1. TV, halftime shows, and the Big Game

Whether or not you like football, the Super Bowl is a spectacle. There's a little something for everyone at your Super Bowl party. Drama in the form of blowouts, comebacks, and controversy for the sports fan. There are the ridiculously expensive ads, some hilarious, others gut-wrenching, thought-provoking, and weird. The half-time shows with the biggest musicians in the world, sometimes riding giant mechanical tigers (https://youtu.be/ZD1QrIe--_Y?t=14) or leaping from the roof of the stadium (https://youtu.be/mjrdywp5nyE?t=62). It's a show, baby. And in this notebook, we're going to find out how some of the elements of this show interact with each other. After exploring and cleaning our data a little, we're going to answer questions like:

- What are the most extreme game outcomes?
- How does the game affect television viewership?
- How have viewership, TV ratings, and ad cost evolved over time?
- Who are the most prolific musicians in terms of halftime show performances?

Left Shark Steals The Show (https://www.flickr.com/photos/huntleypaton/16464994135/in/photostream/). Katy Perry performing at halftime of Super Bowl XLIX. Photo by Huntley Paton. Attribution-ShareAlike 2.0 Generic (CC BY-SA 2.0).

The dataset we'll use was scraped (https://en.wikipedia.org/wiki/Web_scraping) and polished from Wikipedia. It is made up of three CSV files, one with game data (https://en.wikipedia.org/wiki/List_of_Super_Bowl_champions), one with TV data (https://en.wikipedia.org/wiki/Super_Bowl_television_ratings), and one with halftime musician data (https://en.wikipedia.org/wiki/List_of_Super_Bowl_halftime_shows) for all 52 Super Bowls through 2018. Let's take a look, using `display()` instead of `print()` since its output is much prettier in Jupyter

In [126]:

```
# Import pandas
import pandas as pd

# Load the CSV data into DataFrames
super_bowls = pd.read_csv('datasets/super_bowls.csv')
tv = pd.read_csv('datasets/tv.csv')
halftime_musicians = pd.read_csv('datasets/halftime_musicians.csv')

# Display the first five rows of each DataFrame
display(super_bowls.head())
display(tv.head())
display(halftime_musicians.head())
```

	date	super_bowl	venue	city	state	attendance	team_winner	wir
0	2018-02-04	52	U.S. Bank Stadium	Minneapolis	Minnesota	67612	Philadelphia Eagles	41
1	2017-02-05	51	NRG Stadium	Houston	Texas	70807	New England Patriots	34
2	2016-02-07	50	Levi's Stadium	Santa Clara	California	71088	Denver Broncos	24
3	2015-02-01	49	University of Phoenix Stadium	Glendale	Arizona	70288	New England Patriots	28
4	2014-02-02	48	MetLife Stadium	East Rutherford	New Jersey	82529	Seattle Seahawks	43

	super_bowl	network	avg_us_viewers	total_us_viewers	rating_household	share_h
0	52	NBC	103390000	NaN	43.1	68
1	51	Fox	111319000	172000000.0	45.3	73
2	50	CBS	111864000	167000000.0	46.6	72
3	49	NBC	114442000	168000000.0	47.5	71
4	48	Fox	112191000	167000000.0	46.7	69

	super_bowl	musician	num_songs
0	52	Justin Timberlake	11.0
1	52	University of Minnesota Marching Band	1.0
2	51	Lady Gaga	7.0
3	50	Coldplay	6.0
4	50	Beyoncé	3.0

In [127]:

```
%%nose
# %%nose needs to be included at the beginning of every @tests cell

# One or more tests of the student's code
# The @solution should pass the tests
# The purpose of the tests is to try to catch common errors and
# to give the student a hint on how to resolve these errors

def test_pandas_loaded():
    assert 'pd' in globals(), \
    'Did you import the pandas module under the alias pd?'

def test_super_bowls_correctly_loaded():
    correct_super_bowls = pd.read_csv('datasets/super_bowls.csv')
    assert correct_super_bowls.equals(super_bowls), "The variable super_bowls do
es not contain the data in super_bowls.csv."

def test_tv_correctly_loaded():
    correct_tv = pd.read_csv('datasets/tv.csv')
    assert correct_tv.equals(tv), "The variable tv does not contain the data in
 tv.csv."

def test_halftime_musicians_correctly_loaded():
    correct_halftime_musicians = pd.read_csv('datasets/halftime_musicians.csv')
    assert correct_halftime_musicians.equals(halftime_musicians), "The variable
 halftime_musicians does not contain the data in halftime_musicians.csv."
```

Out[127]:

```
4/4 tests passed
```

2. Taking note of dataset issues

For the Super Bowl game data, we can see the dataset appears whole except for missing values in the backup quarterback columns (`qb_winner_2` and `qb_loser_2`), which make sense given most starting QBs in the Super Bowl (`qb_winner_1` and `qb_loser_1`) play the entire game.

From the visual inspection of TV and halftime musicians data, there is only one missing value displayed, but I've got a hunch there are more. The Super Bowl goes all the way back to 1967, and the more granular columns (e.g. the number of songs for halftime musicians) probably weren't tracked reliably over time. Wikipedia is great but not perfect.

An inspection of the `.info()` output for `tv` and `halftime_musicians` shows us that there are multiple columns with null values.

