

TP1-OrgDeDatos

April 17, 2017

```
In [1]: # Alumnos:
        # Mariano Kakazu, 98178
        # Rodrigo Aparicio, 98967
        # Thomas Cordeu, 99288

In [2]: import numpy as np
        import pandas as pd
        import datetime # para convertir a día de la semana
        import calendar # idem

        # plots
        import matplotlib.pyplot as plt
        import seaborn as sns
        %matplotlib inline

In [3]: trips = pd.read_csv('../data/trip.csv', low_memory=False)
        #Se convierte los dates a datetime64[ns].
        trips['start_date'] = pd.to_datetime(trips['start_date'], format='%m/%d/%Y %H:%M')
        trips['end_date'] = pd.to_datetime(trips['end_date'], format='%m/%d/%Y %H:%M')

In [4]: # ejemplo del uso de datetime con día actual
        dia_actual = datetime.datetime.today()
        dia_actual

Out[4]: datetime.datetime(2017, 4, 16, 23, 8, 43, 295355)

In [5]: # lo paso a día de la semana
        dia_actual.weekday()

Out[5]: 6

In [6]: # mejor en palabras que en números
        calendar.day_name[dia_actual.weekday()]

Out[6]: 'Sunday'

In [7]: # función para convertir fecha a día de la semana.
        def fecha_a_dia(fecha):
            return calendar.day_name[fecha.weekday()]
```

```

In [8]: #Se crean columnas con los dias de la semana.
trips['start_day_of_week'] = trips['start_date'].map(fecha_a_dia)
trips['end_day_of_week'] = trips['end_date'].map(fecha_a_dia)

In [9]: trips['horario_inicial'] = trips['start_date'].dt.time
trips['horario_inicial_float'] = trips['start_date'].dt.hour + trips['start_date'].dt.minute

In [10]: trips['horario_final'] = trips['end_date'].dt.time
trips['horario_final_float'] = trips['end_date'].dt.hour + trips['end_date'].dt.minute

In [11]: trips['duracion_viaje'] = trips['end_date'] - trips['start_date']

In [12]: # chequeo tipos
trips.dtypes

Out[12]: id                                int64
duration                                int64
start_date                             datetime64[ns]
start_station_name                      object
start_station_id                       int64
end_date                               datetime64[ns]
end_station_name                       object
end_station_id                         int64
bike_id                               int64
subscription_type                      object
zip_code                              object
start_day_of_week                      object
end_day_of_week                        object
horario_inicial                        object
horario_inicial_float                  float64
horario_final                          object
horario_final_float                    float64
duracion_viaje                        timedelta64[ns]
dtype: object

In [13]: # vista final de cómo quedó el dataframe
trips.head()

Out[13]:
```

	id	duration	start_date	start_station_name	\
0	4576	63	2013-08-29 14:13:00	South Van Ness at Market	
1	4607	70	2013-08-29 14:42:00	San Jose City Hall	
2	4130	71	2013-08-29 10:16:00	Mountain View City Hall	
3	4251	77	2013-08-29 11:29:00	San Jose City Hall	
4	4299	83	2013-08-29 12:02:00	South Van Ness at Market	

	start_station_id	end_date	end_station_name	\
0	66	2013-08-29 14:14:00	South Van Ness at Market	
1	10	2013-08-29 14:43:00	San Jose City Hall	
2	27	2013-08-29 10:17:00	Mountain View City Hall	

3	10	2013-08-29	11:30:00	San Jose City Hall
4	66	2013-08-29	12:04:00	Market at 10th

	end_station_id	bike_id	subscription_type	zip_code	start_day_of_week	\
0	66	520	Subscriber	94127	Thursday	
1	10	661	Subscriber	95138	Thursday	
2	27	48	Subscriber	97214	Thursday	
3	10	26	Subscriber	95060	Thursday	
4	67	319	Subscriber	94103	Thursday	

	end_day_of_week	horario_inicial	horario_inicial_float	horario_final	\
0	Thursday	14:13:00	14.13	14:14:00	
1	Thursday	14:42:00	14.42	14:43:00	
2	Thursday	10:16:00	10.16	10:17:00	
3	Thursday	11:29:00	11.29	11:30:00	
4	Thursday	12:02:00	12.02	12:04:00	

	horario_final_float	duracion_viaje
0	14.14	00:01:00
1	14.43	00:01:00
2	10.17	00:01:00
3	11.30	00:01:00
4	12.04	00:02:00

0.1 ¿Viajes de menos de 3 minutos con misma estación de inicio y fin?

Una primera impresión es que hay viajes "no tomados": duran menos de 3 minutos y las estaciones de inicio y fin son las mismas. Esto nos dice que el usuario no tomó el viaje por algún motivo, se puede pensar que hubo un desperfecto técnico en la bicicleta. Se van a filtrar esos viajes.

```
In [14]: viajes_no_tomados = trips[(trips['duracion_viaje'] <= '00:03:00') & (trips['start_station_id'] == trips['end_station_id'])
trips = trips[~((trips['duracion_viaje'] <= '00:03:00') & (trips['start_station_id'] == trips['end_station_id']))]
```

```
In [15]: viajes_no_tomados.id.count()
```

```
Out[15]: 2601
```

Se filtraron unos 2600 viajes.

0.1.1 Ahora analicemos lo opuesto, qué tan largo puede ser un viaje.

```
In [16]: trips[(trips['duracion_viaje'] > "12:00:00")].head()
```

Out[16]:	id	duration	start_date	start_station_name	\
743	4663	52698	2013-08-29 15:34:00	Mountain View City Hall	
744	4532	84990	2013-08-29 13:43:00	Market at 4th	
745	4521	85385	2013-08-29 13:37:00	Market at 4th	
746	5069	86102	2013-08-29 21:41:00	Embarcadero at Folsom	
747	4505	97713	2013-08-29 13:30:00	Mountain View Caltrain Station	

	start_station_id	end_date	\
743	27	2013-08-30 06:12:00	
744	76	2013-08-30 13:19:00	
745	76	2013-08-30 13:20:00	
746	51	2013-08-30 21:37:00	
747	28	2013-08-30 16:38:00	

	end_station_name	end_station_id	bike_id	\
743	Park at Olive	38	150	
744	Harry Bridges Plaza (Ferry Building)	50	460	
745	Harry Bridges Plaza (Ferry Building)	50	390	
746	Davis at Jackson	42	269	
747	Mountain View City Hall	27	141	

	subscription_type	zip_code	start_day_of_week	end_day_of_week	\
743	Subscriber	94301	Thursday	Friday	
744	Customer	94118	Thursday	Friday	
745	Customer	94118	Thursday	Friday	
746	Customer	94111	Thursday	Friday	
747	Subscriber	94039	Thursday	Friday	

	horario_inicial	horario_inicial_float	horario_final	horario_final_float	\
743	15:34:00	15.34	06:12:00	6.12	
744	13:43:00	13.43	13:19:00	13.19	
745	13:37:00	13.37	13:20:00	13.20	
746	21:41:00	21.41	21:37:00	21.37	
747	13:30:00	13.30	16:38:00	16.38	

	duracion_viaje
743	0 days 14:38:00
744	0 days 23:36:00
745	0 days 23:43:00
746	0 days 23:56:00
747	1 days 03:08:00

0.2 ¿Viajes de más de 12 horas de duración?

Se pueden esperar viajes muy largos de a lo sumo 12 horas de alguien que saliendo desde la mañana decidió recorrer muchos puntos de la ciudad y aparte en el medio ir parando, pero ya cuando se excede esto hasta casos que incluso superan un día de uso creemos que hubo datos mal cargados o algún otro problema como la incorrecta devolución de la bicicleta. Por eso se van a filtrar esos viajes.

```
In [17]: viajes_larguisimos = trips[(trips['duracion_viaje'] > "12:00:00") | ((trips['duracion_v
# lo que está después del or es para los viajes que duran entre 11 y 12hs y no empiezan
trips = trips[-((trips['duracion_viaje'] > "12:00:00") | ((trips['duracion_viaje'] >= "
```

```
In [18]: viajes_larguisimos.id.count()
```

Out[18]: 1250

Se filtraron unos 1250 viajes.

```
In [19]: # ahora los datos quedan más limpios
trips.head()
```

```
Out[19]:
```

	id	duration	start_date	start_station_name	\
4	4299	83	2013-08-29 12:02:00	South Van Ness at Market	
6	4500	109	2013-08-29 13:25:00	Santa Clara at Almaden	
9	4258	114	2013-08-29 11:33:00	San Jose City Hall	
10	4549	125	2013-08-29 13:52:00	Spear at Folsom	
11	4498	126	2013-08-29 13:23:00	San Pedro Square	

	start_station_id	end_date	end_station_name	\
4	66	2013-08-29 12:04:00	Market at 10th	
6	4	2013-08-29 13:27:00	Adobe on Almaden	
9	10	2013-08-29 11:35:00	MLK Library	
10	49	2013-08-29 13:55:00	Embarcadero at Bryant	
11	6	2013-08-29 13:25:00	Santa Clara at Almaden	

	end_station_id	bike_id	subscription_type	zip_code	start_day_of_week	\
4	67	319	Subscriber	94103	Thursday	
6	5	679	Subscriber	95112	Thursday	
9	11	107	Subscriber	95060	Thursday	
10	54	368	Subscriber	94109	Thursday	
11	4	26	Subscriber	95112	Thursday	

	end_day_of_week	horario_inicial	horario_inicial_float	horario_final	\
4	Thursday	12:02:00	12.02	12:04:00	
6	Thursday	13:25:00	13.25	13:27:00	
9	Thursday	11:33:00	11.33	11:35:00	
10	Thursday	13:52:00	13.52	13:55:00	
11	Thursday	13:23:00	13.23	13:25:00	

	horario_final_float	duracion_viaje
4	12.04	00:02:00
6	13.27	00:02:00
9	11.35	00:02:00
10	13.55	00:03:00
11	13.25	00:02:00

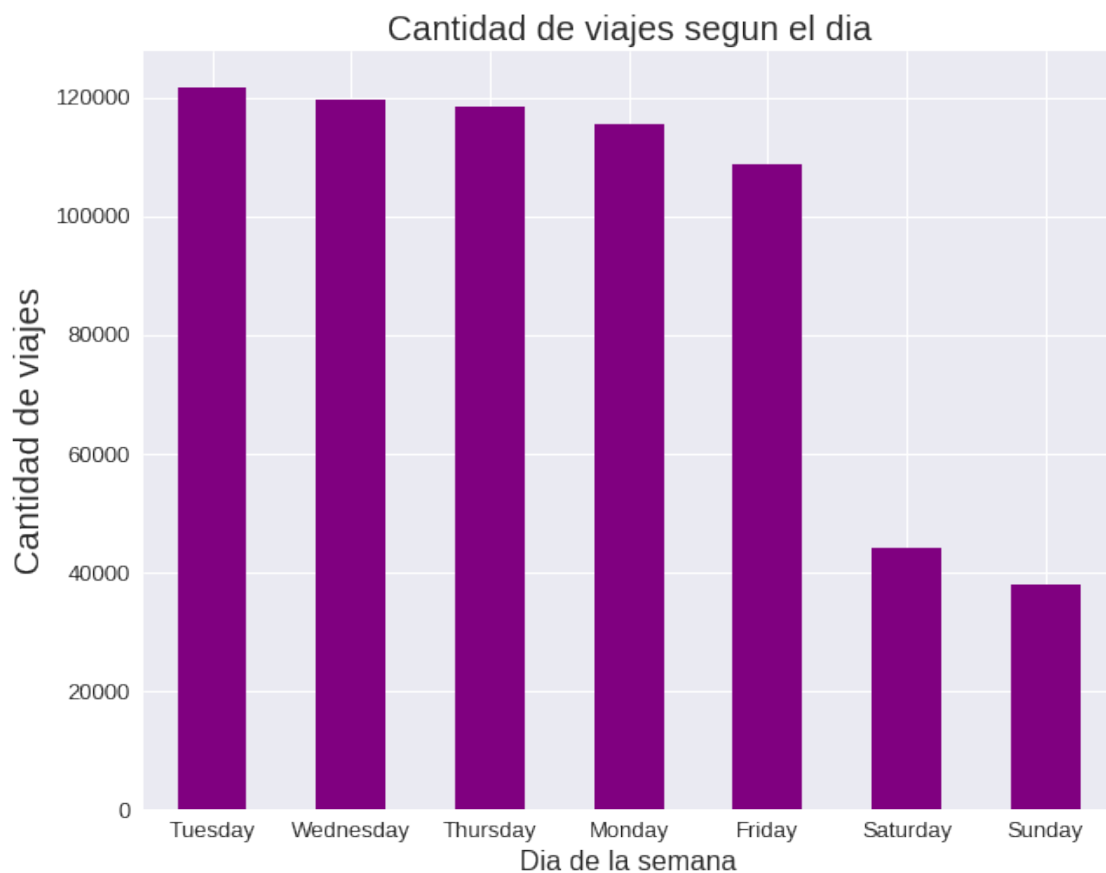
1) 1) Análisis en base a en qué momento se tomó el viaje

2 1.1) Análisis de viajes según día y horario

2.0.1 En estos primeros plots se analizará la relación general entre los viajes y el día/horario de la semana.

2.0.2 ¿El servicio se usa más en días hábiles o los fines de semana?

```
In [20]: trips['start_day_of_week'].value_counts().plot(kind='bar', rot=0, figsize=(10,8), color='purple')
plt.title('Cantidad de viajes segun el dia', fontsize=20);
plt.xlabel('Dia de la semana', fontsize=16);
plt.ylabel('Cantidad de viajes', fontsize=20);
```

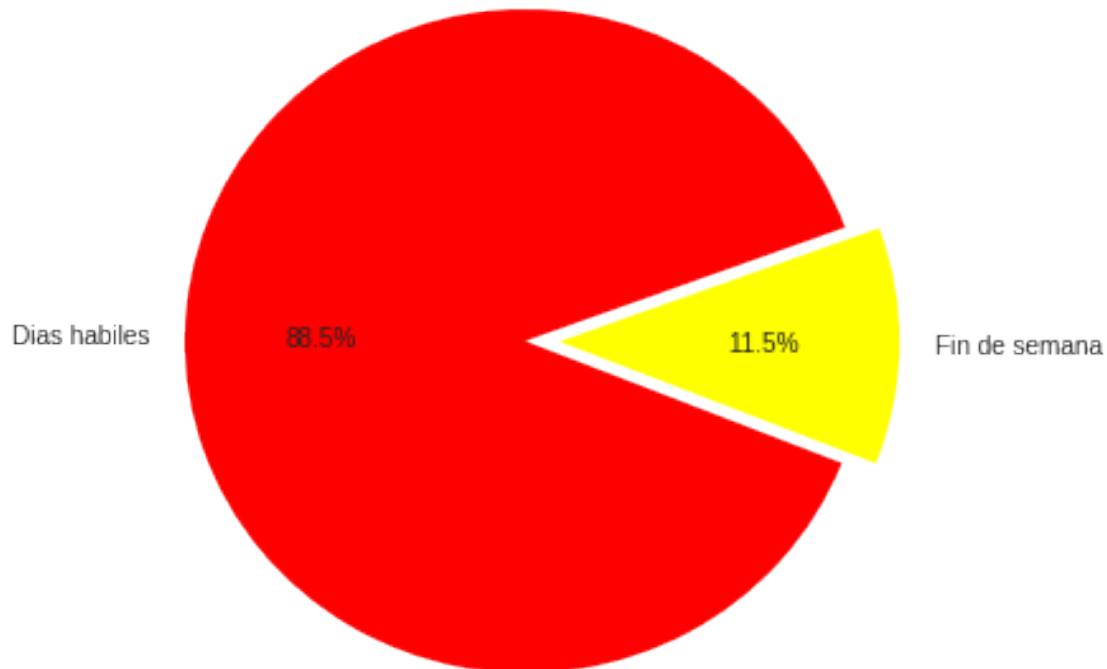


```
In [21]: dias = trips[['start_day_of_week']]
dias_semana = dias[-(dias['start_day_of_week'] == "Saturday")]
dias_semana = dias[-(dias['start_day_of_week'] == "Sunday")]
dias_finde = dias[(dias['start_day_of_week'] == "Saturday") | (dias['start_day_of_week'] == "Sunday")]
```

```
In [22]: sizes = [dias_semana.start_day_of_week.count(), dias_finde.start_day_of_week.count()]
nombres = ['Dias habiles', 'Fin de semana']
```

```
plt.figure(figsize=(6, 6))
plt.title('Distribucion semanal del uso del servicio', fontsize=20)
plt.pie(sizes, labels=nombres, autopct='%1.1f%%', startangle=20, colors=['red', 'yellow'])
plt.show()
```

Distribucion semanal del uso del servicio



Se ve que hay una diferencia drástica en el uso del servicio entre los días hábiles y el fin de semana.

2.0.3 Uso del servicio según el horario:

```
In [23]: semana_entera = trips[['start_day_of_week', 'horario_inicial_float', 'start_station_name']]
semana_entera['horario_inicial_float'] = semana_entera['horario_inicial_float'].map(lambda x: x - 24 if x > 23 else x)
# 24hs = 0hs
semana_entera.head()
```

```
Out [23]:
```

	start_day_of_week	horario_inicial_float	start_station_name
4	Thursday	12.0	South Van Ness at Market
6	Thursday	13.0	Santa Clara at Almaden
9	Thursday	11.0	San Jose City Hall

10	Thursday	14.0	Spear at Folsom
11	Thursday	13.0	San Pedro Square

	end_station_name
4	Market at 10th
6	Adobe on Almaden
9	MLK Library
10	Embarcadero at Bryant
11	Santa Clara at Almaden

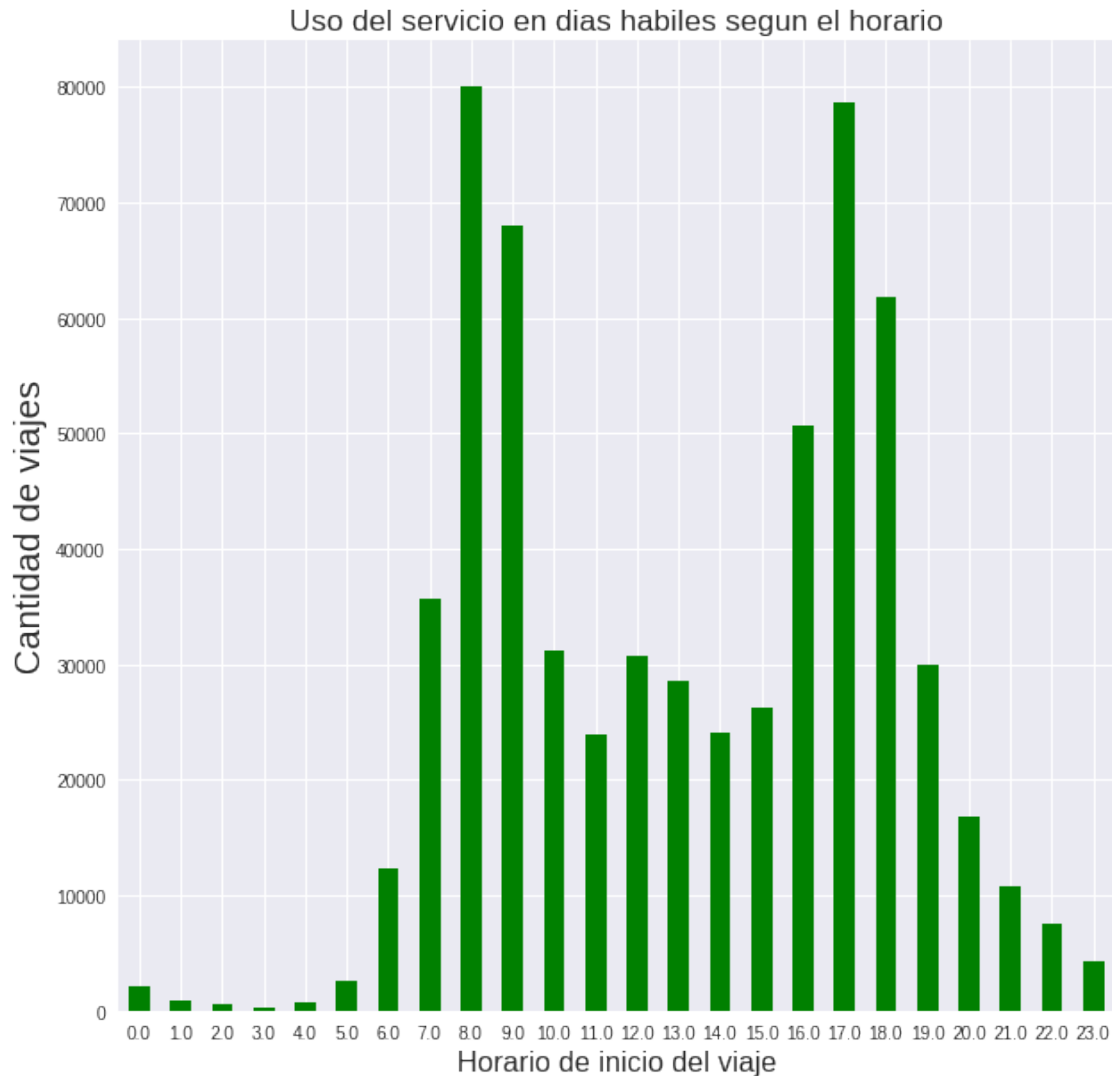
```
In [24]: semana = semana_entera[-(semana_entera['start_day_of_week'] == "Saturday")]
semana = semana_entera[-(semana_entera['start_day_of_week'] == "Sunday")]
```

```
In [25]: semana['apariciones'] = semana['start_day_of_week'].map(lambda x: 1) # seteo todas las
horarios_semana = semana[['horario_inicial_float', 'apariciones']]
semana = semana.drop('apariciones', 1) # vuelvo a dejar el dt como antes
horarios_semana_contador = horarios_semana.groupby('horario_inicial_float').aggregate(s
```

/home/mk/venv/lib/python3.6/site-packages/ipykernel/__main__.py:1: SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: <http://pandas.pydata.org/pandas-docs/stable/indexing.html#>
if __name__ == '__main__':

```
In [26]: horarios_semana_contador.plot.bar(rot=0, figsize=(10,10), color='green', fontsize=10);
plt.ylabel('Cantidad de viajes', fontsize=20)
plt.xlabel('Horario de inicio del viaje', fontsize=16)
plt.title('Uso del servicio en dias habiles segun el horario', fontsize=17)
plt.legend('')
plt.show()
```

Se aprecia que durante los días hábiles los horarios pico son a las 8 y 9, y a las 17 y 18, particularmente cuando la gente va y cuando regresa al trabajo, escuela, etc.

```
In [27]: finde = semana_entera[(semana_entera['start_day_of_week'] == "Saturday") | (semana_entera['start_day_of_week'] == "Sunday")]
```

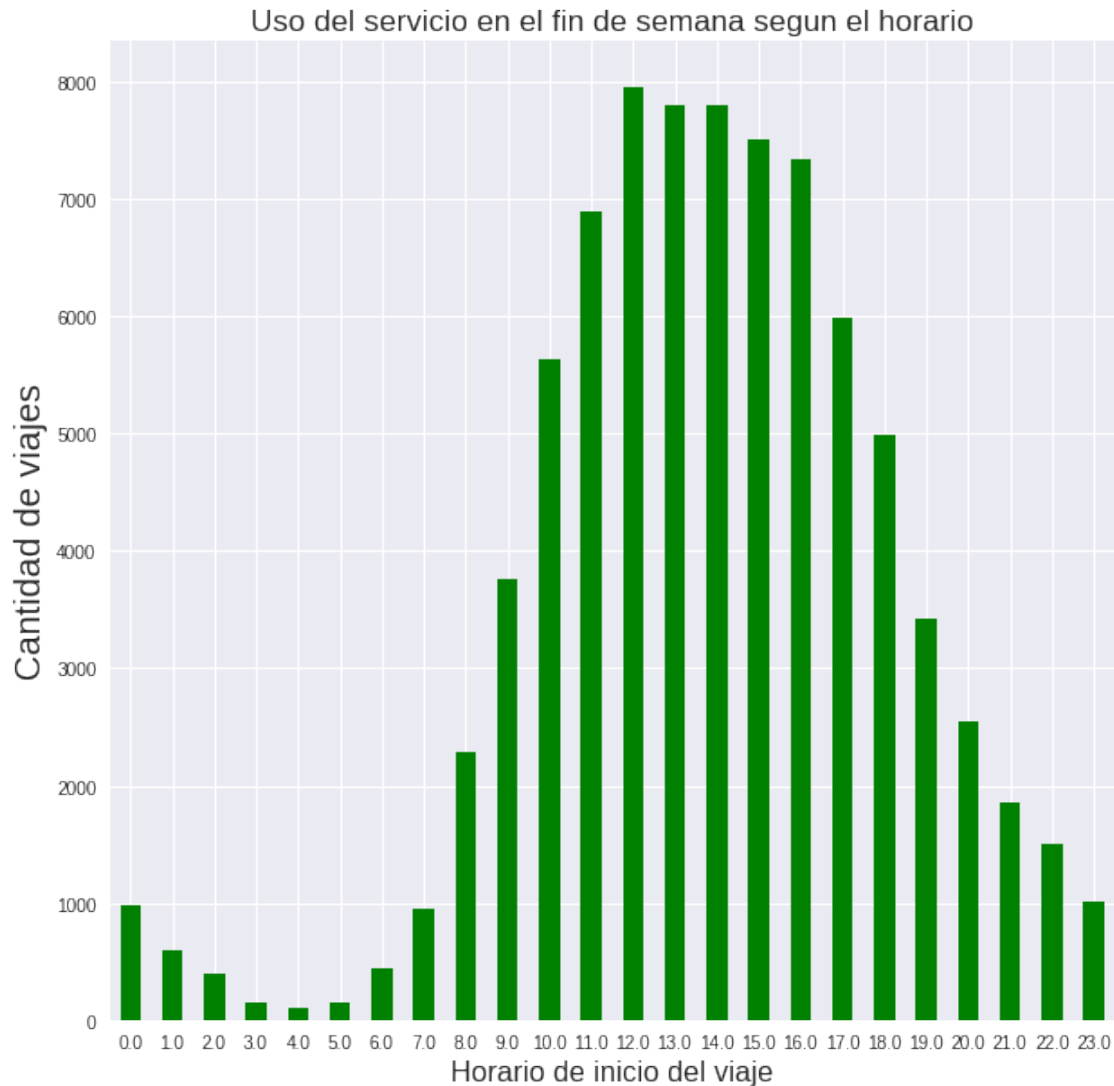
```
In [28]: finde['apariciones'] = finde['start_day_of_week'].map(lambda x: 1) # seteo todas las apariciones a 1
horarios_finde = finde[['horario_inicial_float', 'apariciones']]
finde = finde.drop('apariciones', 1) # vuelvo a dejar el dt como antes
horarios_finde_contador = horarios_finde.groupby('horario_inicial_float').aggregate(sum)
```

