Data Bootcamp Final Project: NYC Film Permits

A film permit for May 14th, 2018 in Manhattan's Gramercy Neighborhood

Author: Matt Robinson

Email: matthew.robinson@stern.nyu.edu

NYU Students frequently pass by film sets while walking around Greenwich Village and the surrounding area. Actor & actress trailers can be found on a regular basis, and brands such as Cadillac are known to frequently shoot in the SoHo area. In 2010, then-Mayor Michael Bloomberg enacted a $300 processing fee (https://www.telegraph.co.uk/culture/film/film-news/7638566/New-York-to-charge-300-for-filming-permits.html) for film permits. This was met with some backlash, but nevertheless, NYC has remained a hot location to film.

NYC has a vibrant arts scene, and that includes film & television. Since January 1st 2012 there have been over 49,000 film permits filed with the NYC Mayor's office. That's an average of over 20 per day. Because Lower Manhattan is regarded as a very affluent area within NYC, one might hypothesize that this means that more filming is done in this area.

With that in mind, this project will explore the question: **Where are people filming in Manhattan?**

More specifically, this project will explore at a zip code level what the most popular zip codes are for filming. This will be done year-to-year. Furthermore, knowing how different seasons provide different opportunities to film, this project will also see how much shifting there is between popular zip codes by season. Going back to the hypothesis that filming is being predominantly done in higher-income areas, this project will look at how strong of a correlation there is—if any—between zip code income and amount of permits. Finally, this project will evaluate shifts in the data, comparing the first three years of data to the most recent three years. This will provide a glimpse into which areas are growing and declining in popularity.

This project will utilize the database of NYC film permits (https://data.cityofnewyork.us/City-Government/Film-Permits/tg4x-b46p/data), sourced from the NYC Open Data website. This database is updated frequently, and will still work with the setup below barring any unusual changes.

DataFrames

This project will create 3 primary DataFrames.

1. **film_Manhattan_refined** is a DataFrame that will contain all the permit entries being analyzed. Data from this DataFrame will be fed into the next two DataFrames.
2. **sum_counts_manhattan** is a *GeoDataFrame* that maintains geometries for Manhattan zip codes and corresponding figures for yearly and seasonal permit data. This GeoDataFrame will be used to map in conjunction with the GeoPandas package.
3. **df_sum_counts** is a DataFrame version of the second GeoDataFrame that does not contain zip code geometries. It will be used to generate visualizations other than maps.

In addition to these 3 DataFrames, numerous helper DataFrames will also be created.

Important Notes before running:

In addition to the standard packages that should already be installed, there are two more resources required:

1. Please download the shapefile for NYC (https://data.cityofnewyork.us/download/i8iw-xf4u/application%2Fzip) and save it in your current working directory. This file can also be found in the Github repository.
2. Please also install the Seaborn package by entering "pip install seaborn" in the Terminal window.

Data Packages

This project will use a variety of packages to manipulate data and produce insightful graphics. A brief description of how each package will be used is given.

In [4982]:
```
import pandas as pd #main package for dataset and visualizations
import matplotlib.pyplot as plt #visualization package
import numpy as np #for numerical functions
import os #to access files from the computer

from IPython.display import display, Image #display package

import requests #for accessing NYC Open Data API

from census import Census #this is the most up-to-date U.S. census data
from us import states #returns census data at a more local level

import fiona #package required for mapping
import geopandas as gpd #main mapping tool
from shapely.geometry import Point, Polygon #for mapping, this package w
ill plot shapes

import seaborn as sns #for regression plot visualization
import matplotlib.ticker as mtick #for axes unit formatting on plots
```

In [4983]:
```
#not a data package, but an important switch
#It looks more professional to hide these warnings once it is known that
 the entire notebook functions properly
pd.options.mode.chained_assignment = None #setting this to "warn" or com
menting out this line will enable warnings
```

Fetch the Film Dataset

Create a DataFrame from the raw film permit data.

In [4984]:
```
#This is the url location where the data's API lives
#A $limit modifier was added to this SODA API to request more than the d
efault 1,000 items
data_url = 'https://data.cityofnewyork.us/resource/6aka-uima.json?$limit
=100000'
r = requests.get(data_url)

df_raw_permits = pd.DataFrame(r.json())

#make sure ~50,000 rows and 14 columns are imported
df_raw_permits.shape
```

Out[4984]: (49063, 14)

Get rid of unnecessary columns and create better column headers for those that remain.

In [4985]:
```
#drop irrelevant columns
df_raw_permits.drop(['startdatetime','policeprecinct_s','country','event
agency',
                     'enteredon','communityboard_s'], axis = 1, inplace
 = True)

#properly title remaining ones
df_raw_permits.columns = ["Borough", "Category", "End","ID","Type",
                          "Streets","Subcategory","Zips"]

#convert the date column to a datetime
df_raw_permits['Datetime']=pd.to_datetime(df_raw_permits['End'])

#we don't care about the time the permit was entered
df_raw_permits['Date'] = df_raw_permits['Datetime'].dt.date

#now that the date has been reformatted, we can drop these extra columns
df_raw_permits.drop(['Datetime','End'], axis=1, inplace = True)

#rename
df_raw_permits = df_raw_permits[["ID","Date","Zips","Borough","Type",
                                 "Category","Subcategory","Streets"]]
```

Identify the season of each entry

The following Python function converts a date into whatever day of the year it was. There are established ranges for which day of the year is what season.

In [4986]:
```
def season(date):

    #days of year that constitue each season
    #winter is excluded because we can consider it to be 'everything els
e'
    spring = range(79, 172)
    summer = range(173, 264)
    fall = range(265, 355)

    #convert to the day of the year type
    to_date = pd.to_datetime(date)
    day_of = to_date.dayofyear

    #return corresponding season
    if day_of in spring:
        return 'Spring'
    elif day_of in summer:
        return 'Summer'
    elif day_of in fall:
        return 'Autumn'
    else:
        return 'Winter'
```

Going row by row, each date is converted into a season object which is appended to the DataFrame as a new column.

In [4987]:
```
df_raw_permits['Season'] = df_raw_permits['Date'].apply(lambda x: season
(x))
```

Clean DataFrame and manipulate zip code column

Some entries have multiple zip codes listed. To dfeal with this problem, a helper DataFrame is generated. This DataFrame will take the unique ID value of each permit and the list of zip codes associated. The zip code list will then be divided up into individual lines, and reworked back into the main DataFrame. The result of this, is that each entry will hold only one zip code. An entry that originally held more than one zip code will be duplicated into one entry per zip code.

In [4988]:
```
Zip_Codes = df_raw_permits["Zips"].str.split(', ', expand = True)

Zip_Codes.replace(to_replace = [None], value = np.nan, inplace = True)

Zip_Codes = Zip_Codes.join(df_raw_permits["ID"])

Zip_Codes = Zip_Codes.melt(id_vars = "ID")

Zip_Codes.astype(int, errors = "ignore")

film_permits = pd.merge(df_raw_permits, Zip_Codes, on = "ID", how = "out
er")

film_permits.dropna(inplace=True)
```

Make the table look nice again: re-order, drop some columns, and rename for consistency. The DataFrame is organized and previewed below.

In [4989]:
```
film_permits.sort_values(by=['Date','ID'], ascending = False, inplace = True)

film_permits.drop(['Zips','variable'], axis = 1, inplace = True)

film_permits.rename(columns={"value": "Zip Code"}, inplace=True)

film_permits.reset_index(drop=True, inplace=True)

film_permits.head(5)
```

Out[4989]:

	ID	Date	Borough	Type	Category	Subcategory	Streets	Season	Co
0	395641	2018-02-24	Manhattan	Shooting Permit	Television	Not Applicable	AMSTERDAM AVENUE between WEST 62 STREET and ...	Winter	100
1	393730	2018-02-17	Brooklyn	Shooting Permit	Television	Variety	NOBLE STREET between WEST STREET and DEAD END	Winter	112
2	396378	2018-02-16	Manhattan	Theater Load in and Load Outs	Theater	Theater	WEST 52 STREET between 6 AVENUE and 7 AVENUE	Winter	100
3	396868	2018-02-14	Queens	Shooting Permit	Television	Episodic series	SKILLMAN AVENUE between QUEENS BOULEVARD and 3...	Winter	111
4	396868	2018-02-14	Queens	Shooting Permit	Television	Episodic series	SKILLMAN AVENUE between QUEENS BOULEVARD and 3...	Winter	112

Entries where the zip code entry is not a number need to be thrown out. All zip codes need to be number objects.

Some zip codes are errantly input as '0' or '83.' Eliminating all rows witha zip code value under 10000 will ensure none of these rogue zip codes stay.

In [4990]:
```
film_permits['Zip Code'] = pd.to_numeric(film_permits['Zip Code'], error
s='coerce')

film_permits = film_permits[film_permits['Zip Code'].map(abs) > 9999]
```

Isolate the relevant Manhattan permits into DataFrame #1

Get all the Manhattan entries

In [4991]:
```
#each film permit entry is classified by borough
film_borough = film_permits.groupby(["Borough"])

film_Manhattan = film_borough.get_group('Manhattan')
```

The next filtering to be done is to sort out the different groups. Theater load in and load outs are not relevant to this project. Shooting, rigging and DCAS are (DCAS is the type of permit used to film inside a municipal building in NYC).

In [4992]:
```
#These are the different types of permits in the system.
film_Manhattan['Type'].value_counts()
```

Out[4992]:
```
Shooting Permit                   33396
Theater Load in and Load Outs      4316
Rigging Permit                      815
DCAS Prep/Shoot/Wrap Permit         607
Name: Type, dtype: int64
```

Bring the relevant groups together to create a new DataFrame. A **shooting permit** is a standard filming permit, and the most popular kind. A **rigging permit** is slightly different, but for a similar purpose. A **DCAS permit** is for filming in municipal areas. And finally, a **theater load** permit is for moving equipment into and out of venues. The last category will not be used in this project.

In [4993]:
```
film_Manhattan_types = film_Manhattan.groupby(['Type'])

film_Manhattan_sp = film_Manhattan_types.get_group('Shooting Permit')
film_Manhattan_dcas = film_Manhattan_types.get_group('DCAS Prep/Shoot/Wr
ap Permit')
film_Manhattan_rp = film_Manhattan_types.get_group('Rigging Permit')

#add the three groups together into one DataFrame
#This is the first DataFrame as mentioned in the overview
film_Manhattan_refined = pd.concat([film_Manhattan_sp, film_Manhattan_dc
as, film_Manhattan_rp])
```

Incorporate Zip Code Shape and Income Data

Use the American Community Survey from the U.S. census to get median income figures

This section will use the U.S. census package to import median income for Manhattan zip codes. Additionally, it will use a shapefile for NYC sourced from NYC's official website. ***If you are not on a Mac, proceed carefully through the next step.***

In [4994]:
```
#this is my private api access key--use it responsibly
api_key = '2053721469a476f018488f71ca73ba767d8670e8'

#get the current computer file directory location
cwd = os.getcwd()

#WARNING: if you are using a PC, execute the second line instead:
nyc_shape = cwd + "/ZIP_CODE_040114/ZIP_CODE_040114.shx"
#nyc_shape = cwd + "\\shape_files\\NYC\\ZIP_CODE_040114.shx"
```

Fix and clean the resulting raw DataFrame with proper zip code labelling. Manhattan is not just a borough, it's also New York County. The state, city, and county all bear the same name in Manhattan. This isn't found anywhere else in the US.