In [128]:

```
# Summary of the TV data to inspect
tv.info()

print('\n')

# Summary of the halftime musician data to inspect
halftime_musicians.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 53 entries, 0 to 52
Data columns (total 9 columns):
super_bowl          53 non-null int64
network             53 non-null object
avg_us_viewers      53 non-null int64
total_us_viewers    15 non-null float64
rating_household    53 non-null float64
share_household     53 non-null int64
rating_18_49        15 non-null float64
share_18_49         6 non-null float64
ad_cost             53 non-null int64
dtypes: float64(4), int64(4), object(1)
memory usage: 3.8+ KB


<class 'pandas.core.frame.DataFrame'>
RangeIndex: 134 entries, 0 to 133
Data columns (total 3 columns):
super_bowl    134 non-null int64
musician      134 non-null object
num_songs     88 non-null float64
dtypes: float64(1), int64(1), object(1)
memory usage: 3.2+ KB
```

In [129]:

```
%%nose
# %%nose needs to be included at the beginning of every @tests cell

# One or more tests of the student's code
# The @solution should pass the tests
# The purpose of the tests is to try to catch common errors and
# to give the student a hint on how to resolve these errors

# def test_nothing_task_2():
#     assert True, "Nothing to test."

last_input = In[-2]

def test_not_empty_task_2():
    assert "# ... YOUR CODE FOR TASK" not in last_input, \
        "It appears that # ... YOUR CODE FOR TASK X ... is still in the code cel
l, which suggests that you might not have attempted the code for this task. If y
ou have, please delete # ... YOUR CODE FOR TASK X ... from the cell and resubmi
t."
```

Out[129]:

```
1/1 tests passed
```

3. Combined points distribution

For the TV data, the following columns have missing values and a lot of them:

- `total_us_viewers` (amount of U.S. viewers who watched at least some part of the broadcast)
- `rating_18_49` (average % of U.S. adults 18-49 who live in a household with a TV that were watching for the entire broadcast)
- `share_18_49` (average % of U.S. adults 18-49 who live in a household with a TV *in use* that were watching for the entire broadcast)

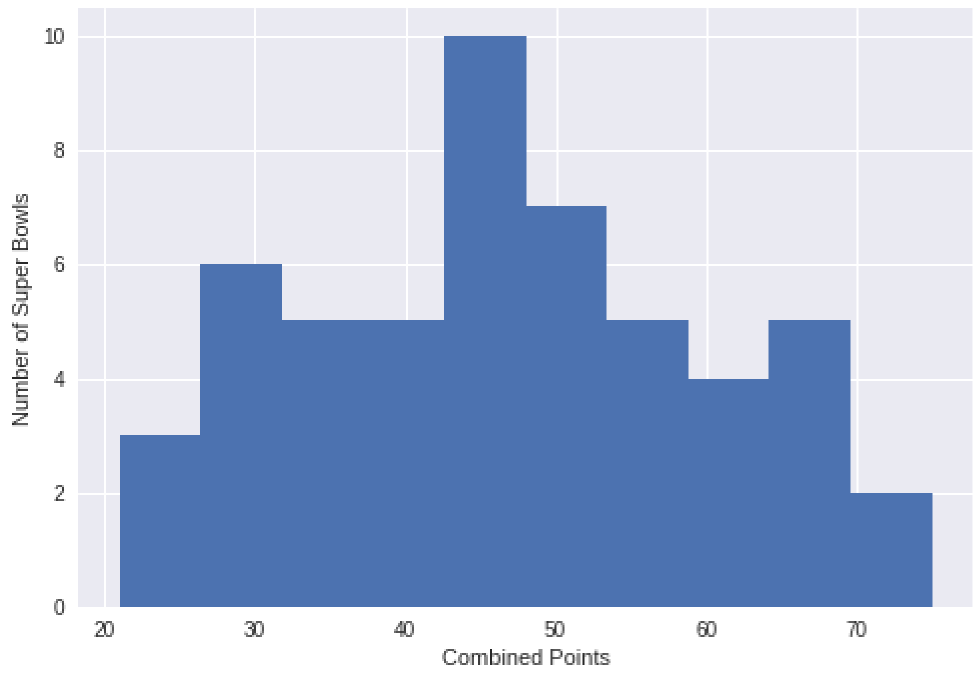
For the halftime musician data, there are missing numbers of songs performed (`num_songs`) for about a third of the performances.

There are a lot of potential reasons for these missing values. Was the data ever tracked? Was it lost in history? Is the research effort to make this data whole worth it? Maybe. Watching every Super Bowl halftime show to get song counts would be pretty fun. But we don't have the time to do that kind of stuff now! Let's take note of where the dataset isn't perfect and start uncovering some insights.

Let's start by looking at combined points for each Super Bowl by visualizing the distribution. Let's also pinpoint the Super Bowls with the highest and lowest scores.