/home/mk/venv/lib/python3.6/site-packages/ipykernel/__main__.py:1: SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: <http://pandas.pydata.org/pandas-docs/stable/indexing.html#>

```
if __name__ == '__main__':
```

```
In [29]: horarios_finde_contador.plot.bar(rot=0, figsize=(10,10), color='green', fontsize=10);
plt.ylabel('Cantidad de viajes', fontsize=20)
plt.xlabel('Horario de inicio del viaje', fontsize=16)
plt.title('Uso del servicio en el fin de semana segun el horario', fontsize=17)
plt.legend('')
plt.show()
```



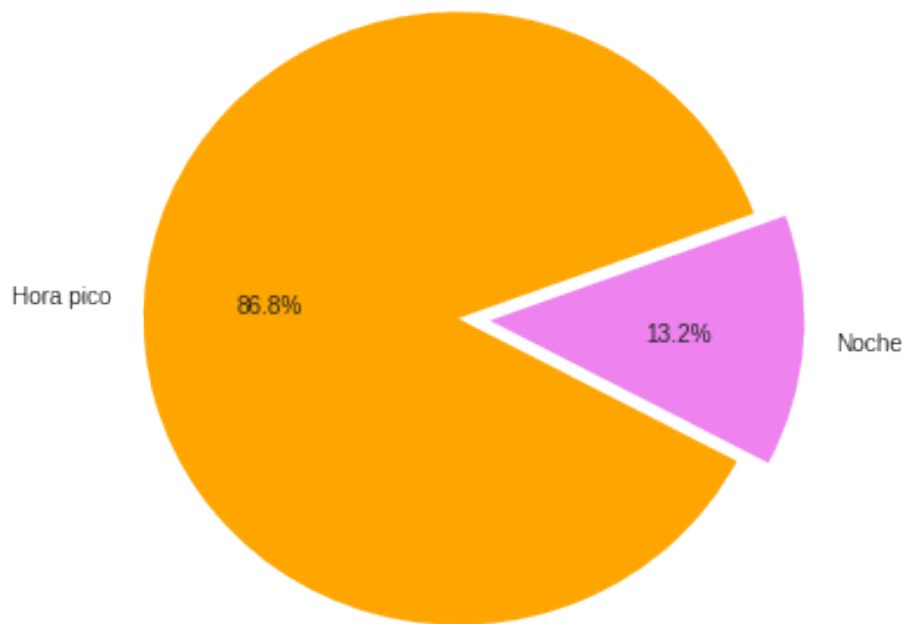
Los horarios pico los fines de semana son de 11 a 16. También lo que se observa es la disminución del uso del servicio al llegar la noche a pesar de que sea fin de semana. En el siguiente plot se va a analizar esa comparación.

```
In [30]: viajes_en_hora_pico_finde = finde[((finde['horario_inicial_float'] >= 11) & (finde['horario_inicial_float'] < 20))
viages_finde_noche = finde[((finde['horario_inicial_float'] >= 20) & (finde['horario_inicial_float'] < 24))
```

```
In [31]: sizes = [viajes_en_hora_pico_finde.start_day_of_week.count(), viajes_finde_noche.start_
nombres = ['Hora pico', 'Noche']

plt.figure(figsize=(6, 6))
plt.title('Distribucion del uso del servicio los fines de semana', fontsize=20)
plt.pie(sizes, labels=nombres, autopct='%1.1f%%', startangle=20, colors=['orange', 'violet'])
plt.show()
```

Distribucion del uso del servicio los fines de semana



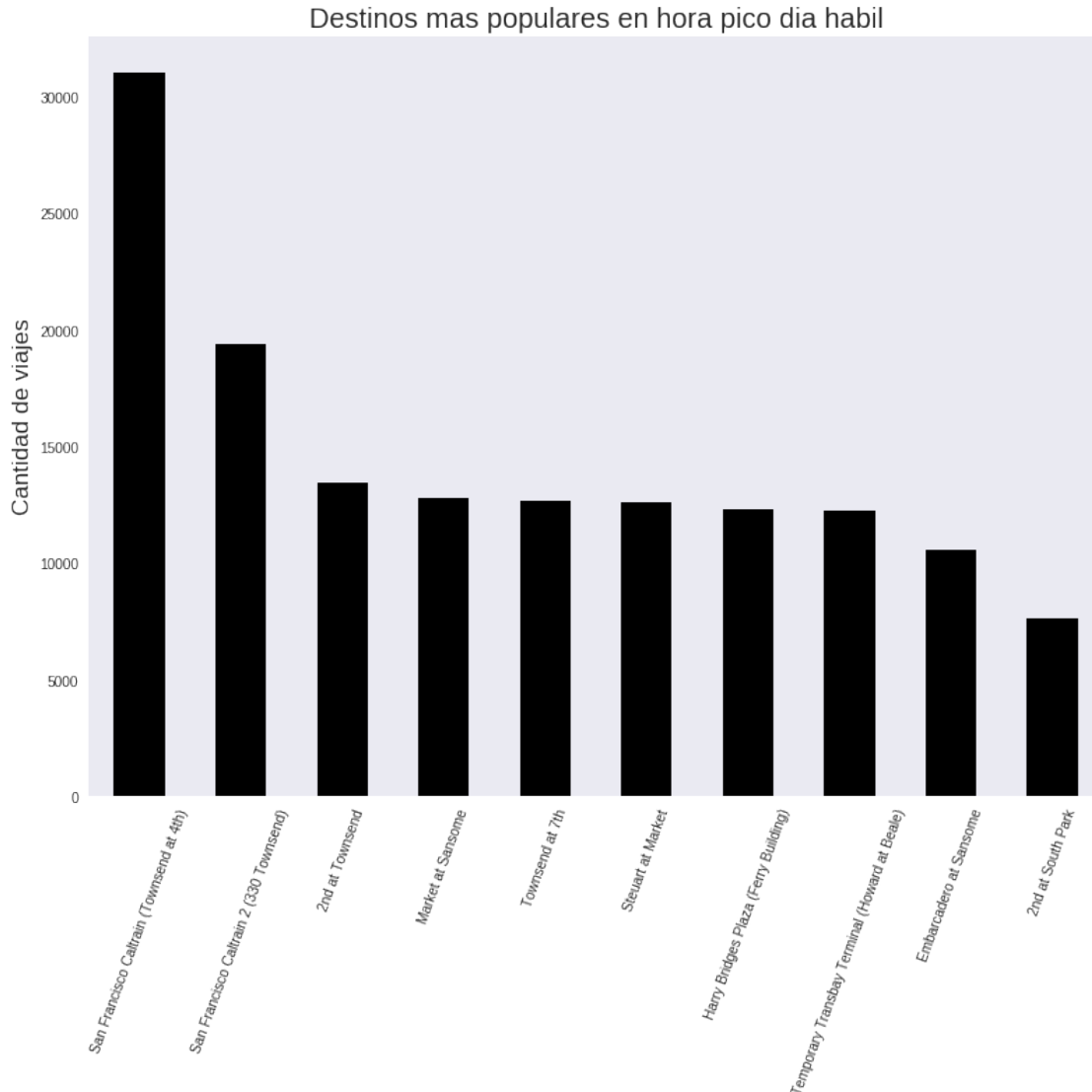
2.1 1.1.A) Análisis de los horarios importantes

2.1.1 Se analizarán los destinos y trayectos más populares en los horarios importantes de los días hábiles y fin de semana.

```
In [32]: viajes_en_hora_pico_semana = semana[((semana['horario_inicial_float'] >= 8) & (semana['
| ((semana['horario_inicial_float'] >= 17) & (semana

In [33]: destinos_mas_populares_hora_pico_semana = viajes_en_hora_pico_semana['end_station_name']
destinos_mas_populares_hora_pico_semana = destinos_mas_populares_hora_pico_semana.head(

In [34]: destinos_mas_populares_hora_pico_semana.plot(kind='bar', rot=70, figsize=(13,10), color=
plt.title('Destinos mas populares en hora pico dia habil', fontsize=20);
plt.ylabel('Cantidad de viajes', fontsize=17);
```



Estos son los 10 destinos más populares durante las horas pico en los días hábiles. En los primeros lugares se destaca el Caltrain, donde una gran cantidad de gente se dirige luego de trabajar para regresar a sus casas. Las demás estaciones presentan áreas de mucho movimiento de gente: empresas, comercios, etc, y también se encuentra el puerto, por lo que lo más probable es que la gente se dirija allí para concurrir a trabajar.

```
In [35]: # función que dado un dataframe con un campo 'start_station_name' y otro 'end_station_name'
# devuelve un diccionario con los start_station como clave y como valor un diccionario
# y valor la cantidad de viajes de ese trayecto. También devuelve una lista con el trayecto
# de viajes junto con el start y end station del mismo. Orden = O(n) siendo n la cantidad de viajes
def contador_viajes(dataframe):
    cont_viajes = {}
    viaje_mas_popular = []
    viaje_mas_popular.append(0)
```

```

viaje_mas_popular.append("")
viaje_mas_popular.append("")
for index,row in dataframe.iterrows():

    if row['start_station_name'] not in cont_viajes:
        cont_viajes[row['start_station_name']] = {}

    if row['end_station_name'] not in cont_viajes[row['start_station_name']]:
        cont_viajes[row['start_station_name']][row['end_station_name']] = 1
    else:
        cont_viajes[row['start_station_name']][row['end_station_name']] += 1

    if cont_viajes[row['start_station_name']][row['end_station_name']] > viaje_mas_popular[0]:
        viaje_mas_popular[0] = cont_viajes[row['start_station_name']][row['end_station_name']]
        viaje_mas_popular[1] = row['start_station_name']
        viaje_mas_popular[2] = row['end_station_name']

return cont_viajes,viaje_mas_popular

```

```

In [36]: contador_de_viajes_hora_pico_semana,viaje_mas_popular_hora_pico_semana = contador_viajes_hora_pico_semana, viaje_mas_popular_hora_pico_semana

```

```

Out[36]: [3004, 'San Francisco Caltrain 2 (330 Townsend)', 'Townsend at 7th']

```

Con esto se ve que el viaje más realizado en hora pico durante los días hábiles es el trayecto San Francisco Caltrain 2 (330 Townsend) - Townsend at 7th. Precisamente como se mencionó anteriormente, lo más probable es que se trate de las personas que van a trabajar y se transportan desde el Caltrain hasta la zona en cuestión.

```

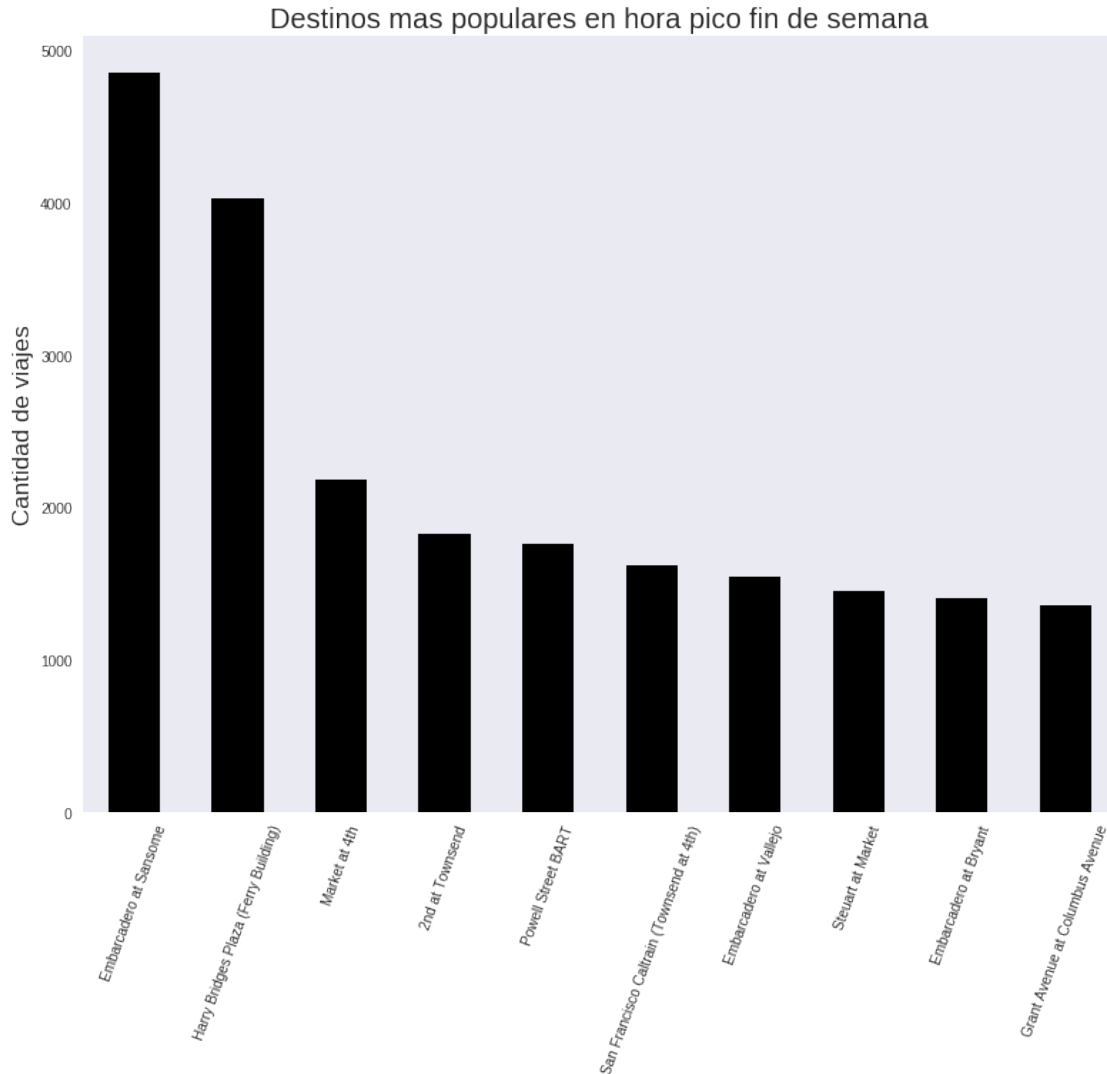
In [37]: destinos_mas_populares_hora_pico_finde = viajes_en_hora_pico_finde['end_station_name'].value_counts()
destinos_mas_populares_hora_pico_finde = destinos_mas_populares_hora_pico_finde.head(10)

```

```

In [38]: destinos_mas_populares_hora_pico_finde.plot(kind='bar', rot=70, figsize=(13,10), color='red')
plt.title('Destinos mas populares en hora pico fin de semana', fontsize=20);
plt.ylabel('Cantidad de viajes', fontsize=17);

```



Aqui se encuentran los 10 destinos más populares para las horas pico del fin de semana. Se puede apreciar que los mismos se caracterizan por ser lugares muy atractivos para pasear y hacer actividades de ocio.

```
In [39]: contador_de_viajes_hora_pico_finde, viaje_mas_popular_hora_pico_finde = contador_viajes(
        viaje_mas_popular_hora_pico_finde
```

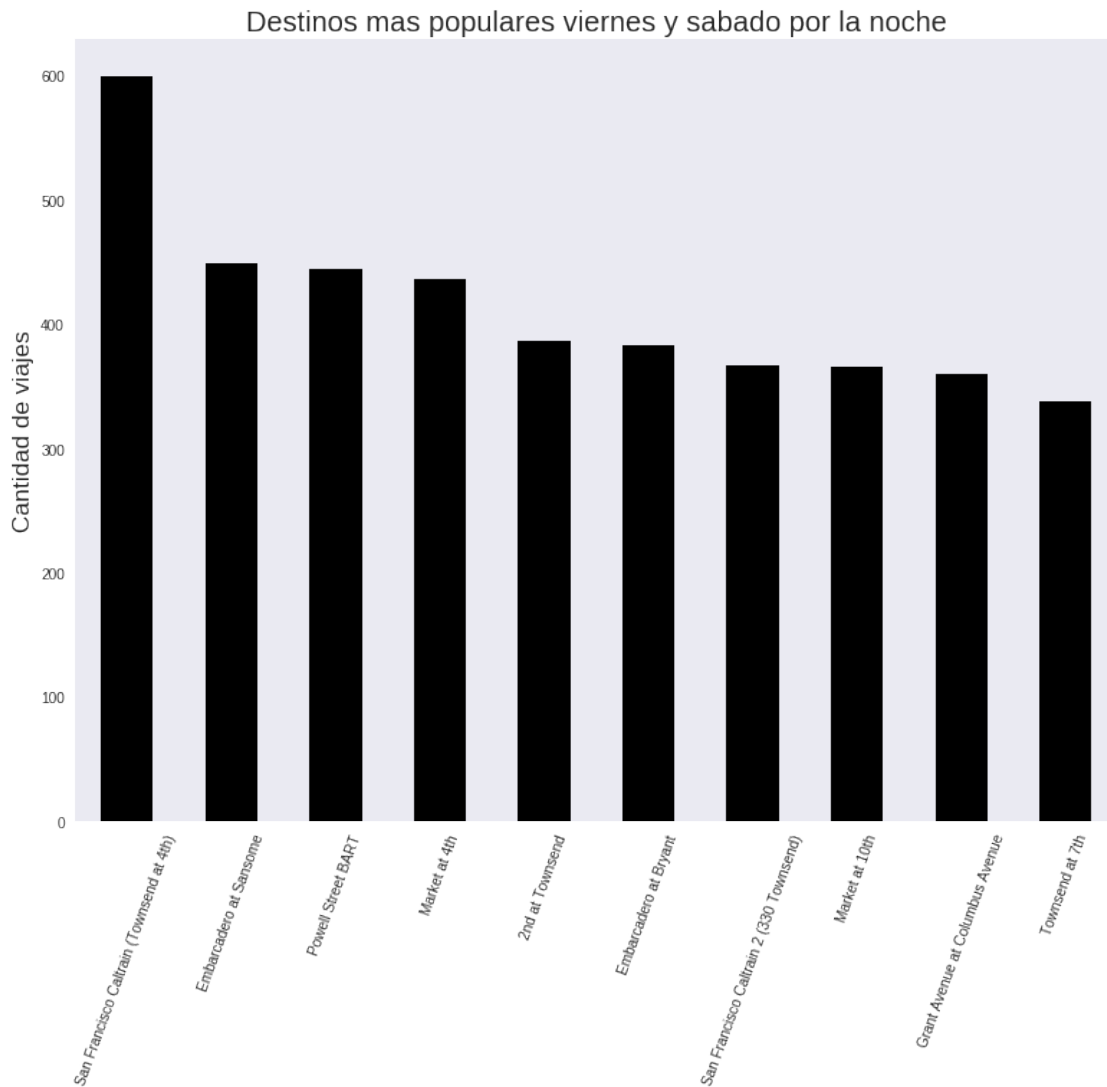
```
Out[39]: [1068, 'Harry Bridges Plaza (Ferry Building)', 'Embarcadero at Sansome']
```

Con esto se ve que el viaje más realizado en hora pico durante el fin de semana es el trayecto Harry Bridges Plaza (Ferry Building) - Embarcadero at Sansome. Además estos dos son los destinos más concurridos las horas pico de los fines de semana. Un posible uso de esta información podría ser para fines comerciales, ya que esto da la pauta que por esta zona es donde más concentración de gente se encuentra.

```
In [40]: viernes_y_sab = semana_entera[(semana_entera['start_day_of_week'] == "Saturday") | (seman
        viernes_y_sab_noche = viernes_y_sab[(viernes_y_sab['horario_inicial_float'] >= 20) & (v
```

```
In [41]: destinos_mas_populares_viernes_y_sab_noche = viernes_y_sab_noche['end_station_name'].va
destinos_mas_populares_viernes_y_sab_noche = destinos_mas_populares_viernes_y_sab_noche

In [42]: destinos_mas_populares_viernes_y_sab_noche.plot(kind='bar', rot=70, figsize=(13,10), co
plt.title('Destinos mas populares viernes y sabado por la noche', fontsize=20);
plt.ylabel('Cantidad de viajes', fontsize=17);
```



Estos son los destinos más frecuentados los viernes y sábados por la noche. Se analizó aparte del domingo ya que estos son los días que al día siguiente por lo general no se trabaja/concurre a estudiar, por lo que la gente se podría dormir más tarde y planear otro tipo de salida. Se destacan entre los lugares más frecuentados las estaciones Powell Street BART y Market at 4th, las cuales están rodeadas de shoppings, restaurantes y demás.

2.2 1.1.B) Análisis relacionado a las duraciones de los viajes

2.2.1 B.1) - ¿Viajes más largos en días hábiles o el fin de semana?

```
In [43]: semana_entera_con_duracion = trips[['start_day_of_week', 'horario_inicial_float', 'start_station_name', 'end_station_name', 'duracion_viaje', 'duration']]
semana_entera_con_duracion.head()
```

```
Out[43]:
```

	start_day_of_week	horario_inicial_float	start_station_name	end_station_name	duracion_viaje	duration
4	Thursday	12.0	South Van Ness at Market	Market at 10th	00:02:00	83
6	Thursday	13.0	Santa Clara at Almaden	Adobe on Almaden	00:02:00	109
9	Thursday	11.0	San Jose City Hall	MLK Library	00:02:00	114
10	Thursday	14.0	Spear at Folsom	Embarcadero at Bryant	00:03:00	125
11	Thursday	13.0	San Pedro Square	Santa Clara at Almaden	00:02:00	126

```
In [44]: semana_con_duracion = semana_entera_con_duracion[-(semana_entera_con_duracion['start_day_of_week'] == 'S')]
semana_con_duracion = semana_entera_con_duracion[-(semana_entera_con_duracion['start_day_of_week'] == 'S')]
```

```
In [45]: finde_con_duracion = semana_entera_con_duracion[(semana_entera_con_duracion['start_day_of_week'] == 'S')]
```

```
In [46]: semana_con_duracion.duration.mean() / 60 # resultado en minutos
```

```
Out[46]: 13.909346143479581
```

```
In [47]: finde_con_duracion.duration.mean() / 60 # resultado en minutos
```

```
Out[47]: 31.908434371772668
```

Los viajes en promedio duran más los fines de semana. Esto tiene lógica ya que los fines de semana la gente puede usar más tiempo para pasear con la bicicleta sin tener un destino en particular cuando durante los días hábiles la gente busca ir al trabajo o lugar donde ejercer sus obligaciones. Igualmente hay que considerar que esta estadística podría estar dañada por muchos viajes "largos" en el fin de semana, por ejemplo de más de dos horas, que harían subir el promedio general, sin haber tantos viajes con un horario parecido al promedio calculado. A continuación se analizará eso:

```
In [48]: semana_con_duracion[semana_con_duracion['duracion_viaje'] > "02:00:00"].start_day_of_week
```

```
Out[48]: 8776
```

```
In [49]: finde_con_duracion[finde_con_duracion['duracion_viaje'] > "02:00:00"].start_day_of_week
```

```
Out[49]: 5094
```

Se observa que tanto en los días hábiles como los fines de semana hay viajes de más de 2 horas, y que ya que los días hábiles son más que el fin de semana y la estadística muestra que son casi el doble de viajes, en promedio esos viajes largos se dan por igual todos los días, por lo que el problema de que haya viajes largos no afecta el resultado de la estadística anterior.

