Project: No-show Appointments Data Analysis

Table of Contents

Introduction

Having a biomedical background and working for a pharmaceutical company, I am always very interested in the healthcare, so here I choose this dataset, no-show appointments (Reference 1) for my data analysis project.

In this dataset, there are 110527 observations (a medical appointment per record) and 14 variables including 1 dependent variable, a patient shows up to his/her appointment or not, and 13 independent variables with patients' characteristics. The following table describes the variable names with their definition. For the variable, Scholarship, means whether a patient enrolls in Brasilian welfare program or not. For more details about the program, you can learn more in the reference 2.

Variable Name	Definition
PatientId	Identification of a patient
AppointmentID	Identification of each appointment
Gender	M = Male; F = Female
ScheduledDay	The Date of patient set up their appointment.
AppointmentDay	The Date of the appointment
Age	The age of the patient
Neighbourhood	The location of the hospital
Scholarship	Indicates whether or not the patient was enrolled in Brasilian welfare program (Reference 2)
Hipertension	0: non-hypertension 1: hypertension
Diabetes	0: non-diabetes 1: diabetes
Alcoholism	0: non-alcoholic 1: alcoholic
Handcap	0: False 1: True
SMS_received	0: did not send any message to the patient 1: 1 or more messages sent to the patient
No-show	'Yes': the patient did not show up to their appointment ‘No’: the patient showed up.

Based on the above table, the question I am interested in is

What are the characteristics that a patient who shows up for his/her scheduled appointment demonstrate?

I will perform a exploratory data analysis to answer this question. Before loading the dataset, I install the following libraries I will use in the project.

In [275]:
```
# Install the libraries for this analysis
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
from scipy import stats

%matplotlib inline
```

Data Wrangling

General Properties

> I load the No-show dataset into the Jupyter notebook and take a look at the first 5 rows to get a sense of the values for each variable and then I check the numbers of rows, columns, missing values and data types.

In [276]:
```
# Load the dataset
df = pd.read_csv('noshowappointments-kagglev2-may-2016.csv')
```

In [277]:
```
# Take a look at the first 5 rows
df.head()
```

Out[277]:

	PatientId	AppointmentID	Gender	ScheduledDay	AppointmentDay	Age	Neighbourhood
0	2.987250e+13	5642903	F	2016-04-29T18:38:08Z	2016-04-29T00:00:00Z	62	JARDIM DA PENHA
1	5.589978e+14	5642503	M	2016-04-29T16:08:27Z	2016-04-29T00:00:00Z	56	JARDIM DA PENHA
2	4.262962e+12	5642549	F	2016-04-29T16:19:04Z	2016-04-29T00:00:00Z	62	MATA DA PRAIA
3	8.679512e+11	5642828	F	2016-04-29T17:29:31Z	2016-04-29T00:00:00Z	8	PONTAL DE CAMBURI
4	8.841186e+12	5642494	F	2016-04-29T16:07:23Z	2016-04-29T00:00:00Z	56	JARDIM DA PENHA

In [7]:
```
# Find the numbers of rows and columns
df.shape
```

Out[7]:
```
(110527, 14)
```

In [18]:
```
# Find data types and missing values in each column
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 110527 entries, 0 to 110526
Data columns (total 14 columns):
PatientId         110527 non-null float64
AppointmentID     110527 non-null int64
Gender            110527 non-null object
ScheduledDay      110527 non-null object
AppointmentDay    110527 non-null object
Age               110527 non-null int64
Neighbourhood     110527 non-null object
Scholarship       110527 non-null int64
Hipertension      110527 non-null int64
Diabetes          110527 non-null int64
Alcoholism        110527 non-null int64
Handcap           110527 non-null int64
SMS_received      110527 non-null int64
No-show           110527 non-null object
dtypes: float64(1), int64(8), object(5)
memory usage: 11.8+ MB
```

In [7]:
```
# Find missing values in each column
df.isnull().sum()
```

Out[7]:
```
PatientId         0
AppointmentID     0
Gender            0
ScheduledDay      0
AppointmentDay    0
Age               0
Neighbourhood     0
Scholarship       0
Hipertension      0
Diabetes          0
Alcoholism        0
Handcap           0
SMS_received      0
No-show           0
dtype: int64
```

In [6]:
```
# Find distributions for continuous variables, Age
df.describe()
```

Out[6]:

	PatientId	AppointmentID	Age	Scholarship	Hipertension	Diabete
count	1.105270e+05	1.105270e+05	110527.000000	110527.000000	110527.000000	110527.00000
mean	1.474963e+14	5.675305e+06	37.088874	0.098266	0.197246	0.07186
std	2.560949e+14	7.129575e+04	23.110205	0.297675	0.397921	0.25826
min	3.921784e+04	5.030230e+06	-1.000000	0.000000	0.000000	0.00000
25%	4.172614e+12	5.640286e+06	18.000000	0.000000	0.000000	0.00000
50%	3.173184e+13	5.680573e+06	37.000000	0.000000	0.000000	0.00000
75%	9.439172e+13	5.725524e+06	55.000000	0.000000	0.000000	0.00000
max	9.999816e+14	5.790484e+06	115.000000	1.000000	1.000000	1.00000

Luckily, there is no missing values in each column. The data type of ScheduledDay and AppointmentDay are all string. I will convert them to the dates in the next session. In the describe table, the minimum value of Age is -1 which does not make any senses. I will check how many rows with this value and see if I need to remove them from the dataset. Also, I found that the maximum value of Handcap is 4, but in the defintion, the values are True (1) or False (0). I am curious what other values that this variable has.

In [134]:
```
# Check the values for Handcap
df['Handcap'].value_counts()
```

Out[134]:
```
0    108286
1      2042
2       183
3        13
4         3
Name: Handcap, dtype: int64
```

There are 5 groups in this variable and I am not sure about what this means, so I will exclude this variable from this analysis.

Data Cleaning

> After checking the properties in the previous session, in this session, I would like to clean the dataset and prepare for exploratory data analysis. First, I convert ScheduledDay and AppointmentDay from string to time format, and label them into the weekdays. Second, the minimum of age is -1 which does not make any senses. I will find how many age is equal to -1 and if not much, I treat them as outliers (error) and delete them from the dataset.