In [4995]:
```
#read in the shapefile
nyc_map = gpd.read_file(nyc_shape)

#PO_NAME is county
boroughs = nyc_map.groupby(["PO_NAME"])

#New York County is Manhattan
manhattan_map = boroughs.get_group("New York")

#convert zip codes to integers and operate to get a list for further use
manhattan_map.ZIPCODE = nyc_map.ZIPCODE.astype(int)

manhattan_zip_codes = manhattan_map.ZIPCODE.tolist()

manhattan_zip_codes = "".join(str(manhattan_zip_codes))

manhattan_zip_codes = manhattan_zip_codes[1:-1]

manhattan_map.rename(columns={"ZIPCODE": "Zip Code"}, inplace=True)
```

In [4996]:
```
#these are the variables to use for getting median income data for each
 "zip code tabulation area"
code = ("NAME","B19013_001E")

manhattan_zip_data = pd.DataFrame(census.acs5.get(
    code, {'for': 'zip code tabulation area:' + manhattan_zip_codes }, y
ear=2015))

#rename coded income name
manhattan_zip_data.rename(columns={"B19013_001E": "Income"}, inplace=Tru
e)

#NaN income figures are for P.O. zip codes
manhattan_zip_data.dropna(axis=0, how = 'any', inplace=True)

#drop and reformat as needed
manhattan_zip_data.drop(['NAME'], axis=1, inplace = True)

manhattan_zip_data.columns = ['Income', 'Zip Code']

manhattan_zip_data['Zip Code'] = manhattan_zip_data['Zip Code'].astype(i
nt)
```

Generate GeoDataFrame

Create a new GeoDataFrame with only the needed elements: zip code, income, and the shape info. This has to be done by merging with the GeoDataFrame before extracting the relevant columns, simply merging to the manhattan_zip_data DataFrame would mean the geometry column is no longer part of a GeoDataFrame, and will no longer plot.

A preview of the new GeoDataFrame is shown below.

In [4997]:
```
#make zip codes from the film dataset into integers for the merging
film_Manhattan_refined['Zip Code'].astype(int, copy = False, errors='ign
ore', inplace=True)

#bring income onto the current GeoDataFrame
manhattan_map = manhattan_map.merge(manhattan_zip_data[['Zip Code','Inco
me']], on = ['Zip Code'])

#create new GeoDataFrame to be used throughout the remainder of the proj
ect
sum_counts_manhattan = manhattan_map[['Zip Code','Income','geometry']]

#zip code 10162 is a P.O. zip code that arroneously has an income value
sum_counts_manhattan = sum_counts_manhattan[sum_counts_manhattan['Zip Co
de'].map(abs) != 10162]

sum_counts_manhattan.head(5)
```

Out[4997]:

	Zip Code	Income	geometry
0	10034	42581.0	POLYGON ((1006203.169008225 257345.6566181332,...
1	10033	44933.0	POLYGON ((1003020.86467731 256049.1618358046, ...
2	10040	46200.0	POLYGON ((1002564.135275811 253724.7731118798,...
3	10032	37280.0	POLYGON ((998935.8528342247 249154.259943217, ...
4	10031	39774.0	POLYGON ((1000830.038678393 241801.5896092206,...

Cleaning the GeoDataFrame and adding the permit totals

Create a DataFrame with the number of times each zip code is found in the DataFrame. There are some entries that are non-Manhattan zip codes due to issues in the data. However, these counts will not be included in the resulting DataFrame because it will be merged onto the manhattan_info DataFrame which will not contain those zip codes.

In [4998]:
```
#counts the number of occurrences of each zip code in the first DataFram
e which listed all the permits
zip_counts = film_Manhattan_refined['Zip Code'].value_counts()

#generate a DataFrmae
df_zip_counts = zip_counts.rename_axis('Zip Code').reset_index(name = 'P
ermits')

#merge the count totals to match their respective zip codes on the GeoDa
taFrame
sum_counts_manhattan = sum_counts_manhattan.merge(df_zip_counts[['Zip Co
de','Permits']], on = ['Zip Code'])
```

DataFrame from GeoDataFrame above

In [4999]:
```
#pull the necessary columns
df_sum_counts = sum_counts_manhattan[['Zip Code', 'Income','Permits']]
```

Since this DataFrame will not be used for mapping, the three additional rows for 10004 and the one additional for 10035 can be dropped. These zip codes are not contiguous so they require multiple shape geometries, thus taking up multiple lines in the GeoDataFrame.

A preview of the DataFrame is shown below.

In [5000]:
```
#get rid of duplicate zip codes from GeoDataFrame
df_sum_counts.drop_duplicates(subset=['Zip Code'], inplace=True)

df_sum_counts.head(5)
```

Out[5000]:

	Zip Code	Income	Permits
0	10034	42581.0	97
1	10033	44933.0	232
2	10040	46200.0	92
3	10032	37280.0	379
4	10031	39774.0	423

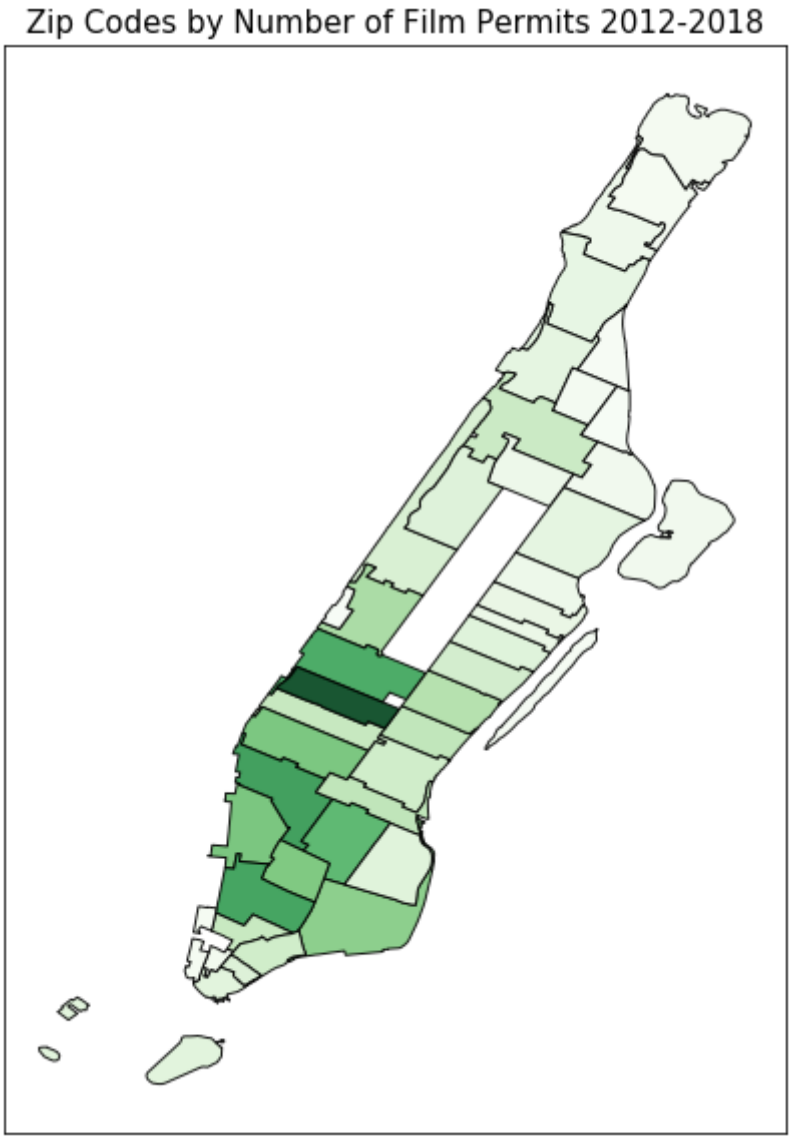
Visualizing the Data

Overall data

First, we'll look at the data in its entirety—the permit totals across all the data.

The map below demonstrates how many film permits are from each zip code overall. The darker the zip code, the more permits it has.

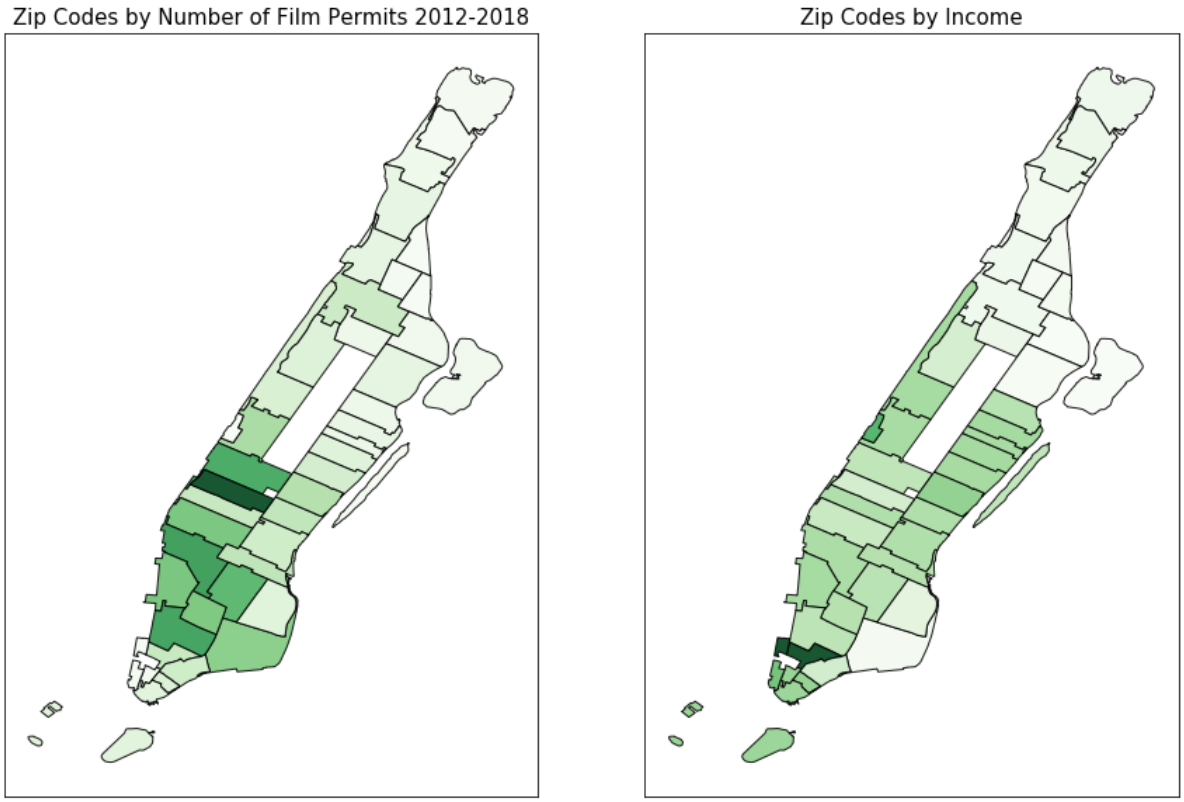
In [5001]:
```
#we'll use the fig-ax method to plot throughout
fig, ax = plt.subplots(figsize = (7,10))

#plot the permits by zip code
#a green color scale looks nice
sum_counts_manhattan.plot(ax = ax, column='Permits', edgecolor='black',
cmap='Greens', alpha = .9)

#hide axes for picture frame look
ax.get_xaxis().set_visible(False)
ax.get_yaxis().set_visible(False)

#make a title and size
ax.set_title('Zip Codes by Number of Film Permits 2012-2018')
ax.title.set_size(15)

#display
plt.show()
```