2.2.2 B.2) - Trayectos interesantes

2.2.3 A continuación se analizarán los trayectos más frecuentes según un criterio de duración y demás fitros.

```
In [50]: finde_viajes_largos = finde_con_duracion[finde_con_duracion['duracion_viaje'] > "00:30:00"]
```

```
In [51]: contador_de_viajes_largos_finde, viaje_largo_mas_popular_finde = contador_viajes(finde_viajes_largos)
viaje_largo_mas_popular_finde
```

```
Out [51]: [561,
           'Harry Bridges Plaza (Ferry Building)',
           'Harry Bridges Plaza (Ferry Building)']
```

Con esto se ve que los viajes mayores a 30 minutos los fines de semana son tanto de inicio como fin la estación Harry Bridges Plaza (Ferry Building). El objetivo de este análisis fue encontrar el trayecto más popular de los fines de semana que sea orientado a pasear (por eso el filtro de la duración). El resultado muestra que el viaje más realizado es empezar en dicha estación, recorrer por más de 30 minutos, y luego volver a esta misma dejar la bicicleta.

```
In [52]: harry_harry = finde_viajes_largos[(finde_viajes_largos['start_station_name'] == 'Harry Bridges Plaza (Ferry Building)')]
harry_harry.describe()
```

```
Out [52]:
```

	horario_inicial_float		duracion_viaje	duration
count	561.000000		561	561.000000
mean	13.178253	0 days 02:35:52.085561		9349.684492
std	3.028488	0 days 01:49:11.726171		6551.425814
min	5.000000	0 days 00:31:00		1811.000000
25%	11.000000	0 days 01:13:00		4392.000000
50%	13.000000	0 days 01:59:00		7126.000000
75%	15.000000	0 days 03:54:00		14073.000000
max	21.000000	0 days 08:13:00		29629.000000

Como se puede apreciar, estos viajes se caracterizan por su duración promedio de 2 horas y media y que en promedio salen a las 13hs. Esto da la pauta de qué elige la mayoría de gente a la hora de pasear por un largo rato aprovechando el clima de recién entrada la tarde.

```
In [53]: semana_viajes_cortos = semana_con_duracion[(semana_con_duracion['duracion_viaje'] > "00:30:00")]
```

```
In [54]: contador_de_viajes_cortos_semana, viaje_corto_mas_popular_semana = contador_viajes(semana_viajes_cortos)
viaje_corto_mas_popular_semana
```

```
Out [54]: [3247, 'Steuart at Market', 'San Francisco Caltrain (Townsend at 4th)']
```

```
In [55]: steuart_caltrain = semana_viajes_cortos[(semana_viajes_cortos['start_station_name'] == 'Steuart at Market')]
steuart_caltrain.describe()
```

```
Out [55]:
```

	horario_inicial_float		duracion_viaje	duration
count	3247.000000		3247	3247.000000
mean	14.825685	0 days 00:11:38.158299		697.412073
std	3.959800	0 days 00:01:58.426460		116.468743

min	0.000000	0 days 00:09:00	486.000000
25%	14.000000	0 days 00:10:00	615.000000
50%	16.000000	0 days 00:11:00	673.000000
75%	17.000000	0 days 00:13:00	757.000000
max	23.000000	0 days 00:19:00	1182.000000

Con esto se ve que el viaje entre 8 y 20 minutos más popular en días hábiles es el trayecto Steuart at Market - San Francisco Caltrain (Townsend at 4th). El objetivo de este análisis fue encontrar el trayecto con una duración normal que más se haga considerando el tráfico que presenta un día hábil. Una posible conclusión debido a la hora promedio de comienzo de los viajes (15hs) es que la gente al terminar de trabajar se dirige desde Steuart at Market (un lugar con mucho movimiento por lo que deducimos que mucha gente trabaja por esa zona) en bicicleta hasta el Caltrain, y toma este transporte para volver a sus casas. Cabe mencionar que este trayecto tiene más viajes que el visto anteriormente en hora pico de días hábiles (que posee 3004), por lo que se concluye que para volver del trabajo a sus hogares se usa más el servicio que al ir al mismo. Esto tiene lógica ya que a la mañana generalmente se tiende a llegar sobre la hora, y aparte se requiere un esfuerzo energético.

3 1.2) Análisis de la influencia de ciertas fechas en los viajes

A continuación se estudiará el comportamiento del servicio en fechas particulares. Para eso se hará un plot con los viajes a medida que transcurre el año. Se tomó el 2014 debido a que el 2013 y 2015 no tienen datos de todo el año entero.

```
In [56]: # funciones para operar con un formato fecha (anio-mes-dia hora:minutos:segundos)
def obtener_dia(fecha):
    return fecha.day
def obtener_mes(fecha):
    return fecha.month
def fecha_sin_hora(fecha):
    return (str(fecha.year) + "-" + str(fecha.month) + "-" + str(fecha.day))
```

```
In [57]: anio_2014 = trips[['start_date', 'start_day_of_week', 'horario_inicial_float', 'start_sta
anio_2014['horario_inicial_float'] = anio_2014['horario_inicial_float'].map(lambda x: x
anio_2014 = anio_2014[(anio_2014['start_date'].dt.year) == 2014]
anio_2014['fecha_sin_horario'] = anio_2014['start_date'].map(fecha_sin_hora)
anio_2014['fecha_sin_horario'] = pd.to_datetime(anio_2014['fecha_sin_horario'])
anio_2014['dia'] = anio_2014['start_date'].map(obtener_dia)
anio_2014['mes'] = anio_2014['start_date'].map(obtener_mes)
anio_2014 = anio_2014.sort_values(by='start_date')
anio_2014.head()
```

```
Out[57]:
```

	start_date	start_day_of_week	horario_inicial_float	\
100563	2014-01-01 00:14:00	Wednesday	0.0	
100564	2014-01-01 00:14:00	Wednesday	0.0	
100565	2014-01-01 00:17:00	Wednesday	0.0	
100566	2014-01-01 00:23:00	Wednesday	0.0	
100567	2014-01-01 00:23:00	Wednesday	0.0	

	start_station_name	end_station_name	duracion_viaje	duration \
100563	San Francisco City Hall	Townsend at 7th	00:07:00	435
100564	San Francisco City Hall	Townsend at 7th	00:07:00	432
100565	Embarcadero at Sansome	Beale at Market	00:25:00	1523
100566	Steuart at Market	Powell Street BART	00:27:00	1620
100567	Steuart at Market	Powell Street BART	00:27:00	1617

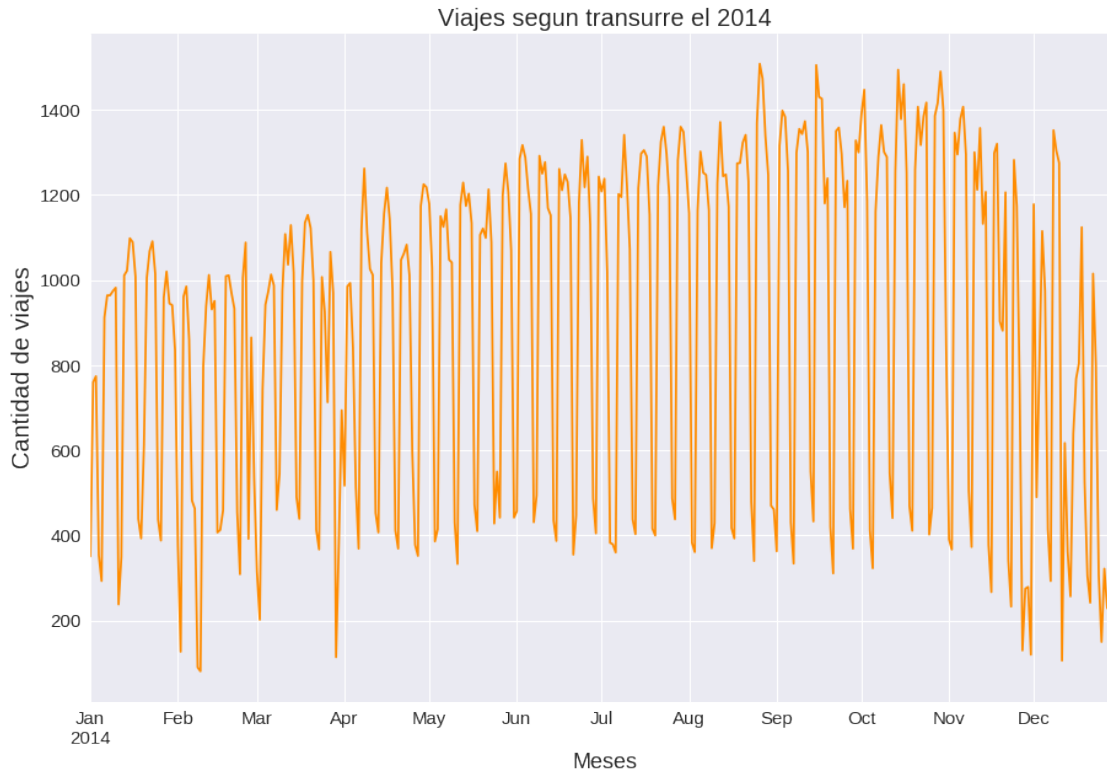
	subscription_type	fecha_sin_horario	dia	mes
100563	Subscriber	2014-01-01	1	1
100564	Subscriber	2014-01-01	1	1
100565	Subscriber	2014-01-01	1	1
100566	Customer	2014-01-01	1	1
100567	Customer	2014-01-01	1	1

```
In [58]: anio_2014['viaje'] = anio_2014['start_day_of_week'].map(lambda x: 1) # seteo todas las
viajes_segun_dia = anio_2014[['fecha_sin_horario', 'viaje']]
anio_2014 = anio_2014.drop('viaje', 1) # vuelvo a dejar el dt como antes
viajes_segun_dia_contador = viajes_segun_dia.groupby('fecha_sin_horario').aggregate(sum)
viajes_segun_dia_contador.head()
```

```
Out[58]:
```

fecha_sin_horario	viaje
2014-01-01	351
2014-01-02	760
2014-01-03	774
2014-01-04	354
2014-01-05	293

```
In [59]: viajes_segun_dia_contador.plot.line(figsize=(15,10), color='darkorange', fontsize=15);
plt.xlabel('Meses', fontsize=18)
plt.ylabel('Cantidad de viajes', fontsize=20)
plt.title('Viajes segun transurre el 2014', fontsize=20);
plt.grid(True)
plt.legend('');
plt.show()
```



En primer lugar se pueden observar picos en fechas como Halloween (fines octubre) y Homecoming (principios octubre). Luego se puede notar como decae el uso del servicio en feriados como por ejemplo en día de acción de gracias (fines noviembre), navidad, semana santa (fines marzo) y demás. También otra observación es la diferencia de viajes que se hacen en invierno en comparación al resto del año.

```
In [60]: viajes_segun_dia_contador[viages_segun_dia_contador['viaje'] > 1400]
```

```
Out[60]:
```

fecha_sin_horario	viaje
2014-08-26	1508
2014-08-27	1473
2014-09-15	1505
2014-09-16	1430
2014-09-17	1426
2014-10-02	1447
2014-10-14	1494
2014-10-16	1460
2014-10-21	1407
2014-10-24	1417
2014-10-28	1416
2014-10-29	1490
2014-11-06	1407

```
In [61]: viajes_segun_dia_contador[viajes_segun_dia_contador['viaje'] < 200]
```

```
Out[61]:
```

fecha_sin_horario	viaje
2014-02-02	127
2014-02-08	90
2014-02-09	81
2014-03-29	114
2014-11-27	130
2014-11-30	120
2014-12-11	106
2014-12-25	150

3.1 ¿Los días con picos de viajes atraen a clientes no suscriptos al servicio?

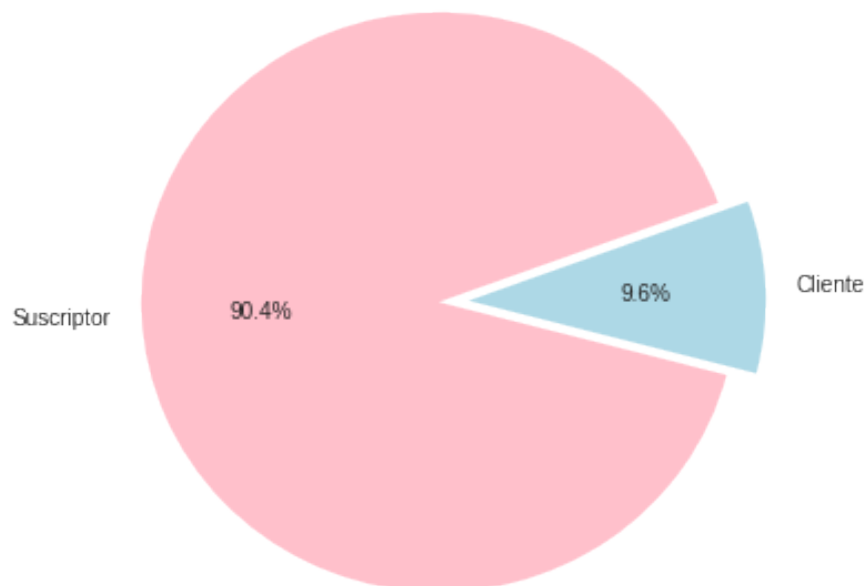
Se analizará cuánta gente no suscripta usa el servicio en días festivos como Halloween.

```
In [62]: halloween = anio_2014[anio_2014['fecha_sin_horario'] == "2014-10-29"].subscription_type
```

```
In [63]: sizes = [halloween.Subscriber, halloween.Customer]  
nombres = ['Suscriptor', 'Cliente']
```

```
plt.figure(figsize=(6, 6))  
plt.title('Tipos de suscripciones en los viajes durante Halloween', fontsize=20)  
plt.pie(sizes, labels=nombres, autopct='%1.1f%%', startangle=20, colors=['pink', 'lightblue'])  
plt.show()
```

Tipos de suscripciones en los viajes durante Halloween



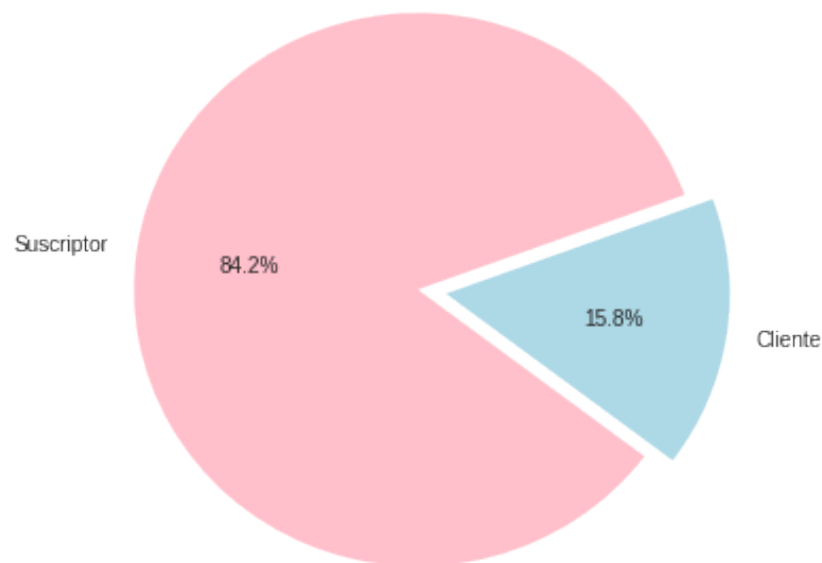
En una primera instancia el valor es bajo a pesar de ser Halloween. Ahora se va a buscar como es normalmente la relación Suscriptor-Cliente en porcentajes y también la diferencia en cantidad de viajes para comparar. Tomamos el promedio de un mes el cual se mantenga estable, sin picos, como por ejemplo junio.

```
In [64]: junio = anio_2014[anio_2014['mes'] == 6]
        suscripciones_junio = junio.subscription_type.value_counts()
        dia_promedio_junio = suscripciones_junio / 30

In [65]: sizes = [dia_promedio_junio.Subscriber, dia_promedio_junio.Customer]
        nombres = ['Suscriptor', 'Cliente']

        plt.figure(figsize=(6, 6))
        plt.title('Tipos de suscripciones en los viajes en dia promedio junio', fontsize=20)
        plt.pie(sizes, labels=nombres, autopct='%1.1f%%', startangle=20, colors=['pink', 'lightblue'])
        plt.show()
```

Tipos de suscripciones en los viajes en dia promedio junio



```
In [66]: halloween
```

```
Out[66]: Subscriber    1347
         Customer       143
         Name: subscription_type, dtype: int64
```

```
In [67]: dia_promedio_junio
```

```
Out[67]: Subscriber      835.466667  
         Customer        157.300000  
         Name: subscription_type, dtype: float64
```

Se ve que en promedio los días comunes tienen un mayor porcentaje de clientes. Y tampoco influye que en Halloween sean más viajes ya que el aumento sólo se nota en suscriptores y no en clientes. Al contrario de lo que uno esperaría, en Halloween la gente no suscripta al servicio no lo toma en cuenta como una opción.

4 2) Relación Entre Viajes En Bicicleta y el Clima

```
In [78]: trips = pd.read_csv('../data/trip.csv', low_memory=False)  
         weather = pd.read_csv('../data/weather.csv', low_memory=False)  
         station = pd.read_csv('../data/station.csv', low_memory=False)  
         #Se convierte los dates a datetime64[ns].  
         trips['start_date'] = pd.to_datetime(trips['start_date'])  
         weather['date'] = pd.to_datetime(weather['date'])
```

```
In [79]: #Se agrega una nueva columna date que coincide con weather.  
         trips['date'] = trips['start_date'].apply(lambda x: x.date())  
         #Se convierte date a datetime64[ns].  
         trips['date'] = pd.to_datetime(trips['date'])
```

```
In [80]: #Formula para convertir F a C.  
         def f_to_c(f_temp):  
             return round((f_temp - 32) / 1.8, 2)
```

```
In [81]: #Se crean columnas con las temperaturas en C.  
         weather['max_temperature_c'] = weather['max_temperature_f'].map(f_to_c)  
         weather['mean_temperature_c'] = weather['mean_temperature_f'].map(f_to_c)  
         weather['min_temperature_c'] = weather['min_temperature_f'].map(f_to_c)
```

```
In [82]: #Se crean columnas con visibilidad en Km.  
         weather['max_visibility_km'] = weather['max_visibility_miles'].map(lambda x: x * 1.6)  
         weather['mean_visibility_km'] = weather['mean_visibility_miles'].map(lambda x: x * 1.6)  
         weather['min_visibility_km'] = weather['min_visibility_miles'].map(lambda x: x * 1.6)
```

```
In [83]: #Funcion para convertir la duracion de segundos a minutos.  
         def s_to_m(time):  
             return (time / 60)  
         #Funcion para convertir la duracion de segundos a horas redondeo a 3 decimales.  
         def s_to_h(time):  
             return round((time / 3600),3)
```

```
In [84]: #Se crea una columna con la duracion en minutos y la duracion en horas.  
         trips['duration_m'] = trips['duration'].map(s_to_m)  
         trips['duration_h'] = trips['duration'].map(s_to_h)
```

```

In [85]: #Funcion para clasificar estaciones climaticas.
def estacion(date):
    if date.month >= 3 and date.month <= 5:
        return 'Primavera'
    elif date.month >= 6 and date.month <= 8:
        return 'Verano'
    elif date.month >= 9 and date.month <= 11:
        return 'Otoño'
    else:
        return 'Invierno'

In [86]: #Se crea la columna con la estacion climatica.
trips['estacion_clima'] = trips['date'].map(estacion)

In [87]: #Se filtra lo mencionado anteriormente,
# las duraciones menores o iguales a 3 minutos con la misma estacion de salida y llegada
# y los viajes de mas de 12 horas (12 * 3600 = 43200 segundos) y los de entre 11 y 12hs
# a la mañana
trips = trips[-((trips['duration_m'] <= 3.0) & (trips['start_station_id'] == trips['end_station_id']) & (trips['start_hour'] == 11) & (trips['start_hour'] <= 12))]
trips['start_hour'] = trips['start_date'].map(lambda x: x.hour)
trips = trips[-((trips['duration'] > 43200) | ((trips['duration'] > 39600) & ((trips['start_hour'] > 12) & (trips['start_hour'] <= 11))))

In [88]: #Se hace join de trips y station para obtener la ciudad en la cual comenzo el viaje.
#Se renombra el id de trips a id_trip.
trips.rename(columns={'id': 'id_trip'}, inplace=True)
station_aux = station[['id', 'city']]
joined_trips_station = trips.merge(station, left_on=['start_station_id'], right_on=['id'])

In [89]: #Funcion para clasificar la ciudad dependiendo del zipcode.
#La clasificacion se basa en los zip_codes obtenidos en https://www.unitedstateszipcode.org/zip-codes
def zip_ciudad(zip_code):
    if zip_code == 95113:
        return 'San Jose'
    elif zip_code == 94301:
        return 'Palo Alto'
    elif zip_code == 94107:
        return 'San Francisco'
    elif zip_code == 94063:
        return 'Redwood City'
    else:
        return 'Mountain View'

In [90]: #Se crea una columna city en weather para que coindida con joined_trips_station.
weather['city'] = weather['zip_code'].map(zip_ciudad)

In [91]: #Se mergean los DataFrames Weather y joined_trips_station en uno solo.
joined = joined_trips_station.merge(weather, left_on=['date', 'city'], right_on=['date', 'city'])

```

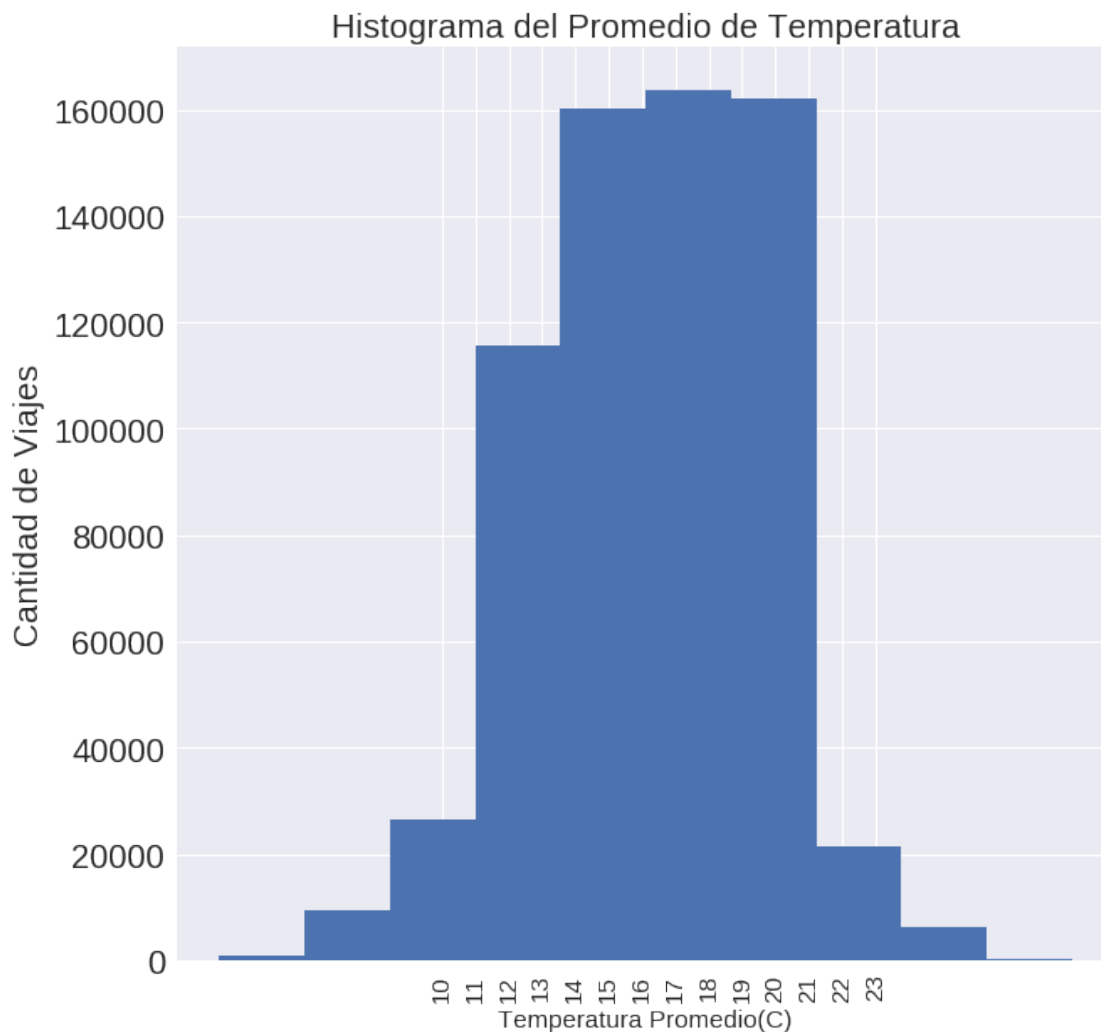

4.1 2.1) ¿Cómo se relacionan los viajes en bicicleta y la temperatura?

En esta serie de plots se analizará si hay una correlación entre los viajes en bicicleta y la temperatura.

Como los registros del clima están en forma diurna, es necesario tomar las fechas del viaje sin horarios. Para todos los plots se toma como fecha del viaje a la fecha de inicio del mismo, esto se debe a que es el momento en el cual la persona toma en cuenta las condiciones climáticas para decidir si realizar un viaje o no. Además, siguiendo el mismo criterio, se utiliza como localización para el clima a la localización de la estación de partida.

