How to Tell a Story Using Data Project

I have decided to open a robot-run cafe in L.A. This project is a market analysis dedicated to help me and other investors make the right choices for opening this kind of establishment. Will we be able to maintain your success when the novelty of robot waiters wears off?

Main Goals

1. Find out which establishments are more popuular in L.A;
2. Find out if there are more chain establishments or non-chain;
3. Find optimal number of seats for the restaurant;
4. Find best streets for openning a café;

Table of Contents

Step 1. Data Preprocessing

In [1]:
```
#load libraries

import matplotlib.pyplot as plt
import matplotlib as mpl
import re
import numpy as np
import pandas as pd
import seaborn as sns
import warnings; warnings.simplefilter('ignore')
import plotly.express as px
!pip install -q usaddress
import usaddress


from functools import reduce
from math import factorial
from scipy import stats as st
from statistics import mean
from IPython.display import display
from plotly import graph_objects as go

pd.set_option('display.max_columns', 500)
```

```
WARNING: The scripts futurize and pasteurize are installed in '/home/jovyan/.local/bin' which is not on PATH.
Consider adding this directory to PATH or, if you prefer to suppress this warning, use --no-warn-script-location.
```

In [2]:
```
#load data

#rest = pd.read_csv('rest_data_us.csv') #if local

rest = pd.read_csv('/datasets/rest_data_us.csv') #if on server
```

In [3]:
```
#check the data
rest.info()
display(rest.describe(include='all'))
display(rest.head())
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 9651 entries, 0 to 9650
Data columns (total 6 columns):
id             9651 non-null int64
object_name    9651 non-null object
address        9651 non-null object
chain          9648 non-null object
object_type    9651 non-null object
number         9651 non-null int64
dtypes: int64(2), object(4)
memory usage: 452.5+ KB
```

	id	object_name	address	chain	object_type	number
count	9651.000000	9651	9651	9648	9651	9651.000000
unique	NaN	8672	8517	2	6	NaN
top	NaN	THE COFFEE BEAN & TEA LEAF	3607 TROUSDALE PKWY	False	Restaurant	NaN
freq	NaN	47	11	5972	7255	NaN
mean	16611.000000	NaN	NaN	NaN	NaN	43.695161
std	2786.148058	NaN	NaN	NaN	NaN	47.622874
min	11786.000000	NaN	NaN	NaN	NaN	1.000000
25%	14198.500000	NaN	NaN	NaN	NaN	14.000000
50%	16611.000000	NaN	NaN	NaN	NaN	27.000000
75%	19023.500000	NaN	NaN	NaN	NaN	46.000000
max	21436.000000	NaN	NaN	NaN	NaN	229.000000

	id	object_name	address	chain	object_type	number
0	11786	HABITAT COFFEE SHOP	3708 N EAGLE ROCK BLVD	False	Cafe	26
1	11787	REILLY'S	100 WORLD WAY # 120	False	Restaurant	9
2	11788	STREET CHURROS	6801 HOLLYWOOD BLVD # 253	False	Fast Food	20
3	11789	TRINITI ECHO PARK	1814 W SUNSET BLVD	False	Restaurant	22
4	11790	POLLEN	2100 ECHO PARK AVE	False	Restaurant	20

There are 9651 restaurants in the datasets.

Fill empty values in 'chain' column and make it boolian.

In [4]:
```
rest['chain'] = rest.chain.fillna(False)
rest['chain'] = (rest.chain == True)
rest.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 9651 entries, 0 to 9650
Data columns (total 6 columns):
id             9651 non-null int64
object_name    9651 non-null object
address        9651 non-null object
chain          9651 non-null bool
object_type    9651 non-null object
number         9651 non-null int64
dtypes: bool(1), int64(2), object(3)
memory usage: 386.5+ KB
```

Check if there're any duplicated enteries (rows that have the same name and address).

In [5]:
```
print ('Amount of duplicated rows: ',rest[rest.duplicated(['object_name','address'])].shape[0])
```

```
Amount of duplicated rows:  0
```

There seem to be a huge mess with restaurant names, let's try to clean them a bit.

In [6]:
```
#Let's look at some establishments with many branches
rest.groupby('object_name')[['id']].count().sort_values('id',ascending=False).head(40)
```

Out[6]:

object_name	id
THE COFFEE BEAN & TEA LEAF	47
SUBWAY	31
DOMINO'S PIZZA	15
KENTUCKY FRIED CHICKEN	14
WABA GRILL	14
TRIMANA	13
MCDONALD'S	13
YOGURTLAND	12
STARBUCKS	12
PAPA JOHN'S PIZZA	12
HONG KONG EXPRESS	12
SUBWAY SANDWICHES	11
CHIPOTLE MEXICAN GRILL	10
LOUISIANA FRIED CHICKEN	10
WINGSTOP	10
EL POLLO LOCO	10
KFC	9
BLUE BOTTLE COFFEE	9
CARL'S JR	8
BASKIN ROBBINS	8
JERSEY MIKE'S SUBS	8
PINKBERRY	7
CHINA EXPRESS	7
WETZEL'S PRETZELS	6
LITTLE CAESARS	6
WHOLE FOODS MARKET	6
TACO BELL	6
LA MONARCA BAKERY	6
PANDA EXPRESS	6
FATBURGER	6
POLLO CAMPERO	6
CHINATOWN EXPRESS	6
THE FLAME BROILER	5
PARIS BAGUETTE	5
BAJA FRESH	5
EDIBLE ARRANGEMENTS	5
HONG KONG BOWL	5
EL SUPER	5
I LOVE BOBA	5
MENDOCINO FARMS	5

I'll try to make some of the most popular establishments have the same name, but I'll also try to apply some of more general changes. Unfortunatly it's almost impossible to clean all this data, therefore we're just going to do the best that we can

In [7]:
```python
#drop all numbers that come not in the start of the string
rest['object_name'] = rest['object_name'].replace('#([0-9+ ]+)$','',regex=True)
#drop all numbers of establishmeints that come with "#" sign and with not less than 2 numbers
rest['object_name'] = rest['object_name'].replace('#[0-9][0-9]','',regex=True)
#make all MCDONALD's have the same name
rest['object_name'] = rest['object_name'].replace('MCDONALD[\w \'!@#$%^&*()\/\\\|]+','MCDONALD\'S',regex=True)
#make all STARBUCKS have the same name
rest['object_name'] = rest['object_name'].replace('STARBUCK[\w \'!@#$%^&*()\/\\\|-]+','STARBUCKS',regex=True)
#make all Subway have the same name
rest['object_name'] = rest['object_name'].replace('SUBWAY[\w \'!@#$%^&*()\/\\\|-]+','SUBWAY',regex=True)
#make all Burger King have the same name
rest['object_name'] = rest['object_name'].replace('BURGER KING[\w \'!@#$%^&*()\/\\\|-]+','BURGER KING',regex=True)
#make all Domino's pizza have the same name
rest['object_name'] = rest['object_name'].replace('DOMINO[\w \'!@#$%^&*()\/\\\|-]+','DOMINO\'S PIZZA',regex=True)
#make all AFC chain have the same name
rest['object_name'] = rest['object_name'].replace("AFC[\w @#&\']+","AFC SUSHI",regex=True)
#drop anything that comes after "," (like INC, LCC etc.)
rest['object_name'] = rest['object_name'].replace("[,][\w .,!@#$%^&*-]+","",regex=True)
#make baskin robbins have the same name
rest['object_name'] = rest['object_name'].replace("BASKIN ROBBINS[\w ]+","BASKIN ROBBINS",regex=True)
#make BIG MAMAS & PAPAS PIZZERIA look have the same name
rest['object_name'] = rest['object_name'].replace("BIG MAMA[\w!@#$%^&*\' ]+PAPA[\w!@#$%^&*\'\\\/ ]+","BIG MAMAS & PAPA
S PIZZERIA",regex=True)
#drop all INC
rest['object_name'] = rest['object_name'].replace("INC","",regex=True)
#drop all LCC
rest['object_name'] = rest['object_name'].replace("LCC","",regex=True)
#drop spaces in the end
rest['object_name'] = rest['object_name'].replace("[ ]$","",regex=True)
#replace KENTUCKY FRIED CHICKEN with KFC
rest['object_name'] = rest['object_name'].replace("KENTUCKY FRIED CHICKEN","KFC",regex=True)
#group all CHINA EXPRESS
rest['object_name'] = rest['object_name'].replace("[\w .,]*CHINA EXPRESS[\w., ]*","CHINA EXPRESS",regex=True)
rest.groupby('object_name')[['id']].count().sort_values('id',ascending=False).head(40)
```