In [196]:

```
# 1.1 Convert ScheduledDay from string to time format and label to weekd
ays
df['ScheduledDay'] = pd.to_datetime(df['ScheduledDay'])
df['Sched_Week'] = df['ScheduledDay'].apply(lambda x:x.weekday())
dmap = {0:'Mon',1:'Tue',2:'Wed',3:'Thu',4:'Fri',5:'Sat',6:'Sun'}
df['Sched_Weekf'] = df['Sched_Week'].map(dmap)
```

In [197]:

```
# 1.2 Convert AppointmentDay from strings to time format and label to we
ekdays
df['AppointmentDay'] = pd.to_datetime(df['AppointmentDay'])
df['App_Week'] = df['AppointmentDay'].apply(lambda x:x.weekday())
df['App_Weekf'] = df['App_Week'].map(dmap)
```

In [93]:
```
# Confirm the conversion
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 110527 entries, 0 to 110526
Data columns (total 18 columns):
PatientId         110527 non-null float64
AppointmentID     110527 non-null int64
Gender            110527 non-null object
ScheduledDay      110527 non-null datetime64[ns]
AppointmentDay    110527 non-null datetime64[ns]
Age               110527 non-null int64
Neighbourhood     110527 non-null object
Scholarship       110527 non-null int64
Hipertension      110527 non-null int64
Diabetes          110527 non-null int64
Alcoholism        110527 non-null int64
Handcap           110527 non-null int64
SMS_received      110527 non-null int64
No-show           110527 non-null object
Sched_Week        110527 non-null int64
Sched_Weekf       110527 non-null object
App_Week          110527 non-null int64
App_Weekf         110527 non-null object
dtypes: datetime64[ns](2), float64(1), int64(10), object(5)
memory usage: 15.2+ MB
```

In [97]:
```
# 2. Find Age = -1 and how many of it
df.query('Age == -1').Age.value_counts()
```

Out[97]:
```
-1    1
Name: Age, dtype: int64
```

In [198]:
```
# Remove Age = -1 from the dataset and check the change in the number of
 observations
df2 = df.query('Age >= 0')
df2.shape
```

Out[198]:
```
(110526, 18)
```

In [199]:
```
# 3. Remove Handcap
df2 = df2.drop(['Handcap'], axis=1)
```

In [200]:
```
df2.head()
```

Out[200]:

	PatientId	AppointmentID	Gender	ScheduledDay	AppointmentDay	Age	Neighbourhood
0	2.987250e+13	5642903	F	2016-04-29 18:38:08	2016-04-29	62	JARDIM DA PENHA
1	5.589978e+14	5642503	M	2016-04-29 16:08:27	2016-04-29	56	JARDIM DA PENHA
2	4.262962e+12	5642549	F	2016-04-29 16:19:04	2016-04-29	62	MATA DA PRAIA
3	8.679512e+11	5642828	F	2016-04-29 17:29:31	2016-04-29	8	PONTAL DE CAMBURI
4	8.841186e+12	5642494	F	2016-04-29 16:07:23	2016-04-29	56	JARDIM DA PENHA

> After finishing above 3 steps, the dataset is ready for exploratory data analysis.

Exploratory Data Analysis

> Here, I relist the question I asked in the introduction session.

What are the characteristics that a patient who shows up for his/her scheduled appointment demonstrate?

1. Analyze binomial variable by using 2 * 2 table method

I would like to take a look at binomial variables. There are 6 binomial variables in this dataset: Gender, Scholarship, Hipertension, Diabetes, Alcoholism, and SMS_received. I will perform the same method to these variables. First, I find the count and proportion in each group of a binomial variable and no-show, create a 2 *2 table, and calculate raletive risk and odds ratio. There is a article about interpreting the result in 2*2 groups (Reference 3).

Here, I am using Gender * No-Show as an example:

Gender/App	No-Show (Yes)	Show (No)	Tot 2
Female	a	b	a+b
Male	c	d	c+d
Tot 1	a+c	b+d	n

Definitions: Risk describes the probability of No show
P0 = probability of No-show (Yes) for males
P1 = probability of No-show (Yes) for females
P0 = c / (c + d)
P1 = a / (a + b)
Risk difference: RD = P1 – P0

Relative risk: RR = P1/P0
O0 = odds for males: O0= P0/(1 – P0)
O1 = odds for females: O1= P1/(1 – P1)
OR = odds ratio = O1 / O0 = (P1 / [1 – P1]) / P0 / [1 – P0]) = (a x d) / (b x c)
If (a + c)/n is “small,” RR and OR have similar values.

A relative risk of 1 (RR = 1) means that females and males have the same risk of No-show. If RR is greater than 1, this means that female have a higher risk than males.
If RR = 1.5, this means that the risk of females is 50% greater than that of males.
If RR = 2, this means that the risk is doubled.
If RR = 0.5, this means that females only have half the risk of males. This can also be referred to as a “protective factor.” It is important to bear in mind which groups are used as reference.
Odds ratio (OR) is used as a measure of association. The odds ratio is the quotient of the chances (odds) of a disease (cure) for persons with or without exposure (therapy). Here, it's no-show for females or males.

a. Gender by No-Show

> From the count, apparently, the number of females is way more than the number of males, so I calculate the No-show and show percentages in females and males separately.

In [103]:
```
# The count in each group of gender and no-show
df2.groupby('No-show')['Gender'].value_counts()
```

Out[103]:
```
No-show  Gender
No       F         57245
         M         30962
Yes      F         14594
         M          7725
Name: Gender, dtype: int64
```

In [104]:
```
# Create a 2*2 table to see the percentage in each group
# Note: denominators are calculated by each gender group
sex_app_tb = pd.crosstab(df2['Gender'], df2['No-show']).apply(lambda r:
r/r.sum(), axis=1)
sex_app_tb
```

Out[104]:

No-show Gender	No	Yes
F	0.796851	0.203149
M	0.800321	0.199679

In [105]:
```
# Calculate Relative Risk
RR = sex_app_tb['Yes']['F']/sex_app_tb['Yes']['M']
RR
```

Out[105]:
```
1.017373986806438
```

In [106]:
```
# Odds for males and females
sex_app_tb['Odds'] = sex_app_tb['Yes'] * (1-sex_app_tb['Yes'])
```

In [107]:
```
sex_app_tb
```

Out[107]:

No-show Gender	No	Yes	Odds
F	0.796851	0.203149	0.161879
M	0.800321	0.199679	0.159808

In [108]:
```
# Odds ratio
ORR = sex_app_tb['Odds']['F']/sex_app_tb['Odds']['M']
ORR
```

Out[108]:
```
1.0129638750184298
```

b. Scholarship by No-show

> Based on the count of scholarship, the number of patients who enroll in the program is less than the number of patients who do not. Then I calculate the percentage.