4.1.1 2.1.A) Histograma del Promedio de Temperatura

```
In [92]: joined.hist(column='mean_temperature_c', grid=True, figsize=(10,10), xrot=90, xlabelsize=15,
plt.xticks(range(10,24,1));
plt.xlabel('Temperatura Promedio(C)', fontsize=15);
plt.ylabel('Cantidad de Viajes', fontsize=20)
plt.title('Histograma del Promedio de Temperatura', fontsize=20);
```

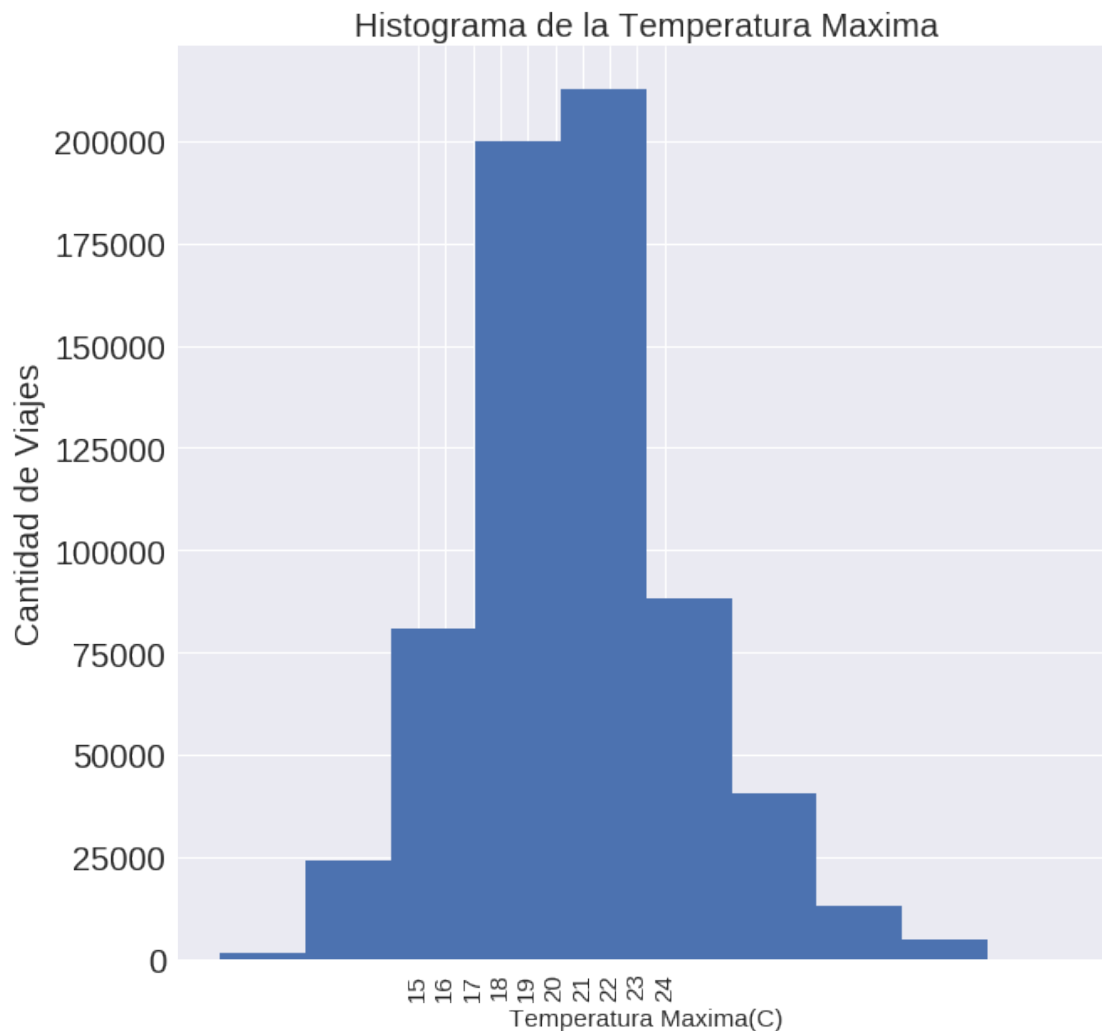


En este histograma se puede apreciar que la mayor cantidad de los viajes se realizan cuando la temperatura promedio esta entre 11°C y 21°C.

Si bien puede parecer que la temperatura es algo baja, hay que tener en cuenta que esto es un promedio de la temperatura de todo el día y como San Francisco es una ciudad costera la temperatura suele bajar bastante sobre la noche.

4.1.2 2.1.B) Histograma de la Temperatura Máxima

```
In [93]: joined.hist(column='max_temperature_c', grid=True, figsize=(10,10), xrot=90, xlabelsize=
plt.xticks(range(15,25,1));
plt.xlabel('Temperatura Maxima(C)', fontsize=15);
plt.ylabel('Cantidad de Viajes', fontsize=20)
plt.title('Histograma de la Temperatura Maxima', fontsize=20);
```



De este histograma se puede obtener que la mayoría de los viajes se realizan cuando la temperatura máxima está entre 17°C y 23°C.

Si se toman en cuenta los dos histogramas en conjunto, se puede apreciar que la mayoría de las personas buscan temperaturas templadas a la hora de realizar los viajes.

No es necesario analizar la temperatura mínima, ya que como se dijo antes, la temperatura suele bajar mucho sobre la noche y daría resultados engañosos.

4.1.3 2.1.C) Días con mayor uso simultáneo de Bicicletas y la Temperatura

```
In [94]: #Funciones para clasificar.
```

```
def f_st(row):
    if row['event'] == 'start_date':
        val = 1
    else:
        val = 0
    return val
```

```
def f_en(row):
    if row['event'] == 'end_date':
        val = 1
    else:
        val = 0
    return val
```

```
In [95]: trips_station_aux = joined_trips_station[['id_trip', 'start_date', 'end_date', 'city']]
trips_station_melt = pd.melt(trips_station_aux, id_vars=['id_trip', 'city'], value_vars=
trips_station_melt['time'] = pd.to_datetime(trips_station_melt['time']))
```

```
In [96]: #Se obtiene la cantidad de bicicletas en uso al mismo tiempo.
```

```
trips_station_ord = trips_station_melt.sort_values('time', ascending=True)
trips_station_ord['start_counter'] = trips_station_ord.apply(f_st, axis=1)
trips_station_ord['end_counter'] = trips_station_ord.apply(f_en, axis=1)
trips_station_ord['start'] = trips_station_ord['start_counter'].cumsum()
trips_station_ord['end'] = trips_station_ord['end_counter'].cumsum()
trips_station_ord = trips_station_ord[['id_trip', 'city', 'time', 'start', 'end']]
trips_station_ord['in_use'] = trips_station_ord['start'] - trips_station_ord['end']
trips_station_ord = trips_station_ord.sort_values('in_use', ascending=False)
```

```
In [97]: #Se eliminan los horarios para coincidir con weather.csv.
```

```
trips_station_ord['time'] = trips_station_ord['time'].apply(lambda x: x.date())
#Se convierte time a datetime64[ns].
trips_station_ord['time'] = pd.to_datetime(trips_station_ord['time'])
```

```
In [98]: #Se combinan los Dataframes.
```

```
joined_simul = trips_station_ord.merge(weather, left_on=['time', 'city'], right_on=['da
```

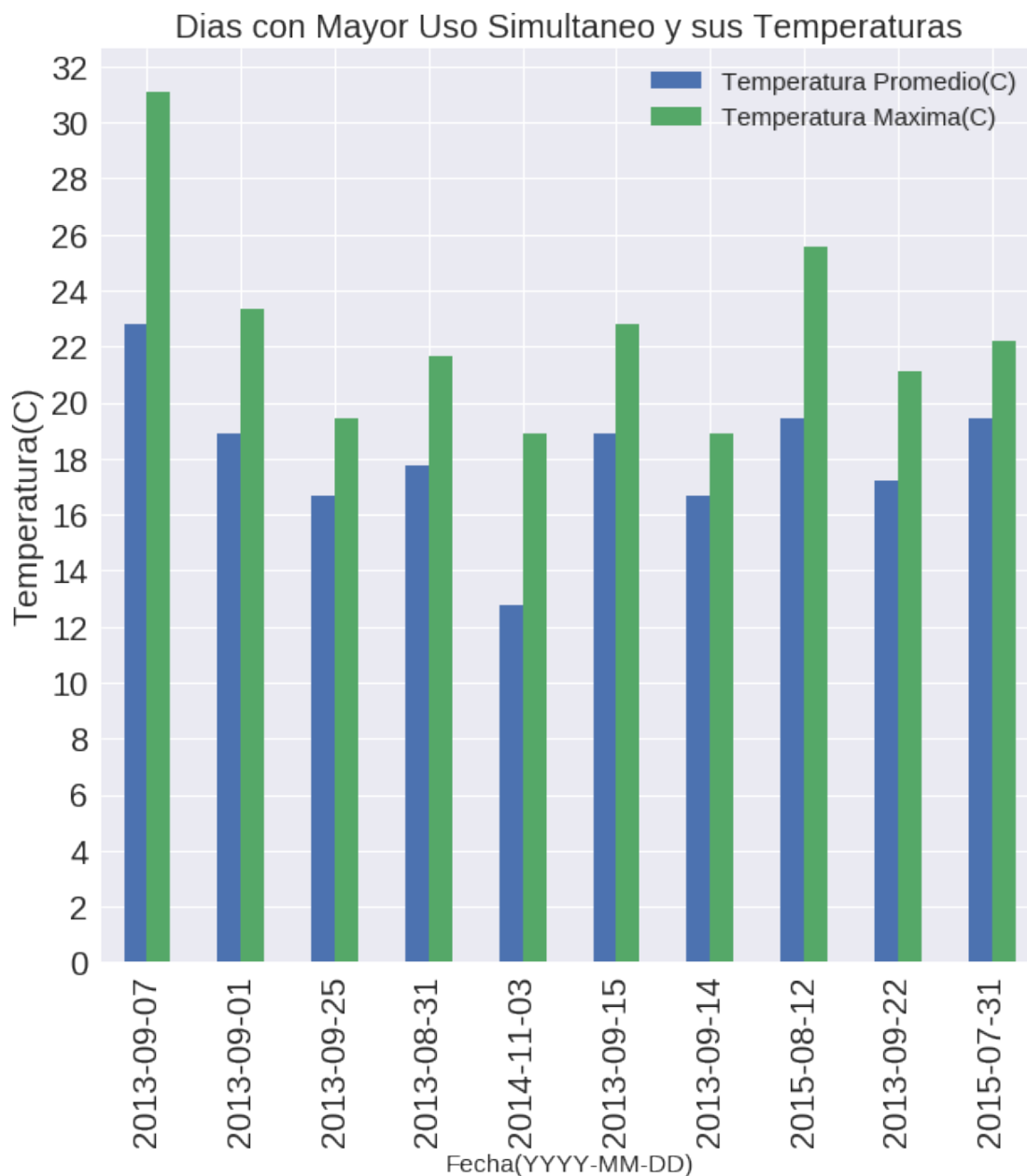
```
In [99]: #Solo hay que quedarse con el máximo de bicicletas simultáneas para ese día.
```

```
joined_max_simul = joined_simul.drop_duplicates(subset=['time'], keep='first')
```

```
In [100]: #Nos quedamos con los 10 valores maximos y las columnas que interesan.
joined_max_simul_bar = joined_max_simul[:10]
joined_max_simul_bar = joined_max_simul_bar[['time', 'mean_temperature_c', 'max_temper
joined_max_simul_bar.set_index('time', inplace=True)
```

En el siguiente gráfico de barras se pueden apreciar los 10 días (de los cuales hay datos de temperatura) con mayor uso simultaneo de bicicletas(ordenados de forma descendente), junto con su temperatura promedio y temperatura máxima. La localización tomada para el clima de cada día, es donde se produce su respectivo pico máximo de bicicletas en uso.

```
In [101]: bar = joined_max_simul_bar.plot.bar(figsize=(10,10), fontsize=20);
#Elimina el 00:00:00 del plot.
bar.set_xticklabels(joined_max_simul_bar.index.format());
plt.yticks(range(0,33,2));
plt.xlabel('Fecha(YYYY-MM-DD)', fontsize=15);
plt.ylabel('Temperatura(C)', fontsize=20);
plt.title('Dias con Mayor Uso Simultaneo y sus Temperaturas', fontsize=20);
plt.legend(['Temperatura Promedio(C)', 'Temperatura Maxima(C)'], fontsize=15);
```



Del este plot podemos obtener los siguientes puntos:

- * El primer punto importante es ver que el segundo día con mayor uso simultaneo tiene una temperatura máxima de 31°C y una temperatura promedio de 23°C. Este día claramente tiene temperaturas mucho más elevadas que los días restantes, por lo tanto, se puede tomar como una anomalía.
- * El segundo punto importante a notar es que la mayoría de estos días tienen temperaturas templadas, es decir, sus temperaturas promedio varían entre 13°C y 19°C mientras que sus temperaturas máximas están entre 19°C y 23°C.
- * Por ultimo, es evidente que si bien la cantidad de bicicletas en uso simultáneamente decrece a lo largo del gráfico de barras, las temperaturas se mantienen siempre dentro de un mismo rango.

4.1.4 2.1.D) Conclusión de Viajes y Temperatura

Usando los plots anteriores se puede concluir que la mayor cantidad de viajes se realizan cuando la temperatura es templada, es decir, cuando esta alrededor de los 20°C. Para describir este fenómeno hay que tener en cuenta los motivos por los cuales se podría realizar un viaje, para simplificar se tomaran dos casos: 1. En el caso de que se quiera realizar un viaje por la necesidad de trasladarse, la temperatura no afecta mucho, excepto en casos de muy bajas o muy altas temperaturas. 2. El caso en el cual se quiere realizar un viaje por placer es el que interesa, ya que es aquí cuando la temperatura juega un rol importante. Con temperaturas templadas las personas estarán mas predisuestas a realizar viajes en bicicleta, como muestran los plots obtenidos.

Por ultimo, es importante notar que las mejores temperaturas se suelen presentar sobre la media mañana, ya que el sol todavía no esta en su punto más alto, y sobre la tarde, ya que es cuando comienza a bajar el sol, por eso no seria extraño que la mayor cantidad de los viajes se realicen en horarios de la media mañana o de la tarde.

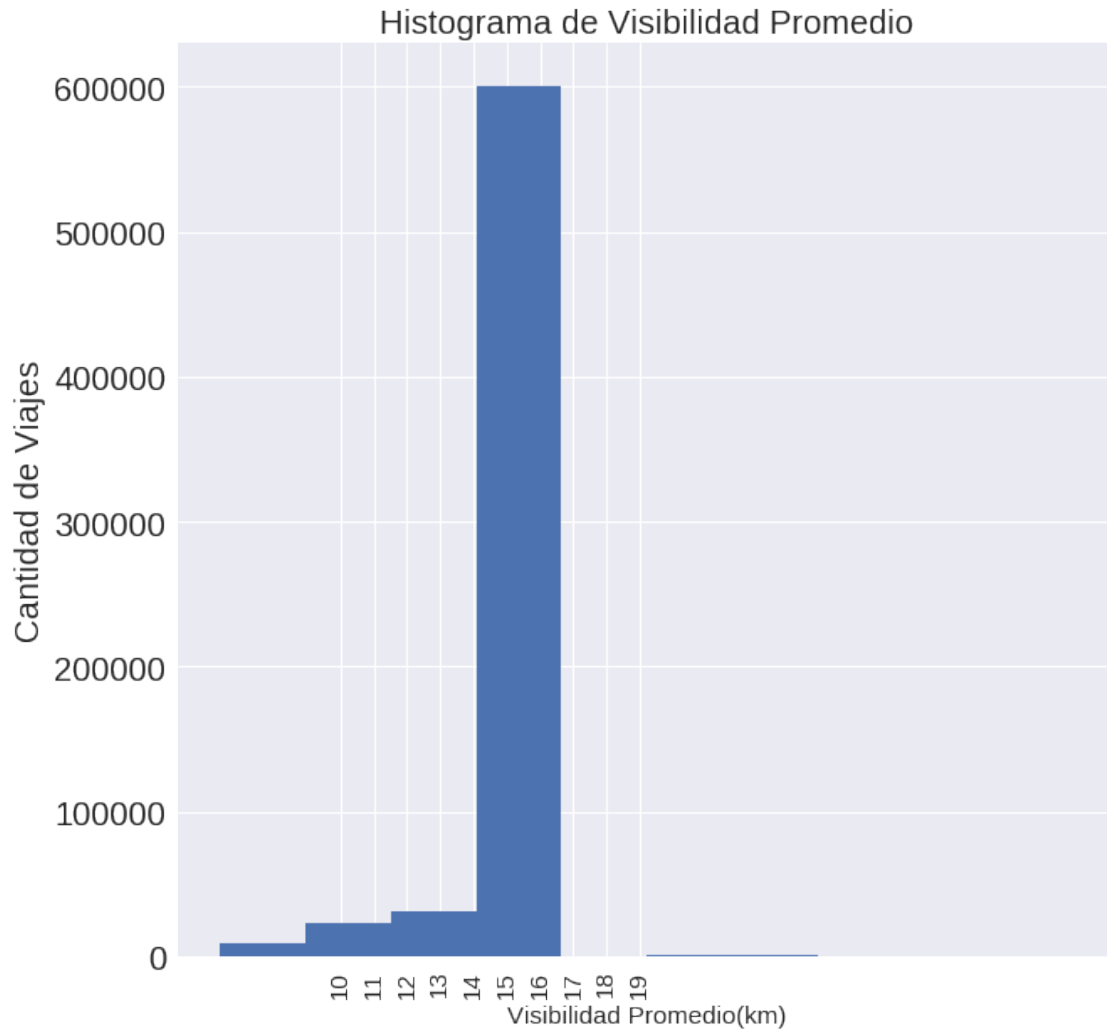
4.2 2.2) ¿Como se relacionan los viajes y la visibilidad?

En la siguiente serie de plots se tratara de analizar si hay una correlación entre los viajes en bicicletas y la visibilidad.

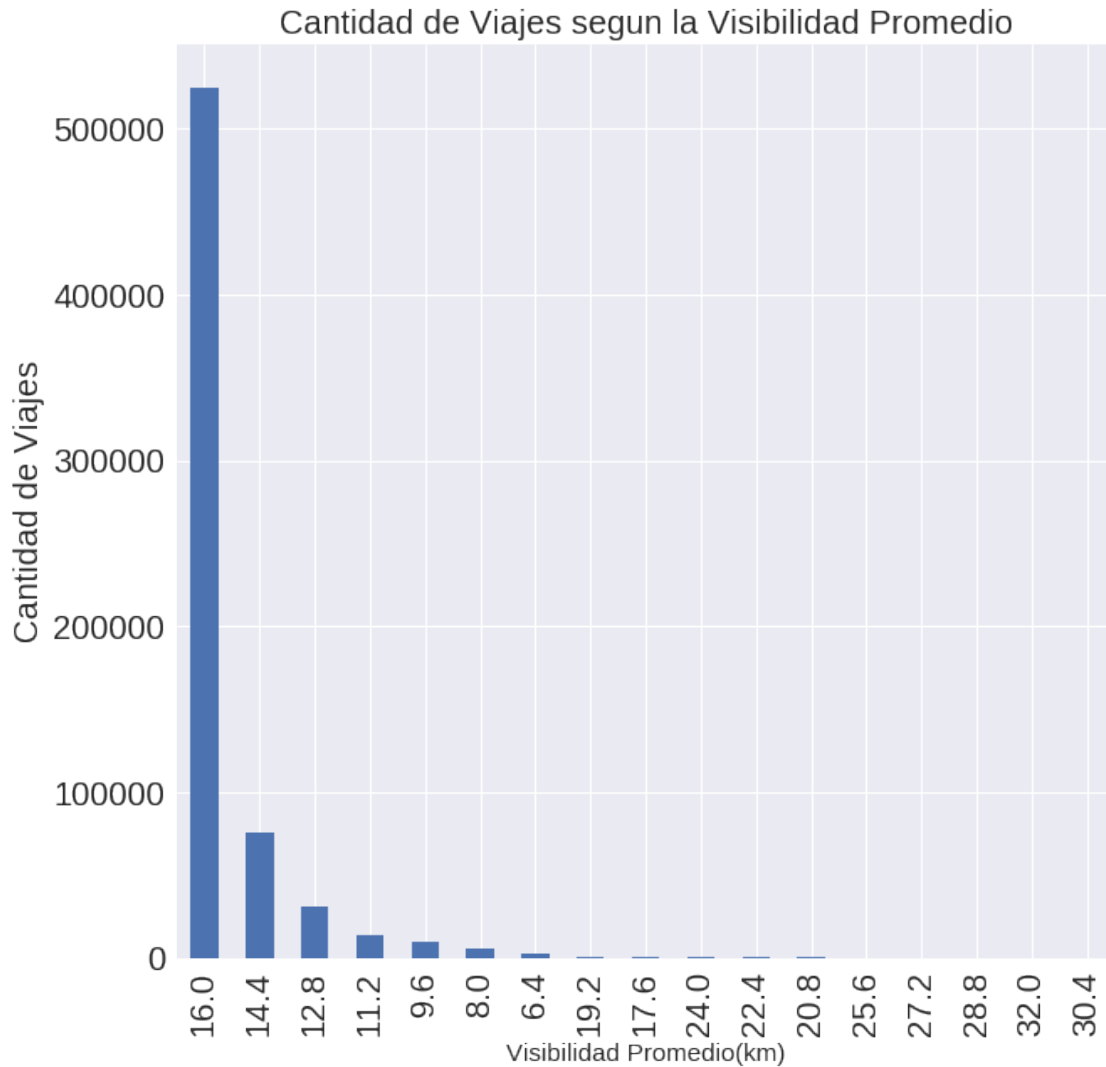
Como los registros del clima están en forma diurna, es necesario tomar las fechas del viaje sin horarios. Para todos los plots se toma como fecha del viaje a la fecha de inicio del mismo, esto se debe a que es el momento en el cual la persona toma en cuenta las condiciones climáticas para decidir si realizar un viaje o no. Además, siguiendo el mismo criterio, se utiliza como localización para el clima a la localización de la estación de partida.

4.2.1 2.2.A) Grafico de Barras de Visibilidad Promedio y Histograma de Visibilidad Promedio

```
In [102]: joined.hist(column='mean_visibility_km', grid=True, figsize=(10,10), xrot=90, xlabelsi
plt.xticks(range(10,20,1));
plt.xlabel('Visibilidad Promedio(km)', fontsize=15);
plt.ylabel('Cantidad de Viajes', fontsize=20)
plt.title('Histograma de Visibilidad Promedio', fontsize=20);
```



```
In [103]: joined['mean_visibility_km'].value_counts(sort=True).plot.bar(figsize=(10,10), fontsize=15);  
plt.xlabel('Visibilidad Promedio(km)', fontsize=15);  
plt.ylabel('Cantidad de Viajes', fontsize=20)  
plt.title('Cantidad de Viajes segun la Visibilidad Promedio', fontsize=20);
```



De este plot se pueden extraer tres puntos importantes:

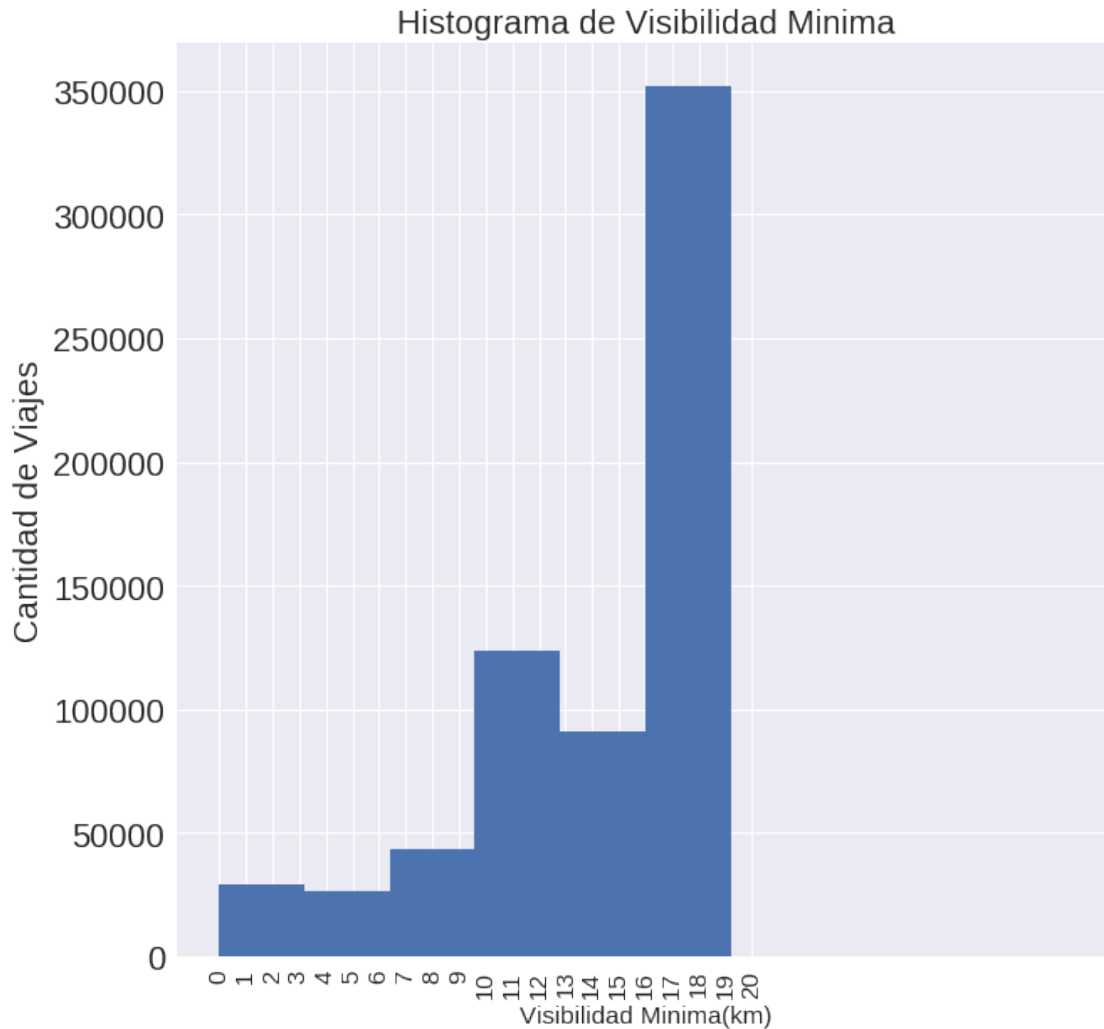
* La mayoría de los viajes en bicicleta se realizan cuando la visibilidad promedio es de 16km. * Por encima de los 16km de visibilidad promedio hay escasez de datos, por lo tanto, se puede intuir que no es normal que en San Francisco se presenten visibilidades promedio tan elevadas.