We can see that the film permits are concentrated in lower Manhattan. This aligns with the hypothesis that film permits might be correlated to income levels. Let's pull up an income map and compare.

Permit map versus income map

In [5002]:
```
fig, ax = plt.subplots(1,2,figsize = (15,10))

######Left Graph: Plot total permits on each zip code
sum_counts_manhattan.plot(ax = ax[0], column='Permits', edgecolor='blac
k', cmap='Greens', alpha = 0.9)

#hide axes, just leave it as a 'picture frame'
ax[0].get_xaxis().set_visible(False)
ax[0].get_yaxis().set_visible(False)

#create a title
ax[0].set_title('Zip Codes by Number of Film Permits 2012-2018')
ax[0].title.set_size(15)

######Right Graph: Plot median income on each zip code
sum_counts_manhattan.plot(ax = ax[1], column='Income', edgecolor='black'
, cmap='Greens', alpha = 0.9)

#hide axes, just leave it as a 'picture frame'
ax[1].get_xaxis().set_visible(False)
ax[1].get_yaxis().set_visible(False)

#create a title
ax[1].set_title('Zip Codes by Income')
ax[1].title.set_size(15)


plt.show()
```

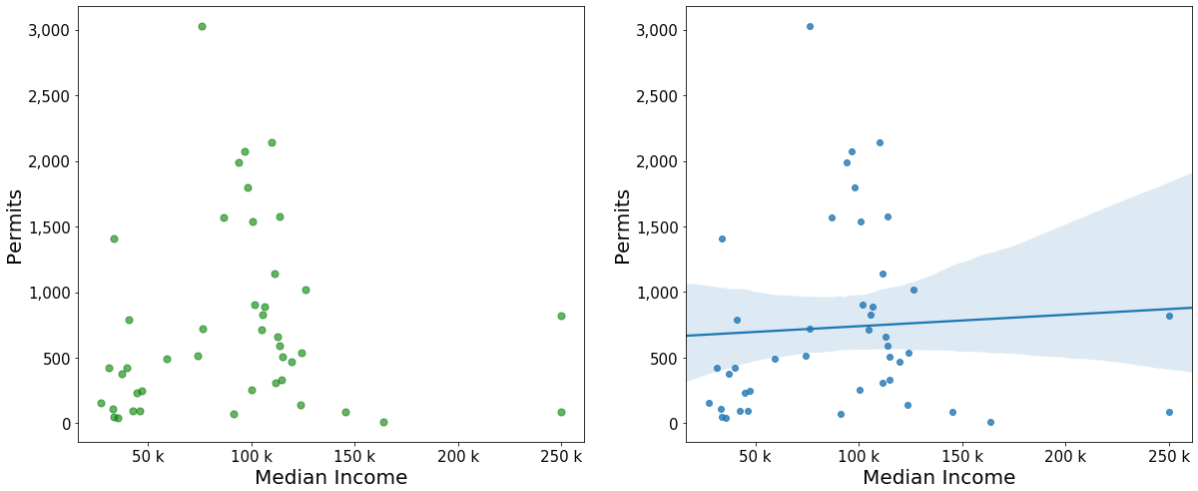


It might not be as correlated as previously thought. Regardless, considering how the lighter sections on both seem concentrated at the top of Manhattan, there is still a chance that the data is correlated.

Now, we will plot median income against number of permits for each zip code. This section will also utilize the seaborn package to generate a regression plot alongside the scatter plot.

Permit versus income scatter and regression plots

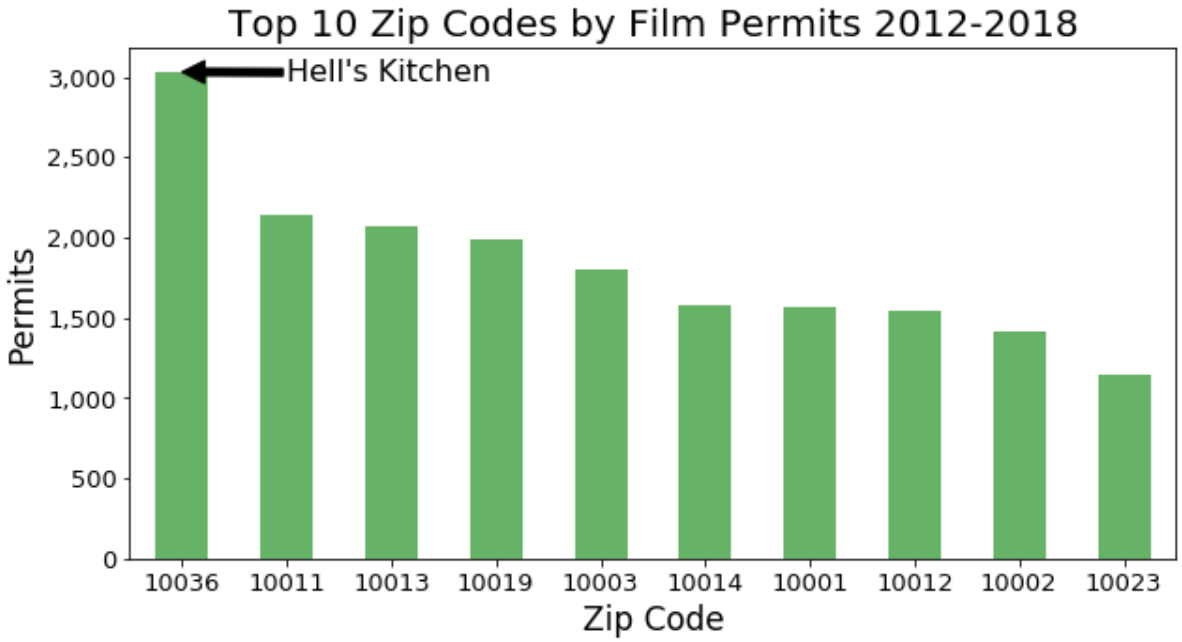
In [5003]:
```
fig,ax = plt.subplots(1,2,figsize=(20,8))

#plot each zip code's median income against its number of permits on the
 left
df_sum_counts.plot(ax=ax[0],kind='scatter',x='Income',y='Permits',s=50,
color='green',alpha=0.6)

#Set the axes labels and increase their size
ax[0].set_xlabel('Median Income',fontsize=20)
ax[0].set_ylabel('Permits',fontsize=20)

#increase the size of the axes units
ax[0].tick_params(axis='x',labelsize=15)
ax[0].tick_params(axis='y',labelsize=15)

#properly format the axes labels
ax[0].xaxis.set_major_formatter(mtick.EngFormatter())
ax[0].yaxis.set_major_formatter(mtick.StrMethodFormatter('{x:,.0f}'))

#use the special plotting data package to make a regression plot on the
 right
sns.regplot(x="Income",y="Permits", data=df_sum_counts,ax=ax[1])

#Set the axes labels and increase their size
ax[1].set_xlabel('Median Income',fontsize=20)
ax[1].set_ylabel('Permits',fontsize=20)

#increase the size of the axes units
ax[1].tick_params(axis='x',labelsize=15)
ax[1].tick_params(axis='y',labelsize=15)

#properly format the axes labels
ax[1].xaxis.set_major_formatter(mtick.EngFormatter())
ax[1].yaxis.set_major_formatter(mtick.StrMethodFormatter('{x:,.0f}'))

plt.show()
```



The result is obvious now. There is not much correlation *at all* between median income and number of permits per zip code. The regression plot on the right demonstrates how insignificant the correlation really is.

Moving on from the fact that the hypothesized relationship between income and permit quantity doesn't exist, let's look at what zip codes are the most popular. The top 10 are plotted below.

In [5004]:
```
fig, ax = plt.subplots(figsize = (10,5))

#sort the DataFrame by number of overall permits
df_sum_counts.sort_values('Permits', ascending = False, inplace = True)
#plot only the top 10 values (indices 0-10 exclusive)
df_sum_counts.iloc[:10].plot.bar(ax=ax, x='Zip Code', y='Permits', color
='green',alpha=0.6,legend=False)

#set title and font size
ax.set_title('Top 10 Zip Codes by Film Permits 2012-2018')
ax.title.set_size(20)

#set axes labels & font sizes
ax.set_xlabel('Zip Code',fontsize=17)
ax.set_ylabel('Permits',fontsize=17)

#format axes units
ax.tick_params(axis='x', rotation=0, labelsize=13)
ax.tick_params(axis='y', rotation=0, labelsize=13)
ax.yaxis.set_major_formatter(mtick.StrMethodFormatter('{x:,.0f}'))

#put a note to identify the outlier
ax.annotate('Hell\'s Kitchen',xy=(0, 3030),xytext=(1, 2975),arrowprops=d
ict(facecolor='black'),fontsize=16)

plt.show()
```



This does not mean much yet, but this gives us an idea of what those darkest zip codes were. We can identify the outlier, **10036**, as the Hell's Kitchen neighborhood. In the following sections if these zip codes are the same at a finer level: year to year and season to season.

Year by year data

This section will dig deeper and go year by year to find how consistent or inconsistent film permit concentration has been.