Out[7]:

object_name	id
SUBWAY	152
STARBUCKS	132
MCDONALD'S	83
JACK IN THE BOX	52
THE COFFEE BEAN & TEA LEAF	51
BURGER KING	38
EL POLLO LOCO	35
DOMINO'S PIZZA	34
PIZZA HUT	30
TACO BELL	29
YOSHINOYA	28
KFC	26
PANDA EXPRESS	22
CARL'S JR	22
AFC SUSHI	21
RALPHS MARKET	20
JAMBA JUICE	19
CHIPOTLE MEXICAN GRILL	18
BASKIN ROBBINS	17
WABA GRILL	15
PAPA JOHN'S PIZZA	14
WINGSTOP	14
LITTLE CAESARS	13
TRIMANA	13
HONG KONG EXPRESS	13
CHINATOWN EXPRESS	12
YOGURTLAND	12
LOUISIANA FRIED CHICKEN	11
CHINA EXPRESS	11
WHELL'S DONUTS	10
CHURCH'S FRIED CHICKEN	9
KING TACO	9
PINKBERRY	9
JERSEY MIKE'S SUBS	9
LA PIZZA LOCA	9
FOOD 4 LESS	9
BLUE BOTTLE COFFEE	9
DENNY'S	8
FATBURGER	8
VONS MARKET	8

Now names look a little bit better and our chains seem to be more grouped (for example I have increased number of Mc Donalds' from 13 to 83).

```
In [8]: rest.sample(10)
```

Out[8]:

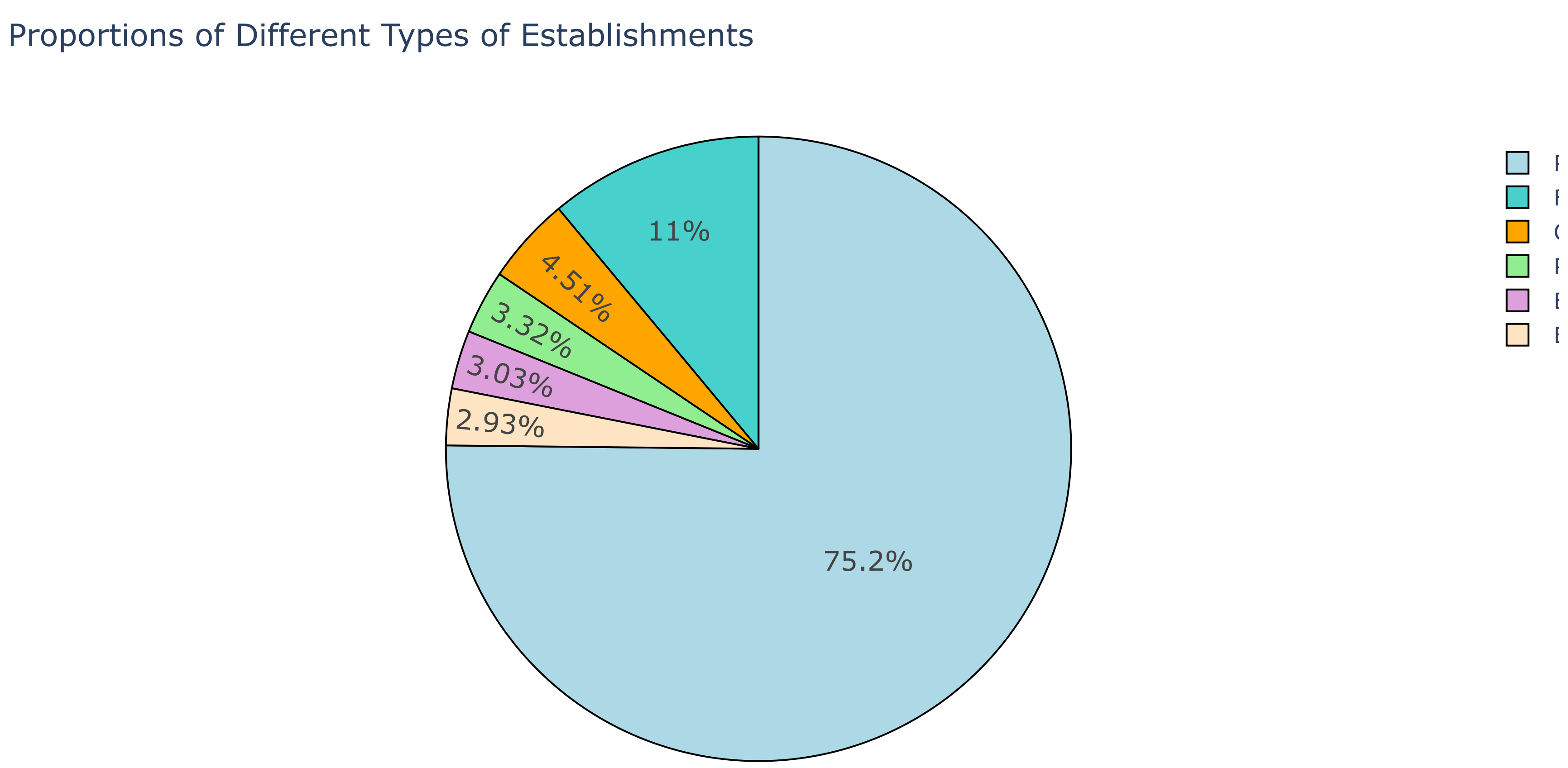
	id	object_name	address	chain	object_type	number
3824	15610	THE CORNER DOOR	12477 W WASHINGTON BLVD	False	Restaurant	43
8619	20405	BEZIAN'S BAKERY	4715 SANTA MONICA BLVD	True	Bakery	3
3696	15482	FOUND COFFEE	1355 COLORADO BLVD	False	Cafe	16
7465	19251	PIZZA BUONA	2100 W SUNSET BLVD	True	Pizza	45
6086	17872	2 FOR 1 PIZZA CO	4707 S BROADWAY	False	Pizza	11
2302	14088	ROSITAS MEXICAN RESTAURANT	2622 N FIGUEROA ST	True	Restaurant	16
4750	16536	JAMBA JUICE	1852 W SLAUSON AVE	True	Restaurant	12
6946	18732	AUNTIE NONA'S RESTAURANT	4463 BEVERLY BLVD STE B	True	Restaurant	29
7913	19699	HOMEBOY DINER	200 N MAIN ST # #210	False	Restaurant	17
9089	20875	ROSE MARKET LLC	3300 OVERLAND AVE # 107	False	Restaurant	24

Now let's get to analysis.

Step 2. Data analysis

Investigate the proportions of the various types of establishments. Plot a graph.