* Sino se toman en cuenta las visibilidades promedio mayores a 16km(porque no hay datos suficientes), la diferencia entre las visibilidades promedio mayores o iguales a 14.4km y las menores a esta es abismal. Esto no es un dato menor, ya que hay que tener una buena visibilidad al realizar un viaje para que este sea seguro y queda claro, por esta diferencia, que el usuario prioriza mucho la seguridad.

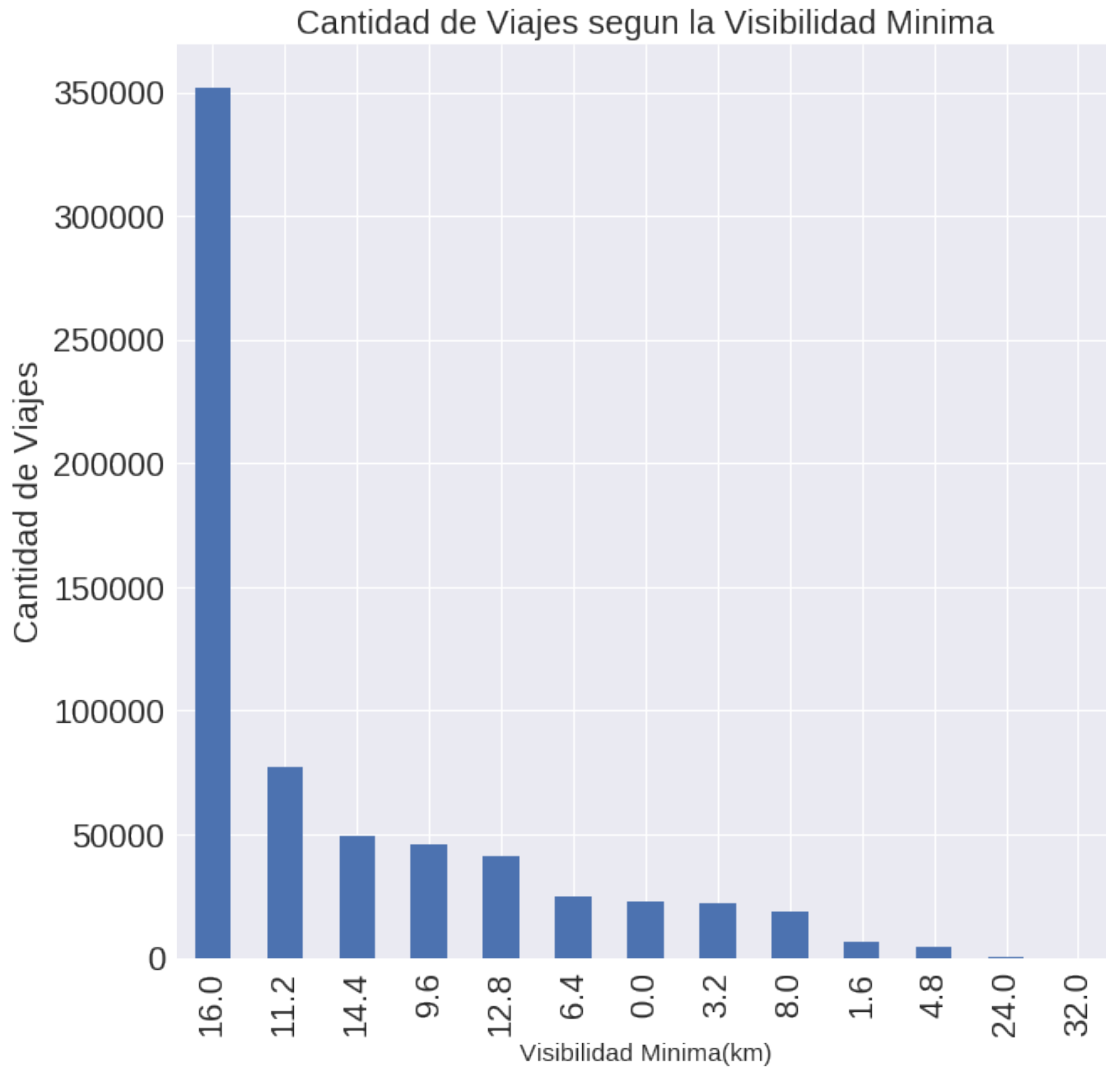
Ahora sería importante analizar que sucede con la visibilidad mínima, ya que por cuestiones de seguridad es importante que haya una buena visibilidad mínima para evitar accidentes.

4.2.2 2.2.B) Grafico de Barras de Visibilidad Mínima y Histograma de Visibilidad Mínima

```
In [104]: joined.hist(column='min_visibility_km', grid=True, figsize=(10,10), xrot=90, xlabelsize=15,
plt.xticks(range(0,21,1));
plt.xlabel('Visibilidad Minima(km)', fontsize=15);
plt.ylabel('Cantidad de Viajes', fontsize=20)
plt.title('Histograma de Visibilidad Minima', fontsize=20);
```



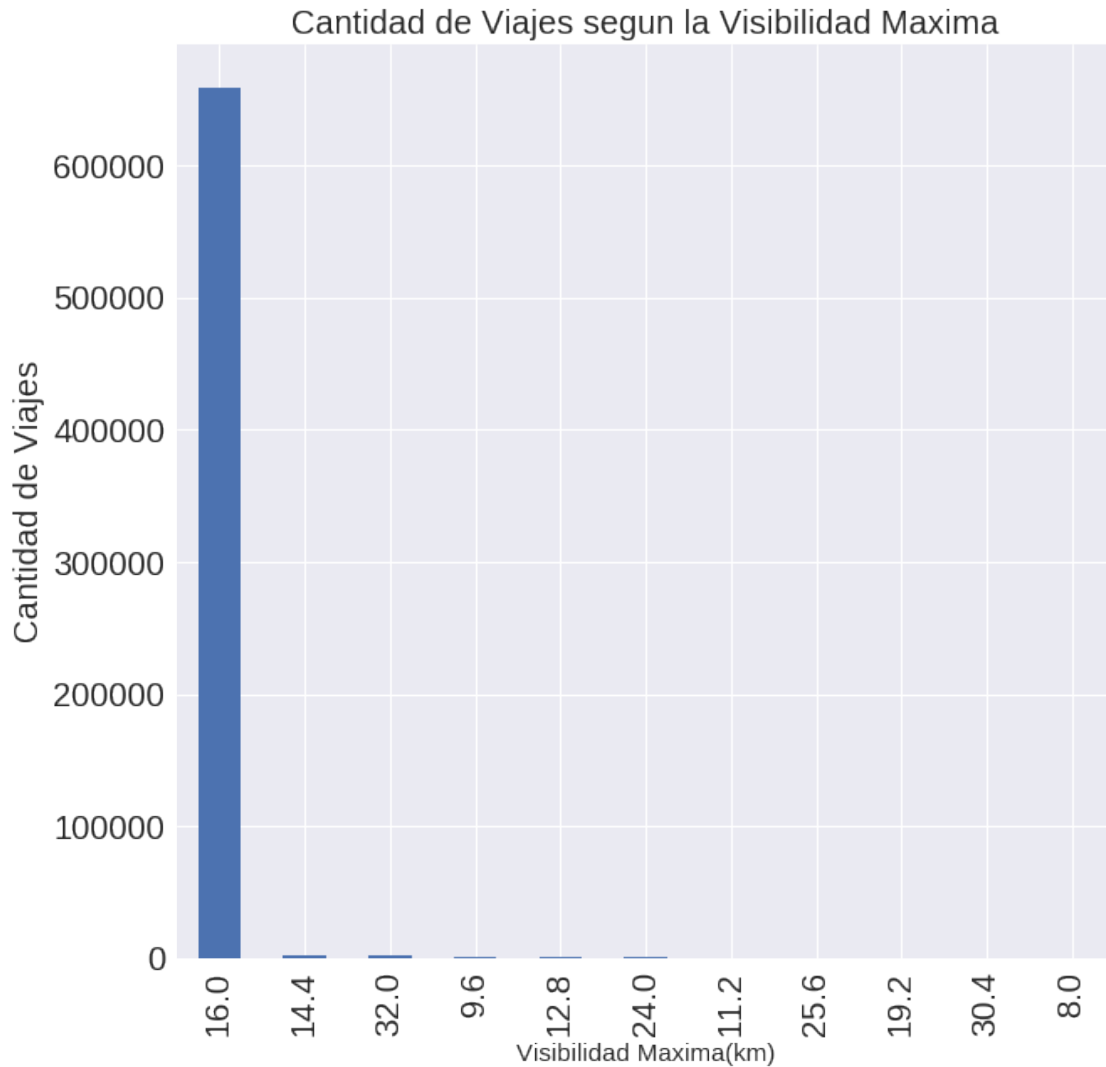
```
In [105]: joined['min_visibility_km'].value_counts(sort=True).plot.bar(figsize=(10,10), fontsize=15,
plt.xlabel('Visibilidad Minima(km)', fontsize=15);
plt.ylabel('Cantidad de Viajes', fontsize=20)
plt.title('Cantidad de Viajes segun la Visibilidad Minima', fontsize=20);
```



En estos dos plots se apreciar de nuevo que la mayor cantidad de viajes se realiza cuando la visibilidad mínima es de 16km. Además es importante notar que de nuevo se manifiesta la gran separación entre la visibilidad de 16km y las que son menores a esta. Por ejemplo, si tomamos la visibilidad mínima de 11.2km podemos ver que hay una diferencia aproximada de 275000 viajes con la visibilidad de 16km.

4.2.3 2.2.C) Grafico de Barras de Visibilidad Máxima

```
In [106]: joined['max_visibility_km'].value_counts(sort=True).plot.bar(figsize=(10,10), fontsize=15);
plt.xlabel('Visibilidad Maxima(km)', fontsize=15);
plt.ylabel('Cantidad de Viajes', fontsize=20)
plt.title('Cantidad de Viajes segun la Visibilidad Maxima', fontsize=20);
```



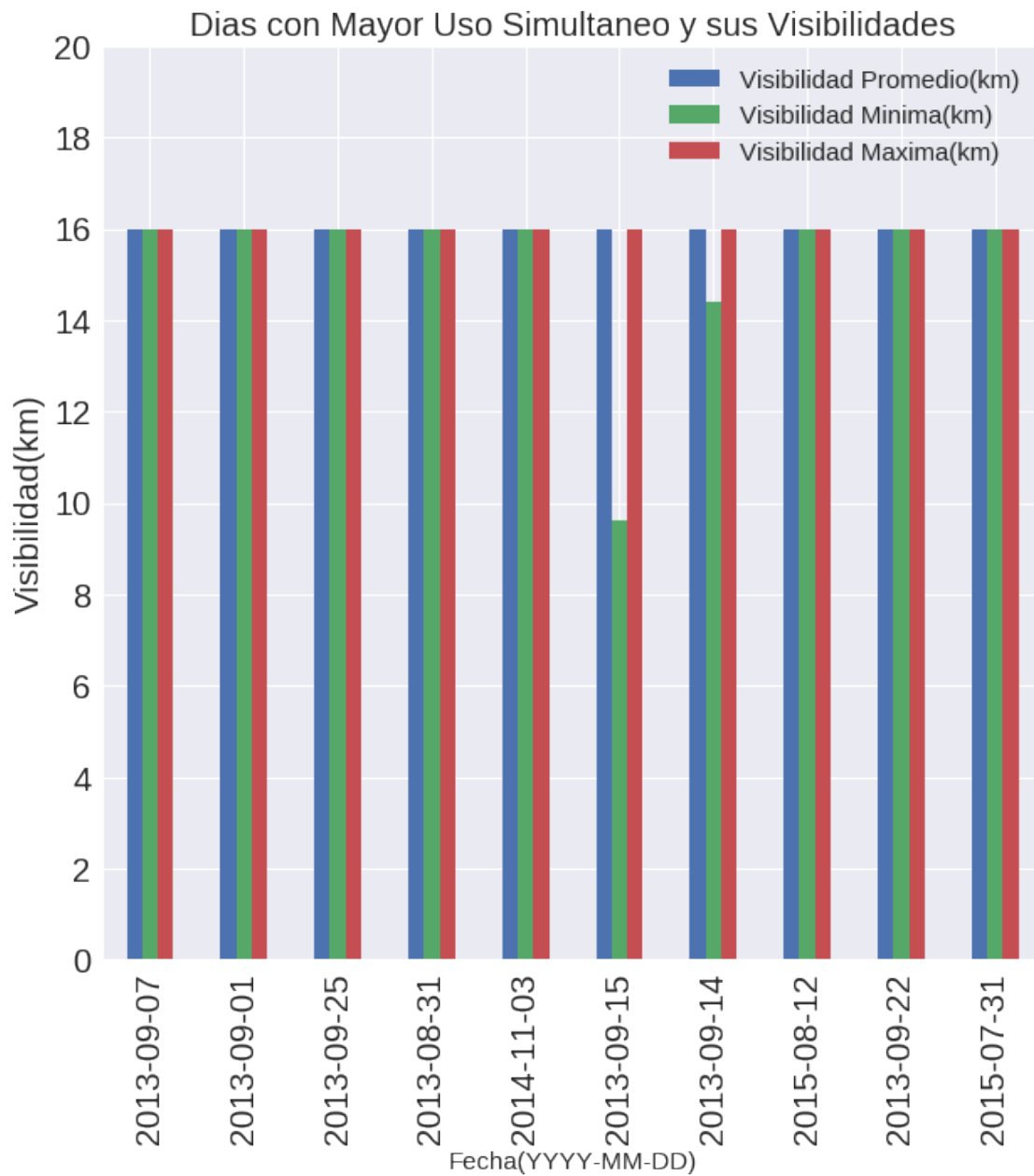
De este plot es muy difícil sacar conclusiones ya que la diferencia de la visibilidad máxima de 16km con las demás es demasiado grande. Lo único importante a destacar, es que hay una ínfima cantidad de viajes realizados con visibilidades máximas menores o iguales a 10km.

4.2.4 2.D) Dias con mayor uso simultaneo de Bicicletas y la Visibilidad

```
In [107]: #Nos quedamos con los 10 valores maximos y las columnas que interesan.
joined_max_simul_vis_bar = joined_max_simul[:10]
joined_max_simul_vis_bar = joined_max_simul_vis_bar[['time', 'mean_visibility_km', 'mi
joined_max_simul_vis_bar.set_index('time', inplace=True)
```

En el siguiente gráfico de barras se pueden apreciar los 10 días (de los cuales hay datos de visibilidad) con mayor uso simultaneo de bicicletas(ordenados de forma descendente), junto con su visibilidad promedio, visibilidad mínima y visibilidad máxima. La localización tomada para el clima de cada día, es donde se produce su respectivo pico máximo de bicicletas en uso.

```
In [108]: bar = joined_max_simul_vis_bar.plot.bar(figsize=(10,10), fontsize=20);
#Elimina el 00:00:00 del plot.
bar.set_xticklabels(joined_max_simul_vis_bar.index.format());
plt.yticks(range(0,22,2));
plt.xlabel('Fecha(YYYY-MM-DD)', fontsize=15);
plt.ylabel('Visibilidad(km)', fontsize=20);
plt.title('Dias con Mayor Uso Simultaneo y sus Visibilidades', fontsize=20);
plt.legend(['Visibilidad Promedio(km)', 'Visibilidad Minima(km)', 'Visibilidad Maxima(km)'])
```



Este plot confirma la tendencia que se había marcado previamente, los 10 días con mayor uso simultaneo de bicicletas presentan visibilidades promedio y visibilidades máximas iguales a los 16km.

Ademas todos la mayoría de los días presentan visibilidades mínimas iguales o cercanas a 16km. El único caso llamativo es del día 2013-09-15 en el cual la visibilidad mínima es de aproximadamente 9km, pero si bien su visibilidad mínima es baja su visibilidad máxima y visibilidad promedio se mantienen dentro de lo esperado. Esto nos indicaría que la mayor cantidad de viajes de ese día se habría producido en horarios donde la visibilidad era más cercana a la máxima que a la mínima.

4.2.5 2.2.E) Conclusión de Viajes y Visibilidad

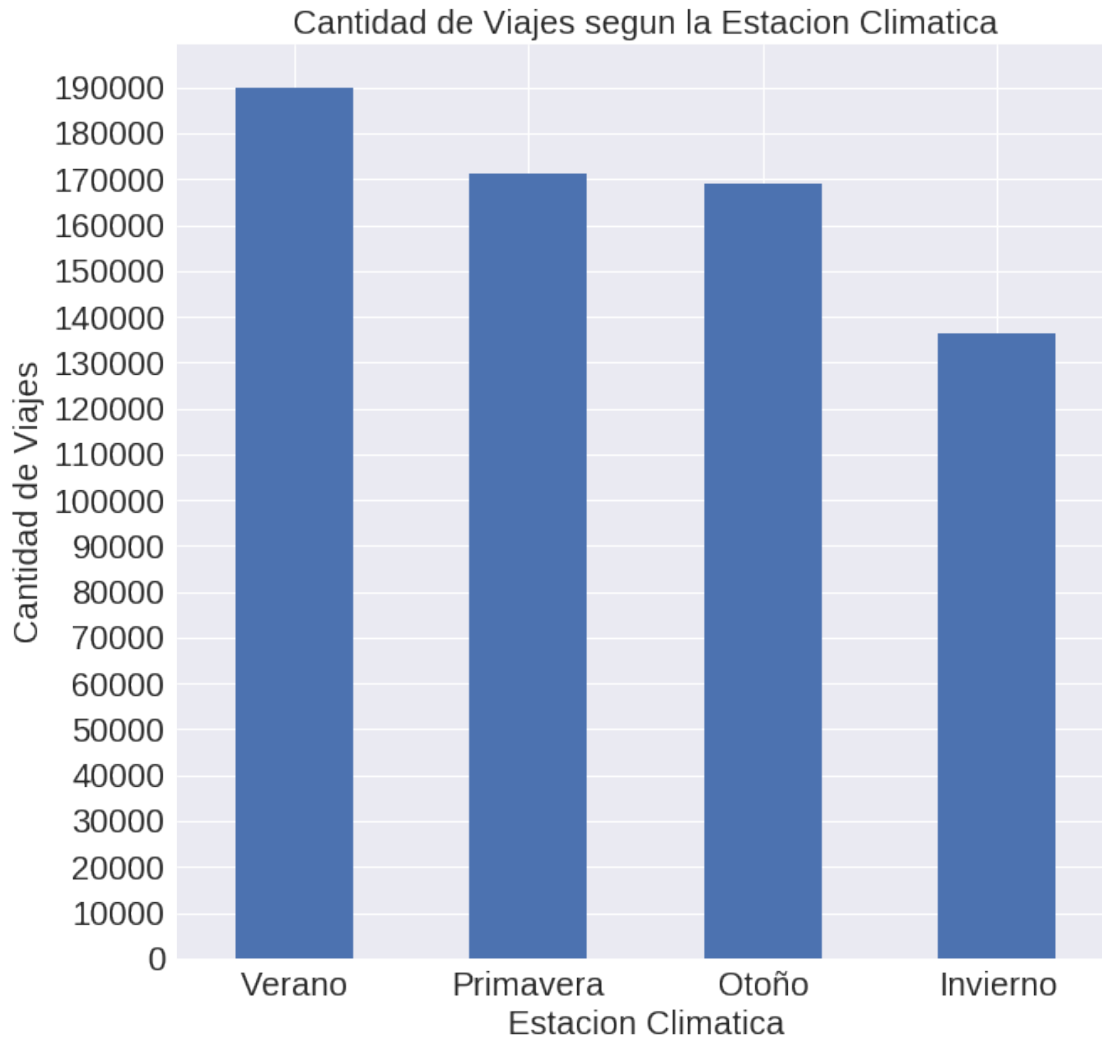
Dados los plots vistos se puede concluir claramente que la visibilidad máxima, mínima y mediana deben ser de, o al menos cercano a, 16km para que se produzca un mayor uso del servicio de bicicletas. Esto claramente esta relacionado, como ya se menciona antes, con la seguridad, los usuarios tienden a poner a la seguridad como uno de los puntos más importantes a la hora de decidir si realizar un viaje en bicicleta o no.

4.3 2.3) ¿Como se relacionan los viajes y las estaciones climáticas?

4.3.1 2.3.A) Cantidad de Viajes y Estaciones Climáticas

En la siguiente serie de plots se analizará si hay una relación entre los viajes en bicicleta y las estaciones climáticas.

```
In [109]: trips['estacion_clima'].value_counts(sort=True).plot.bar(figsize=(10,10), rot=0, fontsize=10)
plt.xticks(range(0,200000,10000))
plt.xlabel('Estacion Climatica', fontsize=20);
plt.ylabel('Cantidad de Viajes', fontsize=20)
plt.title('Cantidad de Viajes segun la Estacion Climatica', fontsize=20);
```



De este plot podemos observar:

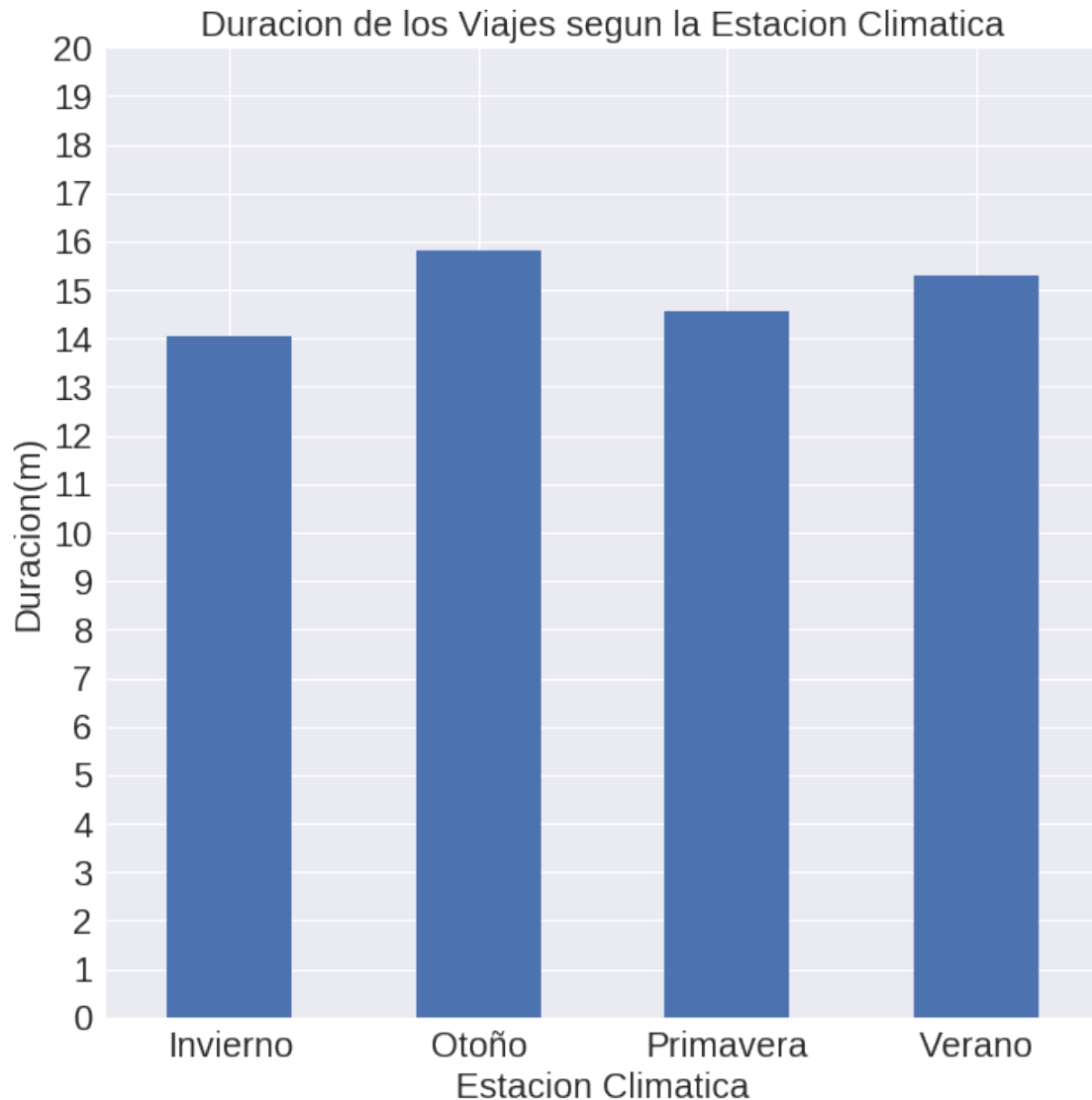
- * La estación climática con la mayor cantidad de viajes es Verano. Esto esta dentro de lo esperado ya que en Verano se llega fácilmente a temperaturas de 20°C.
- * Las siguientes dos estaciones son Primavera y Otoño. Esto de nuevo concuerda con lo dicho previamente, en estas estaciones climáticas hay temperaturas que se acercan a 20°C.
- * Por ultimo, es importante notar que todas las estaciones mantienen un piso de 130000 viajes. Esto va de la mano con lo dicho en la sección 2.1.D y 2.2.E, si alguien quiere utilizar el servicio de bicicletas por un motivo no recreativo, no suele darle demasiada importancia a factores climáticos excepto que conlleven un riesgo para su seguridad.