Including yearly totals

Now, the GeoDataFrame will be extended to include yearly totals for each zip code.

In [5005]:
```
#this function will read each line item's date, which is a DateTime obje
ct, and group by year
year_films = film_Manhattan_refined.groupby(lambda x: film_Manhattan_ref
ined['Date'][x].year)
```

In [5006]:
```
#generate DataFrames for each year
df_2012_films = year_films.get_group(2012)
df_2013_films = year_films.get_group(2013)
df_2014_films = year_films.get_group(2014)
df_2015_films = year_films.get_group(2015)
df_2016_films = year_films.get_group(2016)
df_2017_films = year_films.get_group(2017)
df_2018_films = year_films.get_group(2018)
```

In [5007]:
```
#sum up instances for each zip code in each year's
zip_counts_2012 = df_2012_films['Zip Code'].value_counts()
zip_counts_2013 = df_2013_films['Zip Code'].value_counts()
zip_counts_2014 = df_2014_films['Zip Code'].value_counts()
zip_counts_2015 = df_2015_films['Zip Code'].value_counts()
zip_counts_2016 = df_2016_films['Zip Code'].value_counts()
zip_counts_2017 = df_2017_films['Zip Code'].value_counts()
zip_counts_2018 = df_2018_films['Zip Code'].value_counts()
```

In [5008]:
```
#generate fresh DataFrames for each year's zip code count
df_2012_counts = zip_counts_2012.rename_axis('Zip Code').reset_index(nam
e = '2012')
df_2013_counts = zip_counts_2013.rename_axis('Zip Code').reset_index(nam
e = '2013')
df_2014_counts = zip_counts_2014.rename_axis('Zip Code').reset_index(nam
e = '2014')
df_2015_counts = zip_counts_2015.rename_axis('Zip Code').reset_index(nam
e = '2015')
df_2016_counts = zip_counts_2016.rename_axis('Zip Code').reset_index(nam
e = '2016')
df_2017_counts = zip_counts_2017.rename_axis('Zip Code').reset_index(nam
e = '2017')
df_2018_counts = zip_counts_2018.rename_axis('Zip Code').reset_index(nam
e = '2018')
```

In [5009]:
```
#extend the GeoDataFrame year by year with the corresponding annual sums
sum_counts_manhattan = sum_counts_manhattan.merge(df_2012_counts[['Zip C
ode','2012']],how='left',on = ['Zip Code'])
sum_counts_manhattan = sum_counts_manhattan.merge(df_2013_counts[['Zip C
ode','2013']],how='left',on = ['Zip Code'])
sum_counts_manhattan = sum_counts_manhattan.merge(df_2014_counts[['Zip C
ode','2014']],how='left',on = ['Zip Code'])
sum_counts_manhattan = sum_counts_manhattan.merge(df_2015_counts[['Zip C
ode','2015']],how='left',on = ['Zip Code'])
sum_counts_manhattan = sum_counts_manhattan.merge(df_2016_counts[['Zip C
ode','2016']],how='left',on = ['Zip Code'])
sum_counts_manhattan = sum_counts_manhattan.merge(df_2017_counts[['Zip C
ode','2017']],how='left',on = ['Zip Code'])
sum_counts_manhattan = sum_counts_manhattan.merge(df_2018_counts[['Zip C
ode','2018']],how='left',on = ['Zip Code'])
```

In [5010]:
```
#For years that don't have any permits, replace the default NaN value wi
th zero
sum_counts_manhattan.fillna(value=0,inplace=True,downcast='int')
sum_counts_manhattan = sum_counts_manhattan.astype(int,errors='ignore',i
nplace=True)
```

In [5011]:
```
#Update the DataFrame with the new year columns from the GeoDataFrame
df_sum_counts = df_sum_counts.merge(sum_counts_manhattan
                [['Zip Code','2012','2013','2014','2015','2016','2017',
'2018']], how='left',on = ['Zip Code'])
df_sum_counts.drop_duplicates(subset=['Zip Code'],inplace=True)
df_sum_counts = df_sum_counts.astype(int,errors='ignore',inplace=True)
```

Here is a glimpse of what the GeoDataFrame and DataFrame look like now that yearly sum columns have been added to each zip code.

In [5012]:
```
sum_counts_manhattan.head(5)
```

Out[5012]:

	Zip Code	Income	geometry	Permits	2012	2013	2014	2015	2016	2017	20
0	10034	42581	POLYGON ((1006203.169008225 257345.6566181332,...	97	18	11	13	24	16	15	0
1	10033	44933	POLYGON ((1003020.86467731 256049.1618358046, ...	232	43	34	40	42	41	31	1
2	10040	46200	POLYGON ((1002564.135275811 253724.7731118798,...	92	6	9	20	16	21	18	2
3	10032	37280	POLYGON ((998935.8528342247 249154.259943217, ...	379	63	29	74	101	61	47	4
4	10031	39774	POLYGON ((1000830.038678393 241801.5896092206,...	423	52	93	60	75	59	72	12

In [5013]:
```
df_sum_counts.head(5)
```

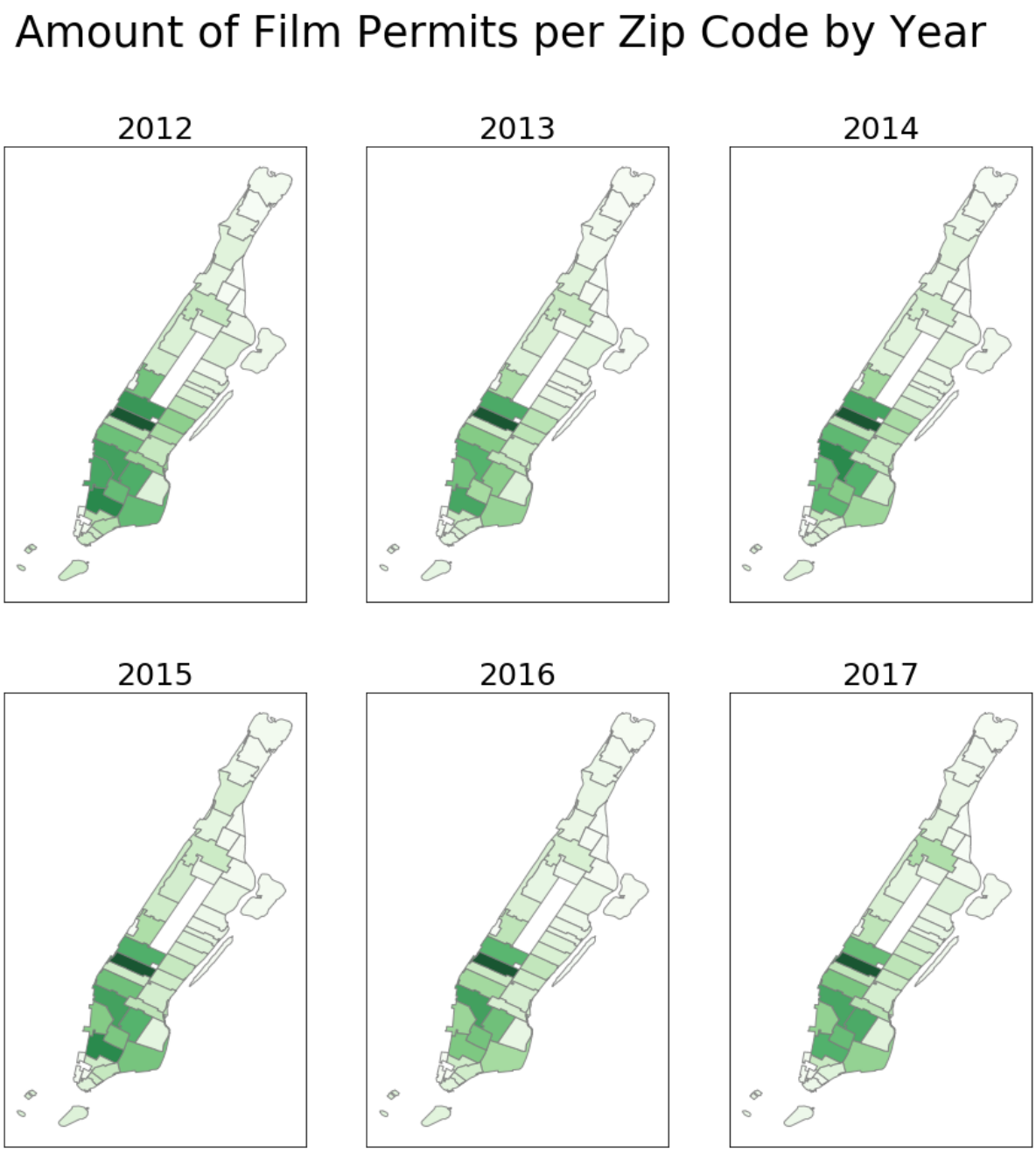
Out[5013]:

	Zip Code	Income	Permits	2012	2013	2014	2015	2016	2017	2018
0	10036	75966	3030	396	573	494	535	524	478	30
1	10011	109818	2142	280	358	398	376	371	325	34
2	10013	96667	2075	323	391	299	432	269	303	58
3	10019	94022	1987	296	381	342	343	320	274	31
4	10003	98151	1801	257	270	309	333	289	317	26

Mapping yearly totals

The following graphs each year 2012-2017. This year's data is not up-to-date and therefore does not have enough data to be statistically significant enough. Thus, 2018 is excluded.