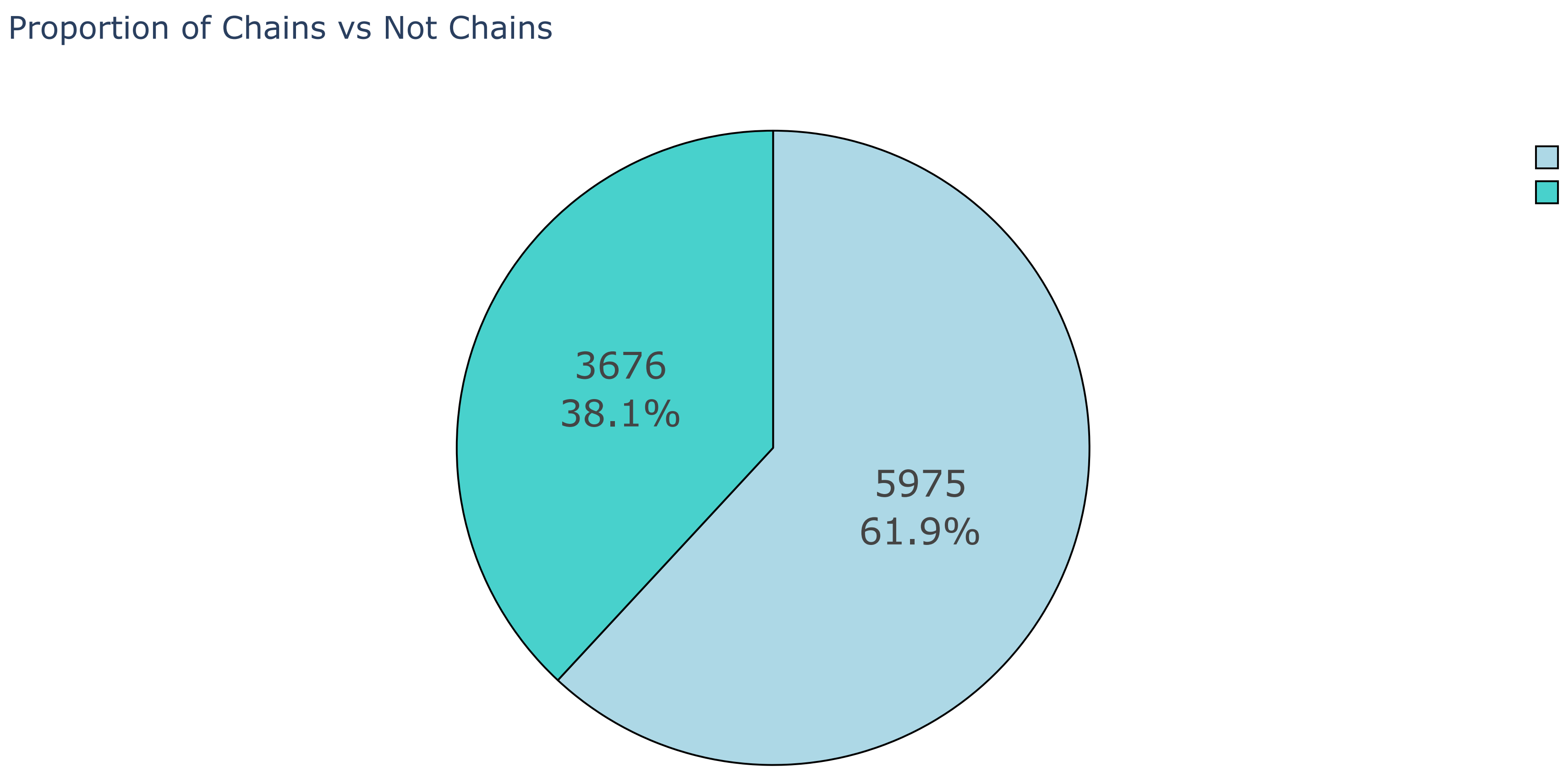
```
In [9]: #define colorpalette
colors = ['lightblue', 'mediumturquoise', 'orange', 'lightgreen', 'plum', 'bisque','lavender',
          'lightcyan','palevioletred']
df = rest.object_type.value_counts()
fig = go.Figure(data=[go.Pie(labels=df.index, values=df)],
                layout_title_text="Proportions of Different Types of Establishments")
fig.update_traces(textfont_size=15,
                  marker=dict(colors=colors, line=dict(color='#000000', width=1)))
fig.show()
```



More than 3/4 of establishments in L.A. are restaurants, they seem to be the most popular place to eat food among residents of the richest U.S. state. Therefore it may be the best choice to open a restaurant.

Investigate the proportions of chain and nonchain establishments. Plot a graph.

In [10]:
```
df = rest.chain.value_counts()
df.index= ['Not Chain', 'Chain']
fig = go.Figure(data=[go.Pie(labels=df.index, values=df)],
                layout_title_text="Proportion of Chains vs Not Chains")
fig.update_traces(textfont_size=20,
                  marker=dict(colors=colors, line=dict(color='#000000', width=1)),textinfo='value+percent')
fig.show()
```



60% of establishments seem to be chains. Maybe we chould open a chain right from the start?

Which type of establishment is typically a chain?

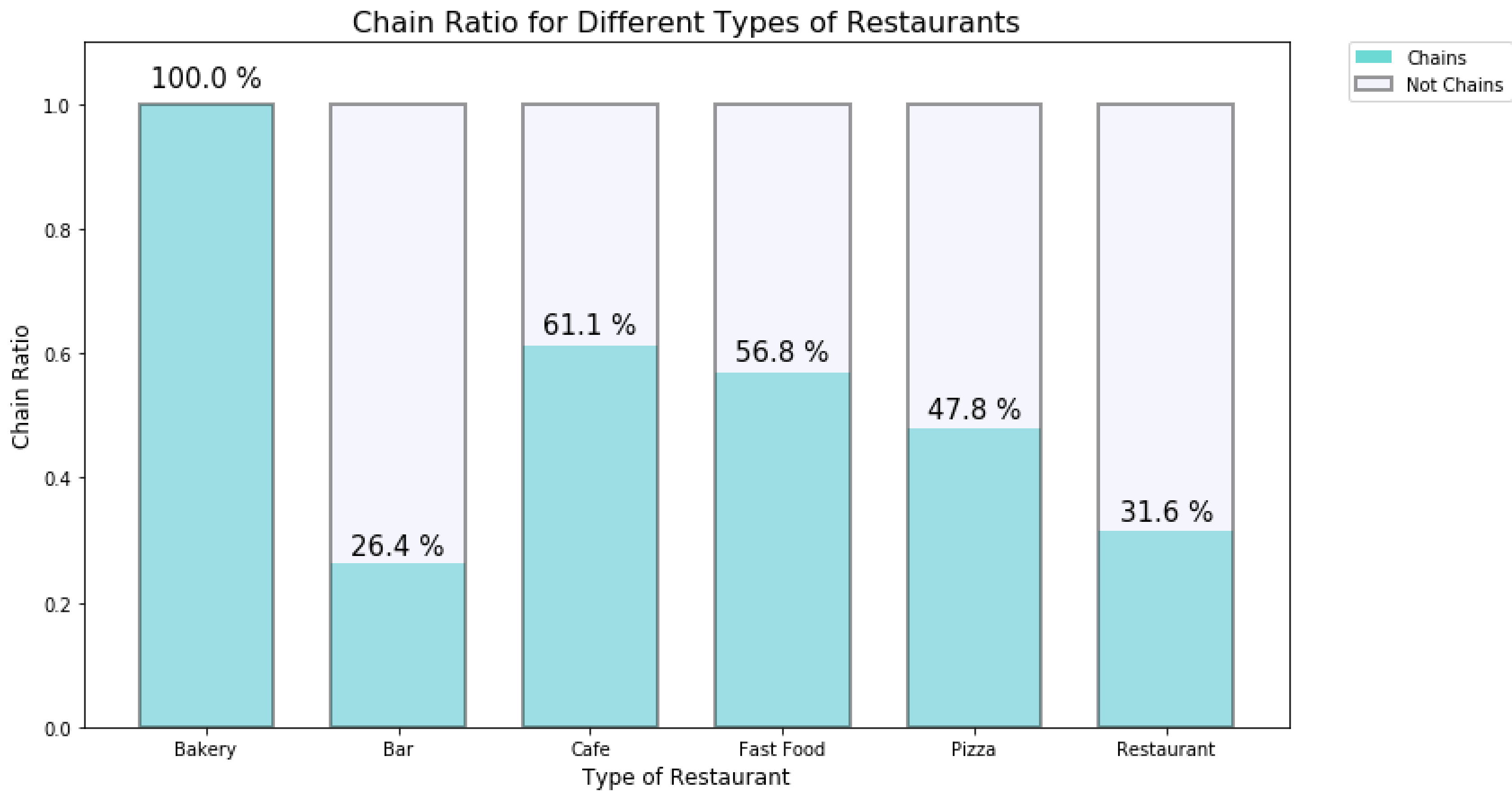
In [11]:
```
df=rest.groupby(['object_type','chain'])[['id']].count().reset_index()
df['chain_ratio'] = (df.id / df.groupby('object_type')['id'].transform('sum'))
chains = df.query('chain==True').chain_ratio

fig, ax = plt.subplots(figsize=(12, 7))
ax.set_title('Chain Ratio for Different Types of Restaurants',fontsize=16)
plt.xlabel('Type of Restaurant',fontsize=12)
plt.ylabel('Chain Ratio',fontsize=12)

#plot
g = plt.bar(df.query('chain==True').object_type, df.query('chain==True').chain_ratio,
        0.7, label='Chains',color=colors[1], alpha=0.8)
g1 = plt.bar(df.query('chain==True').object_type, 1, 0.7,
     label='Not Chains', color=colors[6], alpha=0.4, edgecolor='black',linewidth=2)


#get text above the bar
bar_label = (chains*100).round(1).tolist() #values for text
bar_label = [str(label) for label in bar_label]
def autolabel(rects):
    for idx,rect in enumerate(g):
        height = rect.get_height()
        ax.text(rect.get_x() + rect.get_width()/2., 1.02*height,
                bar_label[idx]+" %",
                ha='center', va='bottom', rotation=0, size=15)
autolabel(g)
plt.ylim(0,1.1)
plt.legend(bbox_to_anchor=(1.05, 1), loc='upper left', borderaxespad=0.)
```