4.3.2 2.3.B) Duración promedio de Viajes y Estaciones Climáticas

En este plot se analizará las duraciones promedio de los viajes en cada estación.

```
In [110]: grouped_season = trips[['estacion_clima', 'duration_m']].groupby('estacion_clima')\
          .aggregate('mean')
```

```
In [111]: grouped_season.plot.bar(figsize=(10,10), rot=0, fontsize=20);
plt.yticks(range(0,21,1));
plt.xlabel('Estacion Climatica', fontsize=20);
plt.ylabel('Duracion(m)', fontsize=20)
plt.title('Duracion de los Viajes segun la Estacion Climatica', fontsize=20);
plt.legend('');
```



De este plot podemos hacer tres observaciones importantes:

* Primero, si bien Invierno es la estación climática con menor cantidad de viajes, es la que tiene la mayor duración promedio. * Segundo, tanto Verano como Primavera, que son las estaciones climáticas con la mayor cantidad de viajes, tienen las menores duraciones promedio. * Por ultimo, todas las duraciones promedio se mantienen dentro de un rango bastante pequeño(17m - 19m).

4.3.3 2.3.C) Conclusión de Viajes y Estaciones Climáticas

Las conclusiones que se pueden realizar a raíz de los plots a analizados son:

- * Hay una mayor cantidad de viaje en estaciones climáticas donde la temperatura es mas cálida. Además la diferencia, en termino de cantidades de viajes, entre Verano e Invierno es bastante grande en favor del Verano.

- * Como se expreso anteriormente, todas las estaciones tienen un piso 130000 viajes. Esto da a entender que una gran parte de los usuarios no pone como prioridad al clima a la hora de realizar un viaje, esto se puede deber a que la mayoría de los usuarios utiliza el servicio de bicicletas como modo de transporte y no como en forma recreativa.

5 2.4) ¿Que sucede con los 5 viajes de mayor duración?

En los siguientes plots se analizara que sucede con la temperatura, visibilidad y estación climática para los 5 viajes con mayor duración. Se utilizan las duraciones menores o iguales a 12 horas porque se considera que un viaje continuado puede durar, como mucho, 12 horas si se permiten interrupciones de duración corta. Además se toman solo los viajes que ocurren durante el día para evitar confusiones con bicicletas que olvidaron entregarse.

Algunos de los motivos por los cuales se pueden realizar viajes largos son, por ejemplo:

- * Realizar un tour de la ciudad de San Francisco.

- * Realizar actividad física intensiva. * Tener que realizar varios trayectos en horarios diferentes.

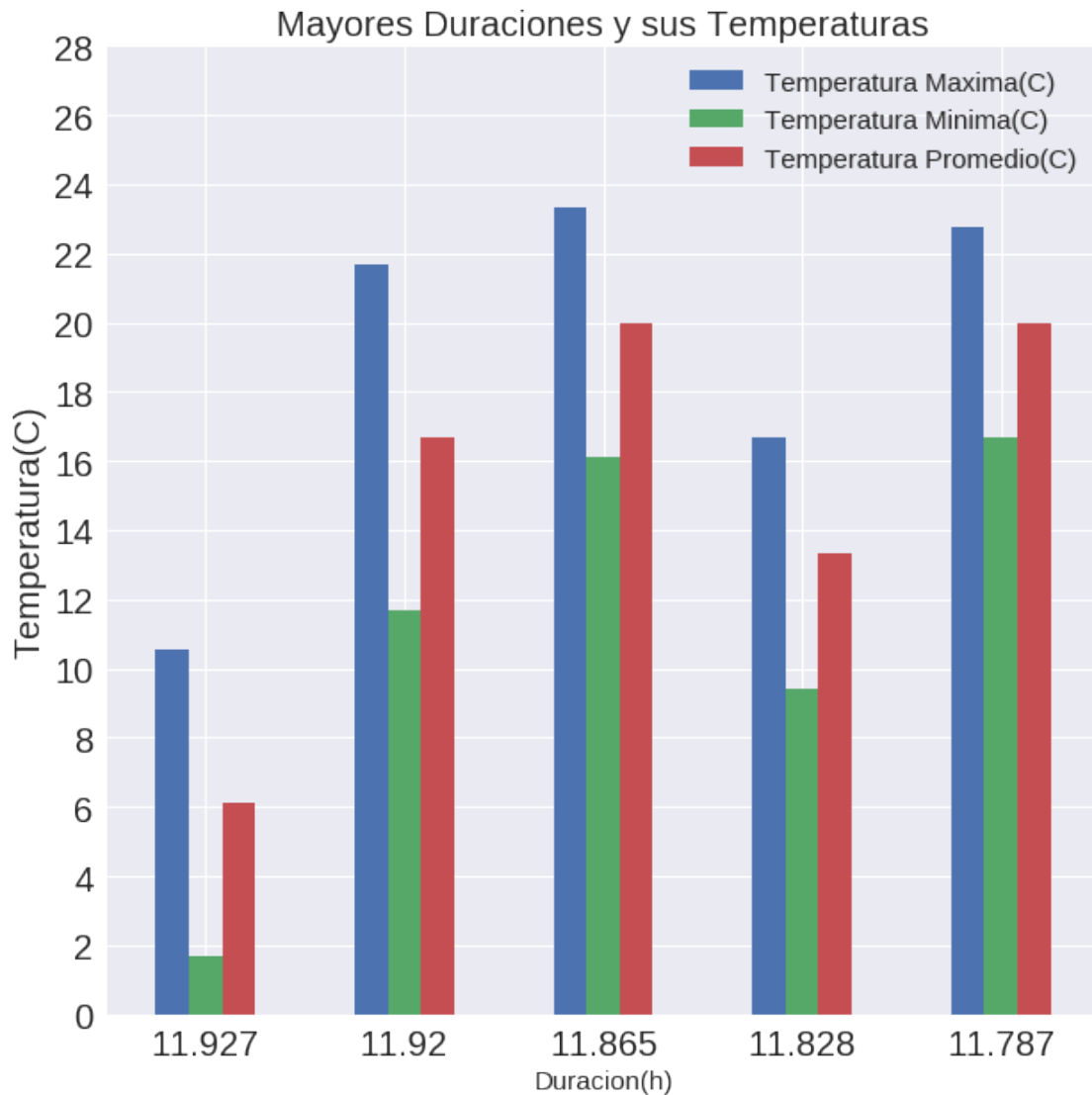
```
In [112]: #Se obtienen los 5 viajes con mayor duracion (que poseen datos climaticos) menores a 12 horas
#que ocurran durante el dia (se toman solo viajes que comiencen a la mañana).
#top_dur = joined[(joined['start_hour'] >= 7) & (joined['start_hour'] <= 11) & (joined['duration_h'] <= 12)]
top_dur = joined[(joined['start_hour'] >= 7) & (joined['start_hour'] <= 11)]
top_dur = top_dur.sort_values('duration_h', ascending=False)[:5]
```

5.0.1 2.4.A) Temperatura de los 5 Viajes con Mayor Duración

```
In [113]: top_dur_temp = top_dur[['duration_h', 'max_temperature_c', 'min_temperature_c', 'mean_temperature_c']]
top_dur_temp.set_index('duration_h', inplace=True)
```

Se muestran los 5 viajes con mayor duración (con sus respectivas duraciones, en horas, como label) ordenados de forma descendente junto con sus respectivas temperaturas máximas, temperaturas mínimas y temperaturas promedio.

```
In [114]: top_dur_temp_bar = top_dur_temp.plot.bar(figsize=(10,10), fontsize=20, rot=0);
plt.xticks(range(0,30,2));
plt.xlabel('Duracion(h)', fontsize=15);
plt.ylabel('Temperatura(C)', fontsize=20);
plt.title('Mayores Duraciones y sus Temperaturas', fontsize=20);
plt.legend(['Temperatura Maxima(C)', 'Temperatura Minima(C)', 'Temperatura Promedio(C)'])
```

Dado este plot se puede decir que las temperaturas varían mucho entre los distintos días:

* La temperatura máxima está entre aproximadamente 10°C y 23°C.

* La temperatura mínima está entre aproximadamente 2°C y 16°C.

* La temperatura promedio está entre aproximadamente 6°C y 20°C.

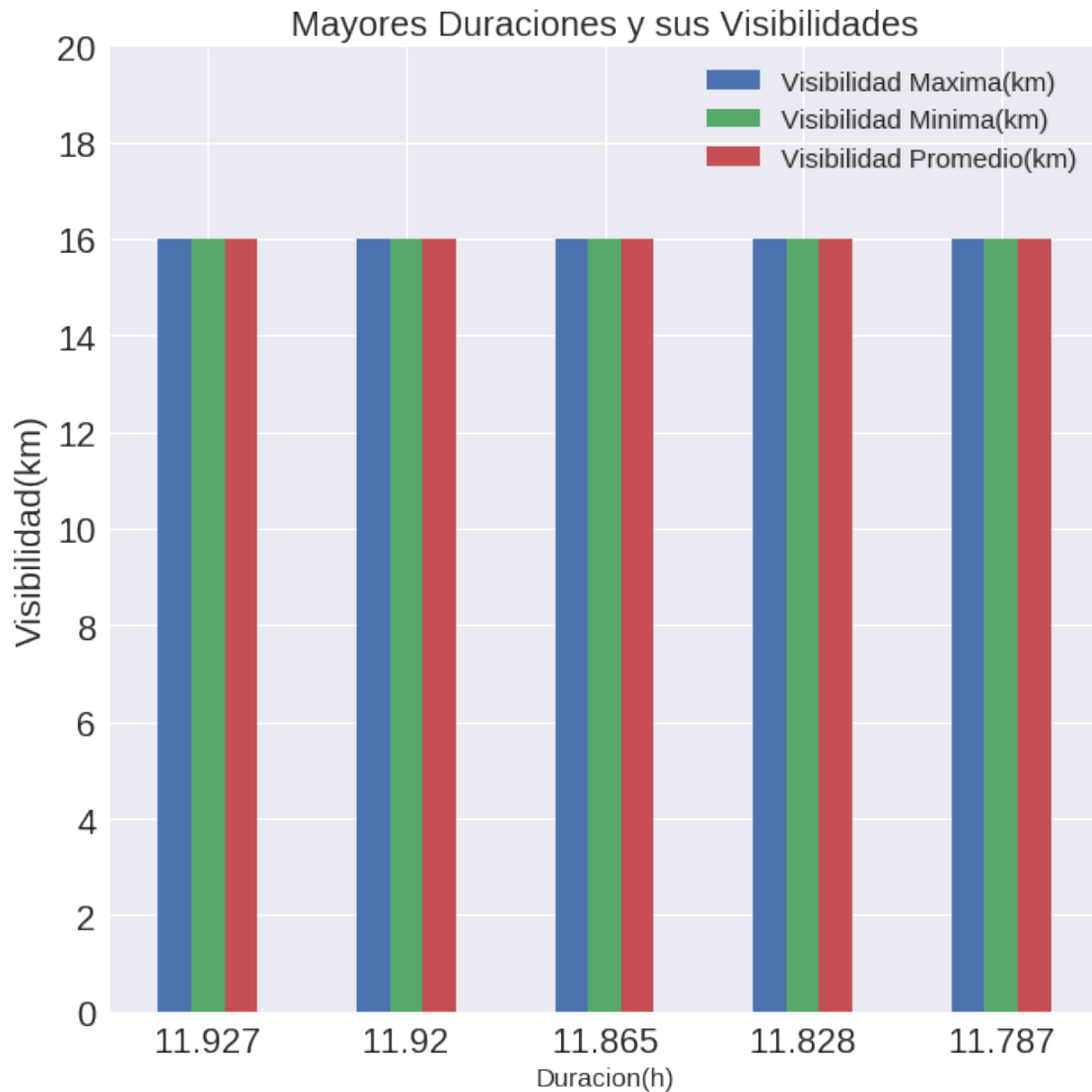
Esto indica que no hay una relación entre los viajes de mayor duración y la temperatura.

5.0.2 2.4.B) Visibilidad de los 5 Viajes con Mayor Duración

```
In [115]: top_dur_vis = top_dur[['duration_h', 'max_visibility_km', 'min_visibility_km', 'mean_v
top_dur_vis.set_index('duration_h', inplace=True)
```

Se muestran los 5 viajes con mayor duración (con sus respectivas duraciones, en horas, como label) ordenados de forma descendente junto con sus respectivas visibilidades máximas, visibilidades mínimas y visibilidades promedio.

```
In [116]: top_dur_vis_bar = top_dur_vis.plot.bar(figsize=(10,10), fontsize=20, rot=0);
plt.yticks(range(0,22,2));
plt.xlabel('Duracion(h)', fontsize=15);
plt.ylabel('Visibilidad(km)', fontsize=20);
plt.title('Mayores Duraciones y sus Visibilidades', fontsize=20);
plt.legend(['Visibilidad Maxima(km)', 'Visibilidad Minima(km)', 'Visibilidad Promedio(km)'])
```



Este plot es muy claro e indica que para que se produzcan viajes de esta duración la visibilidad debe ser muy buena, es decir, visibilidad de 16km. Esto está acorde con lo dicho en la sección 2.2.

5.0.3 2.4.C) Estación Climática de los 5 Viajes con Mayor Duración

```
In [117]: #Se muestran las duracion(en horas) y su respectiva estacion.
top_dur[['duration_h', 'estacion_clima']]
```

```
Out[117]:
```

	duration_h	estacion_clima
75314	11.927	Invierno
626559	11.920	Primavera
329226	11.865	Verano
63402	11.828	Otoño
168239	11.787	Verano

Las estaciones climáticas de los viajes de mayor duración, ordenados de mayor a menor, son:

1. Invierno
2. Primavera
3. Verano
4. Otoño
5. Verano

Dados estos datos se puede decir que:

* La mayor duración de los 5 viajes se produce en Invierno y la menor duración de los 5 se produce en Verano. * La distribución de los 5 viajes en las estaciones climáticas es muy pareja, por lo tanto, se puede decir que la estación climática no afecta mucho en la realización de estos viajes.

5.0.4 2.4.D) Conclusión del Caso Particular

Según la serie de datos analizados podemos decir que para que se produzcan viajes de gran duración el único requerimiento es que la visibilidad debe ser muy buena (igual a 16km). Además si tenemos en cuenta que para realizar estos viajes se requieren muchas horas, podemos decir que estos ocurrirán más en días no laborables o los que realizan este tipo de viajes son turistas.

5.1 2.5) ¿Como se relacionan los viajes en bicicleta y la lluvia?

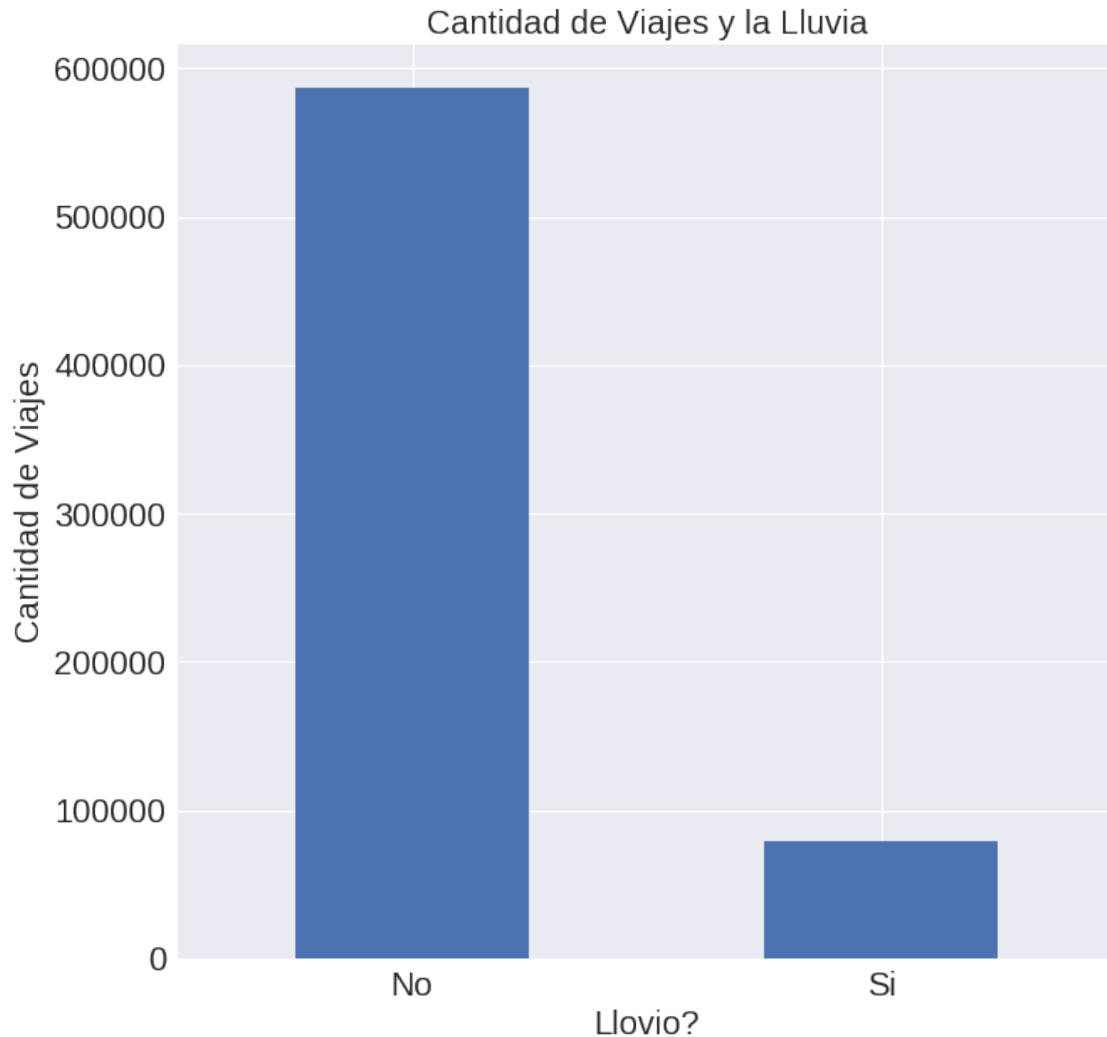
En la siguiente serie de plots se analizara si hay una relación entre los viajes en bicicleta y la lluvia.

```
In [118]: #Funcion para clasificar lluvia.
def lluvia_y_n(event):
    if isinstance(event, float):
        return 'No'
    elif 'rain' in event.lower():
        return 'Si'
    else:
        return 'No'
```

```
In [119]: joined['lluvia'] = joined['events'].map(lluvia_y_n)
```

5.1.1 2.5.A) Cantidad de Viajes con Lluvia Vs. Cantidad de Viajes sin Lluvia

```
In [120]: joined['lluvia'].value_counts().plot.bar(figsize=(10,10), fontsize=20, rot=0);
plt.xlabel('Lluvia?', fontsize=20);
plt.ylabel('Cantidad de Viajes', fontsize=20);
plt.title('Cantidad de Viajes y la Lluvia', fontsize=20);
plt.legend('');
```



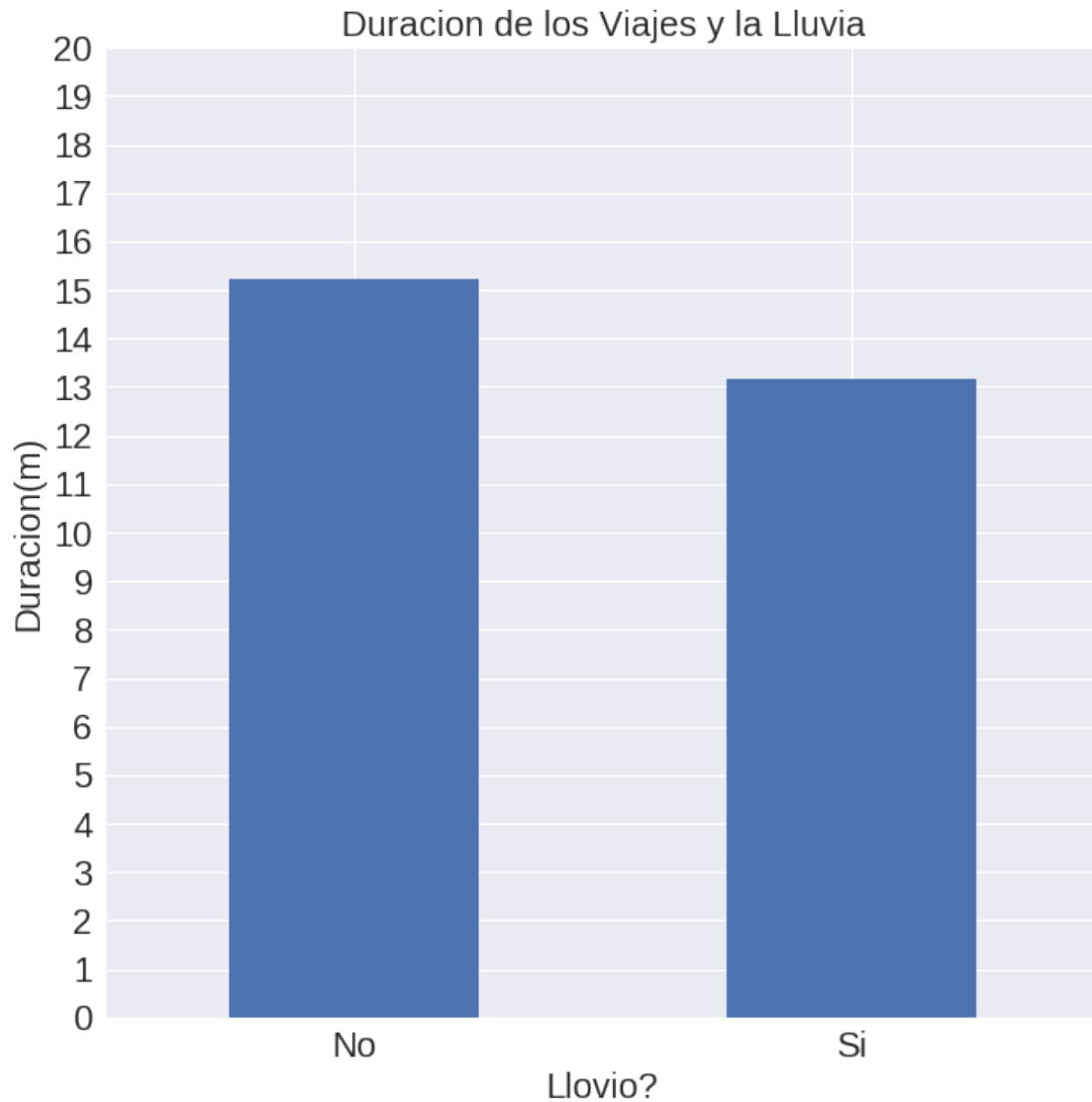
Las observaciones que se hacen de este plot son:

* Hay una diferencia de casi 500000 viajes a favor de los días en los que no llueve. Por ende, se puede decir que se viaja mucho más cuando no llueve.

* Los casi 100000 viajes que se producen cuando llueve se pueden deber a que la intensidad de la lluvia es demasiado baja y, por lo tanto, los usuarios optan por seguir viajando.

5.1.2 2.5.B) Duración promedio de los Viajes con Lluvia Vs. Duración promedio de los Viajes sin Lluvia

```
In [121]: joined[['lluvia', 'duration_m']].groupby('lluvia').aggregate('mean').plot.bar(figsize=
plt.yticks(range(0,21,1));
plt.xlabel('Llovio?', fontsize=20);
plt.ylabel('Duracion(m)', fontsize=20)
plt.title('Duracion de los Viajes y la Lluvia', fontsize=20);
plt.legend('');
```



Es interesante lo que se puede obtener de este plot. Si bien se realizan muchos más viajes cuando no llueve que cuando llueve, la duración promedio de los viajes cuando llueve es mayor que cuando no llueve. Teniendo en cuenta que hay muchos menos viajes con lluvia que sin lluvia y que eso puede afectar el promedio, esto vuelve a indicar que los viajes que se realizan cuando llueve se producirían cuando la intensidad de la lluvia es baja.

5.1.3 2.5.C) Conclusión de Viajes y Lluvia

La conclusión de esta serie de plots es muy clara, se realizan más viajes cuando no llueve que cuando llueve. Pero las duraciones de los viajes que se realizan cuando llueve son mayores que las duraciones de un viaje promedio cuando no llueve. Como se menciono previamente ese dato puede ser engañoso, pero una posible causa por la cual se presenta este fenómeno es que los viajes se realizan cuando la lluvia tiene una intensidad baja.