In [5014]:
```
fig, ax = plt.subplots(2,3, figsize = (15,15))

#create maps for each year
sum_counts_manhattan.plot(ax = ax[0][0], edgecolor='tab:grey', column='2
012', cmap='Greens', alpha = 0.9)
sum_counts_manhattan.plot(ax = ax[0][1], edgecolor='tab:grey', column='2
013', cmap='Greens', alpha = 0.9)
sum_counts_manhattan.plot(ax = ax[0][2], edgecolor='tab:grey', column='2
014', cmap='Greens', alpha = 0.9)
sum_counts_manhattan.plot(ax = ax[1][0], edgecolor='tab:grey', column='2
015', cmap='Greens', alpha = 0.9)
sum_counts_manhattan.plot(ax = ax[1][1], edgecolor='tab:grey', column='2
016', cmap='Greens', alpha = 0.9)
sum_counts_manhattan.plot(ax = ax[1][2], edgecolor='tab:grey', column='2
017', cmap='Greens', alpha = 0.9)

#hide each map's axes for a 'picture frame' effect
ax[0][0].get_xaxis().set_visible(False)
ax[0][0].get_yaxis().set_visible(False)
ax[0][1].get_xaxis().set_visible(False)
ax[0][1].get_yaxis().set_visible(False)
ax[0][2].get_xaxis().set_visible(False)
ax[0][2].get_yaxis().set_visible(False)
ax[1][0].get_xaxis().set_visible(False)
ax[1][0].get_yaxis().set_visible(False)
ax[1][1].get_xaxis().set_visible(False)
ax[1][1].get_yaxis().set_visible(False)
ax[1][2].get_xaxis().set_visible(False)
ax[1][2].get_yaxis().set_visible(False)

#title each map by its respective year
ax[0][0].set_title('2012')
ax[0][1].set_title('2013')
ax[0][2].set_title('2014')
ax[1][0].set_title('2015')
ax[1][1].set_title('2016')
ax[1][2].set_title('2017')

#Enlarge each map's title
ax[0][0].title.set_size(25)
ax[0][1].title.set_size(25)
ax[0][2].title.set_size(25)
ax[1][0].title.set_size(25)
ax[1][1].title.set_size(25)
ax[1][2].title.set_size(25)

#Create an overarching title for the array of maps
fig.suptitle('Amount of Film Permits per Zip Code by Year',fontsize=35)

plt.show()
```



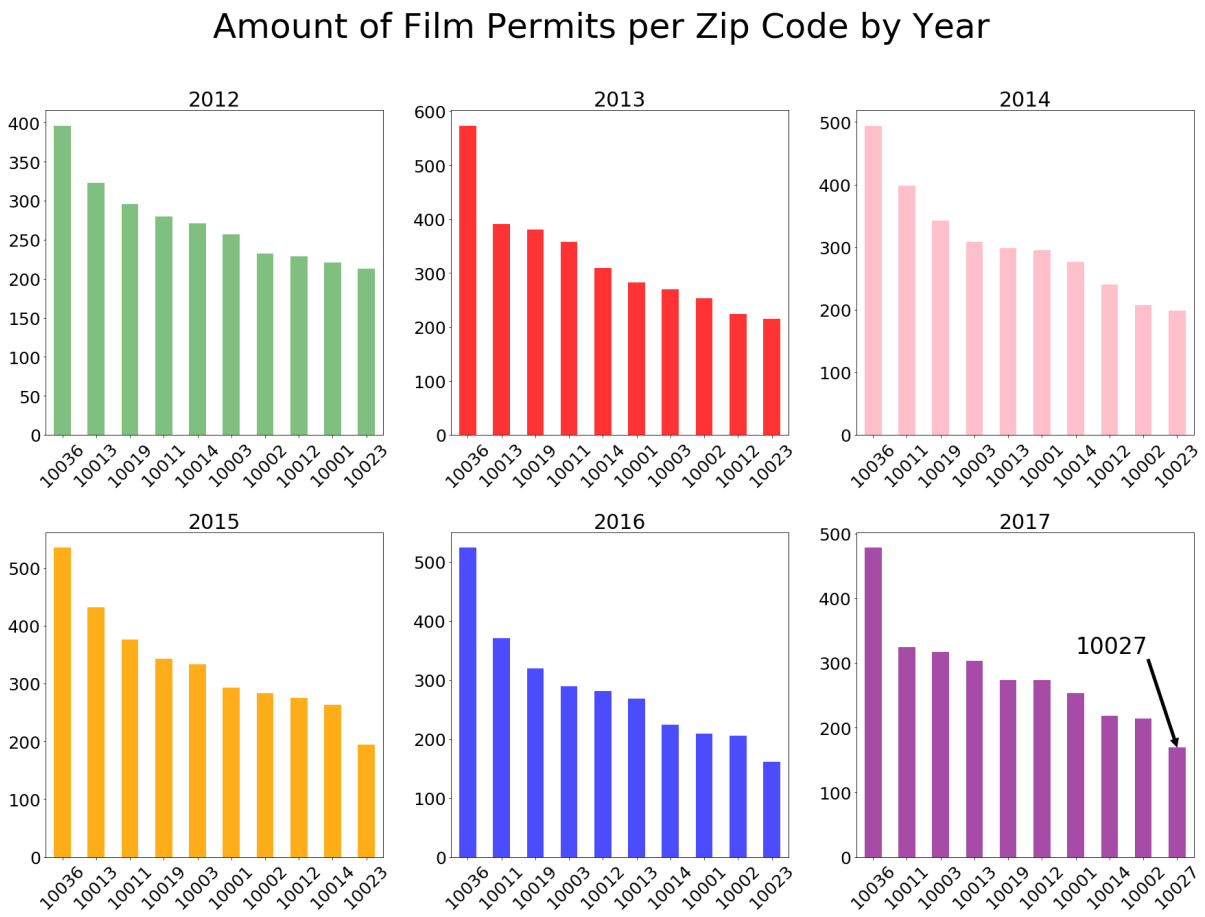
It does not appear that there has been much change between years, but let's take a closer look at how consistent the most popular zip codes are.

Identifying each year's top zip codes

To better distinguish the different shades of green, the top 10 zip codes from each year are plotted on a bar graph below.

In [5015]:
```
fig, ax = plt.subplots(2,3, figsize = (30,20))

#sort the DataFrame by the corresponding year and then graph the top 10
df_sum_counts.sort_values('2012', ascending = False, inplace = True)
df_sum_counts.iloc[:10].plot.bar(ax = ax[0][0], x='Zip Code', y='2012',
color = 'green', alpha = 0.5, legend = False)

#sort the DataFrame by the corresponding year and then graph the top 10
df_sum_counts.sort_values('2013', ascending = False, inplace = True)
df_sum_counts.iloc[:10].plot.bar(ax = ax[0][1], x='Zip Code', y='2013',
color = 'red', alpha=0.8, legend = False)

#sort the DataFrame by the corresponding year and then graph the top 10
df_sum_counts.sort_values('2014', ascending = False, inplace = True)
df_sum_counts.iloc[:10].plot.bar(ax = ax[0][2], x='Zip Code', y='2014',
color = 'pink', alpha=1, legend = False)

#sort the DataFrame by the corresponding year and then graph the top 10
df_sum_counts.sort_values('2015', ascending = False, inplace = True)
df_sum_counts.iloc[:10].plot.bar(ax = ax[1][0], x='Zip Code', y='2015',
color = 'orange', alpha=0.9, legend = False)

#sort the DataFrame by the corresponding year and then graph the top 10
df_sum_counts.sort_values('2016', ascending = False, inplace = True)
df_sum_counts.iloc[:10].plot.bar(ax = ax[1][1], x='Zip Code', y='2016',
color = 'blue', alpha=0.7,legend = False)

#sort the DataFrame by the corresponding year and then graph the top 10
df_sum_counts.sort_values('2017', ascending = False, inplace = True)
df_sum_counts.iloc[:10].plot.bar(ax = ax[1][2], x='Zip Code', y='2017',
color = 'purple', alpha=0.7,legend = False)

#title each graph by its respective year
ax[0][0].set_title('2012')
ax[0][1].set_title('2013')
ax[0][2].set_title('2014')
ax[1][0].set_title('2015')
ax[1][1].set_title('2016')
ax[1][2].set_title('2017')

#adjust the individual title sizes
ax[0][0].title.set_size(30)
ax[0][1].title.set_size(30)
ax[0][2].title.set_size(30)
ax[1][0].title.set_size(30)
ax[1][1].title.set_size(30)
ax[1][2].title.set_size(30)

#throw out all the x-axis labels.
#the overarching title added at the end will suffice
ax[0][0].set_xlabel('')
ax[0][1].set_xlabel('')
ax[0][2].set_xlabel('')
ax[1][0].set_xlabel('')
ax[1][1].set_xlabel('')
ax[1][2].set_xlabel('')
```

```
#format the axes unit labels size and positioning
#tilt the x axis to fit all the zip codes without overlap
ax[0][0].tick_params(axis='x', rotation=45, labelsize=25)
ax[0][0].tick_params(axis='y', labelsize=25)
ax[0][1].tick_params(axis='x', rotation=45, labelsize=25)
ax[0][1].tick_params(axis='y', labelsize=25)
ax[0][2].tick_params(axis='x', rotation=45, labelsize=25)
ax[0][2].tick_params(axis='y', labelsize=25)
ax[1][0].tick_params(axis='x', rotation=45, labelsize=25)
ax[1][0].tick_params(axis='y', labelsize=25)
ax[1][1].tick_params(axis='x', rotation=45, labelsize=25)
ax[1][1].tick_params(axis='y', labelsize=25)
ax[1][2].tick_params(axis='x', rotation=45, labelsize=25)
ax[1][2].tick_params(axis='y', labelsize=25)

#draw an arrow pointing to the notable statistical change
ax[1][2].annotate('10027',xy=(9, 170),xytext=(6, 315),arrowprops=dict(fa
cecolor='black'),fontsize=33,color='black')

#create an overarching title
fig.suptitle('Amount of Film Permits per Zip Code by Year',fontsize=50)

#add a little bit of spacing between the graphs for visual appeal
fig.subplots_adjust(hspace=.3)

plt.show()
```



Phenomenally, the same 10 zip codes remained in the top 10 each year. The only exception to this is in 2017 when zip code **10027** was the 10th highest, while **10023** (which had been 10th each of the previous 5 years), dropped to 13th. As one would guess, these 10 zip codes are the same ones that were seen in the overall plot.

Seasonal Data

While it isn't terribly surprising that there aren't large swings in the top zip codes for filming from year to year, it would be reasonable to assume that there would be some variation from season to season. Certain areas might be considered better during a particular season due to natural scenery (i.e. snow, rain, etc.) or because of seasonal icons (i.e. the Rockefeller Center when the Christmas tree is up).

Including seasonal totals

Create a DataFrame for each season. Then use the same technique as before to create film permit sums for each zip code.