In [109]:
```
# The count in each group of Scholarship and no-show
df2.groupby('No-show')['Scholarship'].value_counts()
```

Out[109]:
```
No-show  Scholarship
No       0              79924
         1               8283
Yes      0              19741
         1               2578
Name: Scholarship, dtype: int64
```

In [110]:
```
# Create a 2*2 table to see the percentage in each group
# Note: denominators are calculated by each Scholarship group
sch_app_tb = pd.crosstab(df2['Scholarship'], df2['No-show']).apply(lambd
a r: r/r.sum(), axis=1)
sch_app_tb
```

Out[110]:

No-show	No	Yes
Scholarship		
0	0.801926	0.198074
1	0.762637	0.237363

In [113]:
```
# Calculate Relative Risk
sch_RR = sch_app_tb['Yes'][1]/sch_app_tb['Yes'][0]
sch_RR
```

Out[113]:
```
1.198358117046747
```

In [114]:
```
# Odds for enroll scholarship and non enroll scholarship
sch_app_tb['Odds'] = sch_app_tb['Yes'] * (1-sch_app_tb['Yes'])
sch_app_tb
```

Out[114]:

No-show	No	Yes	Odds
Scholarship			
0	0.801926	0.198074	0.158840
1	0.762637	0.237363	0.181022

In [115]:
```
# Odds ratio
sch_ORR = sch_app_tb['Odds'][1]/sch_app_tb['Odds'][0]
sch_ORR
```

Out[115]:
```
1.1396458924151784
```

c. Hipertension

> From the counts, the number of hypertension patients is less than the number of non-hypertension patients.

In [116]:
```
# The count in each group of Hipertension and no-show
df2.groupby('No-show')['Hipertension'].value_counts()
```

Out[116]:
```
No-show  Hipertension
No       0               70178
         1               18029
Yes      0               18547
         1                3772
Name: Hipertension, dtype: int64
```

In [117]:
```
# Create a 2*2 table to see the percentage in each group
# Note: denominators are calculated by each Hipertension group
hi_app_tb = pd.crosstab(df2['Hipertension'], df2['No-show']).apply(lambd
a r: r/r.sum(), axis=1)
hi_app_tb
```

Out[117]:

No-show	No	Yes
Hipertension		
0	0.790961	0.209039
1	0.826980	0.173020

In [118]:
```
# Calculate Relative Risk
hi_RR = hi_app_tb['Yes'][1]/hi_app_tb['Yes'][0]
hi_RR
```

Out[118]:
```
0.8276898037794616
```

In [119]:
```
# Odds for with hipertension and without hipertension
hi_app_tb['Odds'] = hi_app_tb['Yes'] * (1-hi_app_tb['Yes'])
hi_app_tb
```

Out[119]:

No-show	No	Yes	Odds
Hipertension			
0	0.790961	0.209039	0.165342
1	0.826980	0.173020	0.143084

In [120]:
```
# Calculate Odds ratio
hi_ORR = hi_app_tb['Odds'][1]/hi_app_tb['Odds'][0]
hi_ORR
```

Out[120]:
```
0.8653819847005272
```

d. Diabetes

> From the counts, the number of diabetes patients is less than the number of non-diabetes patients.

In [121]:
```
# The count in each group of Diabetes and no-show
df2.groupby('No-show')['Diabetes'].value_counts()
```

Out[121]:
```
No-show  Diabetes
No       0           81694
         1            6513
Yes      0           20889
         1            1430
Name: Diabetes, dtype: int64
```

In [122]:
```
# Create a 2*2 table to see the percentage in each group
# Note: denominators are calculated by each Diabetes group
di_app_tb = pd.crosstab(df2['Diabetes'], df2['No-show']).apply(lambda r:
 r/r.sum(), axis=1)
di_app_tb
```

Out[122]:

No-show	No	Yes
Diabetes		
0	0.796370	0.203630
1	0.819967	0.180033

In [123]:
```
# Calculate Relative Risk
di_RR = di_app_tb['Yes'][1]/di_app_tb['Yes'][0]
di_RR
```

Out[123]: 0.8841159400804455

In [124]:
```
# Odds for with diabetes and without diabetes
di_app_tb['Odds'] = di_app_tb['Yes'] * (1-di_app_tb['Yes'])
di_app_tb
```

Out[124]:

No-show	No	Yes	Odds
Diabetes			
0	0.796370	0.203630	0.162165
1	0.819967	0.180033	0.147621

In [125]:
```
# Calculate Odds ratio
di_ORR = di_app_tb['Odds'][1]/di_app_tb['Odds'][0]
di_ORR
```

Out[125]: 0.9103134740150118

e. Alcoholism

> From the counts, the number of alcoholic patients is less than the number of non-alcoholic patients.

In [126]:
```
# The count in each group of Alcoholism and no-show
df2.groupby('No-show')['Alcoholism'].value_counts()
```

Out[126]:
```
No-show  Alcoholism
No       0             85524
         1              2683
Yes      0             21642
         1               677
Name: Alcoholism, dtype: int64
```

In [127]:
```
# Create a 2*2 table to see the percentage in each group
# Note: denominators are calculated by each Alcoholism group
al_app_tb = pd.crosstab(df2['Alcoholism'], df2['No-show']).apply(lambda
r: r/r.sum(), axis=1)
al_app_tb
```

Out[127]:

No-show Alcoholism	No	Yes
0	0.798052	0.201948
1	0.798512	0.201488

In [128]:
```
# Calculate Relative Risk
al_RR = al_app_tb['Yes'][1]/al_app_tb['Yes'][0]
al_RR
```

Out[128]:
```
0.9977207843214912
```

In [129]:
```
# Odds for with Alcoholism and without Alcoholism
al_app_tb['Odds'] = al_app_tb['Yes'] * (1-al_app_tb['Yes'])
al_app_tb
```

Out[129]:

No-show Alcoholism	No	Yes	Odds
0	0.798052	0.201948	0.161165
1	0.798512	0.201488	0.160891

In [130]:
```
# Calculate Odds ratio
al_ORR = al_app_tb['Odds'][1]/al_app_tb['Odds'][0]
al_ORR
```

Out[130]:
```
0.9982962293349484
```

f. SMS_received

> From the counts, the number of patients who received SMS is less than the number of patients who do not.