6 3) Estaciones y Trayectos

6.1 3.0) Lectura y limpieza de datos

```
In [124]: from mpl_toolkits.basemap import Basemap
```

```
stations = pd.read_csv('../data/station.csv', low_memory=False)

stations.rename(columns={'installation_date': 'date', 'long': 'lon'}, inplace=True)

#Se cambia el formato de las fechas
stations['date'] = pd.to_datetime(stations['date'])
```

```
In [125]: trips = pd.read_csv('../data/trip.csv', low_memory=False)
```

```
#Se eliminan los zip-codes ya que no seran relevantes para estos analisis
trips.drop('zip_code', 1, inplace=True)
```

```
trips.rename(columns={'start_date'      : 's_date' ,
                      'end_date'        : 'e_date' ,
                      'start_station_name': 'ss_name',
                      'start_station_id' : 'ss_id' ,
                      'end_station_name'  : 'es_name',
                      'end_station_id'    : 'es_id' ,
                      'subscription_type' : 'subs'
                      }, inplace=True)
```

```
#Se cambia el formato de las fechas y tiempos de inicio y fin de cada viaje
trips['s_date'] = pd.to_datetime(trips['s_date'], format='%m/%d/%Y %H:%M')
trips['e_date'] = pd.to_datetime(trips['e_date'], format='%m/%d/%Y %H:%M')
```

```
#Se filtra lo mencionado anteriormente,
# las duraciones menores o iguales a 3 minutos con la misma estacion de salida y llegada
trips = trips[-((trips['duration'] <= 180) & (trips['ss_id'] == trips['es_id']))]
# los viajes de mas de 12 horas (12 * 3600 = 43200 segundos) y los de entre 11 y 12hs
# a la mañana
trips['start_hour'] = trips['s_date'].map(lambda x: x.hour)
trips = trips[-((trips['duration'] > 43200) | ((trips['duration'] > 39600) & ((trips['start_hour'] < 11 & trips['start_hour'] > 12)))))

#Se ordenan los viajes por id
trips = trips.sort_values(by='id')
```

6.2 3.1) Usuarios del Servicio

Cómo es la población que utiliza el servicio?

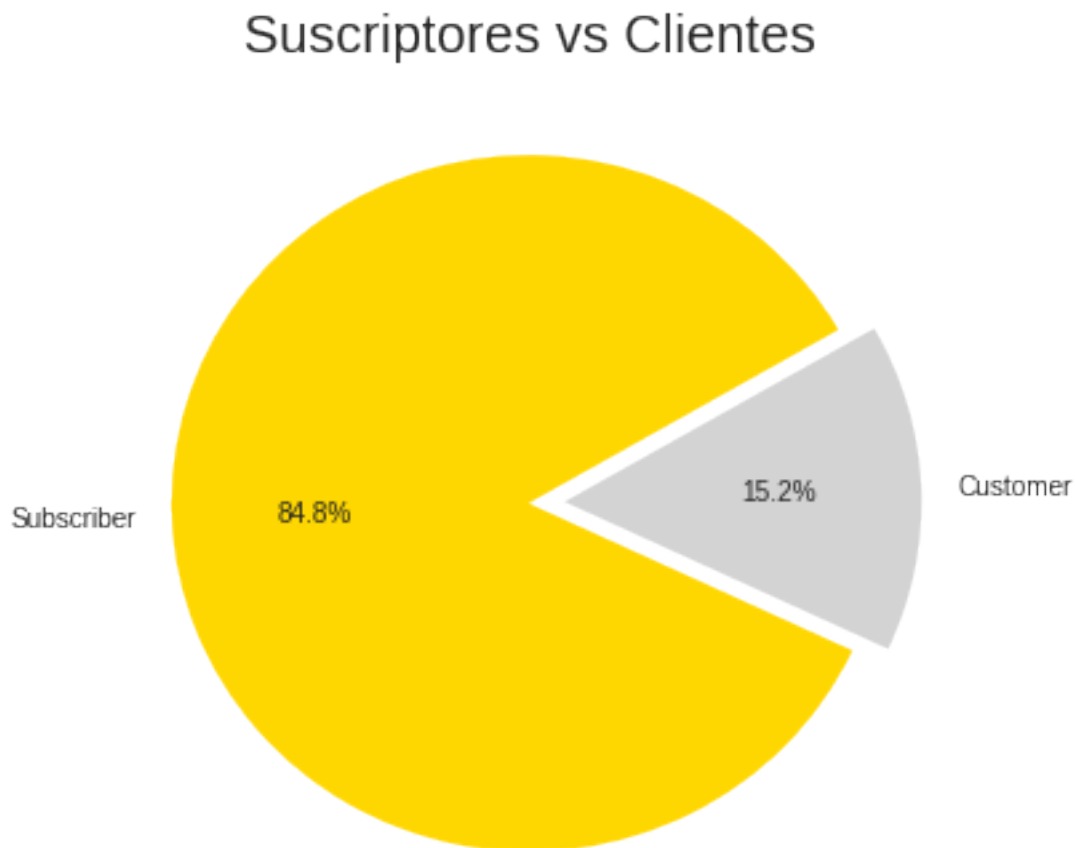
```
In [126]: #Se separan los tipos de suscripción de cada viaje y se juntan los del mismo tipo
subs = trips.groupby('subs').count()[['id']]
lbls = subs.index.values
```

```

vals = subs.values

plt.figure(figsize=(6, 6));
plt.title('Suscriptores vs Clientes', fontsize=20);
plt.pie(vals, explode=(0.1, 0), labels=lbls, colors=['lightgrey', 'gold'], autopct='%1
plt.savefig('../img/usuarios_del_servicio.png');
plt.show();

```



Aquí se ve que la mayoría de los usuarios, están suscriptos al servicio.

Esto puede indicar o refuerza la idea de que los usuarios del servicio, lo utilizan por necesidad y les es conveniente utilizar las bicicletas de este modo.

6.3 3.2) Estaciones más utilizadas

6.3.1 3.2.1) Estaciones más utilizadas por cada año

Vemos que estaciones presenciaron mas trafico de bicicletas por cada año.

```

In [127]: #Se separan los viajes segun el año
trips_2013 = trips['2014' > trips['s_date']]

```

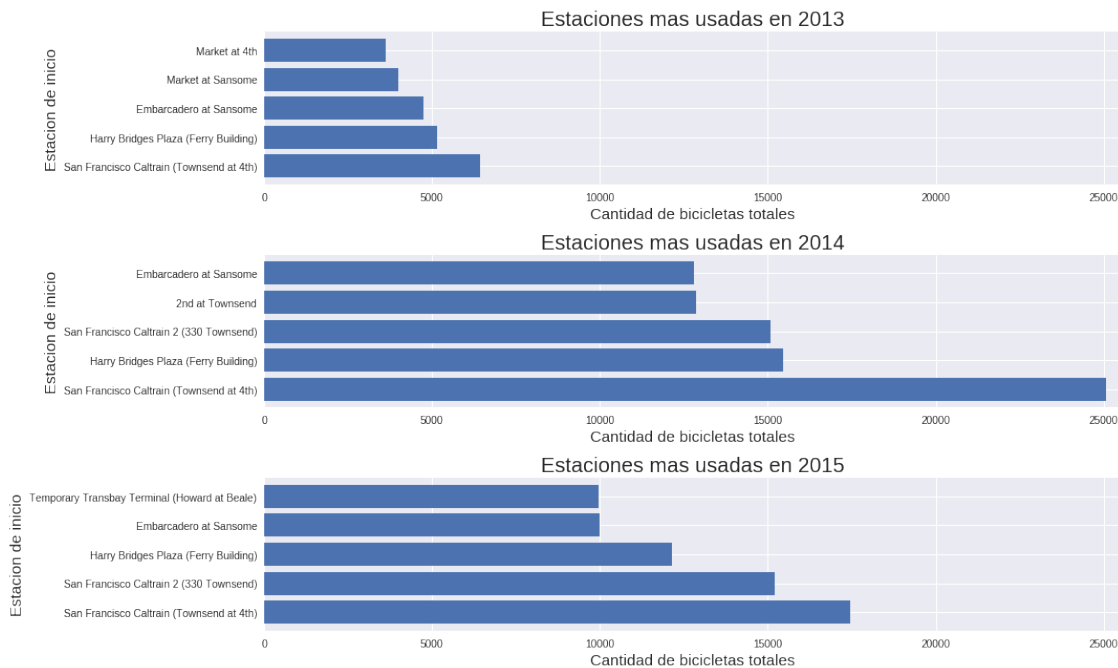
```
trips_2014 = trips[('2014' < trips['s_date']) & (trips['s_date'] < '2015')]
trips_2015 = trips['2015' < trips['s_date']]

viaje_anio = [trips_2013, trips_2014, trips_2015]
anios = ['2013', '2014', '2015']

plt.figure(figsize=(15, 9));

for i in range(len(viaje_anio)):
    #Se calcula la suma de bicicletas que partieron de las estaciones usadas en cada a
    viaje_anio[i] = viaje_anio[i].groupby('ss_name')['id'].count().sort_values(ascendi

    ax = plt.subplot(311 + i);
    ax.set_xlim([0, 25500]);
    plt.title('Estaciones mas usadas en ' + anios[i], fontsize=20);
    plt.xlabel('Cantidad de bicicletas totales', fontsize=15);
    plt.ylabel('Estacion de inicio', fontsize=15);
    ax.barh([j for j in range(5)], [valor for valor in viaje_anio[i][:5].values],
            tick_label=[viaje for viaje in viaje_anio[i][:5].index.values]);
    plt.savefig('../img/estaciones_mas_usadas_en_' + anios[i] + '.png');
plt.tight_layout();
```



La estación 'San Francisco Caltrain (Townsend at 4th)' es en definitiva una de las mas utilizadas por los usuarios.

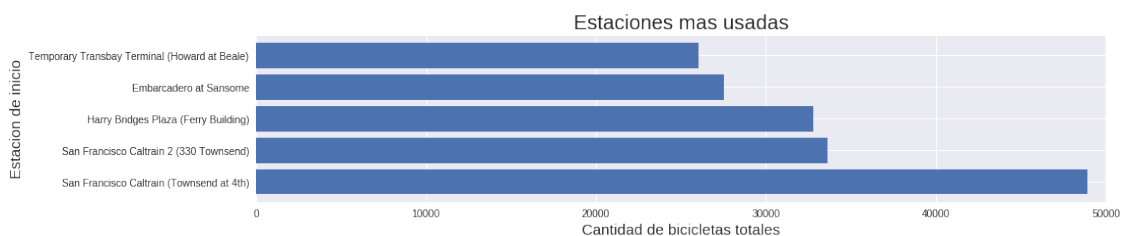
Una observación importante es que del año 2013 estan registrados los viajes entre Agosto y Diciembre, y del año 2015 de Enero a Agosto, por ese motivo se registran menos cantidad de viajes en cada estacion.

Sin embargo, para los 8 meses registrados en 2015, la utilizacion del servicio, si bien es menor a la del año 2014, es parecida. Esto indicaría que hubo un aumento en la cantidad de viajes entre esos años.

6.3.2 3.2.2) Estaciones más utilizadas en todos los años

```
In [128]: trips_13_15 = trips.groupby('ss_name')['id'].count().sort_values(ascending=False)
```

```
plt.figure(figsize=(15, 3));
ax = plt.subplot();
ax.set_xlim([0, 50000]);
plt.title('Estaciones mas usadas', fontsize=20);
plt.xlabel('Cantidad de bicicletas totales', fontsize=15);
plt.ylabel('Estacion de inicio', fontsize=15);
ax.barh([j for j in range(5)],
         [c for c in trips_13_15[:5].values],
         tick_label=[c for c in trips_13_15[:5].index.values]);
plt.savefig('../img/estaciones_mas_usadas.png');
```



6.4 3.3) Viajes por Estación y Ciudad

```
In [129]: from math import pi as PI
```

```
def distancia_grados(dist):
    return (180 * dist) / (6371 * PI)

def distancia_km(angulo):
    return angulo * 6371 * PI / 180

def mean(a, b):
    return (a + b) / 2

def mmap(value, min_in, max_in, min_out, max_out):
    m = (max_out - min_out) / float(max_in - min_in)
    b = min_out - m * min_in
    return m * value + b
```

```
In [130]: #Se calcula la cantidad de bicicletas por cada estacion
bicis_por_estacion = trips.groupby('ss_id').count()[['ss_name']].reset_index()
```

```
bicis_por_estacion.rename(columns={'ss_id':'id', 'ss_name':'count'}, inplace=True)

#Se agrega el nombre de la estacion, la ciudad en donde queda y sus coordenadas
estaciones = pd.merge(stations.drop(['date', 'dock_count'], 1), bicis_por_estacion, on
```

6.4.1 3.3.1) Frecuencia de uso de Estaciones por ciudad

Lo que se quiere ver ahora es, como se distribuyen los viajes realizados entre las estaciones de cada ciudad. (se utiliza la cantidad de bicicletas que parten de una estacion como medida de frecuencia de uso de una estacion)

```
In [131]: #Se separan las estaciones por ciudad
sf = estaciones[estaciones['city'] == 'San Francisco'].reset_index(drop=True)
sj = estaciones[estaciones['city'] == 'San Jose' ].reset_index(drop=True)
mv = estaciones[estaciones['city'] == 'Mountain View'].reset_index(drop=True)
pa = estaciones[estaciones['city'] == 'Palo Alto' ].reset_index(drop=True)
rc = estaciones[estaciones['city'] == 'Redwood City' ].reset_index(drop=True)

ciudades = [sf, sj, mv, pa, rc, estaciones]
nombres = ['San Francisco', 'San Jose', 'Mountain View', 'Palo Alto', 'Redwood City',
#La cantidad de estaciones se va a utilizar para escalar a los tamaños de las estaciones
ciudad_cant_estaciones = []
ciudad_bicis = []

#Listas con latitudes y longitudes de cada ciudad
lats = []
lons = []
#Lista con distancias para visualizar en mapas. Las distancias indican en cierto grado,
offsets = []

#Para cada ciudad, vemos su cantidad de estaciones y la cantidad total de bicicletas
for ciudad in ciudades:
    ciudad_cant_estaciones.append(ciudad.shape[0])
    ciudad_bicis.append(ciudad['count'].sum())
    #Para mostrar a todas las estaciones, tomamos la media entre la coordenada maxima
    lat_max, lat_min = ciudad['lat'].max(), ciudad['lat'].min()
    lon_max, lon_min = ciudad['lon'].max(), ciudad['lon'].min()
    lats.append(mean(lat_max, lat_min))
    lons.append(mean(lon_max, lon_min))
    offsets.append(max((lat_max - lat_min) * 0.6, (lon_max - lon_min) * 0.6))

In [132]: plt.figure(figsize=(20, 20))

for i in range(len(ciudades)):
    plt.subplot(331 + i);
    plt.title('Estaciones en ' + nombres[i], fontsize=20);
    mapa=Basemap(projection='merc', resolution='i', epsg=4326,
        llcrnrlat=lats[i] - offsets[i], llcrnrlon=lons[i] - offsets[i],
```

```

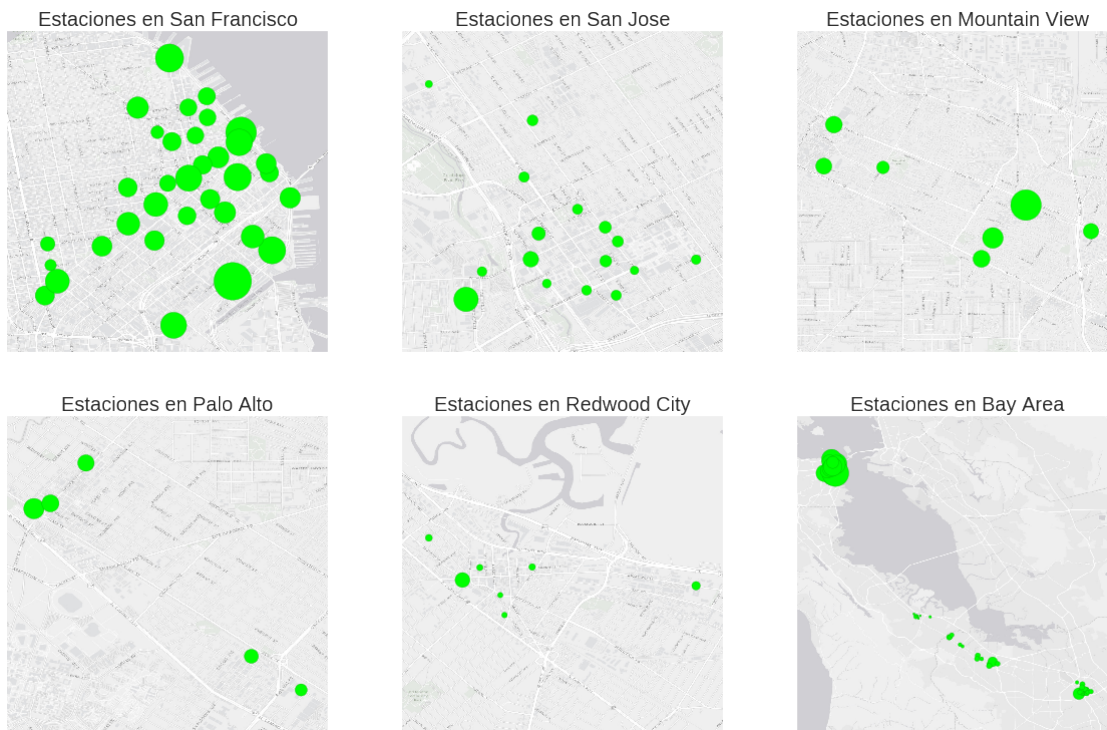
urcrnrlat=lats[i] + offsets[i], urcrnrlon=lons[i] + offsets[i])

#Aqui se escalan los tamaños de las estaciones dependiendo de la ciudad que se mue
mapa.scatter(ciudades[i]['lon'].values, ciudades[i]['lat'].values, marker='o', c='e
s=ciudades[i]['count'].apply(lambda x:x/ciudad_cant_estaciones[i]), e

mapa.arcgisimage(service='Canvas/World_Light_Gray_Base', xpixels=1000, ypixels=100
plt.savefig('../img/' + nombres[i].lower().replace(' ','_') + '_estaciones_uso.png

/home/mk/venv/lib/python3.6/site-packages/mpl_toolkits/basemap/__init__.py:3222: MatplotlibDepre
b = ax.ishold()
/home/mk/venv/lib/python3.6/site-packages/mpl_toolkits/basemap/__init__.py:3231: MatplotlibDepre
See the API Changes document (http://matplotlib.org/api/api\_changes.html)
for more details.
ax.hold(b)
/home/mk/venv/lib/python3.6/site-packages/mpl_toolkits/basemap/__init__.py:3296: MatplotlibDepre
b = ax.ishold()
/home/mk/venv/lib/python3.6/site-packages/mpl_toolkits/basemap/__init__.py:3305: MatplotlibDepre
See the API Changes document (http://matplotlib.org/api/api\_changes.html)
for more details.
ax.hold(b)

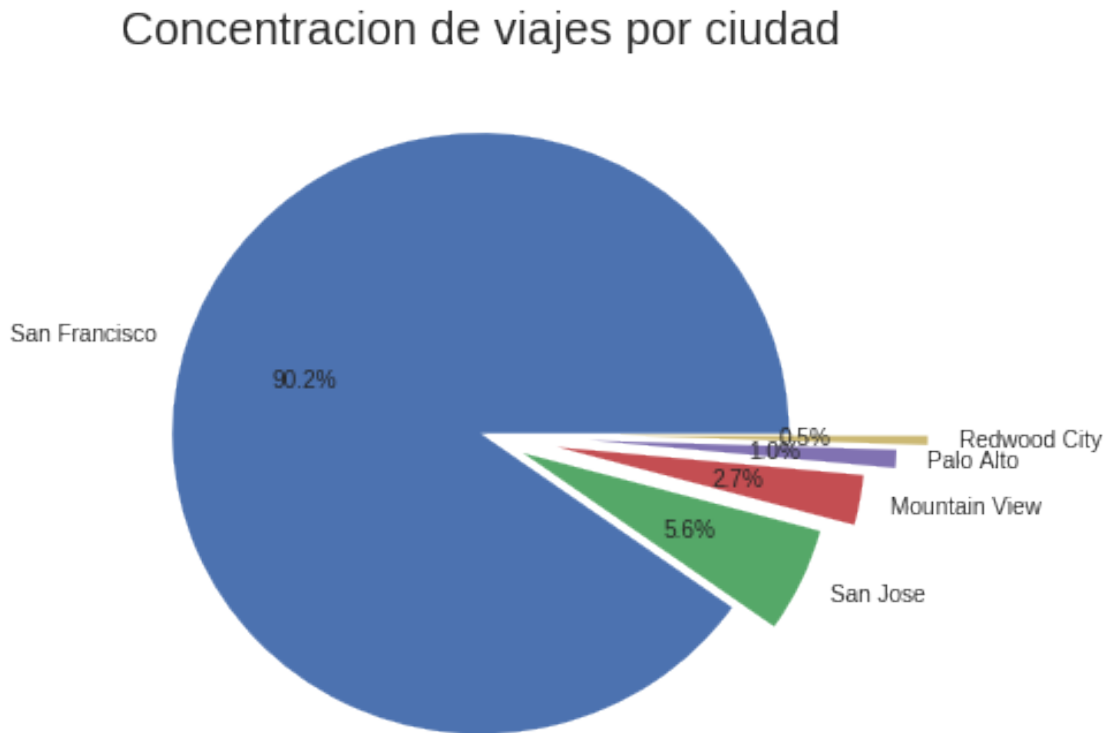
```



6.4.2 3.3.2) Concentración de viajes por ciudad

Qué ciudades son concentran la mayor parte de los viajes?

```
In [133]: plt.figure(figsize=(6, 6))
plt.title('Concentracion de viajes por ciudad', fontsize=20)
plt.pie(ciudad_bicis[:-1:], labels=nombres[:-1:], autopct='%1.1f%%', explode=(0, .15),
plt.savefig('../img/concentracion_viajes_por_ciudad.png')
plt.show()
```



Gran parte de los viajes parten de estaciones ubicadas en la ciudad de San Francisco. Luego le siguen San Jose, Mountain View, Palo Alto y por último Redwood City.