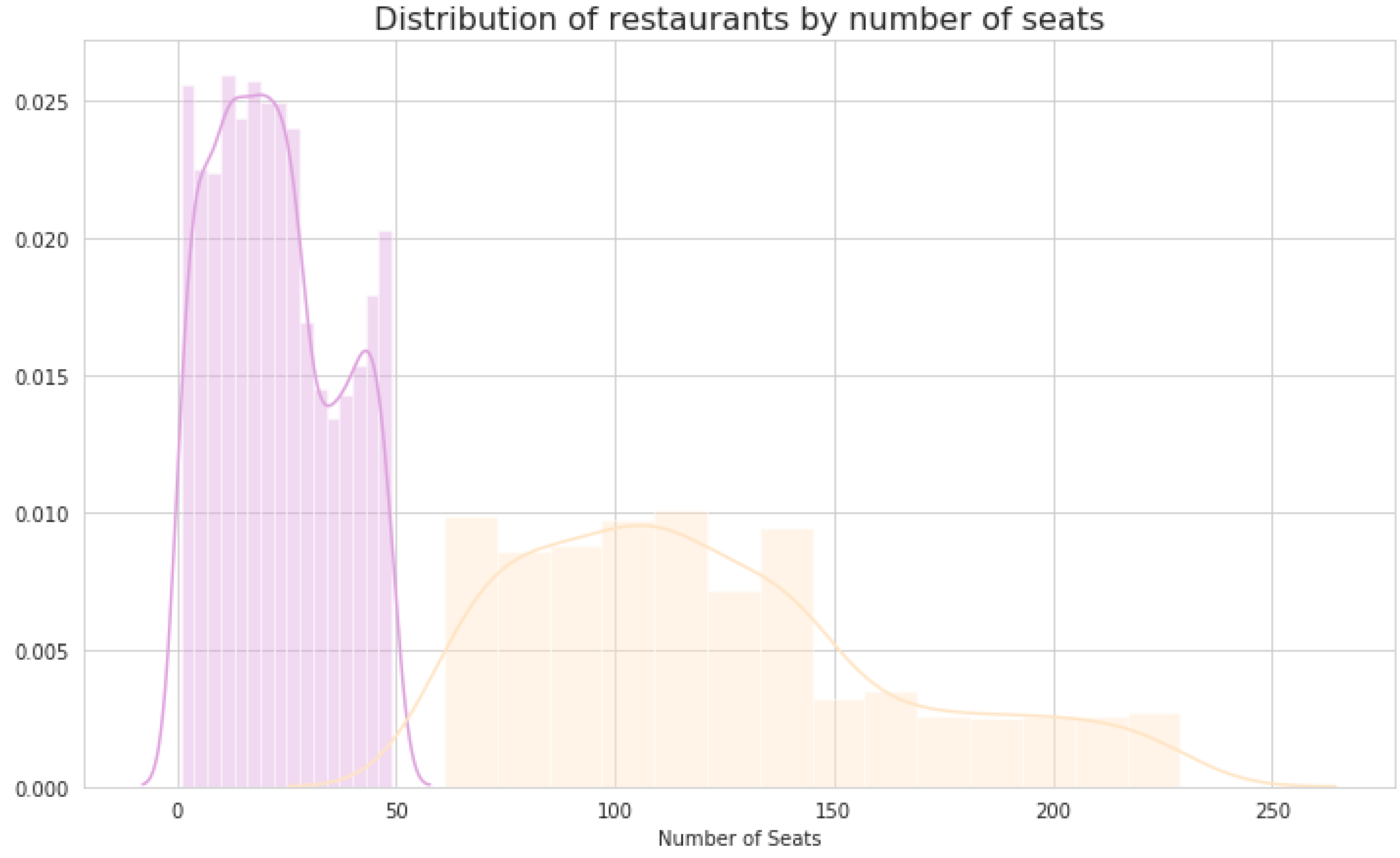
Out[11]: <matplotlib.legend.Legend at 0x7f2f7da19890>



From here I see that all the bakeries are chains. Only 31% of restaurants are chains. Also really high percentage of cafès are chains. We should pay attention to that.

What characterizes chains: many establishments with a small number of seats or a few establishments with a lot of seats?

