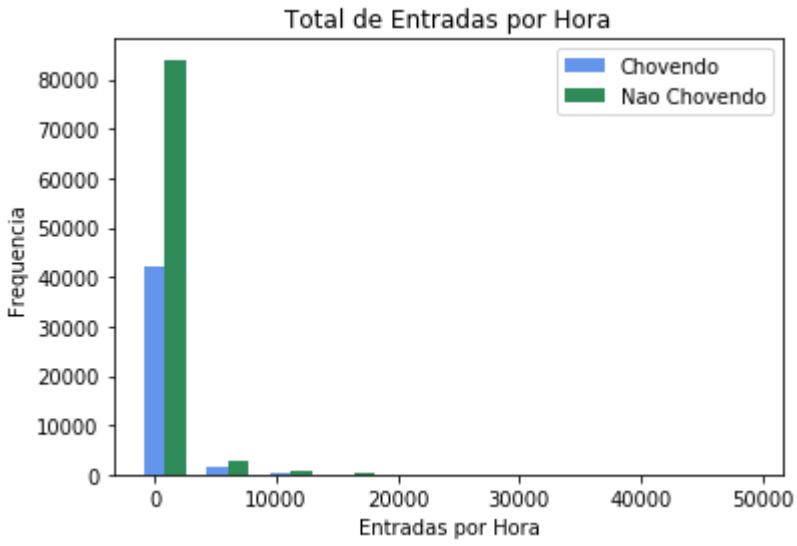
*Exercicio 2.6

Antes de realizar qualquer análise, pode ser útil olhar para os dados que esperamos analisar. Mais especificamente, vamos examinR as entradas por hora em nossos dados do metrô de Nova York para determinar a distribuição dos dados. Estes dados são armazenados na coluna ['ENTRIESn_hourly'].
Trace dois histogramas nos mesmos eixos para mostrar as entradas quando esta chovendo vs quando não está chovendo. Abaixo está um exemplo sobre como traçar histogramas com pandas e matplotlib:

```
Turnstile_weather ['column_to_graph']. Hist ()
```

In [15]:

```
import numpy as np
import pandas
import matplotlib.pyplot as plt

def entries_histogram(turnstile_weather):

    # your code here to plot a historgram for hourly entries when it is raining

    # filtro os registros para dias chuvosos
    turnstile_weather_rain = turnstile_weather[(turnstile_weather['rain'] == 1)]

    # your code here to plot a histogram for hourly entries when it is not raining
    # filtro os registros para dias nao chuvosos
    turnstile_weather_not_rain = turnstile_weather[(turnstile_weather['rain'] == 0)]

    # define as informacoes do grafico
    fig, ax = plt.subplots()

    a_heights, a_bins = np.histogram(turnstile_weather_rain['ENTRIESn_hourly'])
    b_heights, b_bins = np.histogram(turnstile_weather_not_rain['ENTRIESn_hourly'], bin
s=a_bins)

    width = (a_bins[1] - a_bins[0])/3
    ax.bar(a_bins[:-1], a_heights, width=width, facecolor='cornflowerblue',label='Chove
ndo')
    ax.bar(b_bins[:-1]+width, b_heights, width=width, facecolor='seagreen', label='Nao
 Chovendo')
    ax.legend()

    # define as as legendas do grafico
    plt.title('Total de Entradas por Hora')
    plt.ylabel('Frequencia')
    plt.xlabel('Entradas por Hora')

    return plt

plt = entries_histogram(df_weather)
plt.show()
```



*Exercicio 2.7

Os dados que acabou de plotar que tipo de ditribuição? Existe diferença na distribuição entre dias chuvosos e não chuvosos?

Resposta :

- O tipo de distribuição é Obliqua Negativa
- Sim, podemos observar que nos dias de chuva temos uma redução do número de entradas.