In [130]:

```
# Import matplotlib and set plotting style
from matplotlib import pyplot as plt
%matplotlib inline
plt.style.use('seaborn')

# Plot a histogram of combined points
plt.hist(super_bowls.combined_pts)
plt.xlabel('Combined Points')
plt.ylabel('Number of Super Bowls')
plt.show()

# Display the Super Bowls with the highest and lowest combined scores
display(super_bowls[super_bowls['combined_pts'] > 70])
display(super_bowls[super_bowls['combined_pts'] < 25])
```



	date	super_bowl	venue	city	state	attendance	team_winner	wir
0	2018-02-04	52	U.S. Bank Stadium	Minneapolis	Minnesota	67612	Philadelphia Eagles	41
23	1995-01-29	29	Joe Robbie Stadium	Miami Gardens	Florida	74107	San Francisco 49ers	49

	date	super_bowl	venue	city	state	attendance	team_winner	winnin
43	1975-01-12	9	Tulane Stadium	New Orleans	Louisiana	80997	Pittsburgh Steelers	16
45	1973-01-14	7	Memorial Coliseum	Los Angeles	California	90182	Miami Dolphins	14
49	1969-01-12	3	Orange Bowl	Miami	Florida	75389	New York Jets	16

In [131]:

```
%%nose
# %%nose needs to be included at the beginning of every @tests cell

# One or more tests of the student's code
# The @solution should pass the tests
# The purpose of the tests is to try to catch common errors and
# to give the student a hint on how to resolve these errors

def test_matplotlib_loaded():
    assert 'plt' in globals(), \
    'Did you import the pyplot module from matplotlib under the alias plt?'
```

Out[131]:

```
1/1 tests passed
```

4. Point difference distribution

Most combined scores are around 40-50 points, with the extremes being roughly equal distance away in opposite directions. Going up to the highest combined scores at 74 and 75, we find two games featuring dominant quarterback performances. One even happened recently in 2018's Super Bowl LII where Tom Brady's Patriots lost to Nick Foles' underdog Eagles 41-33 for a combined score of 74.

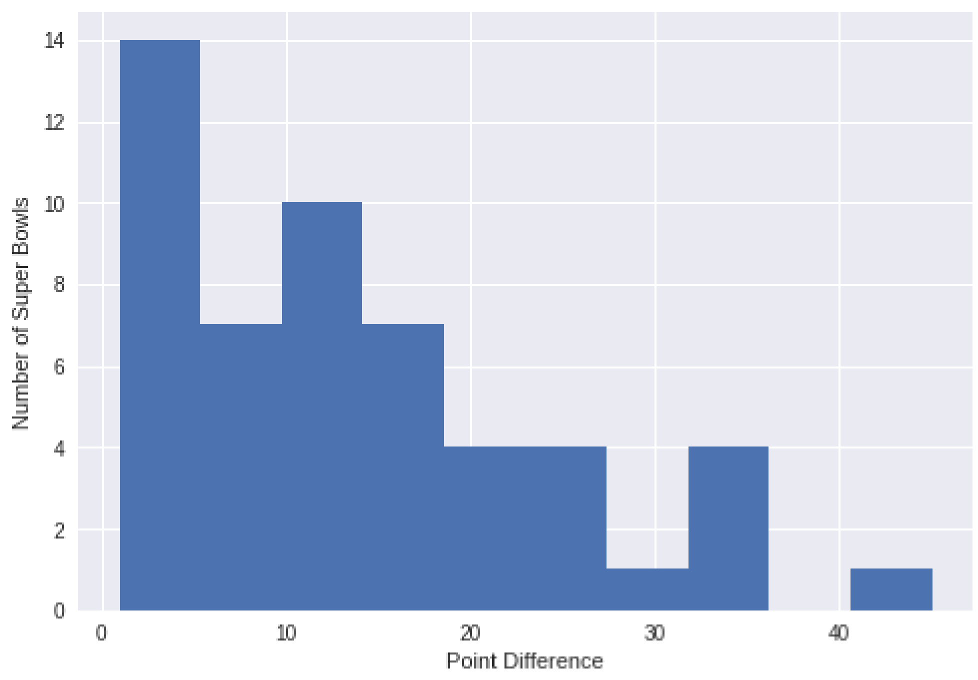
Going down to the lowest combined scores, we have Super Bowl III and VII, which featured tough defenses that dominated. We also have Super Bowl IX in New Orleans in 1975, whose 16-6 score can be attributed to inclement weather. The field was slick from overnight rain, and it was cold at 46 °F (8 °C), making it hard for the Steelers and Vikings to do much offensively. This was the second-coldest Super Bowl ever and the last to be played in inclement weather for over 30 years. The NFL realized people like points, I guess.

UPDATE: In Super Bowl LIII in 2019, the Patriots and Rams broke the record for the lowest-scoring Super Bowl with a combined score of 16 points (13-3 for the Patriots).

Let's take a look at point *difference* now.