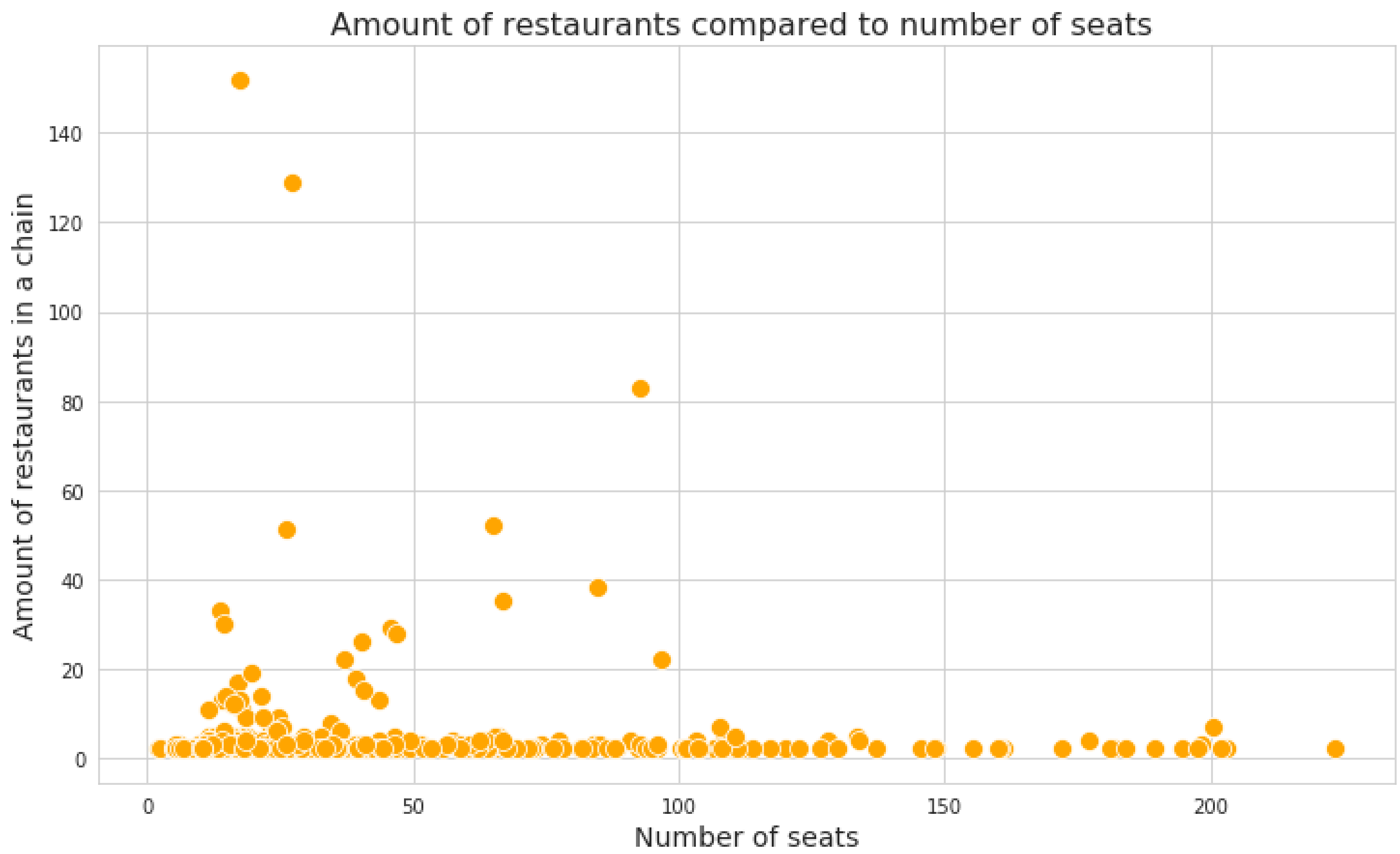
In [12]:
```
sns.set_style('whitegrid')
df = rest.query('chain==True')
fig, ax = plt.subplots(figsize=(12, 7))
sns.distplot(df.query('number <50').number, kde=True, ax=ax, bins='auto', color='plum')
sns.distplot(df.query('number >50').number, kde=True, ax=ax, bins='auto', color='bisque')
ax.set_title('Distribution of restaurants by number of seats',fontsize=16)
ax.set_xlabel('Number of Seats')
plt.grid()
plt.grid()
```



It looks like most of chained restaurants in area don't have very high number of seats, they seem to mostly have from 0 to 50 number of seats.

In [13]:
```
df = (rest
      .query('chain==True')
      .groupby('object_name')[['id','number']]
      .agg({'id':'count','number':'mean'})
      .reset_index()
      .query('id!=1')
     )
fig, ax = plt.subplots(figsize=(12, 7))
ax.set_title('Amount of restaurants compared to number of seats ',fontsize=16)

sns.scatterplot(data=df, y='id',x='number',ax=ax, color=colors[2], s=100)
plt.xlabel('Number of seats',fontsize=14)
plt.ylabel('Amount of restaurants in a chain',fontsize=14)
plt.show()
```

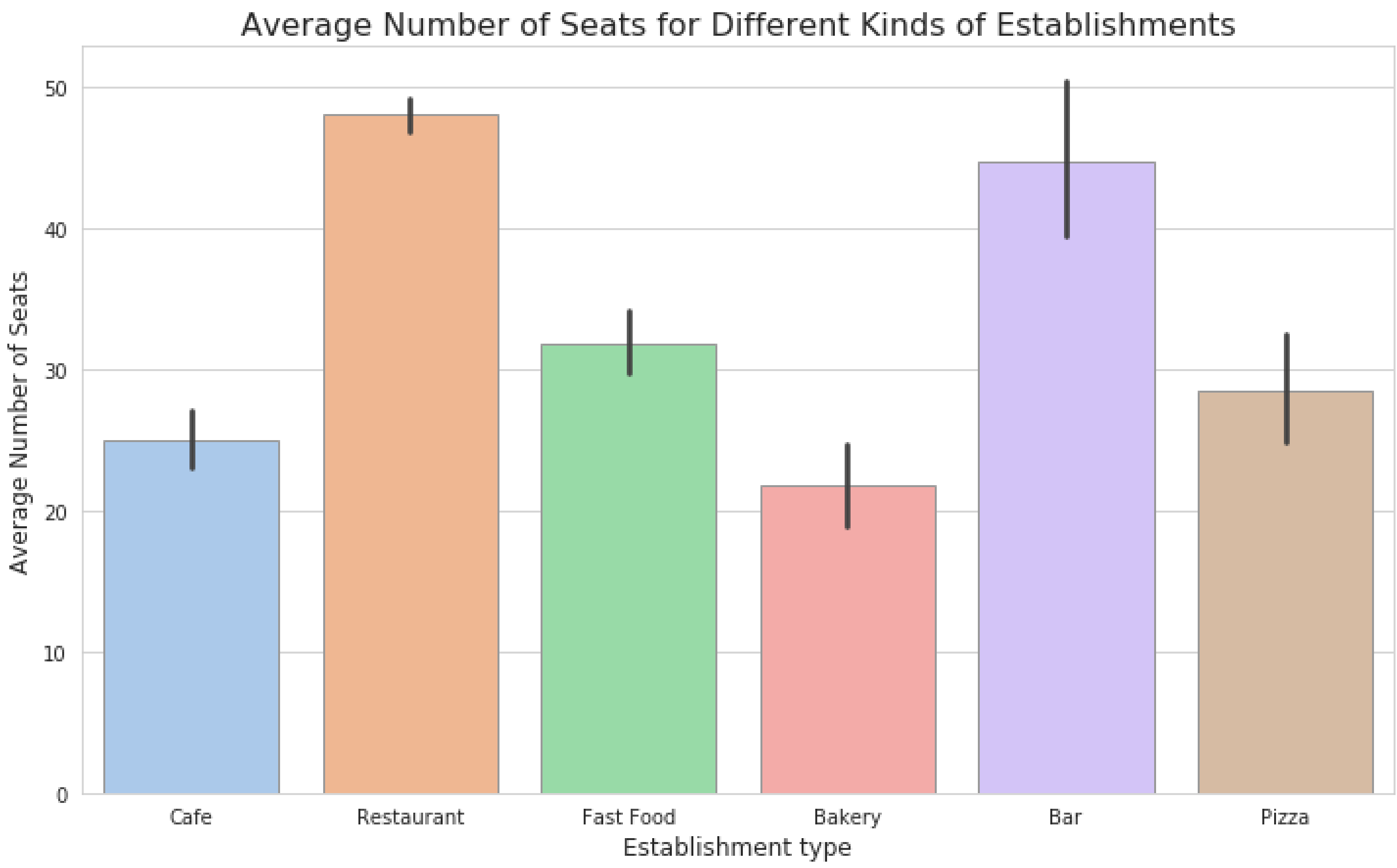


From here I see that most of the restaurants are small chains that have less than 20 branches. But we should also take into considiration that this restaurants are not grouped in the best possible way, and the data may be not entirely accurate. Also most of big chains don't have many seats.

Determine the average number of seats for each type of restaurant. On average, which type of restaurant has the greatest number of seats? Plot graphs.

In [14]:
```
fig, ax = plt.subplots(figsize=(12, 7))
ax.set_title('Average Number of Seats for Different Kinds of Establishments',fontsize=16)
sns.barplot(x='object_type', y='number', data=rest, palette="pastel", edgecolor=".6", ax=ax)
plt.xlabel('Establishment type',fontsize=12)
plt.ylabel('Average Number of Seats',fontsize=12)
display(rest.groupby('object_type')[['number']].mean().reset_index())
```

	object_type	number
0	Bakery	21.773852
1	Bar	44.767123
2	Cafe	25.000000
3	Fast Food	31.837711
4	Pizza	28.459375
5	Restaurant	48.042316



Restaurants seem to have the highest average number of seats, while bakeries show the smallest amount of seats, and that makes sence: people tend not to spend much time in a bakery, but to buy food and drinks to take away.