In [5016]:
```
#group the film permit DataFrame by the season column
season_films = film_Manhattan_refined.groupby(['Season'])

#one DataFrame per season
spring_films = season_films.get_group('Spring')
summer_films = season_films.get_group('Summer')
autumn_films = season_films.get_group('Autumn')
winter_films = season_films.get_group('Winter')
```

In [5017]:
```
#count the number of permits for each zip code within each season's Data
Frame
spring_zip_counts = spring_films['Zip Code'].value_counts()
summer_zip_counts = summer_films['Zip Code'].value_counts()
autumn_zip_counts = autumn_films['Zip Code'].value_counts()
winter_zip_counts = winter_films['Zip Code'].value_counts()
```

In [5018]:
```
#generate a DataFrame with each seasons totals
df_spring_counts = spring_zip_counts.rename_axis('Zip Code').reset_index
(name = 'Spring')
df_summer_counts = summer_zip_counts.rename_axis('Zip Code').reset_index
(name = 'Summer')
df_autumn_counts = autumn_zip_counts.rename_axis('Zip Code').reset_index
(name = 'Autumn')
df_winter_counts = winter_zip_counts.rename_axis('Zip Code').reset_index
(name = 'Winter')
```

In [5019]:
```
#Add each season's count to its respective zip code row in the GeoDataFr
ame
sum_counts_manhattan = sum_counts_manhattan.merge(df_spring_counts[['Zip
 Code','Spring']],how='left',on = ['Zip Code'])
sum_counts_manhattan = sum_counts_manhattan.merge(df_summer_counts[['Zip
 Code','Summer']],how='left',on = ['Zip Code'])
sum_counts_manhattan = sum_counts_manhattan.merge(df_autumn_counts[['Zip
 Code','Autumn']],how='left',on = ['Zip Code'])
sum_counts_manhattan = sum_counts_manhattan.merge(df_winter_counts[['Zip
 Code','Winter']],how='left',on = ['Zip Code'])
```

In [5020]:
```
#For years that don't have any permits, replace the default NaN value wi
th zero
sum_counts_manhattan.fillna(value=0,inplace=True)
sum_counts_manhattan = sum_counts_manhattan.astype(int,errors='ignore',i
nplace=True)
```

In [5021]:
```
df_sum_counts = df_sum_counts.merge(sum_counts_manhattan
                                    [['Zip Code','Spring','Summer','Autumn','Win
ter']], on = ['Zip Code'])
df_sum_counts.drop_duplicates(subset=['Zip Code'],inplace=True)
df_sum_counts = df_sum_counts.astype(int,errors='ignore',inplace=True)
```

Here is a glimpse of what the GeoDataFrame and DataFrame look like now that seasonal sum columns have been added to each zip code.

In [5022]:
```
df_sum_counts.head(5)
```

Out[5022]:

	Zip Code	Income	Permits	2012	2013	2014	2015	2016	2017	2018	Spring	Summer	A
0	10036	75966	3030	396	573	494	535	524	478	30	853	917	75
1	10011	109818	2142	280	358	398	376	371	325	34	566	560	59
2	10003	98151	1801	257	270	309	333	289	317	26	429	455	56
3	10013	96667	2075	323	391	299	432	269	303	58	534	526	61
4	10019	94022	1987	296	381	342	343	320	274	31	584	517	53

In [5023]:
```
sum_counts_manhattan.head(5)
```

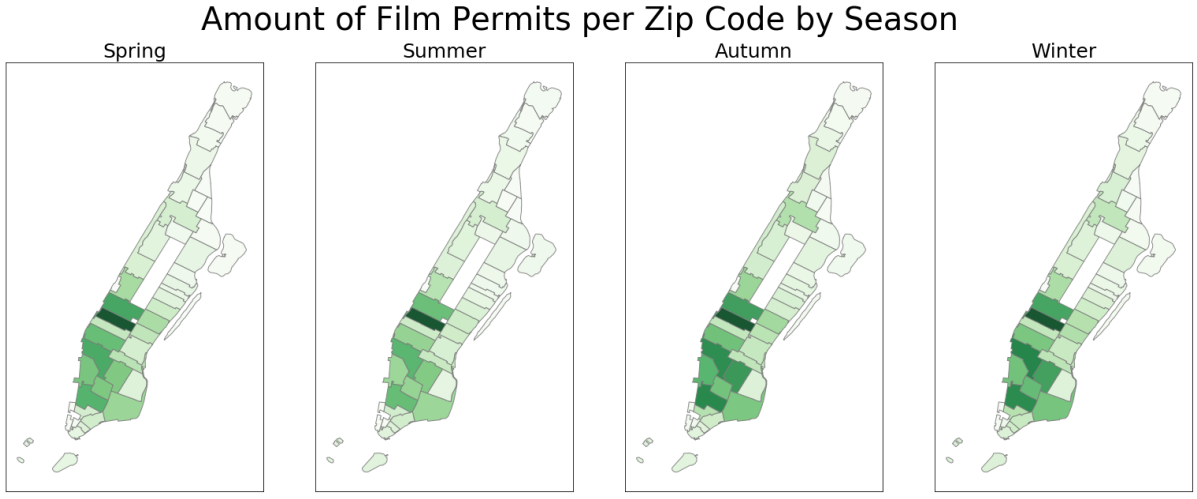
Out[5023]:

	Zip Code	Income	geometry	Permits	2012	2013	2014	2015	2016	2017	20
0	10034	42581	POLYGON ((1006203.169008225 257345.6566181332,...	97	18	11	13	24	16	15	0
1	10033	44933	POLYGON ((1003020.86467731 256049.1618358046, ...	232	43	34	40	42	41	31	1
2	10040	46200	POLYGON ((1002564.135275811 253724.7731118798,...	92	6	9	20	16	21	18	2
3	10032	37280	POLYGON ((998935.8528342247 249154.259943217, ...	379	63	29	74	101	61	47	4
4	10031	39774	POLYGON ((1000830.038678393 241801.5896092206,...	423	52	93	60	75	59	72	12

Mapping each season (all years inclusive)

Below are four heat maps of filming locations for each season.

In [5024]:
```
fig, ax = plt.subplots(1,4, figsize = (27,10))

#draw maps for each season
sum_counts_manhattan.plot(ax = ax[0], edgecolor='tab:grey', column='Spri
ng', cmap='Greens', alpha = 0.9)
sum_counts_manhattan.plot(ax = ax[1], edgecolor='tab:grey', column='Summ
er', cmap='Greens', alpha = 0.9)
sum_counts_manhattan.plot(ax = ax[2], edgecolor='tab:grey', column='Autu
mn', cmap='Greens', alpha = 0.9)
sum_counts_manhattan.plot(ax = ax[3], edgecolor='tab:grey', column='Wint
er', cmap='Greens', alpha = 0.9)

#hide axes for to frame map
ax[0].get_xaxis().set_visible(False)
ax[0].get_yaxis().set_visible(False)
ax[1].get_xaxis().set_visible(False)
ax[1].get_yaxis().set_visible(False)
ax[2].get_xaxis().set_visible(False)
ax[2].get_yaxis().set_visible(False)
ax[3].get_xaxis().set_visible(False)
ax[3].get_yaxis().set_visible(False)

#title each map by its respective season
ax[0].set_title('Spring')
ax[1].set_title('Summer')
ax[2].set_title('Autumn')
ax[3].set_title('Winter')

#enlarge each map's title
ax[0].title.set_size(25)
ax[1].title.set_size(25)
ax[2].title.set_size(25)
ax[3].title.set_size(25)

#create an overarching title
fig.suptitle('Amount of Film Permits per Zip Code by Season',fontsize=40
)

plt.show()
```



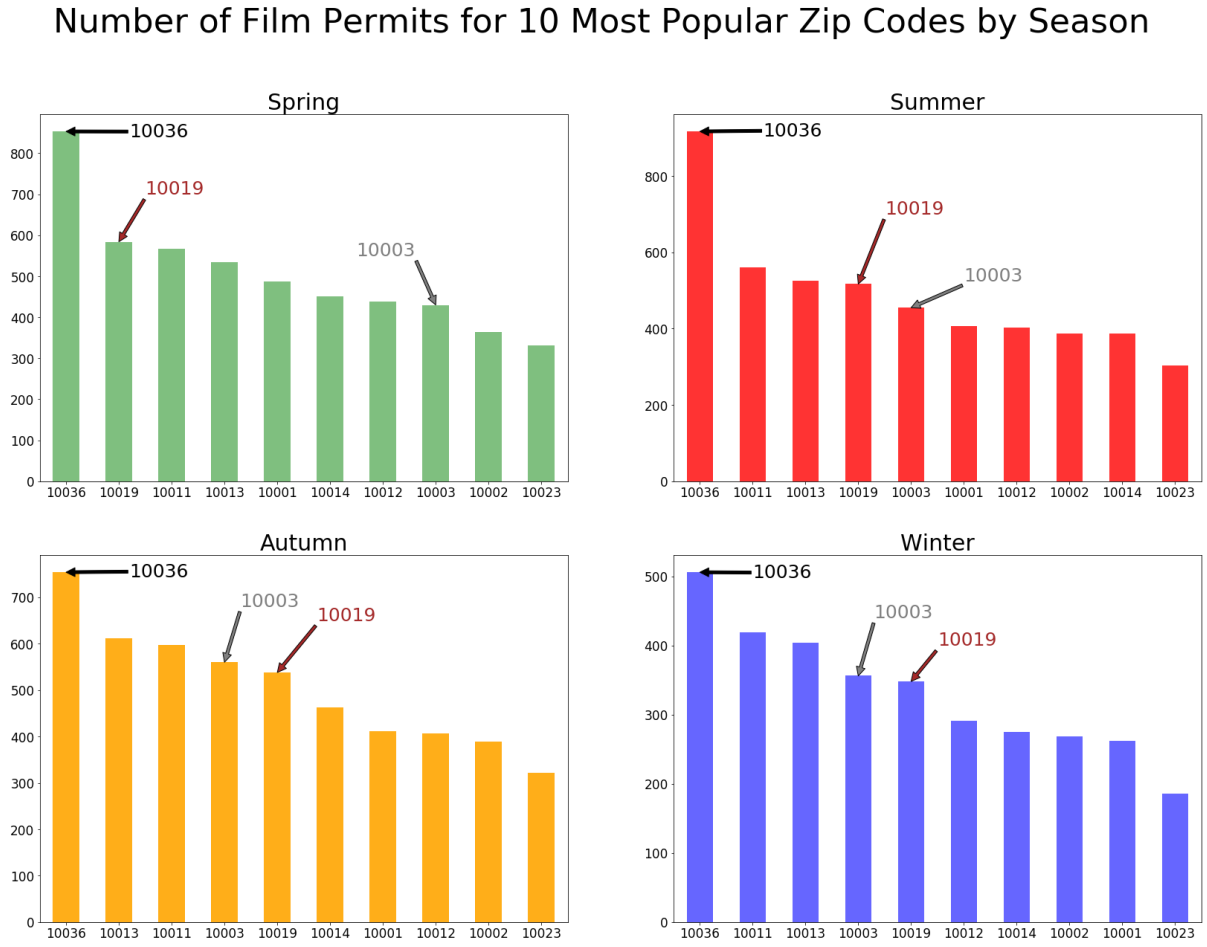
There isn't too much differentiation between the seasons. There are some changes in color, but at a glance, not much shifted.

Identifying each season's top zip codes

Below is a bar graph to better understand the movement or lack thereof. The graphs highlight some zip codes of interest.