In [132]:

```
# Plot a histogram of point differences
plt.hist(super_bowls.difference_pts)
plt.xlabel('Point Difference')
plt.ylabel('Number of Super Bowls')
plt.show()

# Display the closest game(s) and biggest blowouts
display(super_bowls[super_bowls["difference_pts"]==1])
display(super_bowls[super_bowls["difference_pts"]>=35])
```



	date	super_bowl	venue	city	state	attendance	team_winner	winning_pts
27	1991-01-27	25	Tampa Stadium	Tampa	Florida	73813	New York Giants	20

	date	super_bowl	venue	city	state	attendance	team_winner	wi
4	2014-02-02	48	MetLife Stadium	East Rutherford	New Jersey	82529	Seattle Seahawks	43
25	1993-01-31	27	Rose Bowl	Pasadena	California	98374	Dallas Cowboys	52
28	1990-01-28	24	Louisiana Superdome	New Orleans	Louisiana	72919	San Francisco 49ers	55
32	1986-01-26	20	Louisiana Superdome	New Orleans	Louisiana	73818	Chicago Bears	46

In [133]:

```
%%nose
# %%nose needs to be included at the beginning of every @tests cell

# One or more tests of the student's code
# The @solution should pass the tests
# The purpose of the tests is to try to catch common errors and
# to give the student a hint on how to resolve these errors

last_input = In[-2]

def test_not_empty_task_4():
    assert "# ... YOUR CODE FOR TASK" not in last_input, \
        "It appears that # ... YOUR CODE FOR TASK X ... is still in the code cel
l, which suggests that you might not have attempted the code for this task. If y
ou have, please delete # ... YOUR CODE FOR TASK X ... from the cell and resubmi
t."
```

Out[133]:

```
1/1 tests passed
```

5. Do blowouts translate to lost viewers?

The vast majority of Super Bowls are close games. Makes sense. Both teams are likely to be deserving if they've made it this far. The closest game ever was when the Buffalo Bills lost to the New York Giants by 1 point in 1991, which was best remembered for Scott Norwood's last-second missed field goal attempt that went *wide right (https://www.youtube.com/watch?v=RPFZCGgjDSg)*, kicking off four Bills Super Bowl losses in a row. Poor Scott. The biggest point discrepancy ever was 45 points (!) where Hall of Famer Joe Montana's led the San Francisco 49ers to victory in 1990, one year before the closest game ever.

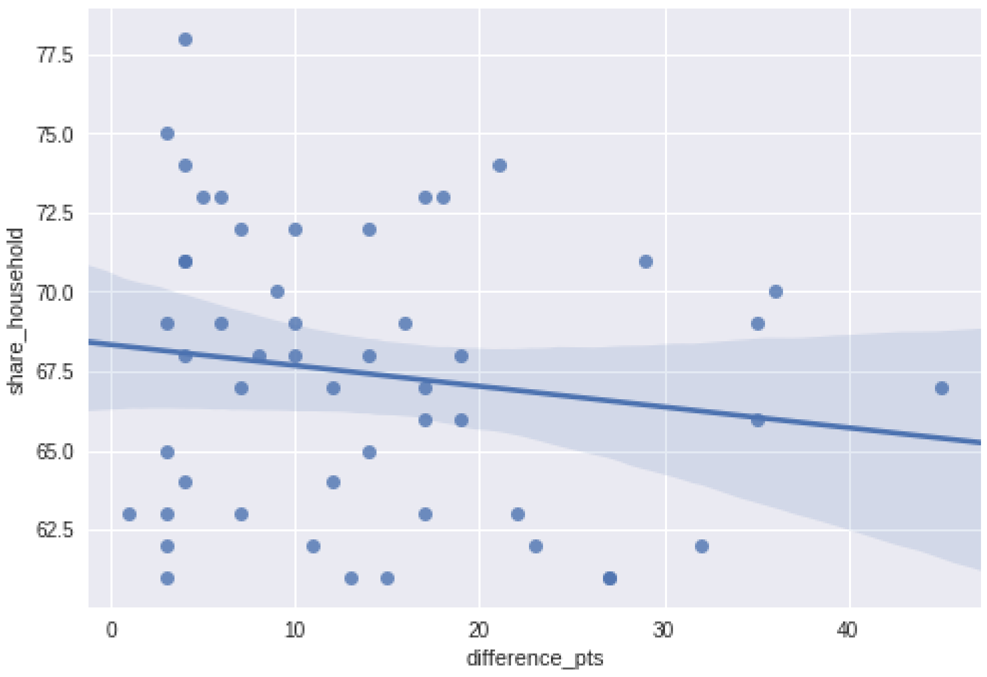
I remember watching the Seahawks crush the Broncos by 35 points (43-8) in 2014, which was a boring experience in my opinion. The game was never really close. I'm pretty sure we changed the channel at the end of the third quarter. Let's combine our game data and TV to see if this is a universal phenomenon. Do large point differences translate to lost viewers? We can plot household share (https://en.wikipedia.org/wiki/Nielsen_ratings) *(average percentage of U.S. households with a TV in use that were watching for the entire broadcast)* vs. point difference to find out.

In [134]:

```
# Join game and TV data, filtering out SB I because it was split over two networ
ks
games_tv = pd.merge(tv[tv['super_bowl'] > 1], super_bowls, on='super_bowl')

# Import seaborn
import seaborn as sns

# Create a scatter plot with a linear regression model fit
sns.regplot(x="difference_pts", y="share_household", data=games_tv)
```