Put the data on street names from the address column in a separate column.

Let's create a function for getting street name out adress.

In [15]:
```
def get_street(row):
    #function for getting street names and street types from US adresses
    try:
        raw_address=usaddress.tag(row)
        try:
            return raw_address[0]['StreetName'] +" "+raw_address[0]['StreetNamePostType']
        except:
            try: return raw_address[0]['StreetNamePreDirectional']+" "+raw_address[0]['StreetName']
            except:
                try: return raw_address[0]['StreetNamePreType']+" "+raw_address[0]['StreetName']
                except:
                    try: return raw_address[0]['PlaceName']+" "+raw_address[0]['StateName']
                    except: return raw_address[0]['StreetName'];

    except:
        try:
            raw_address = usaddress.parse(row)
            dict_address={}
            for i in raw_address:
                dict_address.update({i[1]:i[0]})
            return dict_address['StreetName'] +" "+dict_address['StreetNamePostType']
        except: return 'not found'
```

In [16]:
```
get_street('OLVERA ST 23')
```

Out[16]: 'OLVERA ST'

In [17]:
```
rest['street'] = rest.address.apply(get_street)
rest
```

Out[17]:

	id	object_name	address	chain	object_type	number	street
0	11786	HABITAT COFFEE SHOP	3708 N EAGLE ROCK BLVD	False	Cafe	26	EAGLE ROCK BLVD
1	11787	REILLY'S	100 WORLD WAY # 120	False	Restaurant	9	WORLD WAY
2	11788	STREET CHURROS	6801 HOLLYWOOD BLVD # 253	False	Fast Food	20	HOLLYWOOD BLVD
3	11789	TRINITI ECHO PARK	1814 W SUNSET BLVD	False	Restaurant	22	SUNSET BLVD
4	11790	POLLEN	2100 ECHO PARK AVE	False	Restaurant	20	ECHO PARK AVE
...	...	...	...	...	...	...	...
9646	21432	HALL OF JUSTICE	217 W TEMPLE AVE	False	Restaurant	122	TEMPLE AVE
9647	21433	FIN-MELROSE	5750 MELROSE AVE	False	Restaurant	93	MELROSE AVE
9648	21434	JUICY WINGZ	6741 HOLLYWOOD BLVD	True	Fast Food	15	HOLLYWOOD BLVD
9649	21435	MEDIDATE COFFEE	548 S SPRING ST STE 100	False	Cafe	6	SPRING ST
9650	21436	CAFE SPROUTS	1300 S SAN PEDRO ST STE 111	True	Restaurant	19	SAN PEDRO ST

9651 rows × 7 columns

In [18]:
```
#check if all addresses were parsed
rest.query('street =="not found"')
```

Out[18]:

id	object_name	address	chain	object_type	number	street

Plot a graph of the top ten streets by number of restaurants.

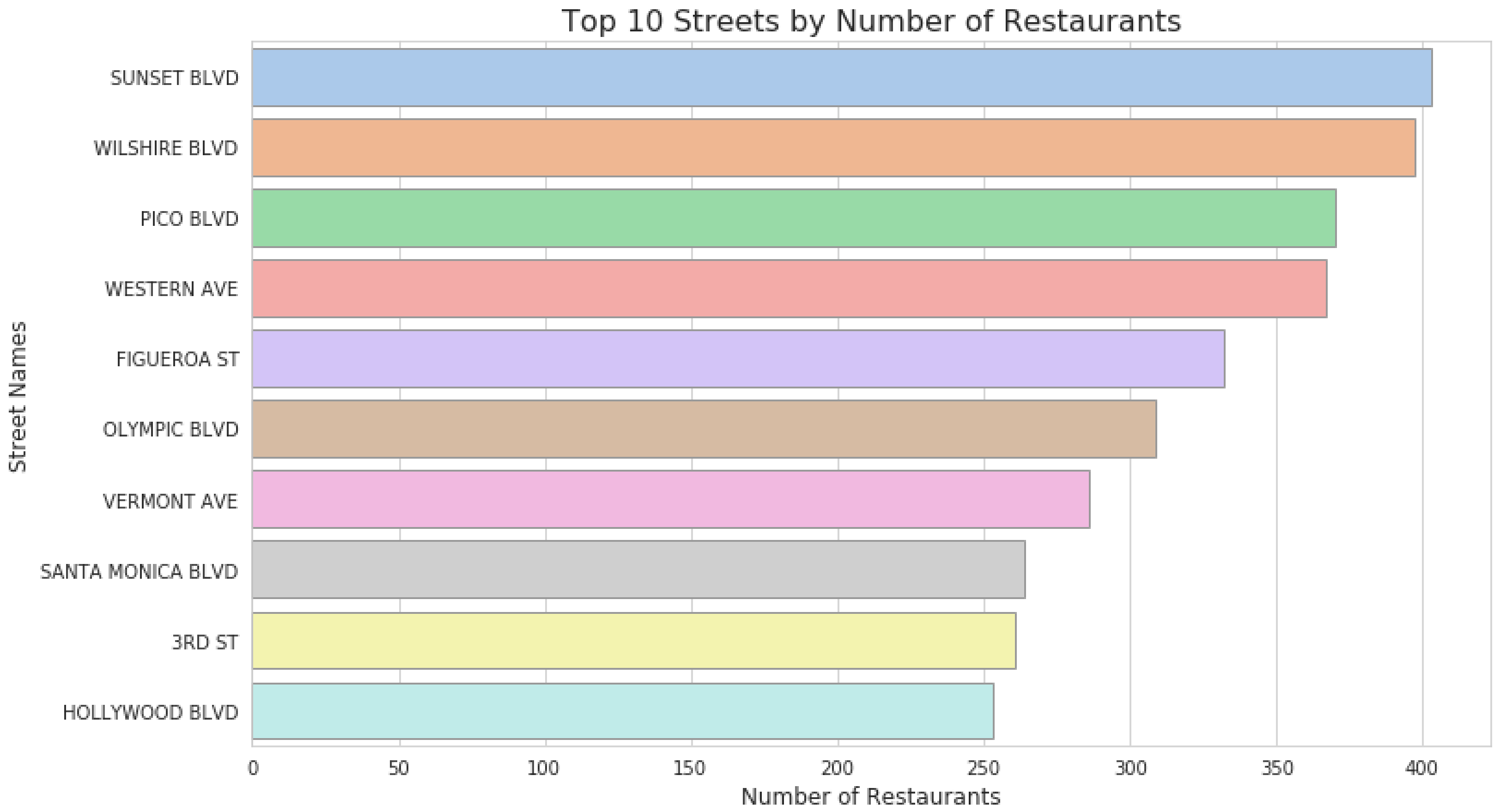
In [19]:
```
rest.groupby('street')[['object_name']].count().reset_index().sort_values('object_name',ascending=False)
```

Out[19]:

	street	object_name
468	SUNSET BLVD	403
538	WILSHIRE BLVD	397
396	PICO BLVD	370
523	WESTERN AVE	367
198	FIGUEROA ST	332
...	...	...
244	HOEFNER AVE	1
240	HILHURST AVE	1
237	HEWITT ST	1
236	HEREFORD DR	1
554	vine ST	1