*Exercicio 2.8

Construa uma função que que retorne:

1. A média das entradas com chuva
2. A média das entradas sem chuva

In [16]:

```
import numpy as np

import pandas

def means(turnstile_weather):

    ### YOUR CODE HERE ###

    # filtra os registros para dias chuvosos
    mean_entries = turnstile_weather[(turnstile_weather['rain'] == 1)]
    # retorna a media para dias chuvosos
    with_rain_mean = mean_entries['ENTRIESn_hourly'].mean()

    # filtra os registros para dias chuvosos
    mean_entries = turnstile_weather[(turnstile_weather['rain'] == 0)]
    # retorna a media para dias nao chuvosos
    without_rain_mean = mean_entries['ENTRIESn_hourly'].mean()

    p = 'grader'

    return with_rain_mean, without_rain_mean, p # leave this line for the grader


with_rain_mean, without_rain_mean, p = means(df_weather)

print "A média das entradas com chuva é = {:.2f}.".format(with_rain_mean)
print "A média das entradas sem chuva é = {:.2f}.".format(without_rain_mean)
```

```
A média das entradas com chuva é = 1105.45.
A média das entradas sem chuva é = 1090.28.
```

Responda as perguntas abaixo de acordo com a saida das suas funções:

1. Qual a média das entradas com chuva?
2. Qual a média das entradas sem chuva?

Resposta :

- A média das entradas com chuva é = 1105.45.
- A média das entradas sem chuva é = 1090.28.

Exercicio 3 - Map Reduce

Exercicio 3.1

A entrada para esse exercício e o mesmo arquivo da seção anterior (Exercicio 2). Você pode baixar o arquivo neste link:

https://s3.amazonaws.com/content.udacity-data.com/courses/ud359/turnstile_data_master_with_weather.csv (https://s3.amazonaws.com/content.udacity-data.com/courses/ud359/turnstile_data_master_with_weather.csv)

Varmos criar um mapeador agora. Para cada linha de entrada, a saída do mapeador deve IMPRIMIR (não retornar) a UNIT como uma chave e o número de ENTRIESn_hourly como o valor. Separe a chave e o valor por uma guia. Por exemplo: 'R002 \ t105105.0'

Exporte seu mapeador em um arquivo chamado mapper_result.txt e envie esse arquivo juntamente com a sua submissão. O código para exportar seu mapeador já está escrito no código abaixo.

In [23]:

```
import sys
import csv

def mapper():

    # your code here

    # carrega o arquivos
    reader = csv.reader(sys.stdin, delimiter=',')
    writer = csv.writer(sys.stdout, delimiter='\t', quotechar='"', quoting=csv.QUOTE_NO
NE)

    skipHeader = True

    for line in reader:
        # pula a linha do header
        if skipHeader:
            skipHeader = False
            continue

        writer.writerow([line[1],line[6]])

sys.stdin = open('turnstile_data_master_with_weather.csv')
sys.stdout = open('mapper_result.txt', 'wb')
mapper()
```

Exercicio 3.2

Agora crie o redutor. Dado o resultado do mapeador do exercicio anterior, o redutor deve imprimir(Não retornar) uma linha por UNIT, juntamente com o número total de ENTRIESn_hourly.Ao longo de maio (que é a duração dos nossos dados), separados por uma guia. Um exemplo de linha de saída do redutor pode ser assim: 'R001 \ t500625.0'

Você pode assumir que a entrada para o redutor está ordenada de tal forma que todas as linhas correspondentes a uma unidade particular são agrupados. No entanto a saida do redutor terá repetição pois existem lojas que aparecem em locais diferentes dos arquivos.

Exporte seu redutor em um arquivo chamado reducer_result.txt e envie esse arquivo juntamente com a sua submissão.

In [25]:

```
import sys
import csv

def reducer():

    # your code here

    # variaveis de controle de chave
    oldKey = None
    entriesTotal = 0

    # carrega arquivos
    reader = csv.reader(sys.stdin, delimiter='\t')
    writer = csv.writer(sys.stdout, delimiter='\t', quotechar='"', quoting=csv.QUOTE_NO
NE)

    for line in reader:

        # obtem os valores da linha
        thisUnit, thisEntry = line

        # faz a sumarizacao se mudou a Unit
        if oldKey and oldKey != thisUnit:
                writer.writerow((oldKey, entriesTotal))
                entriesTotal = 0

        oldKey = thisUnit
        entriesTotal += float(thisEntry)

    # imprime a ultima linha
    if oldKey:
        writer.writerow((oldKey, entriesTotal))

sys.stdin = open('mapper_result.txt')
sys.stdout = open('reducer_result.txt', 'wb')
reducer()
```