Out[134]:

```
<matplotlib.axes._subplots.AxesSubplot at 0x7fdd69720320>
```

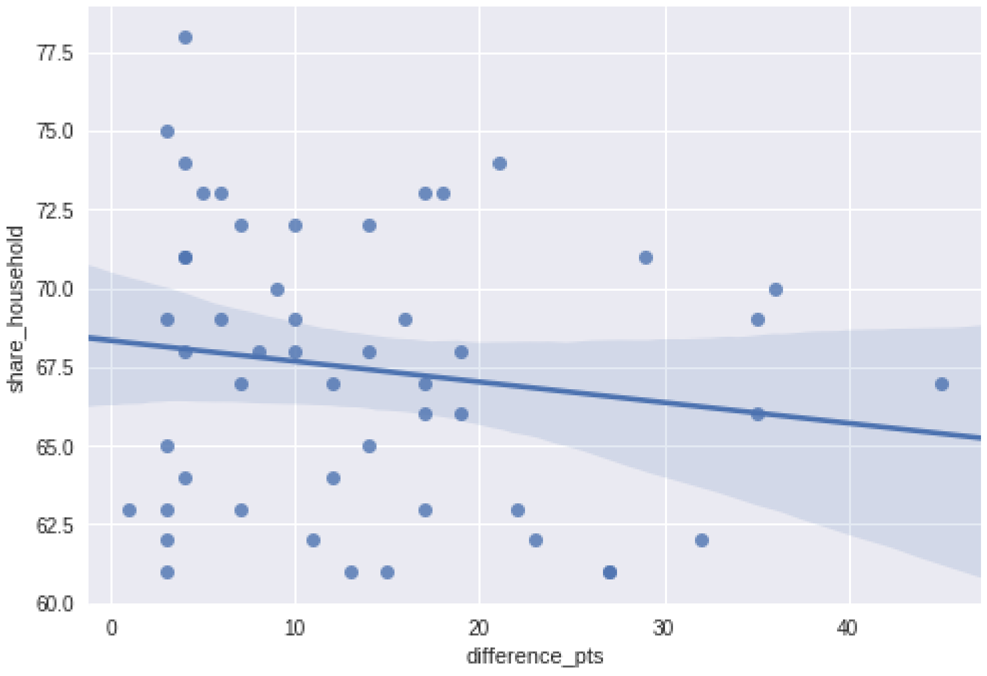


In [135]:

```
%%nose
# %%nose needs to be included at the beginning of every @tests cell

# One or more tests of the student's code
# The @solution should pass the tests
# The purpose of the tests is to try to catch common errors and
# to give the student a hint on how to resolve these errors

last_value = _

def test_seaborn_loaded():
    assert 'sns' in globals(), \
    'Did you import the seaborn module under the alias sns?'

def test_plot_exists_5():
    try:
        assert type(last_value) == type(sns.regplot(x='difference_pts', y='share
_household', data=games_tv))
    except AssertionError:
        assert False, 'A plot was not the last output of the code cell.'
```

Out[135]:

```
2/2 tests passed
```



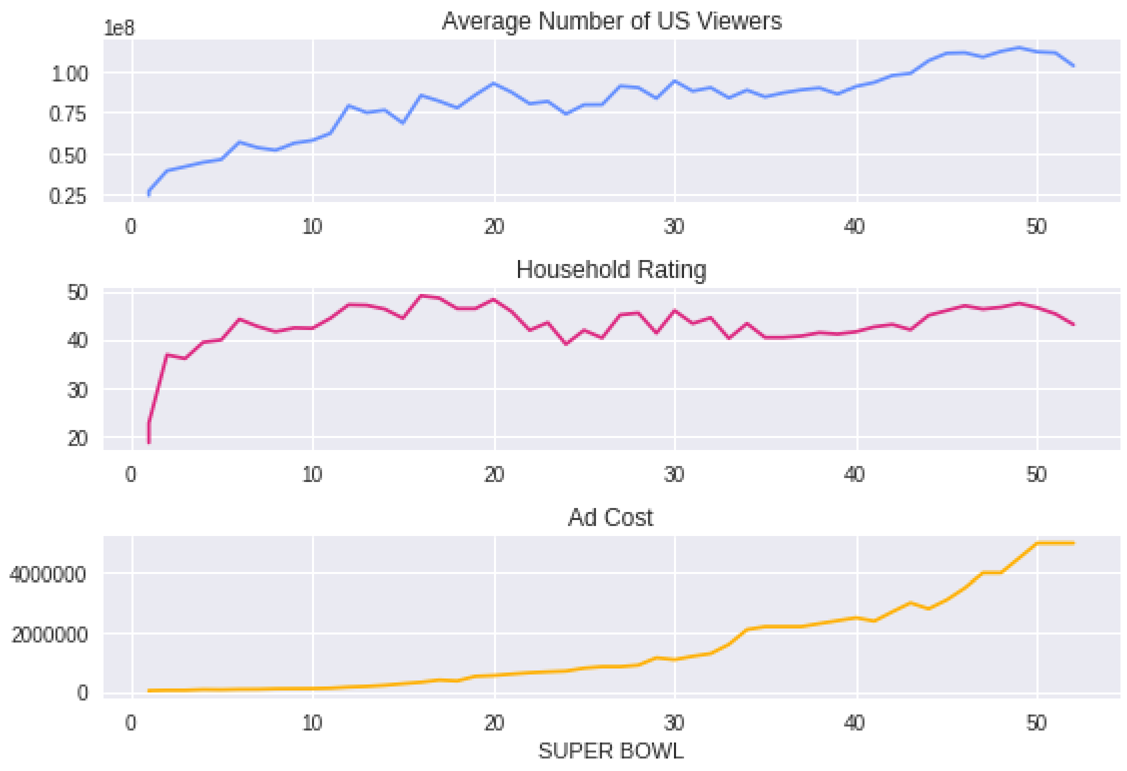
6. Viewership and the ad industry over time

The downward sloping regression line and the 95% confidence interval for that regression *suggest* that bailing on the game if it is a blowout is common. Though it matches our intuition, we must take it with a grain of salt because the linear relationship in the data is weak due to our small sample size of 52 games.