In [278]:
```
# # The count in each group of SMS_received and no-show
df2.groupby('No-show')['SMS_received'].value_counts()
```

Out[278]:
```
No-show  SMS_received
No       0               62509
         1               25698
Yes      0               12535
         1                9784
Name: SMS_received, dtype: int64
```

In [140]:
```
# Create a 2*2 table to see the percentage in each group
# Note: denominators are calculated by each SMS_received group
sms_app_tb = pd.crosstab(df2['SMS_received'], df2['No-show']).apply(lamb
da r: r/r.sum(), axis=1)
sms_app_tb
```

Out[140]:

No-show	No	Yes
SMS_received		
0	0.832965	0.167035
1	0.724255	0.275745

In [141]:
```
# Calculate Relative Risk
sms_RR = sms_app_tb['Yes'][1]/sms_app_tb['Yes'][0]
sms_RR
```

Out[141]: 1.6508210155131384

In [142]:
```
# Odds for with SMS_received and without SMS_received
sms_app_tb['Odds'] = sms_app_tb['Yes'] * (1-sms_app_tb['Yes'])
sms_app_tb
```

Out[142]:

No-show	No	Yes	Odds
SMS_received			
0	0.832965	0.167035	0.139135
1	0.724255	0.275745	0.199710

In [143]:
```python
# Calculate Odds ratio
sms_ORR = sms_app_tb['Odds'][1]/sms_app_tb['Odds'][0]
sms_ORR
```

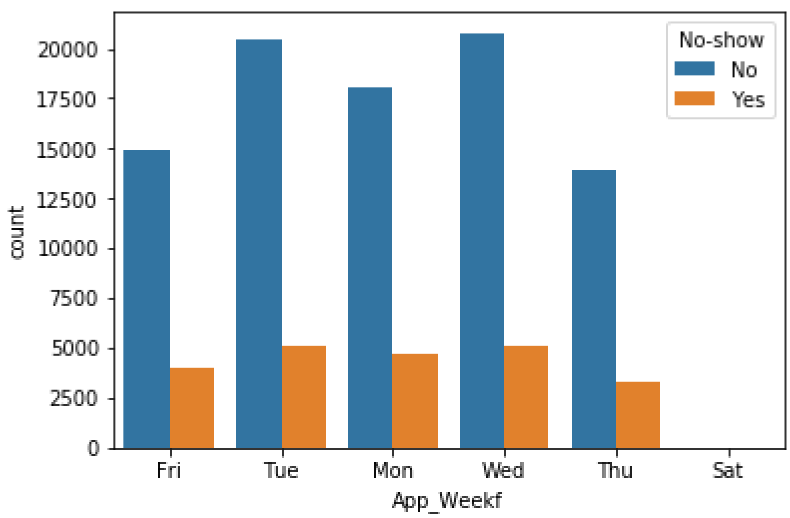
Out[143]: 1.4353725802930137

2. Aanlyze categorical nomial variables

> Then I would like to take a look at ScheduledDay, AppointmentDay, and Neighnourhood. I already derived the weekdays from ScheduledDay and AppointmentDay separately because I am interested in if a patient shows up to the weekend's or weekday's appointment and if the day that a patient makes an appointment that will influences show-up rate or not. Also, I want to know which hospital has a lowerest or highest attandance rate.

a. AppointmentDay (derived weekday variable: App_weekf)

In [204]:
```python
# Appointment Day: Apparently, the number of patients who show up to app
ointments is way higher than
# the number of patients who did not show up. Also, the number on Saturd
ay is too little, so cannot be visualized
# in the plot. Thus, calculating proportions is more reasonable.
sns.countplot(x='App_Weekf', hue='No-show', data=df2);
```



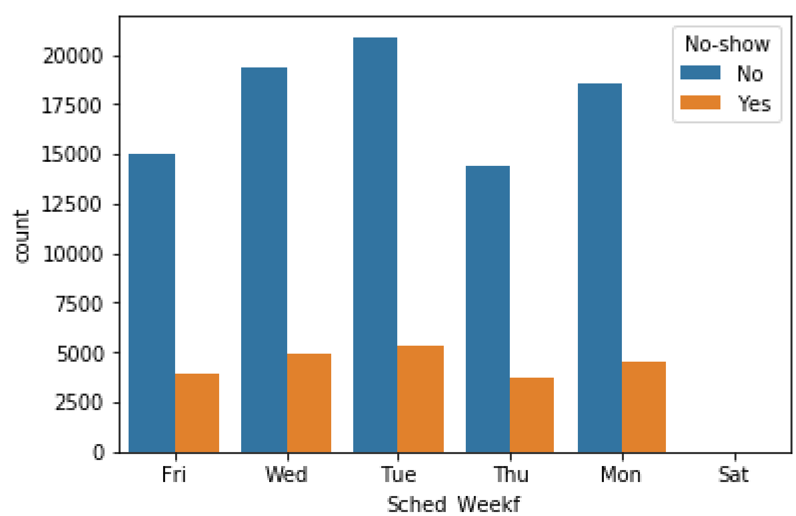
In [273]:
```
# By calculation proportions, we can find that the less people show up t
o appointments on Saturday.
app_tb = pd.crosstab(df2['App_Weekf'],df2['No-show']).apply(lambda r: r/
r.sum(), axis=1)
app_tb
```

Out[273]:

No-show	No	Yes
App_Weekf		
Fri	0.787739	0.212261
Mon	0.793519	0.206481
Sat	0.769231	0.230769
Thu	0.806459	0.193541
Tue	0.799064	0.200936
Wed	0.803108	0.196892

b. ScheduledDay (Sched_weekf)

In [243]:
```
# Do the same thing for ScheduledDay. The number of Saturday is too litt
le so cannot be visualized in the plot.
sns.countplot(x='Sched_Weekf', hue='No-show', data=df2);
```



In [245]:
```
# By calculation proportions, we can patients who make appointments on S
aturday have the highest attendance rate.
sch_tb = pd.crosstab(df2['Sched_Weekf'],df2['No-show']).apply(lambda r:
r/r.sum(), axis=1)
sch_tb
```

Out[245]:

No-show	No	Yes
Sched_Weekf		
Fri	0.794502	0.205498
Mon	0.802417	0.197583
Sat	0.958333	0.041667
Thu	0.795275	0.204725
Tue	0.797806	0.202194
Wed	0.798904	0.201096

c. Neighbourhood

In [252]:
```
# Find the numbers of No-show and Show in each hospital (Neighbourhood)
ctnei = pd.crosstab(df2['Neighbourhood'],df2['No-show']).apply(lambda r:
 r, axis=1)
ctnei.reset_index(level=0, inplace=True)
ctnei
```

Out[252]:

No-show	Neighbourhood	No	Yes
0	AEROPORTO	7	1
1	ANDORINHAS	1741	521
2	ANTÔNIO HONÓRIO	221	50
3	ARIOVALDO FAVALESSA	220	62
4	BARRO VERMELHO	332	91
5	BELA VISTA	1523	384
6	BENTO FERREIRA	665	193
7	BOA VISTA	254	58
8	BONFIM	2223	550
9	CARATOÍRA	1974	591
10	CENTRO	2631	703
11	COMDUSA	254	56
12	CONQUISTA	689	160
13	CONSOLAÇÃO	1139	237
14	CRUZAMENTO	1094	304
15	DA PENHA	1788	429
16	DE LOURDES	258	47
17	DO CABRAL	472	88
18	DO MOSCOSO	321	92
19	DO QUADRO	709	140
20	ENSEADA DO SUÁ	183	52
21	ESTRELINHA	432	106
22	FONTE GRANDE	533	149
23	FORTE SÃO JOÃO	1543	346
24	FRADINHOS	210	48
25	GOIABEIRAS	563	137
26	GRANDE VITÓRIA	854	217
27	GURIGICA	1562	456
28	HORTO	133	42
29	ILHA DAS CAIEIRAS	836	235
...	...	...	...
51	PARQUE INDUSTRIAL	1	0
52	PARQUE MOSCOSO	623	179
53	PIEDADE	364	88

No-show	Neighbourhood	No	Yes
54	PONTAL DE CAMBURI	57	12
55	PRAIA DO CANTO	845	190
56	PRAIA DO SUÁ	994	294
57	REDENÇÃO	1278	275
58	REPÚBLICA	692	143
59	RESISTÊNCIA	3525	906
60	ROMÃO	1740	474
61	SANTA CECÍLIA	325	123
62	SANTA CLARA	372	134
63	SANTA HELENA	141	37
64	SANTA LUÍZA	351	77
65	SANTA LÚCIA	352	86
66	SANTA MARTHA	2635	496
67	SANTA TEREZA	1060	272
68	SANTO ANDRÉ	2063	508
69	SANTO ANTÔNIO	2262	484
70	SANTOS DUMONT	907	369
71	SANTOS REIS	435	112
72	SEGURANÇA DO LAR	117	28
73	SOLON BORGES	400	69
74	SÃO BENEDITO	1152	287
75	SÃO CRISTÓVÃO	1473	363
76	SÃO JOSÉ	1549	428
77	SÃO PEDRO	1933	515
78	TABUAZEIRO	2559	573
79	UNIVERSITÁRIO	120	32
80	VILA RUBIM	710	141

81 rows × 3 columns

In [253]:
```
# There are 81 hospitals in this dataset and find out the percentages of
 No-show and show in each hospital
nei_tb = pd.crosstab(df2['Neighbourhood'],df2['No-show']).apply(lambda r
: r/r.sum(), axis=1)
nei_tb.reset_index(level=0, inplace=True)
nei_tb
```

Out[253]:

No-show	Neighbourhood	No	Yes
0	AEROPORTO	0.875000	0.125000
1	ANDORINHAS	0.769673	0.230327
2	ANTÔNIO HONÓRIO	0.815498	0.184502
3	ARIOVALDO FAVALESSA	0.780142	0.219858
4	BARRO VERMELHO	0.784870	0.215130
5	BELA VISTA	0.798637	0.201363
6	BENTO FERREIRA	0.775058	0.224942
7	BOA VISTA	0.814103	0.185897
8	BONFIM	0.801659	0.198341
9	CARATOÍRA	0.769591	0.230409
10	CENTRO	0.789142	0.210858
11	COMDUSA	0.819355	0.180645
12	CONQUISTA	0.811543	0.188457
13	CONSOLAÇÃO	0.827762	0.172238
14	CRUZAMENTO	0.782546	0.217454
15	DA PENHA	0.806495	0.193505
16	DE LOURDES	0.845902	0.154098
17	DO CABRAL	0.842857	0.157143
18	DO MOSCOSO	0.777240	0.222760
19	DO QUADRO	0.835100	0.164900
20	ENSEADA DO SUÁ	0.778723	0.221277
21	ESTRELINHA	0.802974	0.197026
22	FONTE GRANDE	0.781525	0.218475
23	FORTE SÃO JOÃO	0.816834	0.183166
24	FRADINHOS	0.813953	0.186047
25	GOIABEIRAS	0.804286	0.195714
26	GRANDE VITÓRIA	0.797386	0.202614
27	GURIGICA	0.774034	0.225966
28	HORTO	0.760000	0.240000
29	ILHA DAS CAIEIRAS	0.780579	0.219421
...	...	...	...
51	PARQUE INDUSTRIAL	1.000000	0.000000
52	PARQUE MOSCOSO	0.776808	0.223192
53	PIEDADE	0.805310	0.194690

No-show	Neighbourhood	No	Yes
54	PONTAL DE CAMBURI	0.826087	0.173913
55	PRAIA DO CANTO	0.816425	0.183575
56	PRAIA DO SUÁ	0.771739	0.228261
57	REDENÇÃO	0.822923	0.177077
58	REPÚBLICA	0.828743	0.171257
59	RESISTÊNCIA	0.795531	0.204469
60	ROMÃO	0.785908	0.214092
61	SANTA CECÍLIA	0.725446	0.274554
62	SANTA CLARA	0.735178	0.264822
63	SANTA HELENA	0.792135	0.207865
64	SANTA LUÍZA	0.820093	0.179907
65	SANTA LÚCIA	0.803653	0.196347
66	SANTA MARTHA	0.841584	0.158416
67	SANTA TEREZA	0.795796	0.204204
68	SANTO ANDRÉ	0.802412	0.197588
69	SANTO ANTÔNIO	0.823744	0.176256
70	SANTOS DUMONT	0.710815	0.289185
71	SANTOS REIS	0.795247	0.204753
72	SEGURANÇA DO LAR	0.806897	0.193103
73	SOLON BORGES	0.852878	0.147122
74	SÃO BENEDITO	0.800556	0.199444
75	SÃO CRISTÓVÃO	0.802288	0.197712
76	SÃO JOSÉ	0.783510	0.216490
77	SÃO PEDRO	0.789624	0.210376
78	TABUAZEIRO	0.817050	0.182950
79	UNIVERSITÁRIO	0.789474	0.210526
80	VILA RUBIM	0.834313	0.165687

81 rows × 3 columns

In [255]:
```
# Find out the distribution of no-show and show rate
nei_tb.describe()
```