6.5 3.4) Trayectos más frecuentes

6.5.1 3.4.1) Trayectos por ciudad

Lo que se quiere lograr es, ver cuales son los viajes/trayectos entre estaciones más utilizados, por cada ciudad.

```
In [134]: #Se calculan las frecuencias de cada trayecto
trayectos_frec = trips[['id', 'ss_name', 'es_name']].groupby(['ss_name', 'es_name'], a

trayectos_frec.rename(columns={'id': 'count'}, inplace=True)
```

```

In [135]: #Dataframe con los nombres de todas las estaciones y sus coordenadas
ss_location = stations[['name', 'lat', 'lon', 'city']]
ss_location.rename(columns={'name':'ss_name', 'lat':'s_lat', 'lon':'s_lon', 'city':'s_

es_location = stations[['name', 'lat', 'lon', 'city']]
es_location.rename(columns={'name':'es_name', 'lat':'e_lat', 'lon':'e_lon', 'city':'e_

#Estos dataframes seran combinados con los de trayectos por la columna que tengan en
#Se quiere obtener un dataframe con todos los trayectos, esto es: las estaciones inici
# de las estaciones y la frecuencia del trayecto

/home/mk/venv/lib/python3.6/site-packages/pandas/core/frame.py:2834: SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame

See the caveats in the documentation: http://pandas.pydata.org/pandas-docs/stable/indexing.html#
**kwargs)

```

En el dataframe que tenemos hasta ahora, hay trayectos que estan 'repetidos', es decir que el trayecto tiene como punto inicial y final, las mismas estaciones.

```

In [136]: def reducir_trayectos(trayectos_frec):
    l = []
    #Lista con indice del dataframe del trayecto eliminado
    for i in range(len(trayectos_frec)):
        #Por cada trayecto
        if i not in l:
            #Si esta en el dataframe
            prim = trayectos_frec.loc[i]
            #Se obtiene el trayecto actual
            for j in range(i, len(trayectos_frec)):
                #Buscamos en lo que resta del dataframe
                if j not in l:
                    #Si el trayecto que queremos ver, esta en el dataframe
                    seg = trayectos_frec.loc[j]
                    #Se guarda
                    if prim['ss_name'] == seg['es_name'] and prim['es_name'] == seg['s_
                    #Si coinciden los trayectos
                    trayectos_frec.loc[i]['count'] += trayectos_frec.loc[j]['count']
                    #Se suman las frecuencias
                    trayectos_frec.drop(j, inplace=True)
                    #Se quita al segundo trayecto del dataframe
                    l.append(j)
                    #Se coloca en la lista, para que no halla errores
            return trayectos_frec

#El trayecto A -> B es el mismo que B -> A
#La funcion junta las repeticiones de estos trayectos

```

```
tf = reducir_trayectos(trayectos_frec)
```

```
#Se añade al dataframe, las ubicaciones de cada estacion
```

```
tf = pd.merge(tf, ss_location, on='ss_name')
```

```
tf = pd.merge(tf, es_location, on='es_name')
```

/home/mk/venv/lib/python3.6/site-packages/ipykernel/__main__.py:18: SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame

See the caveats in the documentation: <http://pandas.pydata.org/pandas-docs/stable/indexing.html#>

```
In [137]: #Este diccionario se va a contener, por ciudad:
```

```
# ù lista de tuplas ([lons], [lats], color) que tienen las coord de estaciones de un
```

```
# ù lista de los trayectos de la ciudad
```

```
viajes = {'San Francisco': [], [],  
          'San Jose'      : [], [],  
          'Redwood City' : [], [],  
          'Mountain View': [], [],  
          'Palo Alto'    : [], []  
        }
```

```
for ciudad in viajes:
```

```
#Se filtran los trayectos por ciudad
```

```
#Aquí se quitan los trayectos con estaciones en distintas ciudades
```

```
viajes[ciudad][1] = tf[(tf['s_city'] == ciudad) & \  
                      (tf['e_city'] == ciudad)].drop(['s_city', 'e_city'], 1).res
```

```
#Se calcula la cantidad maxima de viajes por cada trayecto en cada ciudad
```

```
#Esto es para luego asignar colores a trayectos segun su frecuencia
```

```
max_val = viajes[ciudad][1]['count'].max()
```

```
for indice in range(len(viajes[ciudad][1])):
```

```
    trayecto = viajes[ciudad][1].loc[indice]
```

```
#Mapeo de la frecuencia del trayecto al intervalo [0;255]
```

```
    color = abs(int(mmap(trayecto['count'], 0, max_val, 255, 0)))
```

```
#color = abs(color)
```

```
    viajes[ciudad][0].append((trayecto['s_lon'], trayecto['e_lon'],  
                             [trayecto['s_lat'], trayecto['e_lat']], color))
```

```
#Se ordenan los viajes mas frecuentes al final, así son mostrados por encima de los
```

```
viajes[ciudad][0].sort(key=lambda x: -x[2])
```

```
In [138]: #Ahora visualizamos los trayectos por ciudad
```

```
plt.figure(figsize=(20, 12))
```

```
for i in range(len(ciudades)-1):
```

```
    plt.subplot(231 + i);
```

```
    plt.title('Viajes en ' + nombres[i], fontsize=20);
```

```

mapa = Basemap(projection='merc', resolution='i', epsg=4326,
                llcrnrlat=lats[i] - offsets[i], llcrnrlon=lons[i] - offsets[i],
                urcrnrlat=lats[i] + offsets[i], urcrnrlon=lons[i] + offsets[i])

for trayecto in viajes[nombres[i]][0]:
    #Convertimos el color a un formato RGBA valido
    color = trayecto[2]
    color = '%02Xff%02X' % (color, color)
    #Ploteamos el trayecto
    mapa.plot(trayecto[0], trayecto[1], 'o-', color=color)

mapa.arcgisimage(service='Canvas/World_Light_Gray_Base', xpixels=1000, ypixels=1000)
plt.savefig('../img/' + nombres[i].lower().replace(' ', '_') + '_frecuencia_trayecto')

plt.savefig('../img/frecuencia_trayectos.png');

```

```

/home/mk/venv/lib/python3.6/site-packages/mpl_toolkits/basemap/__init__.py:3260: MatplotlibDeprecationWarning:
  b = ax.ishold()
/home/mk/venv/lib/python3.6/site-packages/mpl_toolkits/basemap/__init__.py:3269: MatplotlibDeprecationWarning:
  See the API Changes document (http://matplotlib.org/api/api\_changes.html)
  for more details.
  ax.hold(b)
/home/mk/venv/lib/python3.6/site-packages/mpl_toolkits/basemap/__init__.py:3296: MatplotlibDeprecationWarning:
  b = ax.ishold()
/home/mk/venv/lib/python3.6/site-packages/mpl_toolkits/basemap/__init__.py:3305: MatplotlibDeprecationWarning:
  See the API Changes document (http://matplotlib.org/api/api\_changes.html)
  for more details.
  ax.hold(b)

```



6.5.2 3.4.2) Trayectos de toda la bahia

```
In [139]: plt.figure(figsize=(15, 15))
plt.title('Viajes en toda la Bahia', fontsize=20)

offset = distancia_grados(35)
mapa = Basemap(projection='merc', resolution='i', epsg=4326,
                llcrnrlat=lats[5] - offset, llcrnrlon=lons[5] - offset,
                urcrnrlat=lats[5] + offset, urcrnrlon=lons[5] + offset )

max_val = tf['count'].max()
tf = tf.sort_values(by='count').reset_index(drop=True)

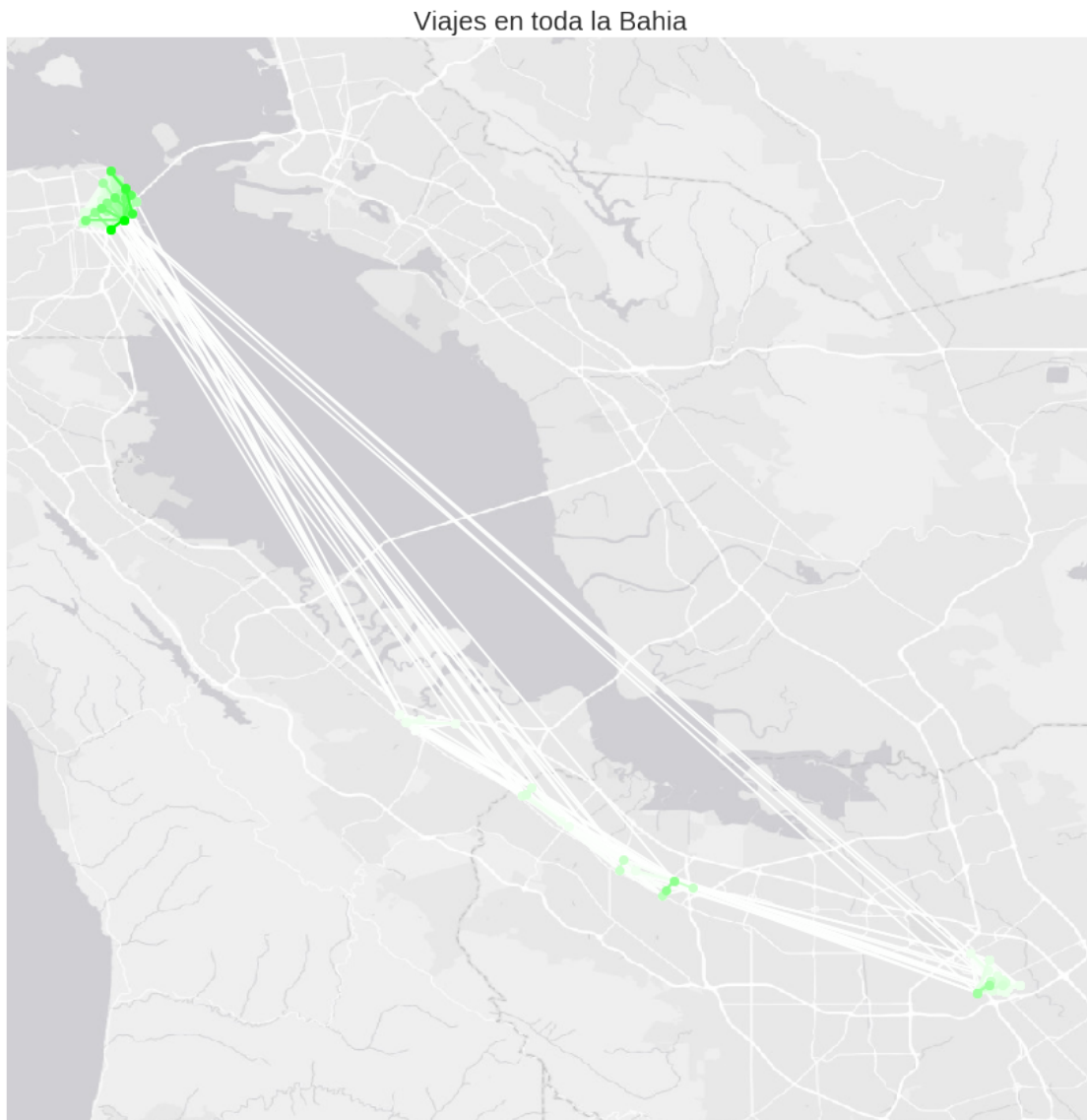
for indice in range(tf.shape[0]):
    trayecto = tf.loc[indice]
    color = abs(int(mmap(trayecto['count'], 0, max_val, 255, 0)))
    mapa.plot([trayecto['s_lon'], trayecto['e_lon']],
              [trayecto['s_lat'], trayecto['e_lat']],
              'o-', color='#02Xff%02X' % (color, color));

mapa.arcgisimage(service='Canvas/World_Light_Gray_Base', xpixels=1000, ypixels=1000);
plt.savefig('../img/bay_area_frecuencia_trayectos.png');

/home/mk/venv/lib/python3.6/site-packages/mpl_toolkits/basemap/___init___py:3260: MatplotlibDeprecationWarning:
  b = ax.ishold()
```



```
/home/mk/venv/lib/python3.6/site-packages/mpl_toolkits/basemap/__init__.py:3269: MatplotlibDeprecationWarning:
  See the API Changes document (http://matplotlib.org/api/api\_changes.html)
  for more details.
  ax.hold(b)
/home/mk/venv/lib/python3.6/site-packages/mpl_toolkits/basemap/__init__.py:3296: MatplotlibDeprecationWarning:
  See the API Changes document (http://matplotlib.org/api/api\_changes.html)
  for more details.
  b = ax.ishold()
/home/mk/venv/lib/python3.6/site-packages/mpl_toolkits/basemap/__init__.py:3305: MatplotlibDeprecationWarning:
  See the API Changes document (http://matplotlib.org/api/api\_changes.html)
  for more details.
  ax.hold(b)
```



6.6 3.5) Cuál es la cantidad total de bicicletas que se utilizó?

6.6.1 3.5.1) Cantidad de bicicletas en total

Cuál es la cantidad de bicicletas disponibles?

```
In [140]: #Se cuentan los valores para cada id de bicicleta diferente
trips['bike_id'].value_counts().count()
```

Out[140]: 700

El total de bicicletas diferentes es de 700. Hay que tener en cuenta que esta cantidad de bicicletas que se registraron en todos los datos, eso incluye los diferentes periodos de cada año, que no es la cantidad de bicicletas disponibles en todo momento.

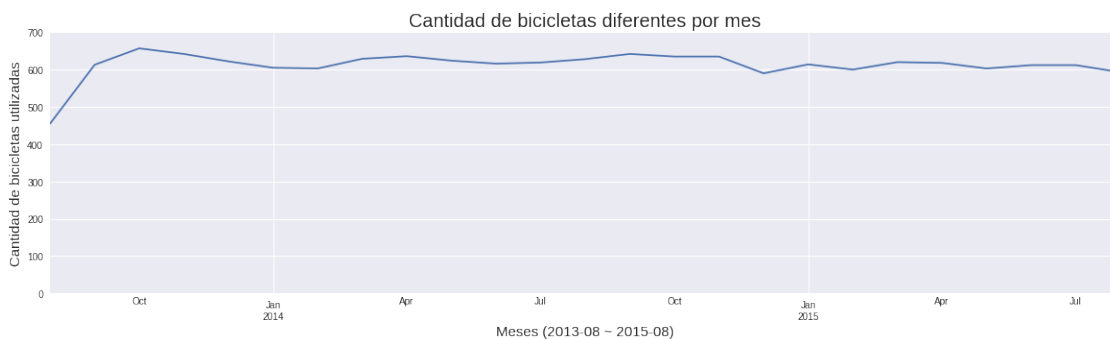
Entonces lo que indica el número, es la cantidad de bicicletas distintas que estuvieron en servicio.

A continuacion vemos la cantidad de bicicletas diferentes que estaban disponibles por mes.

6.6.2 3.5.2) Cantidad de bicicletas disponibles por mes

```
In [141]: trips_fechas = trips[['s_date', 'bike_id', 'id']].reset_index(drop=True)
trips_fechas['s_date'] = pd.to_datetime(trips_fechas['s_date'].dt.strftime('%Y-%m'))

In [142]: trips_fechas.groupby(['s_date', 'bike_id'], as_index=False).count()\
          .groupby('s_date')['bike_id'].count().plot(figsize=(20, 5), ylim=(0,
plt.title('Cantidad de bicicletas diferentes por mes', fontsize=20);
plt.xlabel('Meses (2013-08 ~ 2015-08)', fontsize=15);
plt.ylabel('Cantidad de bicicletas utilizadas', fontsize=15);
```



En el grafico se ve que la cantidad de bicicletas utilizadas es menor a 700, es decir no siempre se utilizan todas las bicicletas.

7 4) Análisis final: análisis a gran escala

7.1 ¿El servicio creció con el paso del tiempo?

A continuación se analizará si con el correr del tiempo el servicio aumentó en cantidad de viajes y de suscriptores.

```
In [143]: viajes_anios = trips[['s_date', 'subs']]
         _2013 = viajes_anios[(viajes_anios['s_date'].dt.year) == 2013]
         _2013 = _2013.sort_values(by='s_date')
         _2014 = viajes_anios[(viajes_anios['s_date'].dt.year) == 2014]
         _2014 = _2014.sort_values(by='s_date')
         _2015 = viajes_anios[(viajes_anios['s_date'].dt.year) == 2015]
         _2015 = _2015.sort_values(by='s_date')
```

```
In [144]: _2013.head()
```

```
Out[144]:
```

	s_date	subs
541	2013-08-29 09:24:00	Subscriber
570	2013-08-29 09:24:00	Subscriber
565	2013-08-29 09:24:00	Subscriber
566	2013-08-29 09:25:00	Subscriber
719	2013-08-29 09:31:00	Customer

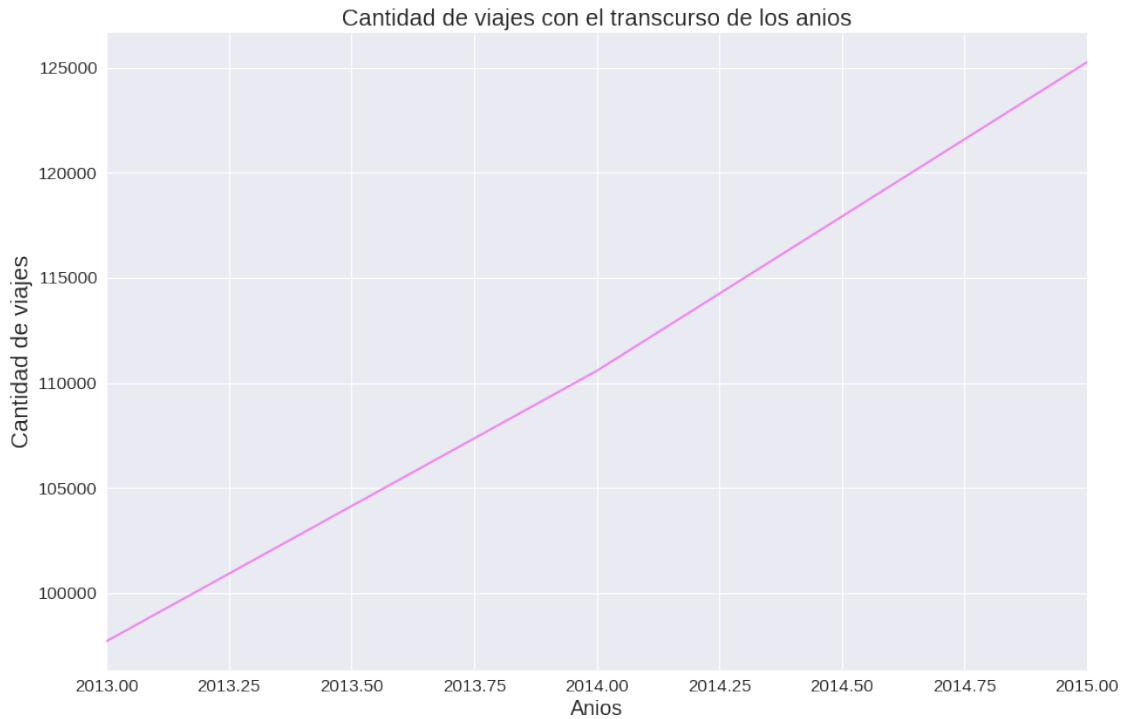
Ya que el año 2013 sólo tiene datos a partir de septiembre y 2015 hasta agosto, se tomarán los últimos 4 meses de cada año para poder realizar la comparación. Cabe mencionar que como se estudió anteriormente, sacando invierno (de enero a marzo), los demás meses presentan una cantidad bastante parecida de viajes, por lo que este análisis no se ve afectado por cuales meses del año se han elegido.

```
In [145]: _2013 = _2013[_2013['s_date'].dt.month >= 9] # elimino los pocos datos de agosto que h
         _2014 = _2014[_2014['s_date'].dt.month >= 9]
         _2015 = _2015[_2015['s_date'].dt.month >= 5]
```

```
In [146]: viajes = [_2013.s_date.count(), _2014.s_date.count(), _2015.s_date.count()]
         anios = [2013,2014,2015]
```

```
In [147]: d = {'cantidad_viajes': viajes, 'anio': anios}
         viajes_por_anio = pd.DataFrame(data=d)
         viajes_por_anio = viajes_por_anio.groupby('anio').aggregate(sum)
```

```
In [148]: viajes_por_anio.plot.line(figsize=(15,10), color='violet', fontsize=15);
         plt.xlabel('Anios', fontsize=18)
         plt.ylabel('Cantidad de viajes', fontsize=20)
         plt.title('Cantidad de viajes con el transcurso de los anios', fontsize=20)
         plt.grid(True)
         plt.legend('');
         plt.show()
```



Como se puede apreciar, con el transcurso de los años los viajes aumentan en forma lineal, por lo que se puede concluir que con el correr del tiempo el servicio cada vez se utiliza más.

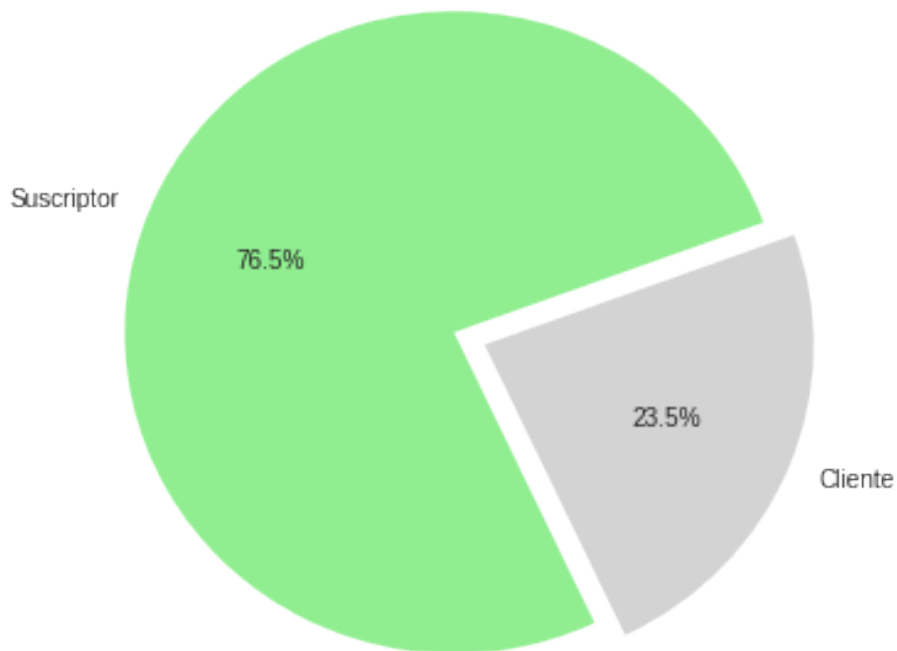
7.2 ¿Cómo cambia el porcentaje de viaje de suscriptos con el correr de los años?

```
In [149]: suscripciones_2013 = _2013.subs.value_counts()
          suscripciones_2014 = _2014.subs.value_counts()
          suscripciones_2015 = _2015.subs.value_counts()
```

```
In [150]: sizes = [suscripciones_2013.Subscriber, suscripciones_2013.Customer]
          nombres = ['Suscriptor', 'Cliente']
```

```
plt.figure(figsize=(6, 6))
plt.title('Tipos de suscripciones en los viajes durante 2013', fontsize=20)
plt.pie(sizes, labels=nombres, autopct='%1.1f%%', startangle=20, colors=['lightgreen', 'lightcoral'],
plt.show()
```

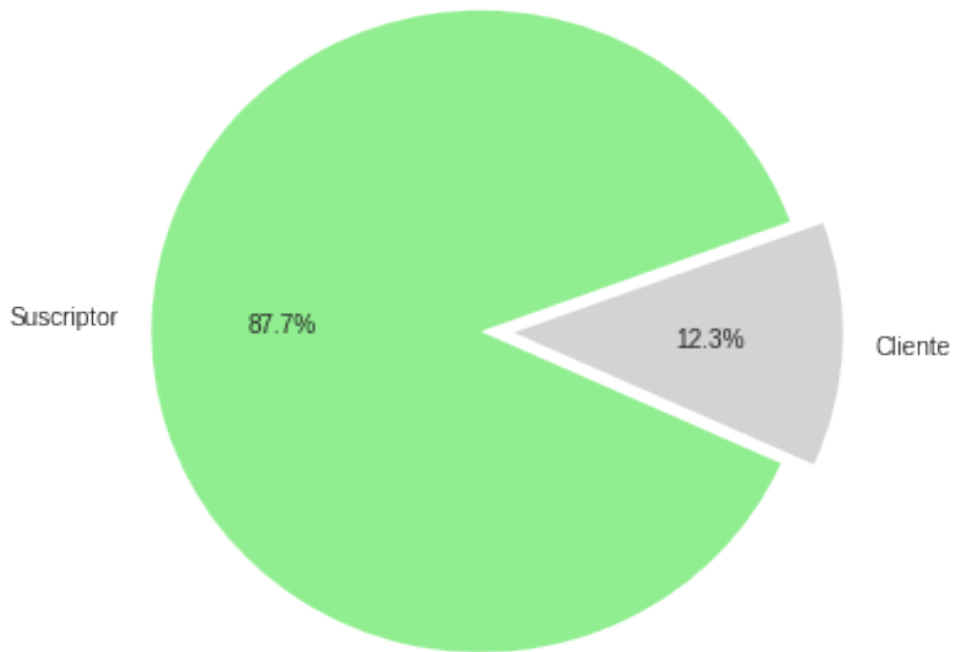
Tipos de suscripciones en los viajes durante 2013



```
In [151]: sizes = [suscripciones_2014.Subscriber, suscripciones_2014.Customer]
          nombres = ['Suscriptor', 'Cliente']

          plt.figure(figsize=(6, 6))
          plt.title('Tipos de suscripciones en los viajes durante 2014', fontsize=20)
          plt.pie(sizes, labels=nombres, autopct='%1.1f%%', startangle=20, colors=['lightgreen',
          plt.show()
```

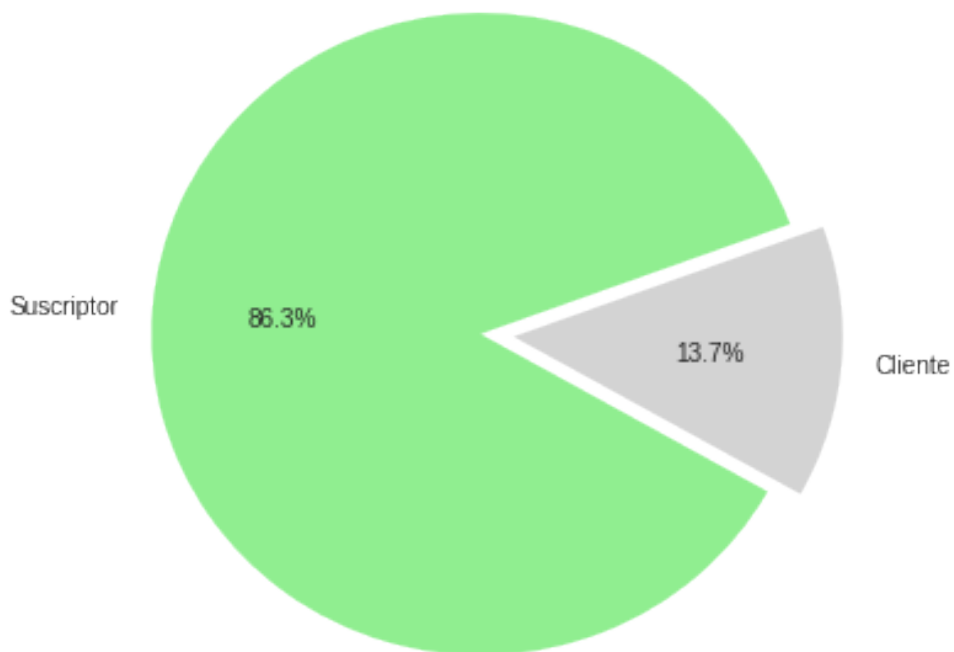
Tipos de suscripciones en los viajes durante 2014



```
In [152]: sizes = [suscripciones_2015.Subscriber, suscripciones_2015.Customer]
          nombres = ['Suscriptor', 'Cliente']

          plt.figure(figsize=(6, 6))
          plt.title('Tipos de suscripciones en los viajes durante 2015', fontsize=20)
          plt.pie(sizes, labels=nombres, autopct='%1.1f%%', startangle=20, colors=['lightgreen',
          plt.show()
```

Tipos de suscripciones en los viajes durante 2015



Se observa que del 2013 al 2014 hubo un importante aumento en la cantidad de viajes realizados por suscriptores (esto también nos podría decir que los mismos también han aumentado), y luego del 2014 al 2015 esa relación se mantuvo igual. Igualmente como vimos anteriormente del 2014 al 2015 hubo un aumento de viajes, y, como el porcentaje de viajes hechos por suscriptores se mantuvo, implica que en cantidad estos viajes aumentaron y por lo tanto también podríamos decir que los suscriptores lo han hecho.