Regardless of the score though, I bet most people stick it out for the halftime show, which is good news for the TV networks and advertisers. A 30-second spot costs a pretty $5 million (https://www.businessinsider.com/super-bowl-commercials-cost-more-than-eagles-quarterback-earns-2018-1) now, but has it always been that way? And how have number of viewers and household ratings trended alongside ad cost? We can find out using line plots that share a "Super Bowl" x-axis.

In [136]:

```
# Create a figure with 3x1 subplot and activate the top subplot
plt.subplot(3, 1, 1)
plt.plot(tv.super_bowl, tv.avg_us_viewers, color='#648FFF')
plt.title('Average Number of US Viewers')

# Activate the middle subplot
plt.subplot(3, 1, 2)
plt.plot(tv.super_bowl, tv.rating_household, color='#DC267F')
plt.title('Household Rating')

# Activate the bottom subplot
plt.subplot(3, 1, 3)
plt.plot(tv.super_bowl,tv.ad_cost, color='#FFB000')
plt.title('Ad Cost')
plt.xlabel('SUPER BOWL')

# Improve the spacing between subplots
plt.tight_layout()
```



In [137]:

```
%%nose
# %%nose needs to be included at the beginning of every @tests cell

# One or more tests of the student's code
# The @solution should pass the tests
# The purpose of the tests is to try to catch common errors and
# to give the student a hint on how to resolve these errors

last_input = In[-2]

def test_not_empty_task_6():
    assert "# ... YOUR CODE FOR TASK" not in last_input, \
        "It appears that # ... YOUR CODE FOR TASK X ... is still in the code cel
l, which suggests that you might not have attempted the code for this task. If y
ou have, please delete # ... YOUR CODE FOR TASK X ... from the cell and resubmi
t."
```

Out[137]:

```
1/1 tests passed
```

7. Halftime shows weren't always this great

We can see viewers increased before ad costs did. Maybe the networks weren't very data savvy and were slow to react? Makes sense since DataCamp didn't exist back then.

Another hypothesis: maybe halftime shows weren't that good in the earlier years? The modern spectacle of the Super Bowl has a lot to do with the cultural prestige of big halftime acts. I went down a YouTube rabbit hole and it turns out the old ones weren't up to today's standards. Some offenders:

- Super Bowl XXVI (https://youtu.be/6wMXHxWO4ns?t=263) in 1992: A Frosty The Snowman rap performed by children.
- Super Bowl XXIII (https://www.youtube.com/watch?v=PKQTL1PYSag) in 1989: An Elvis impersonator that did magic tricks and didn't even sing one Elvis song.
- Super Bowl XXI (https://youtu.be/oSXMNbK2e98?t=436) in 1987: Tap dancing ponies. (Okay, that's pretty awesome actually.)

It turns out Michael Jackson's Super Bowl XXVII performance, one of the most watched events in American TV history, was when the NFL realized the value of Super Bowl airtime and decided they needed to sign big name acts from then on out. The halftime shows before MJ indeed weren't that impressive, which we can see by filtering our `halftime_musician` data.

In [138]:

```
halftime_musicians[halftime_musicians.super_bowl <= 27]
```

Out[138]:

	super_bowl	musician	num_songs
80	27	Michael Jackson	5.0
81	26	Gloria Estefan	2.0
82	26	University of Minnesota Marching Band	NaN
83	25	New Kids on the Block	2.0
84	24	Pete Fountain	1.0
85	24	Doug Kershaw	1.0
86	24	Irma Thomas	1.0
87	24	Pride of Nicholls Marching Band	NaN
88	24	The Human Jukebox	NaN
89	24	Pride of Acadiana	NaN
90	23	Elvis Presto	7.0
91	22	Chubby Checker	2.0
92	22	San Diego State University Marching Aztecs	NaN
93	22	Spirit of Troy	NaN
94	21	Grambling State University Tiger Marching Band	8.0
95	21	Spirit of Troy	8.0
96	20	Up with People	NaN
97	19	Tops In Blue	NaN
98	18	The University of Florida Fightin' Gator March...	7.0
99	18	The Florida State University Marching Chiefs	7.0
100	17	Los Angeles Unified School District All City H...	NaN
101	16	Up with People	NaN
102	15	The Human Jukebox	NaN
103	15	Helen O'Connell	NaN
104	14	Up with People	NaN
105	14	Grambling State University Tiger Marching Band	NaN
106	13	Ken Hamilton	NaN
107	13	Gramacks	NaN
108	12	Tyler Junior College Apache Band	NaN
109	12	Pete Fountain	NaN
110	12	Al Hirt	NaN
111	11	Los Angeles Unified School District All City H...	NaN
112	10	Up with People	NaN