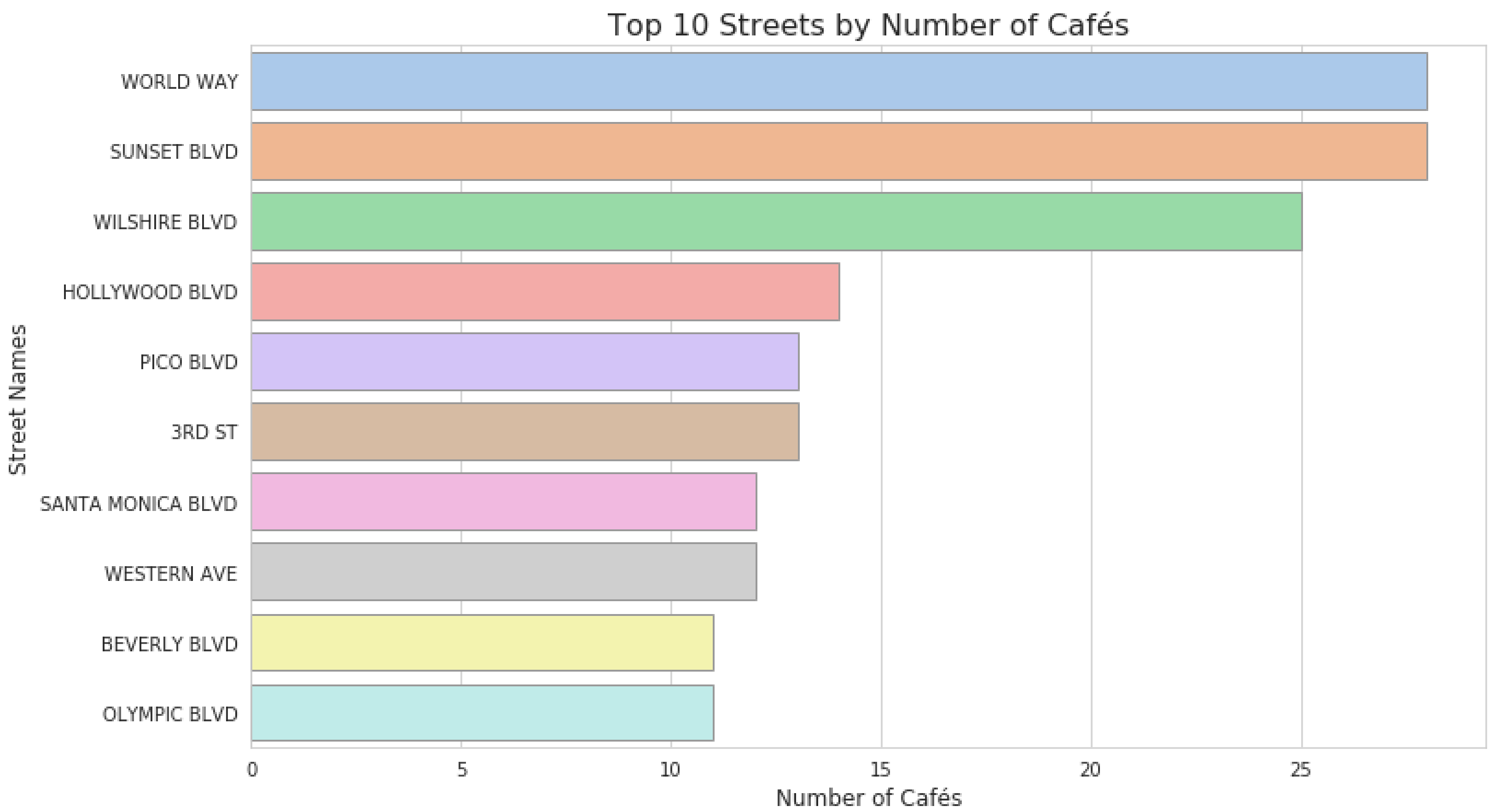
555 rows × 2 columns

In [20]:
```
df = rest.groupby('street')[['object_name']].count().reset_index().sort_values('object_name',ascending=False).head(10)
fig, ax = plt.subplots(figsize=(12, 7))
ax.set_title('Top 10 Streets by Number of Restaurants',fontsize=16)
sns.barplot(y='street', x='object_name', data=df, palette="pastel", edgecolor=".6", ax=ax)
plt.xlabel('Number of Restaurants',fontsize=12)
plt.ylabel('Street Names',fontsize=12)
plt.show()
```



It seems that popular streets tend to have really a lot of restaurants. So it's obvious that people come there a lot, because for such supply there should be simillar demand. Let's also find strets that have lots of cafés.

In [21]:
```
df = (rest.query('object_type == "Cafe"')
      .groupby('street')[['object_name']]
      .count().reset_index()
      .sort_values('object_name',ascending=False).head(10)
     )
fig, ax = plt.subplots(figsize=(12, 7))
ax.set_title('Top 10 Streets by Number of Cafés',fontsize=16)
sns.barplot(y='street', x='object_name', data=df, palette="pastel", edgecolor=".6", ax=ax)
plt.xlabel('Number of Cafés',fontsize=12)
plt.ylabel('Street Names',fontsize=12)
plt.show()
```



Results for cafés differ a little bit from the results for all establishments in general. World Way seem to be the very popular with cafés, while it doesn't have so many restaurants. But still Sunset Boulevard and Wilshire Boulevard are filled with both cafés and other establishments.

Find the number of streets that only have one restaurant.

```
In [22]: one_rest_str = rest.groupby('street')[['id']].count().reset_index().query('id ==1').shape[0]
         print('There are {:.0f} streets in L.A. that have only one restaurant.'.format(one_rest_str))
```

```
There are 251 streets in L.A. that have only one restaurant.
```

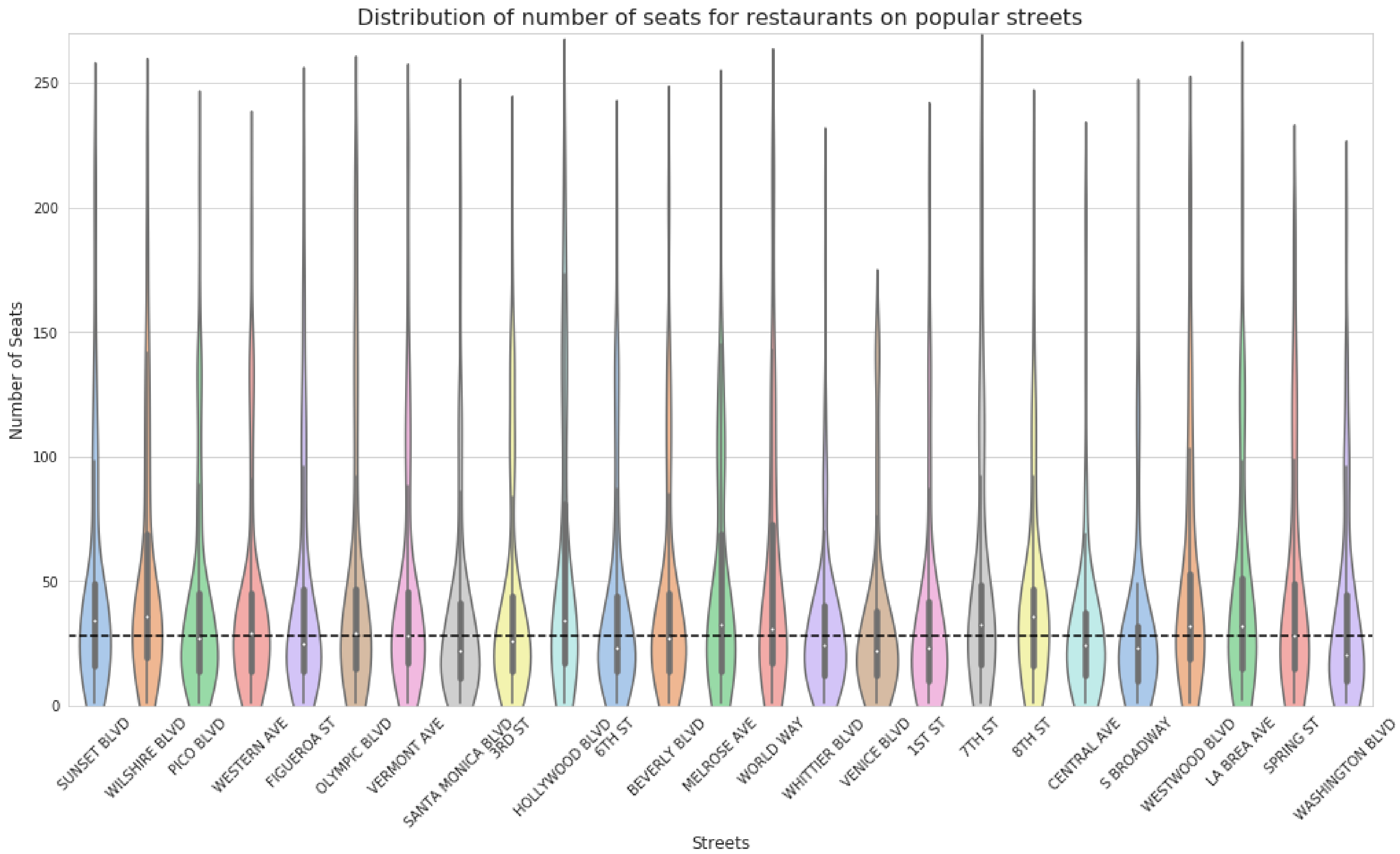
For streets with a lot of restaurants, look at the distribution of the number of seats. What trends can you see?