Out[255]:

No-show	No	Yes
count	81.000000	81.000000
mean	0.794572	0.205428
std	0.097230	0.097230
min	0.000000	0.000000
25%	0.782546	0.179907
50%	0.802412	0.197588
75%	0.820093	0.217454
max	1.000000	1.000000

In [257]:
```
# Find out the top 5 Show hospital
nei_tb.sort_values(by='No', ascending=False).head(5)
```

Out[257]:

No-show	Neighbourhood	No	Yes
51	PARQUE INDUSTRIAL	1.000000	0.000000
31	ILHA DO BOI	0.914286	0.085714
0	AEROPORTO	0.875000	0.125000
48	MÁRIO CYPRESTE	0.854447	0.145553
73	SOLON BORGES	0.852878	0.147122

In [258]:
```
# Take a look at the counts in top 5 hospital
# Only 1 record, the sample size is too little.
ctnei.query('Neighbourhood == "PARQUE INDUSTRIAL"')
```

Out[258]:

No-show	Neighbourhood	No	Yes
51	PARQUE INDUSTRIAL	1	0

In [259]:
```
ctnei.query('Neighbourhood == "ILHA DO BOI"')
```

Out[259]:

No-show	Neighbourhood	No	Yes
31	ILHA DO BOI	32	3

In [260]:
```
ctnei.query('Neighbourhood == "AEROPORTO"')
```

Out[260]:

No-show	Neighbourhood	No	Yes
0	AEROPORTO	7	1

In [261]:
```
ctnei.query('Neighbourhood == "MÁRIO CYPRESTE"')
```

Out[261]:

No-show	Neighbourhood	No	Yes
48	MÁRIO CYPRESTE	317	54

Based on the above calculation, the sample size of top 5 show hospital actually is very small. I am thinking about what is the more reasonable sample size for each hospital. Because we have 110527 records in this dataset, I use 1000 records (1%) as a threshold.

In [262]:
```
ctnei['Hos_sumpt'] = ctnei['No'] + ctnei['Yes']
ctnei.head()
```

Out[262]:

No-show	Neighbourhood	No	Yes	Hos_sumpt
0	AEROPORTO	7	1	8
1	ANDORINHAS	1741	521	2262
2	ANTÔNIO HONÓRIO	221	50	271
3	ARIOVALDO FAVALESSA	220	62	282
4	BARRO VERMELHO	332	91	423

In [264]:
```
# Keep the hospital with over 1000 appointments
ctnei_adj = ctnei.query('Hos_sumpt > 1000')
ctnei_adj.head()
```

Out[264]:

No-show	Neighbourhood	No	Yes	Hos_sumpt
1	ANDORINHAS	1741	521	2262
5	BELA VISTA	1523	384	1907
8	BONFIM	2223	550	2773
9	CARATOÍRA	1974	591	2565
10	CENTRO	2631	703	3334

In [266]:
```
# Only 39 hospitals have over 1000 appointment (around 1% of the records
 in the dataset)
ctnei_adj.shape
```

Out[266]:
```
(39, 4)
```

In [272]:
```
ctnei_adj.describe()
```

Out[272]:

No-show	No	Yes	Hos_sumpt	No_pct	Yes_pct
count	39.000000	39.000000	39.000000	39.000000	39.000000
mean	1907.564103	486.435897	2394.000000	0.795554	0.204446
std	1072.441479	267.272756	1329.344021	0.026671	0.026671
min	836.000000	190.000000	1035.000000	0.710815	0.158416
25%	1160.500000	299.000000	1433.000000	0.781563	0.183370
50%	1734.000000	429.000000	2214.000000	0.798637	0.201363
75%	2190.000000	541.000000	2759.500000	0.816630	0.218437
max	6252.000000	1465.000000	7717.000000	0.841584	0.289185

In [269]:
```
# Calculate No-show and show rate in these 39 hospitals
ctnei_adj['No_pct'] = ctnei_adj['No']/ctnei_adj['Hos_sumpt']
ctnei_adj['Yes_pct'] = ctnei_adj['Yes']/ctnei_adj['Hos_sumpt']
ctnei_adj.head()
```

```
/anaconda3/lib/python3.6/site-packages/ipykernel/__main__.py:1: Setting
WithCopyWarning:
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-d
ocs/stable/indexing.html#indexing-view-versus-copy
  if __name__ == '__main__':
/anaconda3/lib/python3.6/site-packages/ipykernel/__main__.py:2: Setting
WithCopyWarning:
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-d
ocs/stable/indexing.html#indexing-view-versus-copy
  from ipykernel import kernelapp as app
```

Out[269]:

No-show	Neighbourhood	No	Yes	Hos_sumpt	No_pct	Yes_pct
1	ANDORINHAS	1741	521	2262	0.769673	0.230327
5	BELA VISTA	1523	384	1907	0.798637	0.201363
8	BONFIM	2223	550	2773	0.801659	0.198341
9	CARATOÍRA	1974	591	2565	0.769591	0.230409
10	CENTRO	2631	703	3334	0.789142	0.210858

In [270]:
```
ctnei_adj.sort_values(by='Yes_pct', ascending=False).head(5)
```

Out[270]:

No-show	Neighbourhood	No	Yes	Hos_sumpt	No_pct	Yes_pct
70	SANTOS DUMONT	907	369	1276	0.710815	0.289185
36	ITARARÉ	2591	923	3514	0.737336	0.262664
40	JESUS DE NAZARETH	2157	696	2853	0.756046	0.243954
33	ILHA DO PRÍNCIPE	1734	532	2266	0.765225	0.234775
9	CARATOÍRA	1974	591	2565	0.769591	0.230409

In [271]:
```
ctnei_adj.sort_values(by='No_pct', ascending=False).head(5)
```

Out[271]:

No-show	Neighbourhood	No	Yes	Hos_sumpt	No_pct	Yes_pct
66	SANTA MARTHA	2635	496	3131	0.841584	0.158416
39	JARDIM DA PENHA	3246	631	3877	0.837245	0.162755
13	CONSOLAÇÃO	1139	237	1376	0.827762	0.172238
69	SANTO ANTÔNIO	2262	484	2746	0.823744	0.176256
57	REDENÇÃO	1278	275	1553	0.822923	0.177077

Keeping only the hospitals with more than 1000 appointments (1% of the rows in the dataset) and finding out the Top 5 No-Show and Show rates hospitals give us more reasonable outcome. Before doing so, the top 1 hospital has 100% Show rate and 0% No-Show rate. It is not really realistic in the real world.