	super_bowl	musician	num_songs
113	9	Mercer Ellington	NaN
114	9	Grambling State University Tiger Marching Band	NaN
115	8	University of Texas Longhorn Band	NaN
116	8	Judy Mallett	NaN
117	7	University of Michigan Marching Band	NaN
118	7	Woody Herman	NaN
119	7	Andy Williams	NaN
120	6	Ella Fitzgerald	NaN
121	6	Carol Channing	NaN
122	6	Al Hirt	NaN
123	6	United States Air Force Academy Cadet Chorale	NaN
124	5	Southeast Missouri State Marching Band	NaN
125	4	Marguerite Piazza	NaN
126	4	Doc Severinsen	NaN
127	4	Al Hirt	NaN
128	4	The Human Jukebox	NaN
129	3	Florida A&M University Marching 100 Band	NaN
130	2	Grambling State University Tiger Marching Band	NaN
131	1	University of Arizona Symphonic Marching Band	NaN
132	1	Grambling State University Tiger Marching Band	NaN
133	1	Al Hirt	NaN

In [139]:

```
%%nose
# %%nose needs to be included at the beginning of every @tests cell

# One or more tests of the student's code
# The @solution should pass the tests
# The purpose of the tests is to try to catch common errors and
# to give the student a hint on how to resolve these errors

last_value = _

def test_head_output():
    try:
        assert "Wynonna Judd" not in last_value.to_string()
    except AttributeError:
        assert False, "Please do not use the display() or print() functions to d
isplay the filtered DataFrame. Write your line of code as the last line in the c
ell instead."
    except AssertionError:
        assert False, "Hmm, it seems halftime_musicians wasn't filtered correctl
y and/or displayed as the last output of the cell. Michael Jackson's performance
should be the first row displayed."
```

Out[139]:

```
1/1 tests passed
```

8. Who has the most halftime show appearances?

Lots of marching bands. American jazz clarinetist Pete Fountain. Miss Texas 1973 playing a violin. Nothing against those performers, they're just simply not Beyoncé (https://www.youtube.com/watch?v=suIg9kTGBVI). To be fair, no one is.

Let's see all of the musicians that have done more than one halftime show, including their performance counts.

```
In [140]:
```

```
# Count halftime show appearances for each musician and sort them from most to l
east
halftime_appearances = halftime_musicians.groupby('musician').count()['super_bow
l'].reset_index()
halftime_appearances = halftime_appearances.sort_values('super_bowl', ascending=
False)

halftime_appearances[halftime_appearances.super_bowl >=2]
```

```
Out[140]:
```

	musician	super_bowl
28	Grambling State University Tiger Marching Band	6
104	Up with People	4
1	Al Hirt	4
83	The Human Jukebox	3
76	Spirit of Troy	2
25	Florida A&M University Marching 100 Band	2
26	Gloria Estefan	2
102	University of Minnesota Marching Band	2
10	Bruno Mars	2
64	Pete Fountain	2
5	Beyoncé	2
36	Justin Timberlake	2
57	Nelly	2
44	Los Angeles Unified School District All City H...	2

In [141]:

```
%%nose
# %%nose needs to be included at the beginning of every @tests cell

# One or more tests of the student's code
# The @solution should pass the tests
# The purpose of the tests is to try to catch common errors and
# to give the student a hint on how to resolve these errors

last_value = _

def test_filter_correct_8():
    try:
        assert len(last_value) == 14
    except TypeError:
        assert False, "Hmm, it seems halftime_appearances wasn't filtered correc
tly and/or displayed as the last line of code in the cell (i.e., displayed witho
ut the display() or print() functions). There should be 14 repeat halftime show
 acts."
    except AssertionError:
        assert False, "Hmm, it seems halftime_appearances wasn't filtered correc
tly. There should be 14 repeat halftime show acts."
```

Out[141]:

```
1/1 tests passed
```

9. Who performed the most songs in a halftime show?

The world famous Grambling State University Tiger Marching Band (https://www.youtube.com/watch?v=RL_3oqpHiDg) takes the crown with six appearances. Beyoncé, Justin Timberlake, Nelly, and Bruno Mars are the only post-Y2K musicians with multiple appearances (two each).

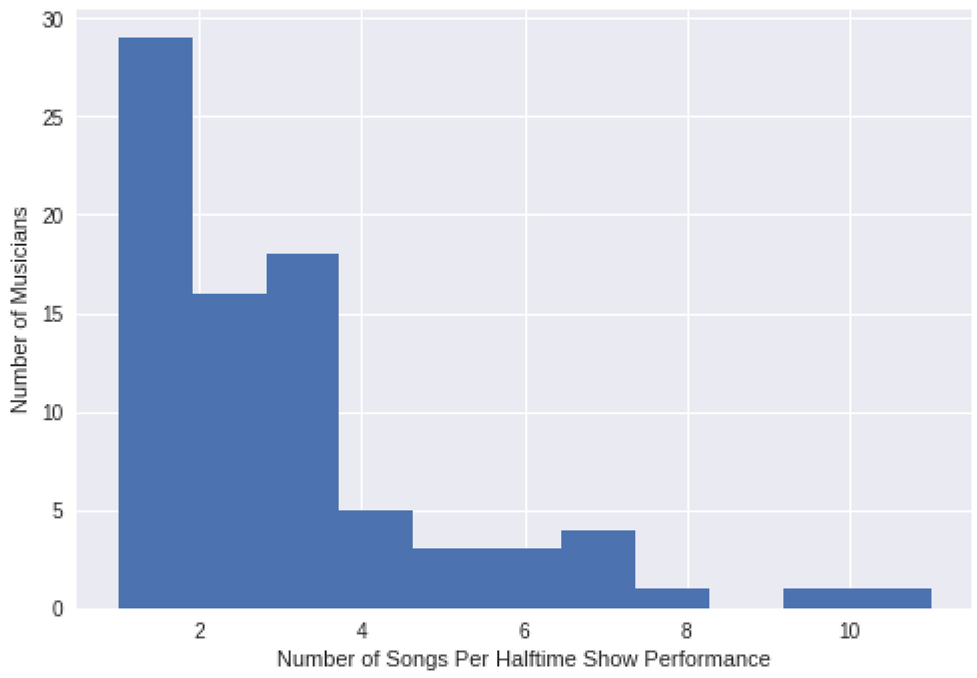
From our previous inspections, the `num_songs` column has lots of missing values:

- A lot of the marching bands don't have `num_songs` entries.
- For non-marching bands, missing data starts occurring at Super Bowl XX.

Let's filter out marching bands by filtering out musicians with the word "Marching" in them and the word "Spirit" (a common naming convention for marching bands is "Spirit of [something]"). Then we'll filter for Super Bowls after Super Bowl XX to address the missing data issue, *then* let's see who has the most number of songs.

In [142]:

```
# Filter out most marching bands
no_bands = halftime_musicians[~halftime_musicians.musician.str.contains('Marchin
g')]
no_bands = no_bands[~no_bands.musician.str.contains('Spirit')]

# Plot a histogram of number of songs per performance
most_songs = int(max(no_bands['num_songs'].values))
plt.hist(no_bands.num_songs.dropna(), bins=most_songs)
plt.xlabel('Number of Songs Per Halftime Show Performance')
plt.ylabel('Number of Musicians')
plt.show()

# Sort the non-band musicians by number of songs per appearance...
no_bands = no_bands.sort_values('num_songs', ascending=False)
# ...and display the top 15
display(no_bands.head(15))
```



	super_bowl	musician	num_songs
0	52	Justin Timberlake	11.0
70	30	Diana Ross	10.0
10	49	Katy Perry	8.0
2	51	Lady Gaga	7.0
90	23	Elvis Presto	7.0
33	41	Prince	7.0
16	47	Beyoncé	7.0
14	48	Bruno Mars	6.0
3	50	Coldplay	6.0
25	45	The Black Eyed Peas	6.0
20	46	Madonna	5.0
30	44	The Who	5.0
80	27	Michael Jackson	5.0
64	32	The Temptations	4.0
36	39	Paul McCartney	4.0

In [143]:

```
%%nose
# %%nose needs to be included at the beginning of every @tests cell

# One or more tests of the student's code
# The @solution should pass the tests
# The purpose of the tests is to try to catch common errors and
# to give the student a hint on how to resolve these errors

last_input = In[-2]

def test_not_empty_task_9():
    assert "# ... YOUR CODE FOR TASK" not in last_input, \
        "It appears that # ... YOUR CODE FOR TASK X ... is still in the code cel
l, which suggests that you might not have attempted the code for this task. If y
ou have, please delete # ... YOUR CODE FOR TASK X ... from the cell and resubmi
t."
```

Out[143]:

```
1/1 tests passed
```

10. Conclusion

So most non-band musicians do 1-3 songs per halftime show. It's important to note that the duration of the halftime show is fixed (roughly 12 minutes) so songs per performance is more a measure of how many hit songs you have. JT went off in 2018, wow. 11 songs! Diana Ross comes in second with 10 in her medley in 1996.

In this notebook, we loaded, cleaned, then explored Super Bowl game, television, and halftime show data. We visualized the distributions of combined points, point differences, and halftime show performances using histograms. We used line plots to see how ad cost increases lagged behind viewership increases. And we discovered that blowouts do appear to lead to a drop in viewers.

This year's Big Game will be here before you know it. Who do you think will win Super Bowl LIII?

UPDATE: Spoiler alert (https://en.wikipedia.org/wiki/Super_Bowl_LIII).

In [144]:

```
# 2018-2019 conference champions
patriots = 'New England Patriots'
rams = 'Los Angeles Rams'

# Who will win Super Bowl LIII?
super_bowl_LIII_winner = patriots
print('The winner of Super Bowl LIII will be the', super_bowl_LIII_winner)
```

```
The winner of Super Bowl LIII will be the New England Patriots
```

In [145]:

```
%%nose
# %%nose needs to be included at the beginning of every @tests cell

# One or more tests of the student's code
# The @solution should pass the tests
# The purpose of the tests is to try to catch common errors and
# to give the student a hint on how to resolve these errors

def test_valid_winner_chosen():
    assert super_bowl_LIII_winner == 'New England Patriots' or super_bowl_LIII_w
inner == 'Los Angeles Rams', \
    "It appears a valid potential winner was not selected. Please assign the pat
riots variable or the rams variable to super_bowl_LIII_winner."
```

Out[145]:

```
1/1 tests passed
```