In [5025]:
```
fig, ax = plt.subplots(2,2, figsize = (28,20))

#create an overarching title and enlarge its size
fig.suptitle('Number of Film Permits for 10 Most Popular Zip Codes by Se
ason', fontsize=45)

#sort the DataFrame by spring permit totals and then plot the top 10
df_sum_counts.sort_values('Spring', ascending = False, inplace = True)
df_sum_counts.iloc[:10].plot.bar(ax = ax[0][0],x='Zip Code',y='Spring',c
olor ='green',alpha=0.5,legend = False)

#sort the DataFrame by summer permit totals and then plot the top 10
df_sum_counts.sort_values('Summer', ascending = False, inplace = True)
df_sum_counts.iloc[:10].plot.bar(ax = ax[0][1], x='Zip Code',y='Summer',
color ='red',alpha=0.8,legend = False)

#sort the DataFrame by autumn permit totals and then plot the top 10
df_sum_counts.sort_values('Autumn', ascending = False, inplace = True)
df_sum_counts.iloc[:10].plot.bar(ax = ax[1][0], x='Zip Code', y='Autumn'
, color = 'orange', alpha=0.9, legend = False)

#sort the DataFrame by winter permit totals and then plot the top 10
df_sum_counts.sort_values('Winter', ascending = False, inplace = True)
df_sum_counts.iloc[:10].plot.bar(ax = ax[1][1], x='Zip Code', y='Winter'
, color = 'blue', alpha=0.6,legend = False)

#title each graph by its respective season
ax[0][0].set_title('Spring')
ax[0][1].set_title('Summer')
ax[1][0].set_title('Autumn')
ax[1][1].set_title('Winter')

#enlarge the graph titles
ax[0][0].title.set_size(30)
ax[0][1].title.set_size(30)
ax[1][0].title.set_size(30)
ax[1][1].title.set_size(30)

#hide x axes titles, the overarching title will suffice
ax[0][0].set_xlabel('')
ax[0][1].set_xlabel('')
ax[1][0].set_xlabel('')
ax[1][1].set_xlabel('')

#position and size the axes unit labels for optimal viewing
ax[0][0].tick_params(axis='x', rotation=0, labelsize=17)
ax[0][0].tick_params(axis='y', labelsize=17)
ax[0][1].tick_params(axis='x', rotation=0, labelsize=17)
ax[0][1].tick_params(axis='y', labelsize=17)
ax[1][0].tick_params(axis='x', rotation=0, labelsize=17)
ax[1][0].tick_params(axis='y', labelsize=17)
ax[1][1].tick_params(axis='x', rotation=0, labelsize=17)
ax[1][1].tick_params(axis='y', labelsize=17)

#use arrow shape to point to zip codes of interest
ax[0][0].annotate('10036',xy=(0, 853),xytext=(1.2, 840),arrowprops=dict(
```

```
facecolor='black'),fontsize=25)
ax[0][0].annotate('10019',xy=(1, 584),xytext=(1.5, 700),arrowprops=dict(
facecolor='brown'),fontsize=25,color='brown')
ax[0][0].annotate('10003',xy=(7, 430),xytext=(5.5, 550),arrowprops=dict(
facecolor='grey'),fontsize=25,color='grey')

#use arrow shape to point to zip codes of interest
ax[0][1].annotate('10036',xy=(0, 917),xytext=(1.2, 905),arrowprops=dict(
facecolor='black'),fontsize=25)
ax[0][1].annotate('10019',xy=(3, 517),xytext=(3.5, 700),arrowprops=dict(
facecolor='brown'),fontsize=25,color='brown')
ax[0][1].annotate('10003', xy=(4, 455),xytext=(5, 525),arrowprops=dict(f
acecolor='grey'),fontsize=25,color='grey')

#use arrow shape to point to zip codes of interest
ax[1][0].annotate('10036',xy=(0, 754),xytext=(1.2, 744),arrowprops=dict(
facecolor='black'),fontsize=25)
ax[1][0].annotate('10019',xy=(4, 538),xytext=(4.75, 650),arrowprops=dict
(facecolor='brown'),fontsize=25,color='brown')
ax[1][0].annotate('10003',xy=(3, 560),xytext=(3.3, 680),arrowprops=dict(
facecolor='grey'),fontsize=25,color='grey')

#use arrow shape to point to zip codes of interest
ax[1][1].annotate('10036',xy=(0, 506),xytext=(1, 498),arrowprops=dict(fa
cecolor='black'),fontsize=25)
ax[1][1].annotate('10019',xy=(4, 348),xytext=(4.5, 400),arrowprops=dict(
facecolor='brown'),fontsize=25,color='brown')
ax[1][1].annotate('10003',xy=(3, 356),xytext=(3.3, 440),arrowprops=dict(
facecolor='grey'),fontsize=25,color='grey')

plt.show()
```



What is most striking about this visual is that it's the same 10 zip codes across all four seasons that rank in the top 10. **10036** reigns superior and then there is little fluctuation between the others. Notably, **10019** is 2nd highest in spring, but is as low as 5th in autumn and winter. In addition, **10003** is 4th in autumn and winter, but drops as low as 8th during spring.

1. **10036** as mentioned before is Hell's Kitchen.
2. **10019** is the zip code directly above 10036 and is primarily in the Hell's Kitchen neighborhood as well.
3. **10003** includes some of East Village, NoHo, and Gramercy.

Aside from those two noticeable differences, the permit variation is incredibly limited between seasons.

Shifts Over Time

We've gone through overall, yearly, and seasonal data. The results are that film permit concentration by zip code hardly shifts between years and between seasons. Hell's Kitchen (zip code 10036) is dominant while other mid-to-lower Manhattan zip codes populate the top 10 locations in each breakdown. In addition, the notion that permits might be correlated to income has been disproven.

The final phase of this project will be to find trends over time. The data will be broken up into the first 3 years, and the most recent 3 years (2018 excluded).

Add new columns to the DataFrame and GeoDataFrame

To analyze the data, we'll tack on a column that sums up the 2012, 2013, and 2014 columns—as well as for 2015, 2016, and 2017. In addition, a percent change column of the two new columns will also be added to the DataFrame.

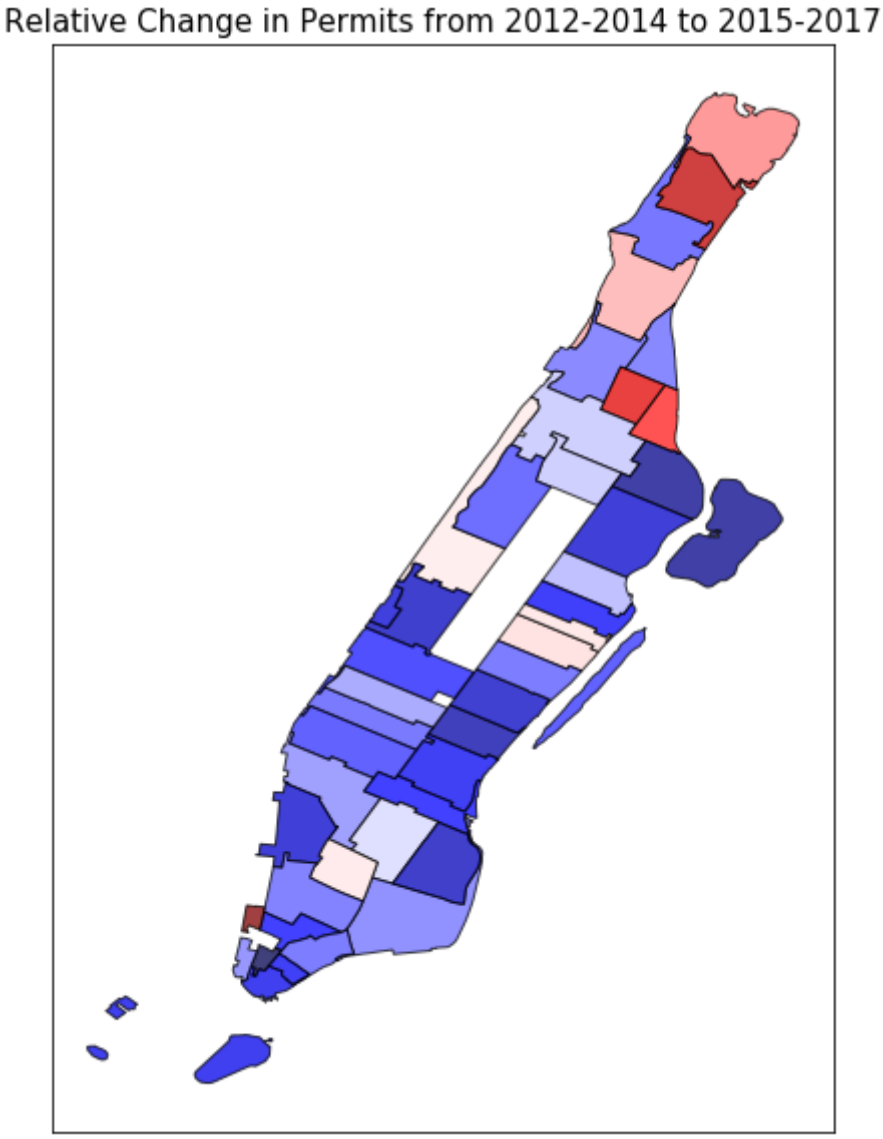
In [5026]:
```
#sum first 3 years
df_sum_counts['2012-2014'] = df_sum_counts['2012']+df_sum_counts['2013']
+df_sum_counts['2014']

#sum most recent complete years
df_sum_counts['2015-2017'] = df_sum_counts['2015']+df_sum_counts['2016']
+df_sum_counts['2017']

#calculate percent change from the first 3 years to the 3 more recent
#this will be displayed as a floating decimal
df_sum_counts['Trend'] = (df_sum_counts['2015-2017'] - df_sum_counts['20
12-2014']) / (df_sum_counts['2012-2014'])

#merge the new DataFrame columns onto the GeoDataFrame
#GeoDataFrame zip codes with multiple entries will be filled in on each,
 as desired
sum_counts_manhattan = sum_counts_manhattan.merge(df_sum_counts
                                  [['Zip Code','2012-2014','2015-2017','T
rend']], on = ['Zip Code'])

df_sum_counts['Trend'].describe()
```

Out[5026]:
```
count    44.000000
mean      0.025191
std       0.224519
min      -0.373494
25%      -0.116643
50%      -0.014668
75%       0.108247
max       0.709677
Name: Trend, dtype: float64
```

The average change is a **2.5%** increase. The data appears to be widely distributed, ranging from **-37.3%** to **71.0%**. Considering how steady the seasonal and yearly data breakdowns were, it's surprising to see level of volatility for the zip codes at a 3-year period. It is also notable that the 50th percentile (or median), is **-1.47%**, meaning that although there was an overall increase, most zip codes decreased.