For this let's get 25 most busy streets.

```
In [23]: busy_streets = rest.groupby('street')['id'].count().sort_values(ascending=False).head(25).index.tolist()
         rest_busy_streets = rest.query('street in @busy_streets')
         rest_busy_streets.number.median()
```

```
Out[23]: 28.0
```

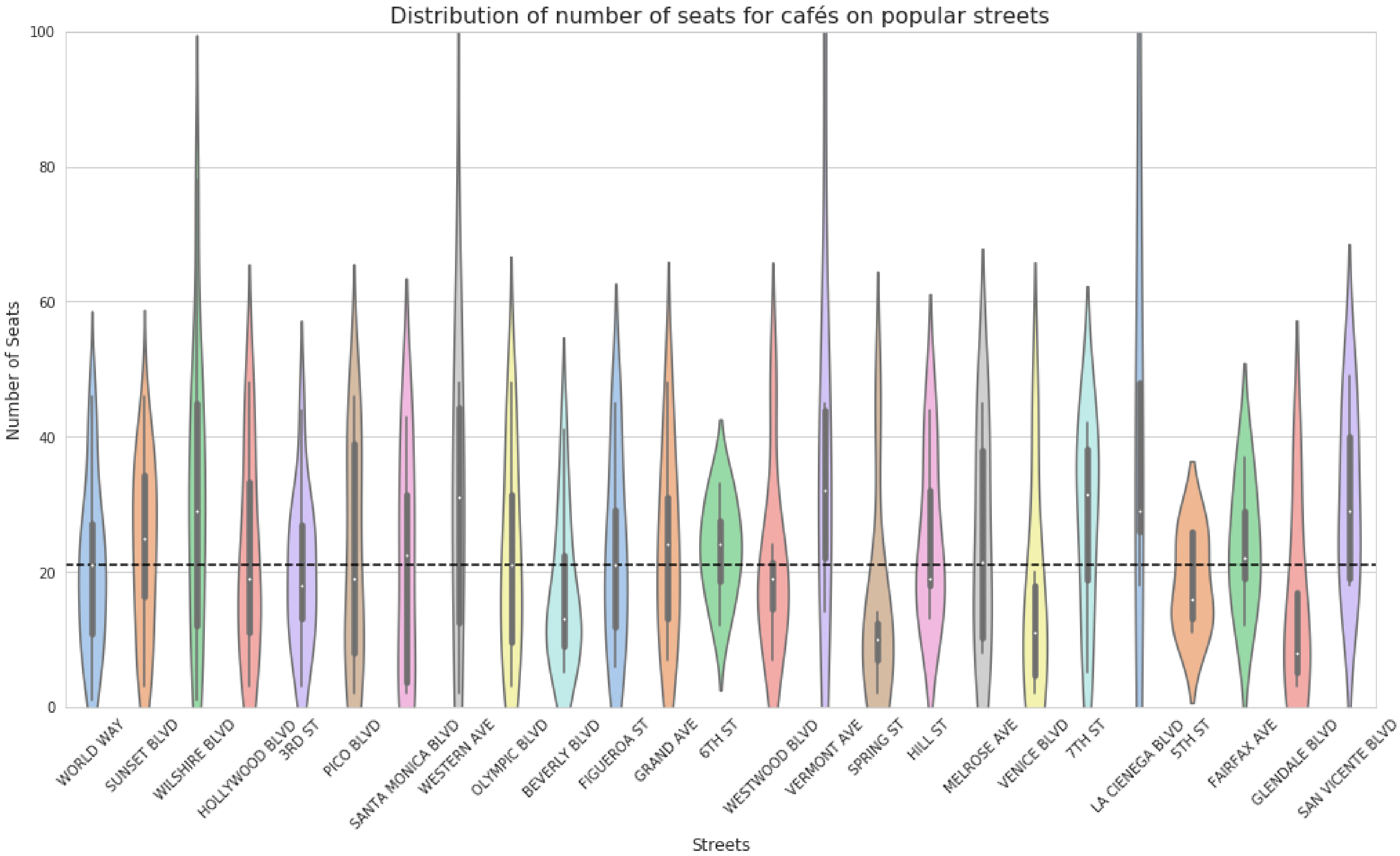
```
In [24]: fig, ax = plt.subplots(figsize=(17, 9))
         sns.violinplot(x='street', y='number',kind="violin",height=16, data=rest_busy_streets, order=busy_streets,
                        palette="pastel", edgecolor=".6", ax=ax)
         ax.set_title('Distribution of number of seats for restaurants on popular streets',fontsize=16)
         plt.xticks(rotation=45)
         plt.ylim(0,270)
         plt.close(2)
         ax.set_xlabel('Streets',fontsize=12)
         ax.set_ylabel('Number of Seats',fontsize=12)
         plt.axhline(y=rest_busy_streets.number.median(), color='black', linestyle='--')
         plt.show()
```



Here we can see that most of the restaurants on popular streets have about 20-30 seats. Seems like it's the best amount of seats for such a crowded place.

Because we are planning on opening a cafe, I'm going to make the same chart but only for cafes on the most popular streets.

In [25]:
```
streets_busy_with_cafes = (rest.query('object_type == "Cafe"')
                          .groupby('street')['id']
                          .count()
                          .sort_values(ascending=False).head(25).index.tolist())
cafes_busy_streets = rest.query('street in @streets_busy_with_cafes and object_type == "Cafe"')
fig, ax = plt.subplots(figsize=(17, 9))
plt.xticks(rotation=45)
plt.ylim(0,100)
ax.set_title('Distribution of number of seats for cafés on popular streets',fontsize=16)
g = sns.violinplot(x='street', y='number',kind="violin",height=16, data=cafes_busy_streets, order=streets_busy_with_ca
fes,
                palette="pastel", edgecolor=".6", ax=ax)
plt.close(2)
ax.set_xlabel('Streets',fontsize=12)
ax.set_ylabel('Number of Seats',fontsize=12)
plt.axhline(y=cafes_busy_streets.number.median(), color='black', linestyle='--')
plt.show()
```



Looks like for cafés average number of seats seem to be even lower than it is for all establishments in general.

Step 3. Prepare a Presentation

Presentation: [Robot Cafe Presentation (https://drive.google.com/file/d/16KU_1Xzs5OBKDml4tDxzVVULcWtYvygt/view?usp=sharing)](https://drive.google.com/file/d/16KU_1Xzs5OBKDml4tDxzVVULcWtYvygt/view?usp=sharing)

In [ ]: