caprinomics (http://www.caprinomics.com/)

KickStarter Consulting

Analysis of KickStarter Success

Kickstarter, the crowdfunding platform, is thinking about providing a consulting service to project founders to help its customers create more successful crowdfunding campaigns. You've been asked to do some initial analysis.

In order to complete this assignment, please follow the link and download the following Kickstarter dataset: http://bit.ly/2cgMGDm (http://bit.ly/2cgMGDm). Use the dataset to complete this task.

A short and simple Google Slide of the results can be seen here:http://bit.ly/2y760Nx (http://bit.ly/2y760Nx)

Loading and Inspecting the Kickstarter Data

In [1]:
```
# Set autosave to 3 minutes
%autosave 180
# ipython magic
%pylab inline
# Import modules
import os
import datetime as dt
from scipy import stats
import numpy as np
import pandas as pd
import seaborn as sns
sns.set(style="whitegrid", color_codes=True)
import matplotlib as mpl
import matplotlib.pyplot as plt
matplotlib.style.use('seaborn-paper')
```

```
Autosaving every 180 seconds
Populating the interactive namespace from numpy and ma
tplotlib
```

In [2]:
```
# The following code reads all the Kickstarter data int
o Pandas DataFrames
# empty data will be filled in as NaN
# The data can be found at http://bit.ly/2cgMGDm
kickstarter_df = pd.read_csv('DSI_kickstarterscrape_dat
aset.csv', keep_default_na=True)
# Rename columns with spaces to _
kickstarter_df.rename(columns={'project id': 'project_i
d', 'funded percentage': 'funded_percentage', 'funded d
ate': 'funded_date', 'reward levels': 'reward_levels'},
inplace=True)
# Set index as 'project_id'
kickstarter_df.set_index(['project_id'], inplace=True)
```

Read all the Kickstarter csv data into Pandas DataFrames. Empty data will be filled in as NaN. The data can be found at http://bit.ly/2cgMGDm (http://bit.ly/2cgMGDm)

In [3]:
```
# Return shape and size to see how much data is being w
orked with
rows, columns = kickstarter_df.shape
print 'Rows:', rows
print 'Columns:', columns
print round(float(os.stat('DSI_kickstarterscrape_datase
t.csv').st_size / 1000000.0), 2), 'MB'
```

```
Rows: 45957
Columns: 16
12.92 MB
```

The dataset is 45,957 rows long, 18 columns wide, and 12.9 MB. Not a very large dataset; but certainly large enough to justify Python over Excel...

```
In [4]: # Show counts and data types for all columns
        kickstarter_df.info()
```

```
Int64Index: 45957 entries, 39409 to 2147460119
Data columns (total 16 columns):
name 45957 non-null object
url 45957 non-null object
category 45957 non-null object
subcategory 45957 non-null object
location 44635 non-null object
status 45957 non-null object
goal 45957 non-null float64
pledged 45945 non-null float64
funded_percentage 45957 non-null float64
backers 45957 non-null int64
funded_date 45957 non-null object
levels 45957 non-null int64
reward_levels 45898 non-null object
updates 45957 non-null int64
comments 45957 non-null int64
duration 45957 non-null float64
dtypes: float64(4), int64(4), object(8)
memory usage: 6.0+ MB
```

```
In [5]: # Find how many missing values and percentage missing i
        n each column
        print 'Missing data in each Kickstarter dataframe colum
        n:'
        for c in kickstarter_df.columns:
         missing_data = len(kickstarter_df) - kickstarter_df[c]
        .count()
         if (missing_data > 0 or missing_data =='NaN'):
         print c, ':', missing_data, 'missing values is', str(r
        ound(float(missing_data / float(len(kickstarter_df))) *
         100, 3)), '% of total'
```

```
Missing data in each Kickstarter dataframe column:
location : 1322 missing values is 2.877 % of total
pledged : 12 missing values is 0.026 % of total
reward_levels : 59 missing values is 0.128 % of total
```

All columns have data except 'location' and 'reward levels'. But still very complete with very little missing values as percentage of whole.

In [6]:
```
# Inspect the data with .head
kickstarter_df.head(5)
```

Out[6]:

	name	url
project_id		
39409	WHILE THE TREES SLEEP	http://www.kickstarter.com/
126581	Educational Online Trading Card Game	http://www.kickstarter.com/
138119	STRUM	http://www.kickstarter.com/
237090	GETTING OVER - One son's search to finally kno...	http://www.kickstarter.com/
246101	The Launch of FlyeGrlRoyalty "The New Nam...	http://www.kickstarter.com/

The .head function gives us a preview of the data without having to load the entire dataset. It shows all the 'columns' of data for each project 'row'. One thing I notice is that the 'status' column is a string of either 'successful, failed, live'.

In [7]:
```
# Get some quick and dirty statistics with pd.describe
kickstarter_df.describe()
```

Out[7]:

	goal	pledged	funded_pe
count	4.595700e+04	4.594500e+04	45957.000
mean	1.194271e+04	4.980750e+03	1.850129
std	1.887583e+05	5.674162e+04	88.492706
min	1.000000e-02	0.000000e+00	0.000000
25%	1.800000e+03	1.960000e+02	0.044000
50%	4.000000e+03	1.310000e+03	1.000000
75%	9.862000e+03	4.165000e+03	1.115640
max	2.147484e+07	1.026684e+07	15066.000

Part 1: Foundational Data Analysis

1.1: What is the mean (total) pledge that projects get? (not per backer)

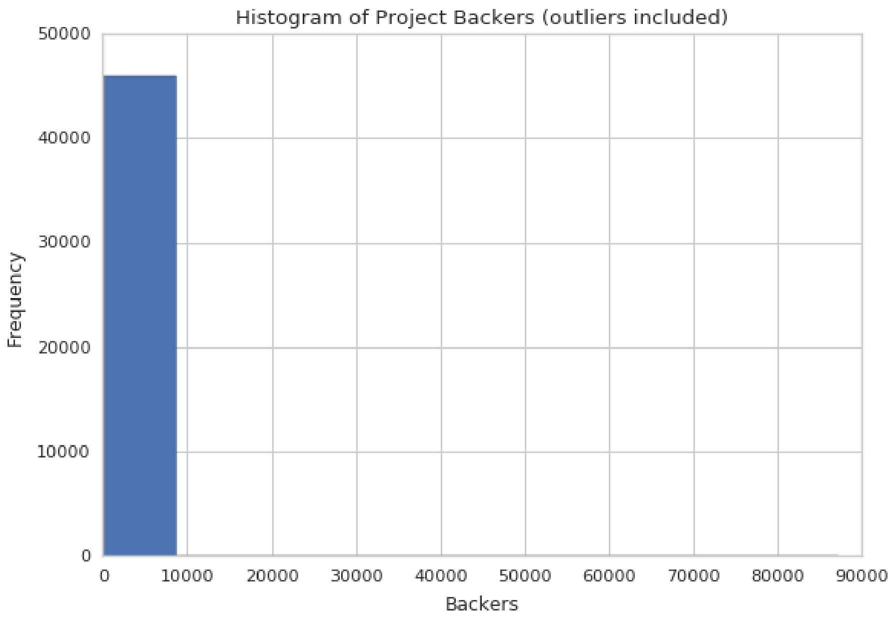
In [8]:
```
# Find mean of 'pledged' column
print "$", round(kickstarter_df.pledged.mean(), 2)
```

```
$ 4980.75
```

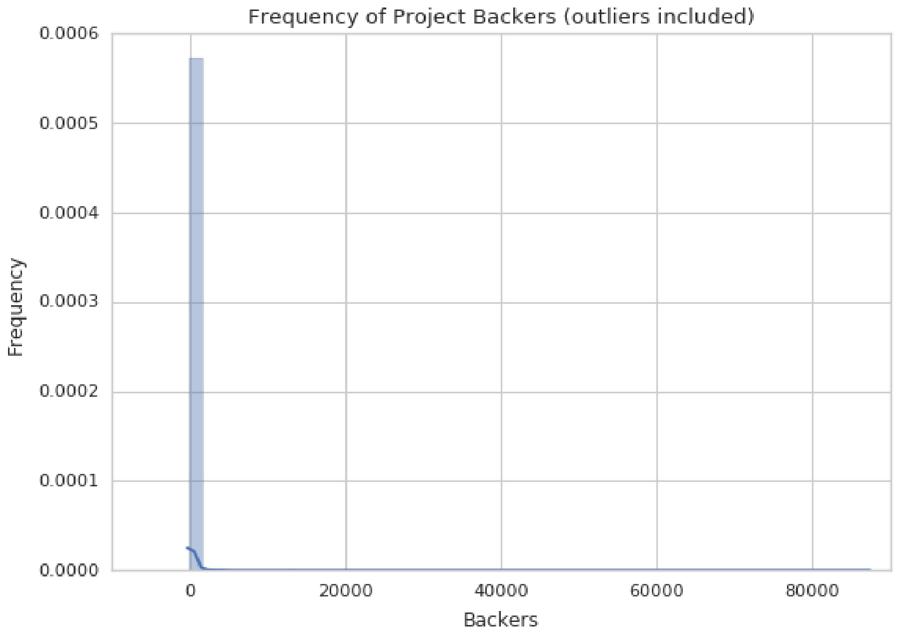
The mean (total) pledge that projects receive is $4980.75.

1.2: Create a historgram that shows the distribution for number of backers. What is the skew of the distribution?

In [50]:
```
# Histogram of 'backers' column
plt.hist(kickstarter_df['backers'])
plt.xlabel('Backers')
plt.ylabel('Frequency')
plt.title('Histogram of Project Backers (outliers inclu
ded)')
plt.show()
```

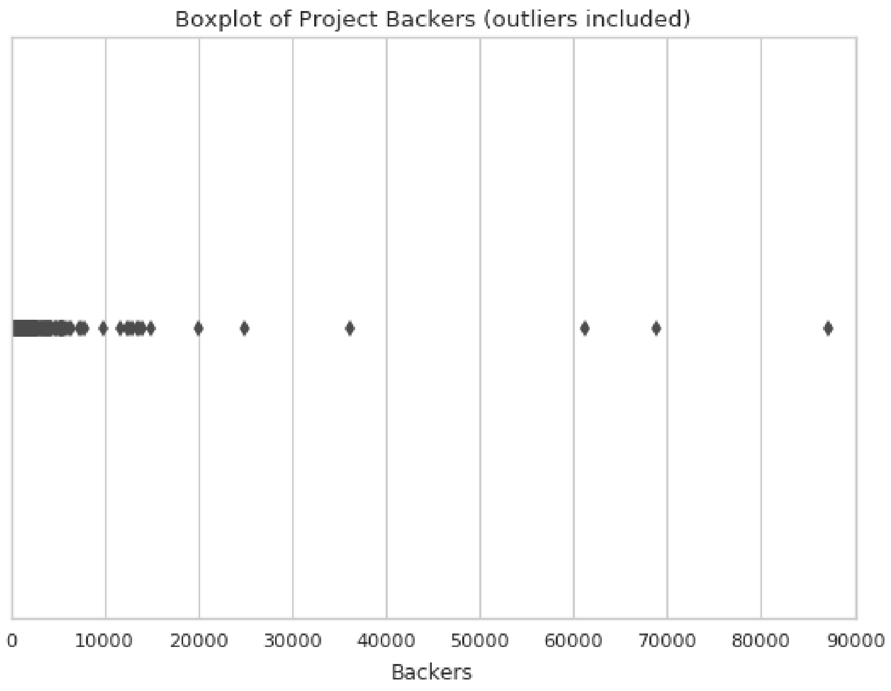


In [51]:
```
# Distribution plot of 'backers' column
sns.distplot(kickstarter_df['backers'])
plt.xlabel('Backers')
plt.ylabel('Frequency')
plt.title('Frequency of Project Backers (outliers inclu
ded)')
plt.show()
```



The above plots are a histogram and a distribution plot of kickstarter backers. It's not very clear. There is **huge** number of projects that receive almost no backers. This would indicate that distribution of kickstarter backers is strongly positive (right-skewed & left leaning). Let's look more into this.

In [52]:
```
# Boxplot of'backers' column
sns.boxplot(x=kickstarter_df['backers'])
plt.xlabel('Backers')
plt.title('Boxplot of Project Backers (outliers include
d)')
plt.show()
```



The boxplot above further proves this skewness. The box (quartiles) is almost not even visible. It appears as a black smudge to the far left of the chart. The extreme outliers are making this data extremely hard to see. Let's look at the numbers and see just how bad the skew is.

In [12]:
```
# Get quick and dirty starts for 'backers'
kickstarter_df['backers'].describe()
```

Out[12]:
```
count 45957.000000
mean 69.973192
std 688.628479
min 0.000000
25% 5.000000
50% 23.000000
75% 59.000000
max 87142.000000
Name: backers, dtype: float64
```

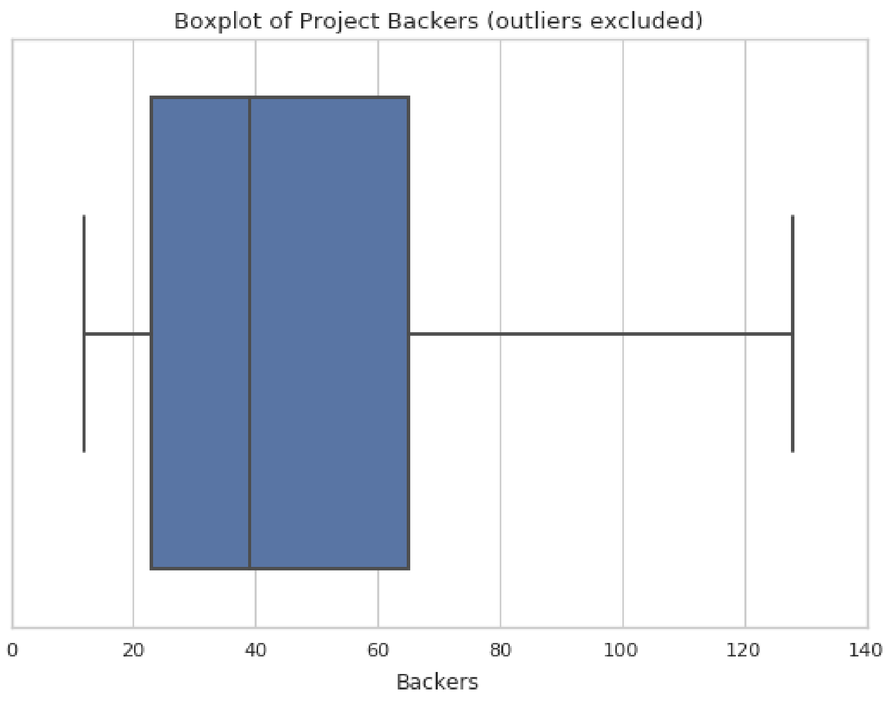
The quick and dirty statistics summary shows some insight as to why. Look at the quartiles. THe lower 25% is 5, the median is 23, the 75% is 59 and the max is a whopping 87142! A standard deviation of 688 show us that the uniform is far from uniform. Let's quickly clean up the outliers so we can see the distribution more clearly.

```
In [13]: # Filter the original kickstarter dataframe to remove l
         ots of outliers
         filtered_df = kickstarter_df[((kickstarter_df.backers -
         kickstarter_df.backers.mean()) / kickstarter_df.backers
         .std()).abs() < .085]
         filtered_df['backers'].describe()
```

```
Out[13]: count 24979.000000
         mean 46.829016
         std 29.133568
         min 12.000000
         25% 23.000000
         50% 39.000000
         75% 65.000000
         max 128.000000
         Name: backers, dtype: float64
```

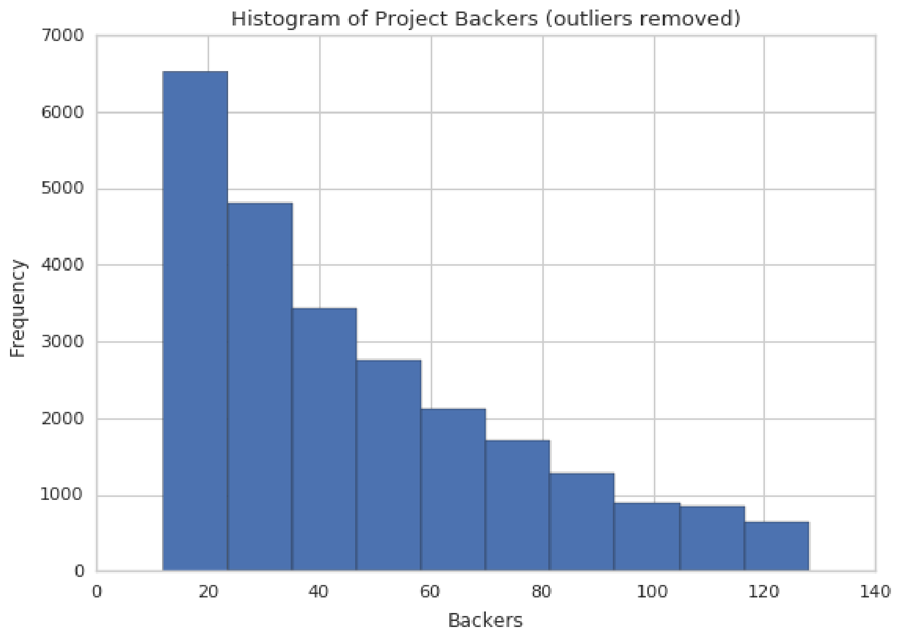
The above summary statistics show that the data has been significantly filtered from statistical outliers. Let's check a box plot next.

```
In [53]: sns.boxplot(x=filtered_df['backers'])
         plt.xlabel('Backers')
         plt.title('Boxplot of Project Backers (outliers exclude
         d)')
         plt.show()
```

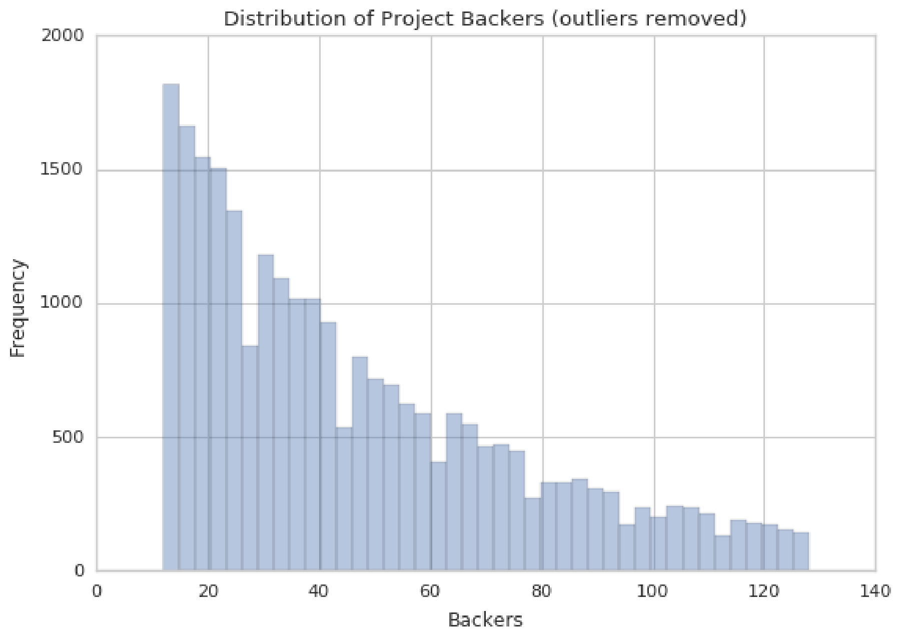


The box plot above shows the extent of the filtering.

In [49]:
```
# Histogram of filtered 'backers' column
plt.hist(filtered_df['backers'])
plt.xlabel('Backers')
plt.ylabel('Frequency')
plt.title('Histogram of Project Backers (outliers remov
ed)')
plt.show()
```

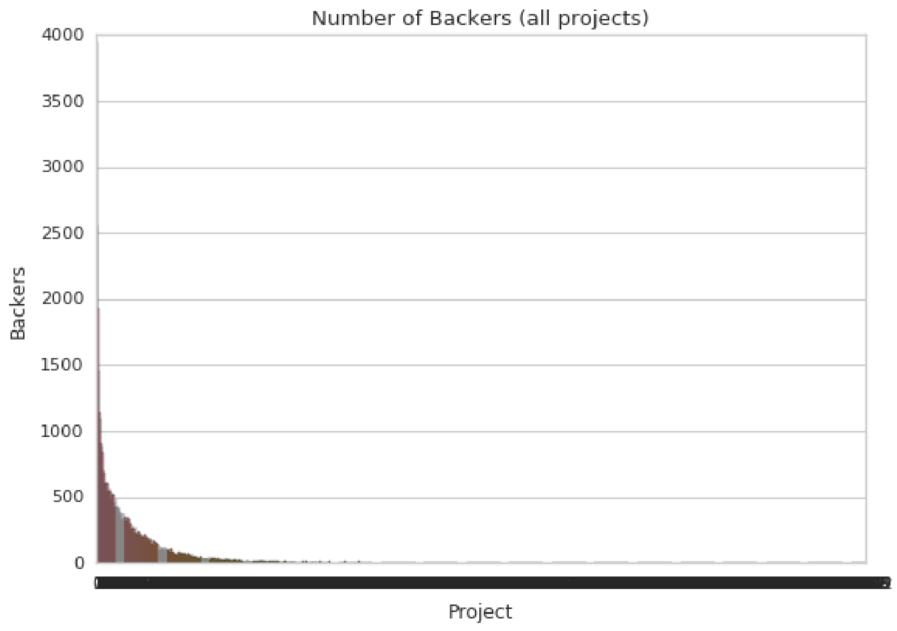


In [54]:
```
# distribution plot of filtered 'backers' column
sns.distplot(filtered_df['backers'], kde=False)
plt.xlabel('Backers')
plt.ylabel('Frequency')
plt.title('Distribution of Project Backers (outliers re
moved)')
plt.show()
```



The histogram above now shows the strongly positive (right-skewed & left leaning) distribution in a way that's easier to understand.

In [56]:
```
# Countplot
sns.countplot(x='backers', data=kickstarter_df)
plt.xlabel('Project')
plt.ylabel('Backers')
plt.title('Number of Backers (all projects)')
plt.show()
```



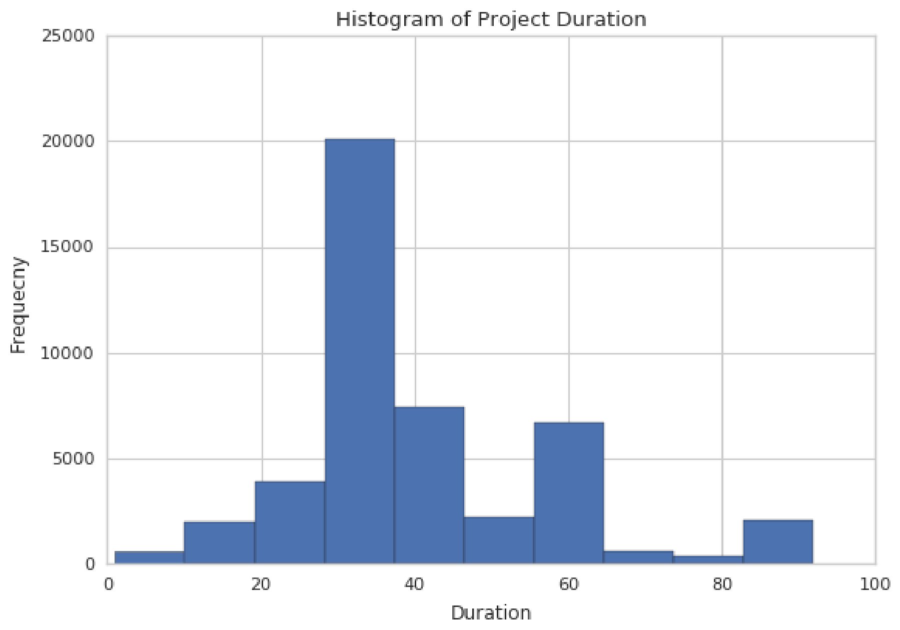
Although not a histogram; the countplot above also shows that very few kickstarter projects get massive amount of backers while the vast majorty receive very little, if any, backers.

1.3a: Is the ‘duration’ variable normally distributed

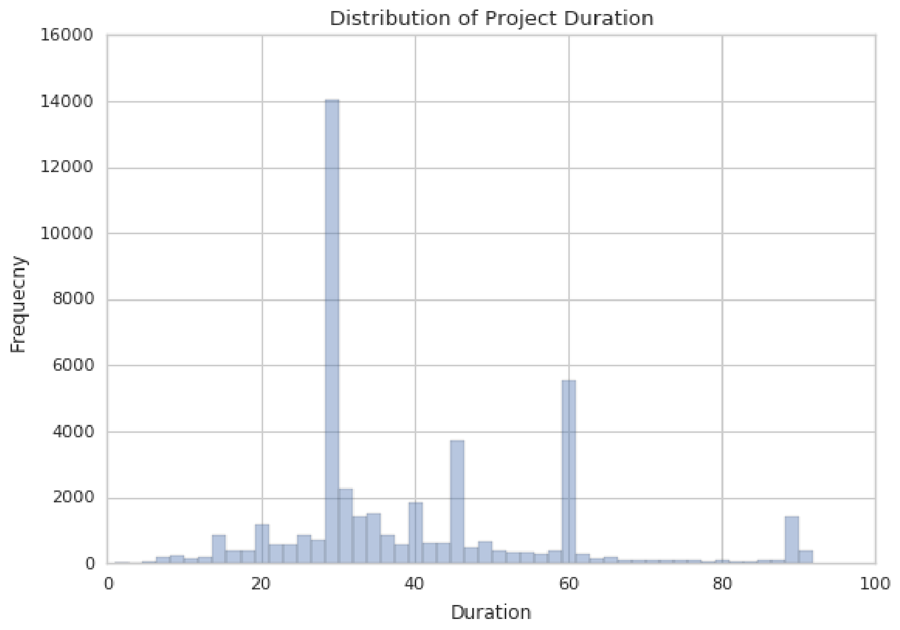
In [18]:
```
# Describe 'duration' column
kickstarter_df['duration'].describe()
```

Out[18]:
```
count 45957.000000
mean 39.995547
std 17.414458
min 1.000000
25% 30.000000
50% 32.000000
75% 48.390000
max 91.960000
Name: duration, dtype: float64
```

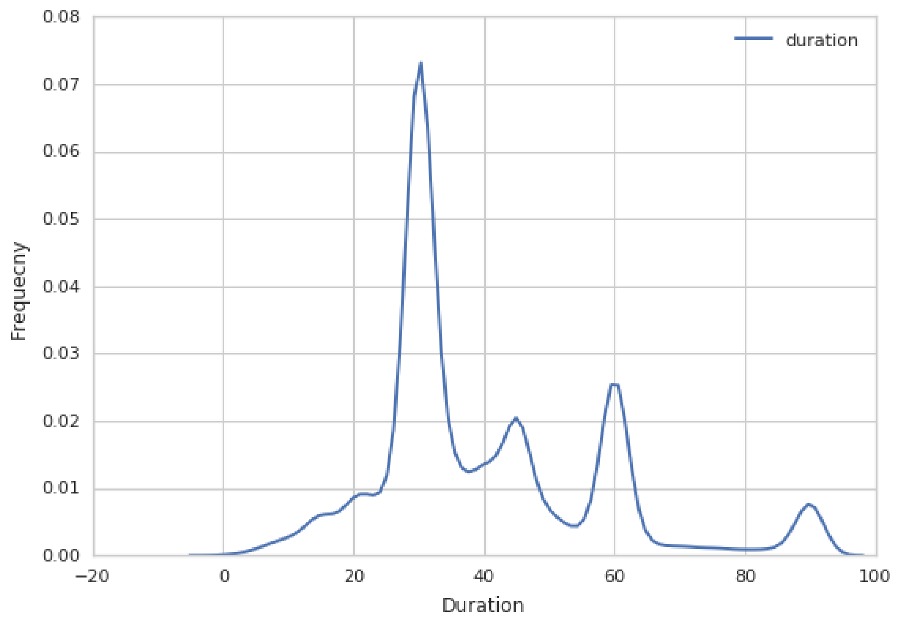
In [57]:
```
# Plot histogram of 'duration' column
plt.hist(kickstarter_df['duration'])
plt.xlabel('Duration')
plt.ylabel('Frequecny')
plt.title('Histogram of Project Duration')
plt.show()
```



In [60]:
```
# Plot distribution plot of 'duration' column
sns.distplot(kickstarter_df['duration'], kde=False)
plt.xlabel('Duration')
plt.ylabel('Frequecny')
plt.title('Distribution of Project Duration')
plt.show()
```

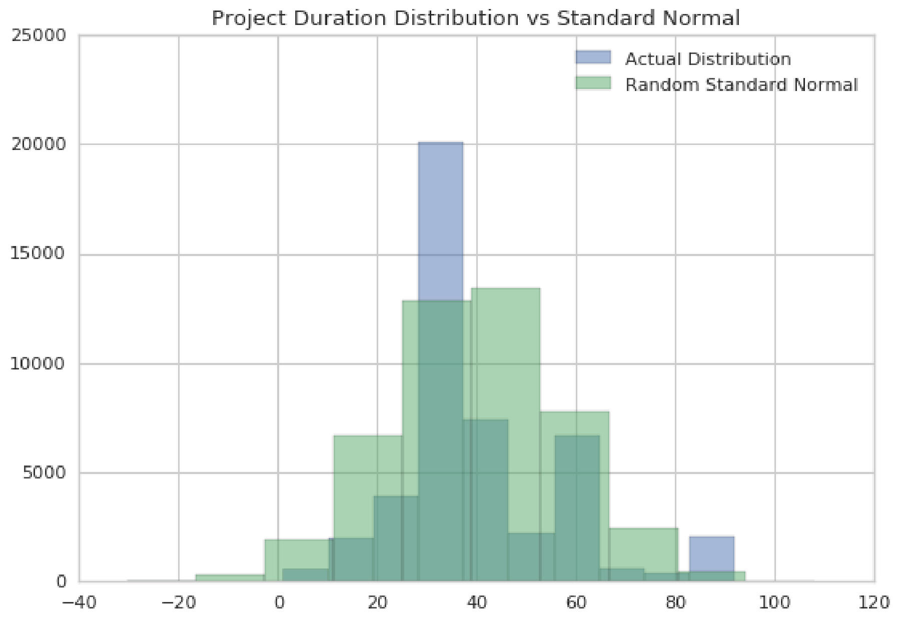


In [46]:
```
# Plot kde of 'duration' column
sns.kdeplot(kickstarter_df['duration'])
plt.xlabel('Duration')
plt.ylabel('Frequecny')
plt.show()
```



The 'duration' column in the dataset looks to be a bit normally distributed looking at it graphically. Let's test it mathematically:

In [61]:
```
# Create normal distribution to overlay on top of actua
l distribution
duration_rand_normal = np.random.normal(loc=kickstarter
_df['duration'].mean(), scale=kickstarter_df['duration'
].std(), size=len(kickstarter_df['duration']))
# Overlay standard normal distribution over actual dist
ribution
pyplot.hist(kickstarter_df['duration'], alpha=0.5, labe
l='Actual Distribution')
pyplot.hist(duration_rand_normal, alpha=0.5, label='Ran
dom Standard Normal')
pyplot.legend(loc='upper right')
pyplot.title('Project Duration Distribution vs Standard
Normal')
pyplot.show()
```



The above chart shows a Gaussian random numbers generated to create a standard normal distribution overlay (green) on top of the actual data (blue). This gives us another idea of how close the actual data is to being standard normal. It's still not conclusive, so let's test it mathematically:

In [23]:
```
# Convert 'duration' column to NumPy array
duration_test = np.array(kickstarter_df['duration'])
# k = z-score, p = p-value
k,p = stats.mstats.normaltest(duration_test)
# If statement to print results of p-test
if p < 0.05:
 print 'P-value =', p, '\nZ-score =', k, '\nDistributio
n is not normal.'
else:
 print 'P-value =', p, '\nZ-score =', k, '\nDistributio
n is normal'
```

```
P-value = 0.0
Z-score = 6985.5754355
Distribution is not normal.
```

1.3b: If you could collect data on another attribute of these projects, what would it be and why?

Although I have not done a very detailed analysis of the data, the one thing that catches my eye is a lack of some sort of way to distinguish which individual backers have contributed to each project. It would be interesting to see what the distribution looks like of each backer to see how many projects they have backed. I imagine this information is in a separate database as it would be very combersome to try and present that information in this .csv. If that was in a separate database or csv some SQL joints could be used for analysis.

Part 2: Qualitative Analysis

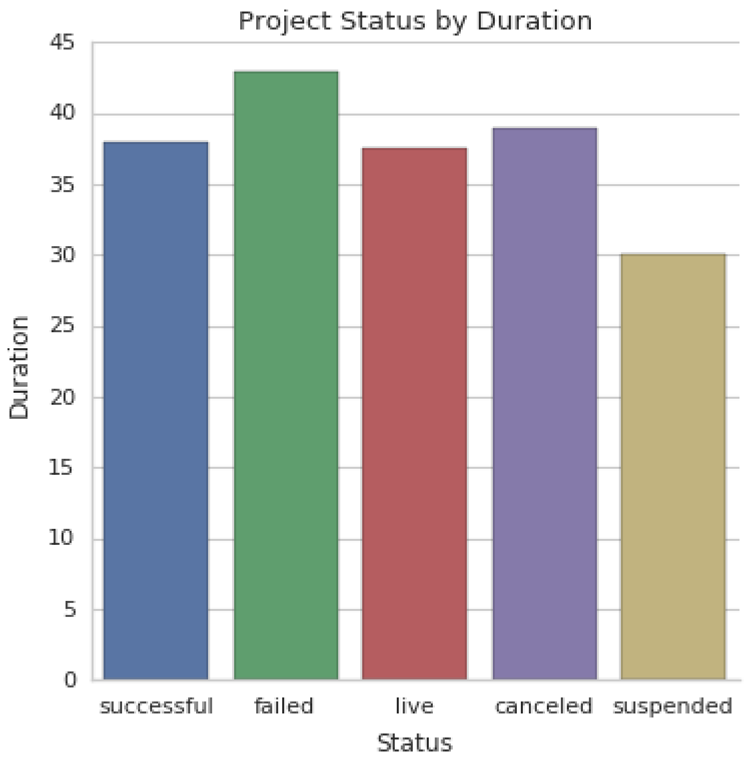
Create a presentation using Google Slides (max. 5 slides) using the data above (and additional data from those tables) that make clear recommendations on how people can create a successful Kickstarter campaign.

In [24]:
```
# Map 'status' column into boolean and make new column
status_bool = {'successful': True, 'failed': False, 'su
spended': False, 'canceled':False}
kickstarter_df['status_bool'] = kickstarter_df['status'
].map(status_bool)
```

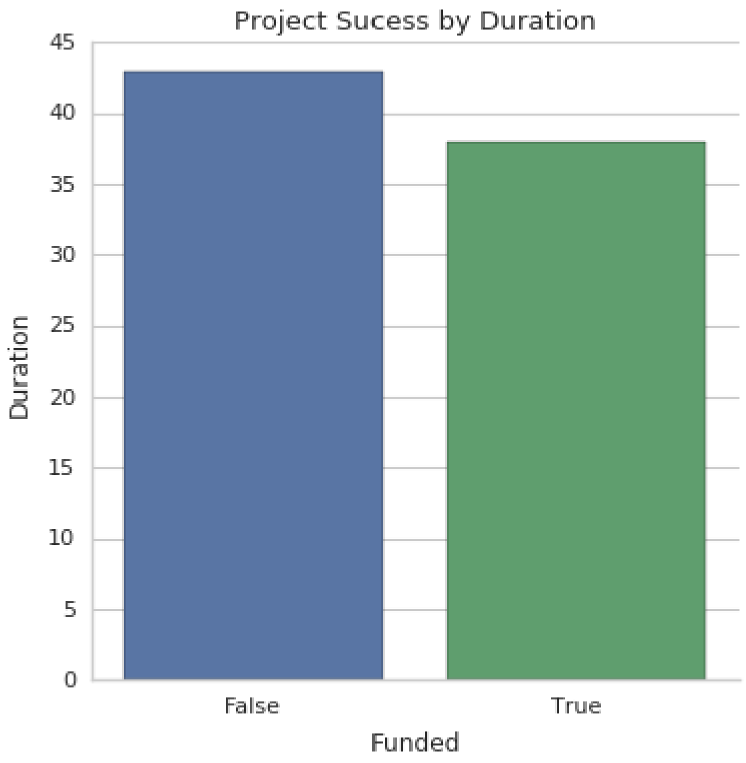
First let's convert the many 'status' options into a boolean of either True or False.

2.1: What's the best length of time to run a campaign?

In [62]:
```
# Plot showing rates of success by duration
sns.factorplot(x='status', y='duration', data=kickstart
er_df, kind='bar', ci=None)
plt.xlabel('Status')
plt.ylabel('Duration')
plt.title('Project Status by Duration')
plt.show()
```



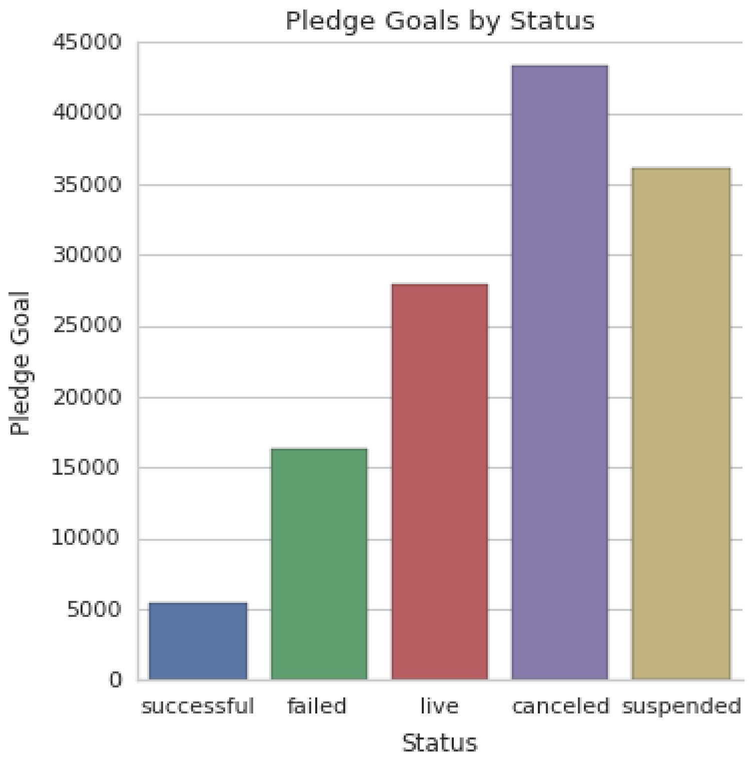
In [63]:
```
# Plot showing rates of success by duration
sns.factorplot(y='duration', x='status_bool', data=kick
starter_df, kind='bar', ci=None)
plt.xlabel('Funded')
plt.ylabel('Duration')
plt.title('Project Sucess by Duration')
plt.show()
```



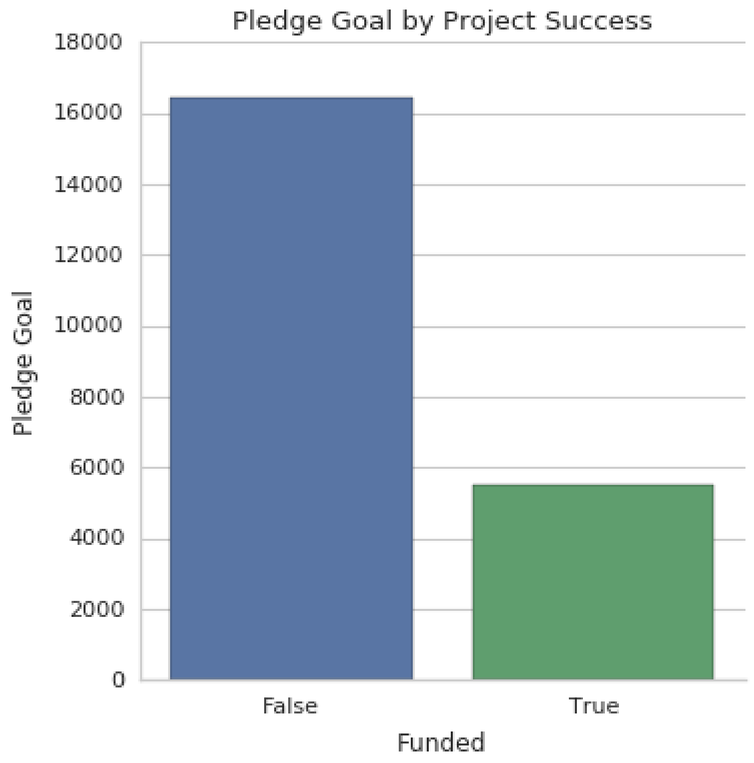
There does not seem to be any correlation between length of time to run a campaign and success. About a month.

2.2: What's the ideal pledge goal?

In [64]:
```
# Plot showing rates of success by duration
sns.factorplot(x='status', y='goal', data=kickstarter_d
f, kind='bar', ci=None)
plt.xlabel('Status')
plt.ylabel('Pledge Goal')
plt.title('Pledge Goals by Status')
plt.show()
```



```
In [65]: # Plot showing rates of success by duration
         sns.factorplot(x='status_bool', y='goal', data=kickstar
         ter_df, kind='bar', ci=None)
         plt.xlabel('Funded')
         plt.ylabel('Pledge Goal')
         plt.title('Pledge Goal by Project Success')
         plt.show()
```



The above plot shots the number of projects organized by status based on their pledge goal. Here one can see that the lower amount the project asks for; the higher chances there are for success.

2.3: What type of projects would be most successful at getting funded?

```
In [29]: # Print categories
         kickstarter_df['category'].unique().tolist()
```

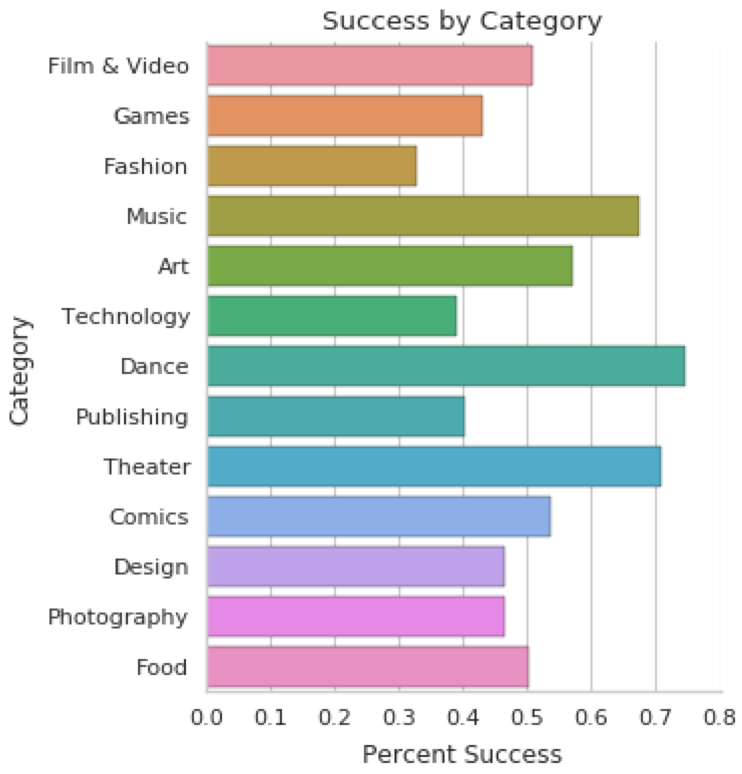
```
Out[29]: ['Film & Video',
          'Games',
          'Fashion',
          'Music',
          'Art',
          'Technology',
          'Dance',
          'Publishing',
          'Theater',
          'Comics',
          'Design',
          'Photography',
          'Food',
          'Film & Video']
```

The above list of unique categories shows there's a category that's showing up twice due to bad data. This needs to be fixed:

```
In [30]: # Replace bad data values in 'category' column
         kickstarter_df['category'] = kickstarter_df['category']
         .replace(['Film & Video'], 'Film & Video')
         kickstarter_df['category'].unique().tolist()
```

```
Out[30]: ['Film & Video',
          'Games',
          'Fashion',
          'Music',
          'Art',
          'Technology',
          'Dance',
          'Publishing',
          'Theater',
          'Comics',
          'Design',
          'Photography',
          'Food']
```

In [68]:
```
# Plot showing rates of success by duration
sns.factorplot(y='category', x='status_bool', data=kick
starter_df, kind='bar', ci=None)
plt.xlabel('Percent Success')
plt.ylabel('Category')
plt.title('Success by Category')
plt.show()
```

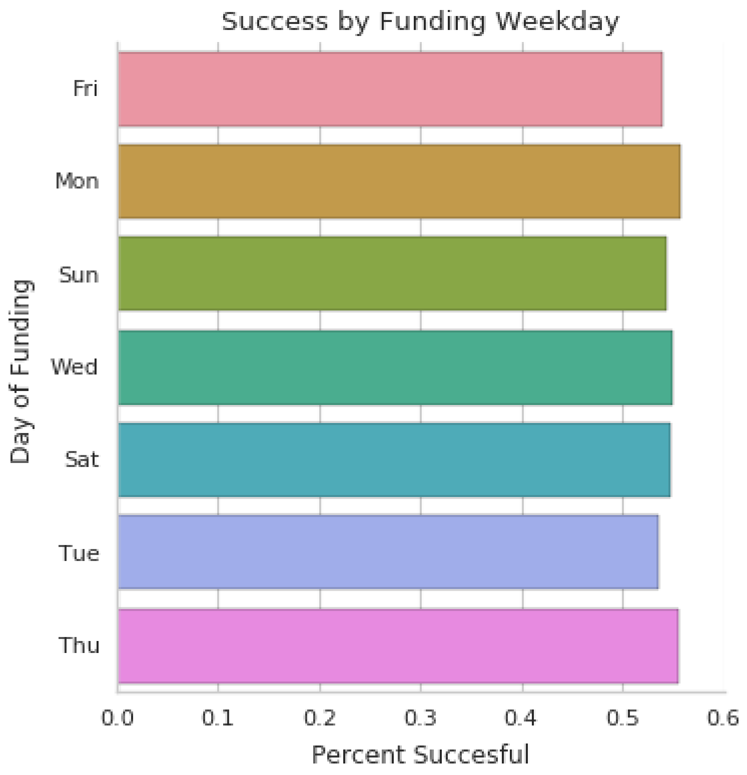


The types of projects most successful to be funded are dance, theater, music, and art. The types of projets least succesful to be funded are fashion, technology, and publishing.

2.4 Is there an ideal month/day/time to launch a campaign?

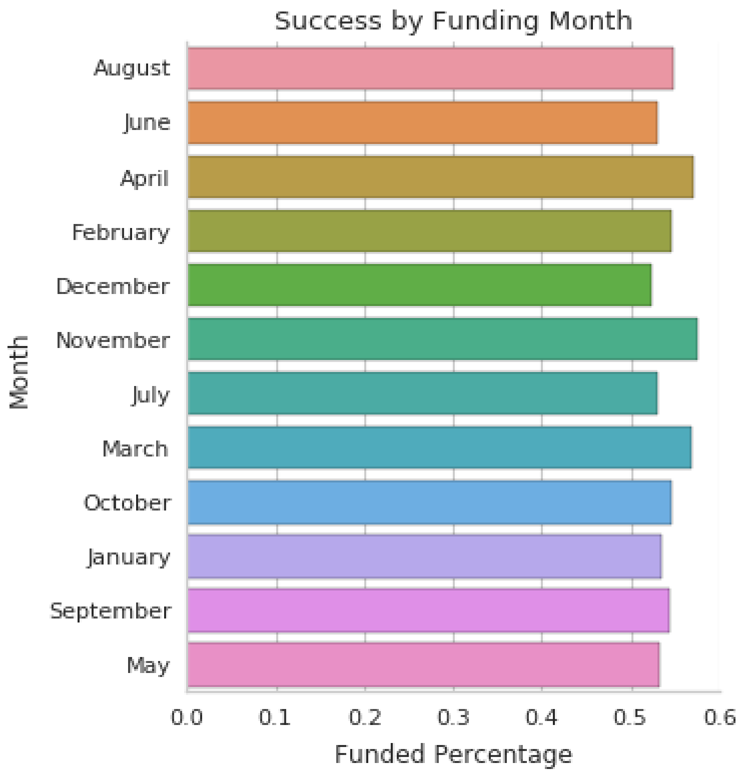
In [32]:
```
# Create 'funded_month' column based on datetime conver
sion from 'funded_date' column
kickstarter_df['funded_month'] = kickstarter_df['funded
_date'].apply(lambda x: dt.datetime.strptime(x, '%a, %d
%b %Y %X -%f').strftime('%B'))
# Create 'funded_weekday' column based on datetime conv
ersion from 'funded_date' column
kickstarter_df['funded_weekday'] = kickstarter_df['fund
ed_date'].apply(lambda x: dt.datetime.strptime(x, '%a,
%d %b %Y %X -%f').strftime('%a'))
```

In [67]:
```
# Plot showing rates of success by weekday
sns.factorplot(y='funded_weekday', x='status_bool', dat
a=kickstarter_df, kind='bar', ci=None)
plt.xlabel('Percent Succesful')
plt.ylabel('Day of Funding')
plt.title('Success by Funding Weekday')
plt.show()
```



There does not appear to be any correlation between success and week day or funding.

In [74]:
```
# Plot showing rates of success by weekday
sns.factorplot(y='funded_month', x='status_bool', data=
kickstarter_df, kind='bar', ci=None)
plt.xlabel('Funded Percentage')
plt.ylabel('Month')
plt.title('Success by Funding Month')
plt.show()
```



There does not appear to be any correlation between success and month of funding.

(https://twitter.com/Caprinomics)