Mapping 3-year percent change by zip code

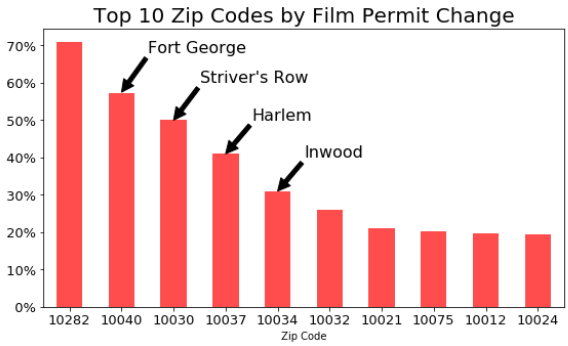
In [5027]:
```
fig, ax = plt.subplots(figsize = (7,10))

#use the seismic cmap color scheme where high figures are more red while
 lower figures are more blue
#alpha used to tone down the colors for a more consistent pastel feel as
 replicated throughout the project
sum_counts_manhattan.plot(ax = ax, column='Trend', edgecolor='black', cm
ap='seismic', alpha = .75)

#hide axes, just leave it as a 'picture frame'
ax.get_xaxis().set_visible(False)
ax.get_yaxis().set_visible(False)

#create a title
ax.set_title('Relative Change in Permits from 2012-2014 to 2015-2017')
ax.title.set_size(15)


plt.show()
```

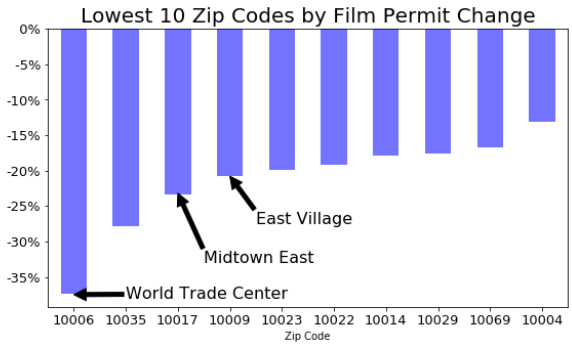


Identifying the 10 zip codes gaining and losing popularity the fastest

While the number has overall decreased, the wide spread of zip code percent change values begs the question, which zip codes are losing popularity the most? And in addition, which ones are gaining the most?

In [5028]:
```
fig, ax = plt.subplots(1,2,figsize = (20,5))

#sort the DataFrame by number of overall permits
df_sum_counts.sort_values('Trend', ascending = False, inplace = True)
#plot only the top 10 values (indices 0-10 exclusive)
df_sum_counts.iloc[:10].plot.bar(ax=ax[0], x='Zip Code', y='Trend', colo
r='red',alpha=0.7,legend=False)

#set title and font size
ax[0].set_title('Top 10 Zip Codes by Film Permit Change')
ax[0].title.set_size(20)

#format axes units
ax[0].tick_params(axis='x', rotation=0, labelsize=13)
ax[0].tick_params(axis='y', rotation=0, labelsize=13)
vals = ax[0].get_yticks()
ax[0].set_yticklabels(['{:3.0f}%'.format(x*100) for x in vals])

#put notes to identify zip codes of interest
ax[0].annotate('Inwood',xy=(4,.31),xytext=(4.5,.4),arrowprops=dict(facec
olor='black'),fontsize=16)
ax[0].annotate('Striver\'s Row',xy=(2,.5),xytext=(2.5,.6),arrowprops=dic
t(facecolor='black'),fontsize=16)
ax[0].annotate('Fort George',xy=(1,.575),xytext=(1.5,.68),arrowprops=dic
t(facecolor='black'),fontsize=16)
ax[0].annotate('Harlem',xy=(3,.41),xytext=(3.5,.5),arrowprops=dict(facec
olor='black'),fontsize=16)

########################################
#sort the DataFrame by number of overall permits
df_sum_counts.sort_values('Trend', ascending = True, inplace = True)
#plot only the top 10 values (indices 0-10 exclusive)
df_sum_counts.iloc[:10].plot.bar(ax=ax[1], x='Zip Code', y='Trend', colo
r='blue',alpha=0.55,legend=False)

#set title and font size
ax[1].set_title('Lowest 10 Zip Codes by Film Permit Change')
ax[1].title.set_size(20)

#format axes units
ax[1].tick_params(axis='x', rotation=0, labelsize=13)
ax[1].tick_params(axis='y', rotation=0, labelsize=13)
vals = ax[1].get_yticks()
ax[1].set_yticklabels(['{:3.0f}%'.format(x*100) for x in vals])

#put a note to identify zip codes of interest
ax[1].annotate('Midtown East',xy=(2,-.232),xytext=(2.5,-.33),arrowprops=
dict(facecolor='black'),fontsize=16)
ax[1].annotate('World Trade Center',xy=(0,-.375),xytext=(1,-.381),arrowp
rops=dict(facecolor='black'),fontsize=16)
ax[1].annotate('East Village',xy=(3,-.208),xytext=(3.5,-.275),arrowprops
=dict(facecolor='black'),fontsize=16)

plt.show()
```





What stands out in the graph on the left is that 4 out of the top 5 zip codes are above 130th Street. To put that in context: they're all above Columbia University, and outside of CitiBike range. From the income map, we know that these zip codes at the top of Manhattan are on the lower end of the median income data.

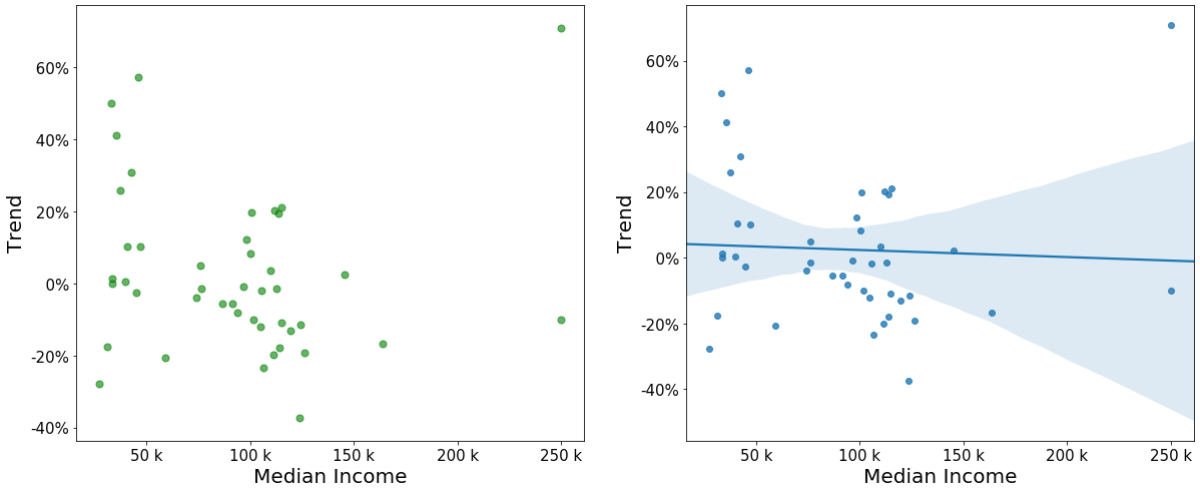
On the right, the top 5 coldest zip codes are a little more dispersed. That said, (thinking back to the income heat map), 3 out of the top 4 (including the top), are identifiably affluent areas.

Looking for correlation between trend and income

So we know that some of the poorer neighborhoods have become more popular, and we also know that some of the richer neighborhoods have become less popular. It's worth one more shot. There might be a statistically significant correlation between the trend column (the 3-year percent change value), and the median income for each zip code.

Below are a scatter plot as well as a regression plot to test this.

In [5029]:
```
fig,ax = plt.subplots(1,2,figsize=(20,8))

#plot each zip code's median income against its number of permits on the
 left
df_sum_counts.plot(ax=ax[0],kind='scatter',x='Income',y='Trend',s=50, co
lor='green',alpha=0.6)

#Set the axes labels and increase their size
ax[0].set_xlabel('Median Income',fontsize=20)
ax[0].set_ylabel('Trend',fontsize=20)

#increase the size of the axes units
ax[0].tick_params(axis='x',labelsize=15)
ax[0].tick_params(axis='y',labelsize=15)

#properly format the axes labels
ax[0].xaxis.set_major_formatter(mtick.EngFormatter())
vals = ax[0].get_yticks()
ax[0].set_yticklabels(['{:3.0f}%'.format(x*100) for x in vals])

#use the special plotting data package to make a regression plot on the
 right
sns.regplot(x="Income",y="Trend", data=df_sum_counts,ax=ax[1])

#Set the axes labels and increase their size
ax[1].set_xlabel('Median Income',fontsize=20)
ax[1].set_ylabel('Trend',fontsize=20)

#increase the size of the axes units
ax[1].tick_params(axis='x',labelsize=15)
ax[1].tick_params(axis='y',labelsize=15)

#properly format the axes labels
ax[1].xaxis.set_major_formatter(mtick.EngFormatter())
vals = ax[1].get_yticks()
ax[1].set_yticklabels(['{:3.0f}%'.format(x*100) for x in vals])

plt.show()
```



Unfortunately, the answer is no. This graph appears to be even less correlated than the one for the overall data.

While there appeared to be little change between years and seasons for each zip code, it was surprising to see in this section how much variation there is when comparing 2012-2014 to 2015-2017. However, it should be noted that--especially for the high-growth zip codes--these numbers are relative. So, even if the zip codes at the top of Manhattan keep growing by a substantial margin, it is unlikely that this will have a noticeable effect on the overall data in the near future.

Conclusion

This project set out to answer the question: *Where are people filming in NYC?*

We took a zip code level approach and looked by year, by season, and as correlated to income. The 7 key findings from this analysis of film permit data for the past six years are:

1. Film Permits have been largely concentrated in the lower half of Manhattan
2. Although this distribution looks similar to a median income map, there is no correlation
3. The top zip code for filming is in Hell's Kitchen
4. The top zip codes have hardly changed over the past 6 years
5. The top 10 zip codes do not change by season with only slight order variation for some zip codes
6. Growing zip code popularity is found most notably at the top of Manhattan
7. There is no correlation between relative permit popularity growth and income level for each zip code

When setting out on this project, the hypothesis was made that there was an inherent correlation between the number of film permits issued for a zip code and the median income. This was quickly disproved. Even among a small survey of NYU peers, the consensus was that there was probably a positively correlated relationship. Looking more carefully at the income distribution along with the permit distribution maps, one starts to get the sense that there really shouldn't be a correlation. This suggests that perhaps our assumptions about filming or income distribution in Manhattan are just incorrect. That said, it's still true that income is concentrated in the lower half of Manhattan, and so are film permits. However, at the more granular level of zip codes, we lose the correlation.