3. Analyze continuous variables

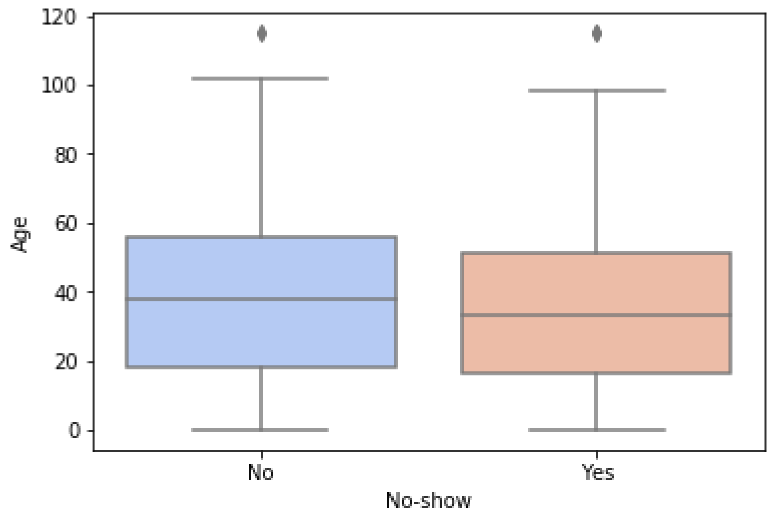
I would like to take a look at Age distribution between No-Show and Show.

In [73]:
```
# Find distributions in Age
df3.groupby('No-show')['Age'].describe()
```

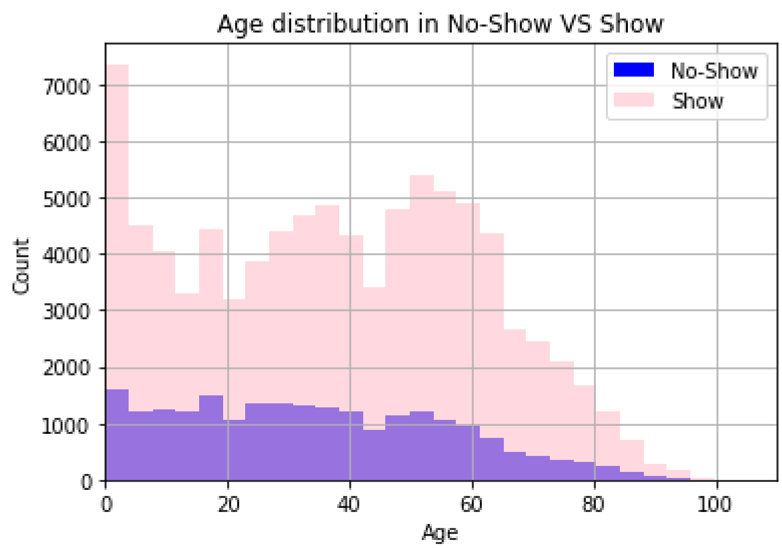
Out[73]:

No-show	count	mean	std	min	25%	50%	75%	max
No	88207.0	37.790504	23.338645	0.0	18.0	38.0	56.0	115.0
Yes	22319.0	34.317667	21.965941	0.0	16.0	33.0	51.0	115.0

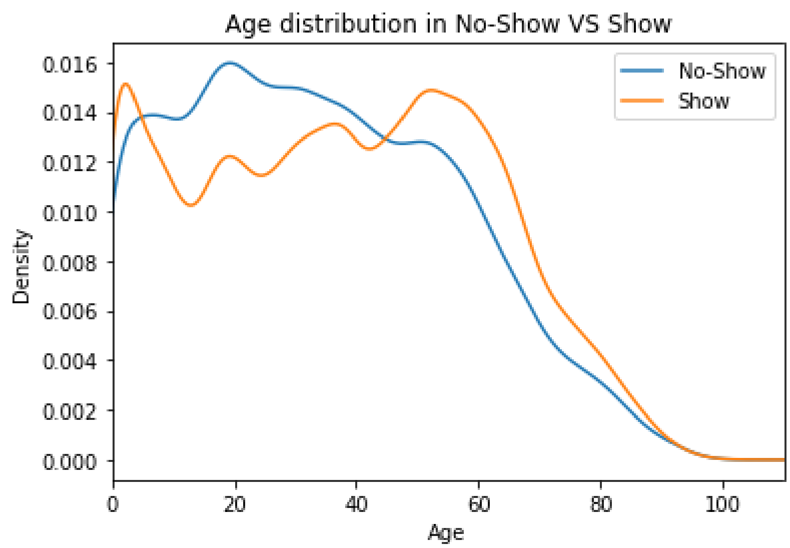
In [167]:
```
# Draw a box plot of Age for No-show and show groups
sns.boxplot(x="No-show", y="Age",data=df3, palette="coolwarm");
```



In [187]:
```
# Draw a histogram to see the disrtibution of Age in No-Show and Show gr
oup
df2[df2['No-show'] == 'Yes']['Age'].hist(label='No-Show', bins=30, color
='blue')
df2[df2['No-show'] == 'No']['Age'].hist(alpha=0.6, label='Show', bins=30
, color='pink')
plt.xlabel('Age')
plt.ylabel('Count')
plt.title('Age distribution in No-Show VS Show')
plt.legend(loc=0)
plt.xlim(0,110);
```



In [182]:
```
# Draw a Density Plot to see the distribution of Age in No-Show and Show
 group
df2[df2['No-show'] == 'Yes']['Age'].plot.kde(label='No-Show')
df2[df2['No-show'] == 'No']['Age'].plot.kde(label='Show')
plt.xlabel('Age')
plt.ylabel('Density')
plt.title('Age distribution in No-Show VS Show')
plt.legend(loc=0)
plt.xlim(0,110);
```



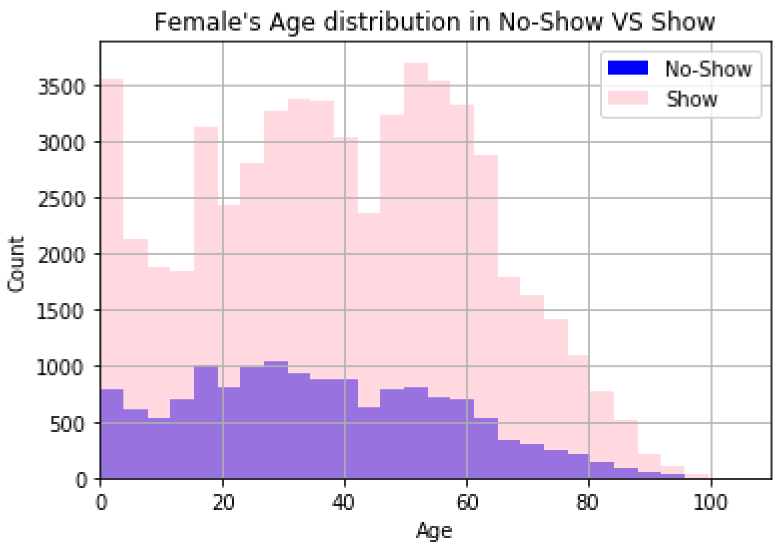
This distribution is not symmetric, more like right skewed, so it's better to use 5 summary numbers

In [279]:
```
# The female's age distribution in No-Show and Show groups.
dff = df2.query('Gender == "F"')
dff.groupby('No-show')['Age'].describe()
```

Out[279]:

	count	mean	std	min	25%	50%	75%	max
No-show								
No	57245.0	39.591126	22.342413	0.0	22.0	40.0	57.0	115.0
Yes	14594.0	36.162190	21.184209	0.0	20.0	34.0	52.0	115.0

In [241]:
```
# Draw a histogram for Female's Age in No-Show and Show groups
dff = df2.query('Gender == "F"')
dff[dff['No-show'] == 'Yes']['Age'].hist(label='No-Show', bins=30, color
='blue')
dff[dff['No-show'] == 'No']['Age'].hist(alpha=0.6, label='Show', bins=30
, color='pink')
plt.xlabel('Age')
plt.ylabel('Count')
plt.title("Female's Age distribution in No-Show VS Show")
plt.legend(loc=0)
plt.xlim(0,110);
```

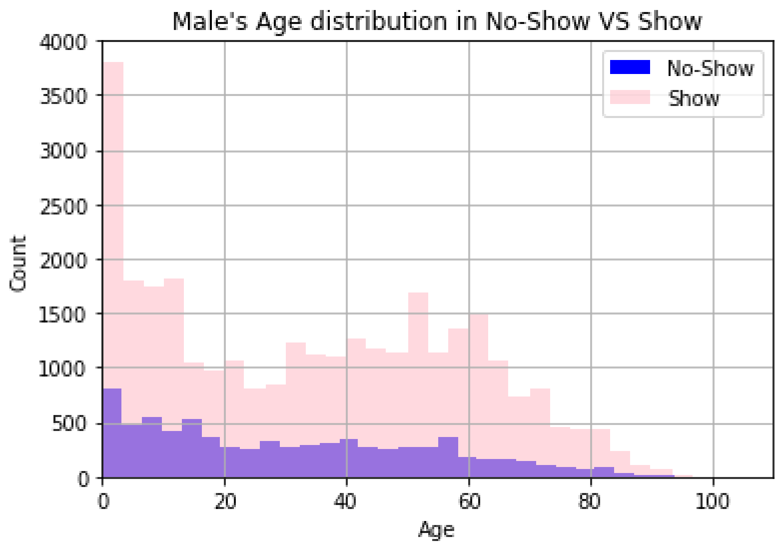


In [280]:
```
# The male's age distribution in No-Show and Show groups.
dff = df2.query('Gender == "M"')
dff.groupby('No-show')['Age'].describe()
```

Out[280]:

	count	mean	std	min	25%	50%	75%	max
No-show								
No	30962.0	34.461372	24.734056	0.0	10.0	34.0	55.0	100.0
Yes	7725.0	30.833010	22.972200	0.0	10.0	28.0	49.0	97.0

In [242]:
```
# Draw a histogram for Male's Age in No-show and Show groups
dfm = df2.query('Gender == "M"')
dfm[dfm['No-show'] == 'Yes']['Age'].hist(label='No-Show', bins=30, color
='blue')
dfm[dfm['No-show'] == 'No']['Age'].hist(alpha=0.6, label='Show', bins=30
, color='pink')
plt.xlabel('Age')
plt.ylabel('Count')
plt.title("Male's Age distribution in No-Show VS Show")
plt.legend(loc=0)
plt.xlim(0,110);
```



Conclusions

1. After calculating and comparing the relative risks and odds ratios, patients who receive SMS have 65% higher showing-up rate than patients who do not receive SMS. Patients who enroll in the Boisa Familia Program have 19.8% higher showing-up rate than patient who do not enroll. Diabete patients and Hipertension tend to not show up to appointments. However, Gender and Alcoholism do not have preferences on No-Show versus Show.

Binomial Variable	Relative Risk	Odds Ratio
Gender	1.0174	1.013
Scholarship	1.1984	1.14
Hipertension	0.8277	0.8654
Diabetes	0.8841	0.9103
Alcoholism	0.9977	0.9983
SMS_received	1.6508	1.4354

1. By calculation proportions, the fewest patients show up to appointments on Saturday. However, patients who make appointments on Saturday have a highest showing-up rate.
2. Only 39 out of 81 hospitals have over 1000 appointments. (I think it is more fair to do the comparison when a hospital has more than a certain number of appointments. Here, I choose 1000 because this number is around 1% of sample size.) In these 39 hospitals, the highest showing up rate is 84.2% at SANTA MARTHA hospital and the highest no-showing rate is 28.9% at SANTOS DUMONT hospital.
3. Based on the Age distribution and 5 summary numbers, younger patients have a higher chance no-show than older patients. The age of "no-show" patients is 33 and the age of "show" patients is 38. If we only see females, the age of "No-show" patients is 34 and the age of "show" patients is 40. In males, the age of "No-show" patients is 28 and the age of "show" patients is 34. It looks like the age of "No-show" males is younger than the age of "No-show" females.

Proposal for the next step

1. Subject level data: in this analysis, I only take a look at data based on appointment level data, so I woule like to take a look at subject level data. If a patient makes more appointments than other, he/she influences on the outcome more.
2. Different time periods: I would like to see a specific time period. For example, does patients show up to appointments in summer more than in winter?
3. Perform statistical models: for the binary outcome, No-show and Show, I would like to build up a logistic regression and see if I can find out more significant insights in the data.

References

1. Original source in Kaggle (https://www.kaggle.com/joniarroba/noshowappointments/home)
2. Bolsa Família (https://en.wikipedia.org/wiki/Bolsa_Fam%C3%ADlia)
3. Interpreting Results in 2 × 2 Tables (https://www.ncbi.nlm.nih.gov/pmc/articles/PMC2797398/)